F 13

```
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          LLLLLLLLLLLLLL
MMM           MMM          TTT          HHH           HHH   RRR           RRR          TTT          LLLLLLLLLLLLLL
MMM           MMM          TTT          HHH           HHH   RRR           RRR          TTT          LLLLLLLLLLLLLL
```

**FILE**ID**MTHDSINCO

C 12

```
MM      MM  TTTTTTTTTT  HH      HH  DDDDDDDD      SSSSSSSS    IIIIII    NN      NN    CCCCCCCC    OOOOOO
MM      MM  TTTTTTTTTT  HH      HH  DDDDDDDD      SSSSSSSS    IIIIII    NN      NN    CCCCCCCC    OOOOOO
MMMM  MMMM      TT      HH      HH  DD      DD  SS              II      NN      NN  CC          OO      OO
MMMM  MMMM      TT      HH      HH  DD      DD  SS              II      NN      NN  CC          OO      OO
MM  MM  MM      TT      HH      HH  DD      DD  SS              II      NNNN    NN  CC          OO      OO
MM  MM  MM      TT      HH      HH  DD      DD  SS              II      NNNN    NN  CC          OO      OO
MM      MM      TT      HHHHHHHHHH  DD      DD    SSSSSS        II      NN  NN  NN  CC          OO      OO
MM      MM      TT      HHHHHHHHHH  DD      DD    SSSSSS        II      NN  NN  NN  CC          OO      OO
MM      MM      TT      HH      HH  DD      DD          SS      II      NN    NNNN  CC          OO      OO
MM      MM      TT      HH      HH  DD      DD          SS      II      NN    NNNN  CC          OO      OO
MM      MM      TT      HH      HH  DD      DD          SS      II      NN      NN  CC          OO      OO    ....
MM      MM      TT      HH      HH  DD      DD          SS      II      NN      NN  CC          OO      OO    ....
MM      MM      TT      HH      HH  DDDDDDDD      SSSSSSSS    IIIIII    NN      NN    CCCCCCCC    OOOOOO      ....
MM      MM      TT      HH      HH  DDDDDDDD      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$DSINCOS      ; Floating Point Sine, Cosine and Sincos
0000                                   ;        Functions
0000          .IDENT /2-007/           ; File: MTHDSINCOS.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$DSIN and MTH$DCOS are functions which return the floating point
0000  ; sine or cosine value of their single precision floating point argu-
0000  ; ment (radians). The call is standard call-by-reference.
0000  ; MTH$DSIN_R7 and MTH$DCOS_R7 are special routines which are the same
0000  ; as MTH$DSIN and MTH$DCOS  except  a faster non-standard JSB call is
0000  ; used with the argument in R0 and no registers are saved.
0000  ;
0000  ; MTH$DSINCOS  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$DSINCOS_R7 is a special
0000  ; routine  which  is  the  same  as  MTH$DSINCOS, except a faster non-
0000  ; standard JSB call is used with the argument in R0 and no  registers
0000  ; are saved.
0000  ;
0000  ; MTH$DSIND and MTH$DCOSD 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$DSIND_R7 and MTH$DCOSD_R7 are special routines which are the same
0000  ; as MTH$DSIND and MTH$DCOSD  except  a faster non-standard JSB call is
0000  ; used with the argument in R0 and no registers are saved.
0000  ;
0000  ; MTH$DSINCOSD is  a routine which returns the floating point sine and
0000  ; cosine value of its single precision floating point degree argument.
0000  ; The call is standard call-by-reference. MTH$DSINCOSD_R7 is a special
0000  ; routine  which  is  the  same  as MTH$DSINCOSD, 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  ; VERSION:     1
0000  ;
0000  ; HISTORY:
0000  ; AUTHOR:
0000  ;           MARY PAYNE & JUD LEONARD, 25-MAY-78:    Version 0
0000  ;
0000  ; MODIFIED BY:
0000  ;
0000  ; 1-1   Tryggve Fossum, 28-May-78
0000  ;
0000  ;
0000  ; VERSION:       2
0000  ;
0000  ; HISTORY:
0000  ; AUTHOR:
0000  ;           BOB HANEK, 25-MAY-78:   Version 2
0000  ;
0000  ;
0000  ; Edit history for Version 2
0000  ;
0000  ; 2-001 - Fixed overflow problem for large radian arguments. RNH 09-Sept-81
0000  ; 2-002 - Included check for A2 = 0 in DSINCOS for small arguments. RNH
0000  ;         22-Sept-81
0000  ; 2-003 - Change DSINCOS so that R6/R7 = |X| instead of X.  RNH 29-Sep-81
0000  ; 2-004 - Modified logic for converting reduced argument from integer to
0000  ;         to floating format to avoid modifying the exponent of a floating
0000  ;         point zero.  RNH 21-Oct-81
0000  ; 2-005 - Modified cosine evaluation logic to check the magnitude of YHI
0000  ;         instead of YLO.  RNH 01-Nov-81
0000  ; 2-006 - Modified negative argument logic for DSINCOSD to eliminate bug
0000  ;         uncovered by FORTRAN QA.
0000  ;       - Modified REDUCE_LARGE logic to fix bug detected in QAR 896.
0000  ;         RNH 14-Jan-82
0000  ; 2-007 - Corrected the FFS and FFC instructions in REDUCE_LARGE to properly
0000  ;         test bits 0 through 20. The loss of accuracy from only testing 20
0000  ;         bits was detected in an SPR. Cleaned up some comments.  JCW 8-JUN-84
```

```
                    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
                    0000          .EXTRN  MTH$$SIGNAL
                    0000          .EXTRN  MTH$K_FLOUNDMAT
                    0000          .EXTRN  MTH$$JACKET_TST
                    0000  ;
                    0000  ; EQUATED SYMBOLS:
                    0000
           0000000B 0000          X_1_OV_45       = ^X0B
                    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  D_PI_OV_4:
68C2A221 0FDA4049   0000          .QUAD   ^X68C2A2210FDA4049      ; 0.7853981633974483E+00
                    0008  D_9_PI_OV_4:
95DAF665 31D541E2   0008          .QUAD   ^X95DAF66531D541E2      ; 0.7068583470577035E+01
                    0010  D_3_PI_OV_4:
0E92F999 CBE34116   0010          .QUAD   ^X0E92F999CBE34116      ; 0.2356194490192345E+01
                    0018  D_5_PI_OV_4:
C2F34AA9 53D1417B   0018          .QUAD   ^XC2F34AA953D1417B      ; 0.3926990816987242E+01
                    0020  D_7_PI_OV_4:
3BAA4DDD EDDF41AF   0020          .QUAD   ^X3BAA4DDDEDDF41AF      ; 0.5497787143782138E+01
                    0028  D_2_OV_PI:
44156E4E F9834022   0028          .QUAD   ^X44156E4EF9834022      ; 0.6366197723675813E+00
                    0030
                    0030  D_45:
00000000 00004334   0030          .QUAD   ^X0000000000004334      ; 0.4500000000000000E+02
                    0038  D_M45:
00000000 0000C334   0038          .QUAD   ^X000000000000C334      ; -.4500000000000000E+02
                    0040  D_SMALLD:
0FBED31E 2EE035E5   0040          .QUAD   ^X0FBED31E2EE035E5      ; 0.4268868231257969E-06
                    0048  D_1_OV_45:
60B6B60B 0B603DB6   0048          .QUAD   ^X60B6B60B0B603DB6      ; 0.2222222222222222E-01
                    0050  D_CONVERT:
9C8B294E A3513BEF   0050          .QUAD   ^X9C8B294EA3513BEF      ; 0.1828292519943295E-02
                    0058  D_90_OV_PI:
0FBED31E 2EE042E5   0058          .QUAD   ^X0FBED31E2EE042E5      ; 0.2864788975654116E+02
```

```
                    0060  D_SMALLEST_DEG:
0FBED31E 2EE00365   0060          .QUAD    ^X0FBED31E2EE00365          ; 0.1683771628589691E-36
                    0068
                    0068
                    0068  PI_OV_2:
                    0068  ; pi/2
68C2A221 0FDA40C9   0068          .QUAD    ^X68C2A2210FDA40C9          ; 0.1570796326794897E+01
03708A2E 131923D3   0070          .QUAD    ^X03708A2E131923D3          ; 0.5721188726109832E-17
44531270 89480766   0078          .QUAD    ^X4453127089480766          ; 0.4335905065061890E-34
                    0080  ; pi
68C2A221 0FDA4149   0080          .QUAD    ^X68C2A2210FDA4149          ; 0.3141592653589793E+01
03708A2E 13192453   0088          .QUAD    ^X03708A2E13192453          ; 0.1144237745221966E-16
44531270 894807E6   0090          .QUAD    ^X44531270894807E6          ; 0.8671810130123781E-34
                    0098  ; 3*pi/2
0E92F999 CBE34196   0098          .QUAD    ^X0E92F999CBE34196          ; 0.4712388980384690E+01
BEB66C2E D8D6A530   00A0          .QUAD    ^XBEB66C2ED8D6A530          ; -.3834758505292833E-16
333E0DD4 E6F6082C   00A8          .QUAD    ^X333E0DD4E6F6082C          ; 0.1300771519518567E-33
                    00B0  ;2*pi
68C2A221 0FDA41C9   00B0          .QUAD    ^X68C2A2210FDA41C9          ; 0.6283185307179586E+01
03708A2E 131924D3   00B8          .QUAD    ^X03708A2E131924D3          ; 0.2288475490443933E-16
44531270 89480866   00C0          .QUAD    ^X4453127089480866          ; 0.1734362026024756E-33
                    00C8
```

```
                   00C8
                   00C8           .SBTTL  COEFFICIENT TABLES       -       Series Coefficients
                   00C8
                   00C8
                   00C8
                   00C8
                   00C8     ;
                   00C8     ; Polynomial Coefficient tables for arguments in radians
                   00C8     ;
                   00C8
                   00C8     COSTBR1:              ; DCOS coefficients for arguments less than 1/2
699DF786 B56AAE47  00C8           .QUAD   ^X699DF786B56AAE47      ; C7 = -.1135212320578394E-10
49E73CCE 74AA320F  00D0           .QUAD   ^X49E73CCE74AA320F      ; C6 = 0.2087555514567788E-08
CC8A7F10 F27BB593  00D8           .QUAD   ^XCC8A7F10F27BB593      ; C5 = -.2755731286569608E-06
B3EDCD6B 0D0038D0  00E0           .QUAD   ^XB3EDCD6B0D0038D0      ; C4 = 0.2480158728289946E-04
B166B609 0B60BBB6  00E8           .QUAD   ^XB166B6090B60BBB6      ; C3 = -.1388888888885896E-02
A99AAAAA AAAA3E2A  00F0           .QUAD   ^XA99AAAAAAAAA3E2A      ; C2 = 0.4166666666666643E-01
FFFFFFFF FFFFBFFF  00F8           .QUAD   ^XFFFFFFFFFFFFBFFF      ; C1 = -.5000000000000000E+00
00000000 00004080  0100           .QUAD   ^X0000000000004080      ; C0 = 0.1000000000000000E+01
         00000008  0108     COSLENR1 = .-COSTBR1/8
                   0108
                   0108     COSTBR2:              ; DCOS coefficients for arguments greater than 1/2
699DF786 B56AAE47  0108           .QUAD   ^X699DF786B56AAE47      ; C7 = -.1135212320578394E-10
49E73CCE 74AA320F  0110           .QUAD   ^X49E73CCE74AA320F      ; C6 = 0.2087555514567788E-08
CC8A7F10 F27BB593  0118           .QUAD   ^XCC8A7F10F27BB593      ; C5 = -.2755731286569608E-06
B3EDCD6B 0D0038D0  0120           .QUAD   ^XB3EDCD6B0D0038D0      ; C4 = 0.2480158728289946E-04
B166B609 0B60BBB6  0128           .QUAD   ^XB166B6090B60BBB6      ; C3 = -.1388888888885896E-02
A99AAAAA AAAA3E2A  0130           .QUAD   ^XA99AAAAAAAAA3E2A      ; C2 = 0.4166666666666643E-01
F7326202 03392404  0138           .QUAD   ^XF732620203392404      ; C1 = 0.7156417079102195E-17
E9809A22 4BFDA029  0140           .QUAD   ^XE9809A224BFDA029      ; C0 = -.3584999999999999E-19
         00000008  0148     COSLENR2 = .-COSTBR2/8
                   0148
                   0148     SINTBR:               ; DSIN coefficients
24F1F2B5 4C4AAC55  0148           .QUAD   ^X24F1F2B54C4AAC55      ; C7 = -.7577867884012712E-12
DA66F085 903A3030  0150           .QUAD   ^XDA66F085903A3030      ; C6 = 0.1605834762322461E-09
2AF0320D 3229B3D7  0158           .QUAD   ^X2AF0320D3229B3D7      ; C5 = -.2505210473826733E-07
D2FC2984 EF1D3738  0160           .QUAD   ^XD2FC2984EF1D3738      ; C4 = 0.2755731921339017E-05
3FDED00C 0D00BA50  0168           .QUAD   ^X3FDED00C0D00BA50      ; C3 = -.1984126984125311E-03
884D8888 88883D08  0170           .QUAD   ^X884D888888883D08      ; C2 = 0.8333333333333320E-02
AAABAAAA AAAABF2A  0178           .QUAD   ^XAAABAAAAAAAABF2A      ; C1 = -.1666666666666667E+00
4800F1E9 077E9E0E  0180           .QUAD   ^X4800F1E9077E9E0E      ; C0 = -.1879741879570161E-20
         00000008  0188     SINLENR = .-SINTBR/8
                   0188
                   0188
                   0188
                   0188
                   0188
                   0188     ;
                   0188     ; Polynomial coefficients for arguments in cycles
                   0188     ;
                   0188
                   0188     COSTBC1:              ; DCOS coefficients for arguments less than 2/pi
0AD5CCAC 2C35ABD9  0188           .QUAD   ^X0AD5CCAC2C35ABD9      ; C7 = -.3857762037200000E-12
6F06EA8A E60A2FFC  0190           .QUAD   ^X6F06EA8AE60A2FFC      ; C6 = 0.1150049702426300E-09
00B11EA4 68F6B3D3  0198           .QUAD   ^X00B11EA468F6B3D3      ; C5 = -.2461136382637005E-07
F91E4181 FA833770  01A0           .QUAD   ^XF91E4181FA833770      ; C4 = 0.3590860445885820E-05
6ADFF1E4 E9E3BAAA  01A8           .QUAD   ^X6ADFF1E4E9E3BAAA      ; C3 = -.3259918869266876E-03
```

```
D54E40DA E0F83D81  01B0              .QUAD    ^XD54E40DAE0F83D81     ; C2 = 0.1585434424381541E-01
2EF24DF2 E9E6BF9D  01B8              .QUAD    ^X2EF24DF2E9E6BF9D     ; C1 = -.3084251375340425E+00
E9809A22 4BFDA029  01C0              .QUAD    ^XE9809A224BFDA029     ; C0 = -.3584999999999999E-19
         00000008  01C8   COSLENC1 = .-COSTBC1/8
                   01C8
                   01C8   COSTBC2:            ; DCOS coefficients for arguments greater than 2/pi
0AD5CCAC 2C35ABD9  01C8              .QUAD    ^X0AD5CCAC2C35ABD9     ; C7 = -.3857762037200000E-12
6F06EA8A E60A2FFC  01D0              .QUAD    ^X6F06EA8AE60A2FFC     ; C6 = 0.1150049702426300E-09
00B11EA4 68F6B3D3  01D8              .QUAD    ^X00B11EA468F6B3D3     ; C5 = -.2461136382637005E-07
F91E4181 FA833770  01E0              .QUAD    ^XF91E4181FA833770     ; C4 = 0.3590860445885820E-05
6ADFF1E4 E9E3BAAA  01E8              .QUAD    ^X6ADFF1E4E9E3BAAA     ; C3 = -.3259918869266876E-03
D54E40DA E0F83D81  01F0              .QUAD    ^XD54E40DAE0F83D81     ; C2 = 0.1585434424381541E-01
77916F91 4F32BE6F  01F8              .QUAD    ^X77916F914F32BE6F     ; C1 = -.5842513753404245E-01
E9809A22 4BFDA029  0200              .QUAD    ^XE9809A224BFDA029     ; C0 = -.3584999999999999E-19
         00000008  0208   COSLENC2 = .-COSTBC2/8
                   0208
                   0208   SINTBC:             ; DSIN coef for arg in cycles
86037C40 2C65A9B6  0208              .QUAD    ^X86037C402C65A9B6     ; C7 = -.2022531292930000E-13
D11BCC06 77632DF4  0210              .QUAD    ^XD11BCC0677632DF4     ; C6 = 0.6948152035052200E-11
0477A15F 83A5B1F1  0218              .QUAD    ^X0477A15F83A5B1F1     ; C5 = -.1757247417617081E-08
693342E1 3C1A35A8  0220              .QUAD    ^X693342E13C1A35A8     ; C4 = 0.3133616889173253E-06
5DE87315 6966B919  0228              .QUAD    ^X5DE873156966B919     ; C3 = -.3657620418214640E-04
56C73BAD 35E33C23  0230              .QUAD    ^X56C73BAD35E33C23     ; C2 = 0.2490394570192716E-02
F296312D 5DE7BEA5  0238              .QUAD    ^XF296312D5DE7BEA5     ; C1 = -.8074551218828078E-01
8C232216 FDAA3E10  0240              .QUAD    ^X8C232216FDAA3E10     ; C0 = 0.3539816339744831E-01
         00000008  0248   SINLENC = .-SINTBC/8
                   0248
                   0248
                   0248
                   0248
                   0248
                   0248   ;
                   0248   ; Polynomial coefficients for arguments in degrees
                   0248   ;
                   0248
                   0248   COSDTB2:            ; DCOS coefficients for arguments less than 90/pi
00C69319 09F3856B  0248              .QUAD    ^X00C6931909F3856B     ; C7 = -.2762868673216389E-35
6EB1B88D 50A40F07  0250              .QUAD    ^X6EB1B88D50A40F07     ; C6 = 0.1667886312398853E-29
8C1C14B5 B11D985F  0258              .QUAD    ^X8C1C14B5B11D985F     ; C5 = -.7227873495985315E-24
8EECE026 1D5A217C  0260              .QUAD    ^X8EECE0261D5A217C     ; C4 = 0.2135494301985905E-18
C57ADDE3 CDC2AA30  0268              .QUAD    ^XC57ADDE3CDC2AA30     ; C3 = -.3925831985734635E-13
E39C541E D88B3284  0270              .QUAD    ^XE39C541ED88B3284     ; C2 = 0.3866323851562972E-08
FDCCD8A0 B50EBA1F  0278              .QUAD    ^XFDCCD8A0B50EBA1F     ; C1 = -.1523087098933543E-03
00000000 00004080  0280              .QUAD    ^X0000000000004080     ; C0 = 0.1000000000000000E+01
         00000007  0288   COSDLN2 = .-COSDTB2/8 - 1
                   0288
                   0288   COSDTB1:            ; DCOS coefficients for arguments greater than 90/pi
00C69319 09F3856B  0288              .QUAD    ^X00C6931909F3856B     ; C7 = -.2762868673216389E-35
6EB1B88D 50A40F07  0290              .QUAD    ^X6EB1B88D50A40F07     ; C6 = 0.1667886312398853E-29
8C1C14B5 B11D985F  0298              .QUAD    ^X8C1C14B5B11D985F     ; C5 = -.7227873495985315E-24
8EECE026 1D5A217C  02A0              .QUAD    ^X8EECE0261D5A217C     ; C4 = 0.2135494301985905E-18
C57ADDE3 CDC2AA30  02A8              .QUAD    ^XC57ADDE3CDC2AA30     ; C3 = -.3925831985734635E-13
E39C541E D88B3284  02B0              .QUAD    ^XE39C541ED88B3284     ; C2 = 0.3866323851562972E-08
EE5FC507 A876B8FD  02B8              .QUAD    ^XEE5FC507A876B8FD     ; C1 = -.3023839739335430E-04
E9809A22 4BFDA029  02C0              .QUAD    ^XE9809A224BFDA029     ; C0 = -.3584999999999999E-19
         00000007  02C8   COSDLN1 = .-COSDTB1/8 - 1
                   02C8
```

```
                   02C8  SINDTB:                 ; DSIN coefficients
BDA56DF7 33F0808C  02C8          .QUAD   ^XBDA56DF733F0808C     ; C7 = -.3218900432111067E-38
19375C16 08060A3A  02D0          .QUAD   ^X19375C1608060A3A     ; C6 = 0.2239270886637075E-32
C2AEE060 B88B93B5  02D8          .QUAD   ^XC2AEE060B88B93B5     ; C5 = -.1146820010579771E-26
81336423 53021CFA  02E0          .QUAD   ^X8133642353021CFA     ; C4 = 0.4141267415665013E-21
B337FD49 B46CA5E1  02E8          .QUAD   ^XB337FD49B46CA5E1     ; C3 = -.9788384861609472E-16
6B1B5708 6CA72E6D  02F0          .QUAD   ^X6B1B57086CA72E6D     ; C2 = 0.1349601623163253E-10
7BC46CA1 DC10B66D  02F8          .QUAD   ^X7BC46CA1DC10B66D     ; C1 = -.8860961557012980E-06
9C8B294E A3513BEF  0300          .QUAD   ^X9C8B294EA3513BEF     ; C0 = 0.1828292519943296E-02
         00000007  0308  SINDLN = .-SINDTB/8 - 1
                   0308
```

```
          0308          .SBTTL  MTH$DSINCOS     -       Radian arguments
          0308
          0308
          0308  ;
          0308  ; FUNCTIONAL DESCRIPTION:
          0308  ;
          0308  ; The DSIN, DCOS and DSINCOS routines are based on octant reduction.  Given an
          0308  ; argument, x, it is written in the form
          0308  ;
          0308  ;               x = I1*(2*pi) + I*(pi/4) + Y1,
          0308  ;
          0308  ; where I1 and I are integers, 0 =< I < 8 and 0 =< Y1 < pi/4.  Since DSIN and
          0308  ; DCOS have a period of 2*pi it follows that
          0308  ;
          0308  ;               DSIN(x) = DSIN(I*(pi/4) + Y1)) and
          0308  ;               DCOS(x) = DCOS(I*(pi/4) + Y1)).
          0308  ;
          0308  ; Using the trigonometric identities for the sum and difference of two angles,
          0308  ; the following table can be generated:
          0308  ;
          0308  ;       If I =                  then DSIN(x) =                  and DCOS(x) =
          0308  ;       ------                  -------------                   -------------
          0308  ;         0                       DSIN(Y1)                        DCOS(Y1)
          0308  ;         1                       DCOS(pi/4-Y1)                   DSIN(pi/4-Y1)
          0308  ;         2                       DCOS(Y1)                       -DSIN(Y1)
          0308  ;         3                       DSIN(pi/4-Y1)                  -DCOS(pi/4-Y1)
          0308  ;         4                      -DSIN(Y1)                       -DCOS(Y1)
          0308  ;         5                      -DCOS(pi/4-Y1)                  -DSIN(pi/4-Y1)
          0308  ;         6                      -DCOS(Y1)                        DSIN(Y1)
          0308  ;         7                      -DSIN(pi/4-Y1)                   DCOS(pi/4-Y1)
          0308  ;
          0308  ; Let Y be defined as Y = Y1 if I is even and Y = pi/4 - Y1, if I is odd, then
          0308  ; each entry of the above table is of the for +/-DSIN(Y) or +/-DCOS(Y).  Based
          0308  ; on the above remarks, the DSIN, DCOS and DSINCOS routines process the input
          0308  ; argument x, to obtain I and Y, and based on I selects a suitable polynomial
          0308  ; approximation, p(Y), to evaluate the desired fuction.
          0308
          0308  ;
          0308  ; INPUT PARAMETERS:
          0308  ;
00000004  0308          LONG    = 4
00000004  0308          x       = 1*LONG                ; x is input angle in radians
00000008  0308          sine    = 2*LONG                ; sine is DSIN(x)
0000000C  0308          cosine  = 3*LONG                ; cosine is DCOS(x)
          0308
```

```
                              0308
                              0308
                              0308  ;
                              0308  ; Return sine and cosine of argument
                              0308  ;
                              0308
                              0308
                        00FC  0308          .ENTRY  MTH$DSINCOS, ^M<R2, R3, R4, R5, R6,R7>
                              030A
                              030A          MTH$FLAG_JACKET
                              030A
        6D  00000000'GF   9E  030A          MOVAB   G^MTH$$JACKET_HND, (FP)
                              0311                                          ; set handler address to jacket
                              0311                                          ; handler
                              0311
                              0311
            50  04 BC     70  0311          MOVD    @x(AP), R0
              00000390'EF 16  0315          JSB     MTH$DSINCOS_R7
              08 BC   50  7D  031B          MOVQ    R0, @sine(AP)
              0C BC   52  7D  031F          MOVQ    R2, @cosine(AP)
                          04  0323          RET
                              0324
                              0324
                              0324
                              0324          .SBTTL  MTH$DSIN
                              0324
                              0324  ;
                              0324  ; Return sine of argument
                              0324  ;
                              0324
                              0324
                        00FC  0324          .ENTRY  MTH$DSIN, ^M<R2, R3, R4, R5, R6, R7>
                              0326
                              0326          MTH$FLAG_JACKET
                              0326
        6D  00000000'GF   9E  0326          MOVAB   G^MTH$$JACKET_HND, (FP)
                              032D                                          ; set handler address to jacket
                              032D                                          ; handler
                              032D
                              032D
            50  04 BC     70  032D          MOVD    @x(AP), R0
              00000493'EF 16  0331          JSB     MTH$DSIN_R7
                          04  0337          RET
                              0338
                              0338
                              0338
                              0338          .SBTTL  MTH$DCOS
                              0338
                              0338  ;
                              0338  ; Return cosine of argument
                              0338  ;
                              0338
                              0338
                        00FC  0338          .ENTRY  MTH$DCOS, ^M<R2, R3, R4, R5, R6, R7>
                              033A
                              033A          MTH$FLAG_JACKET
                              033A
```

```
        6D  00000000'GF   9E  033A          MOVAB   G^MTH$$JACKET_HND, (FP)
                              0341                                          ; set handler address to jacket
                              0341                                          ; handler
                              0341
                              0341    398
           50  04 BC      70  0341    399   MOVD    @x(AP), R0
            0000051A'EF   16  0345    400   JSB     MTH$DCOS_R7
                          04  034B    401   RET
                              034C    402
```

```
              034C          .SBTTL  MTH$DSINCOSD    -       Degrees
              034C
              034C
              034C   ;
              034C   ; FUNCTIONAL DESCRIPTION:
              034C   ;
              034C   ; The DSIND, DCOSD and DSINCOSD routines are based on octant reduction.  Given
              034C   ; an argument, x, it is written in the form
              034C   ;
              034C   ;       x = I1*360 + I*45 + Y1,
              034C   ;
              034C   ; where I1 and I are integers, 0 =< I < 8 and 0 =< Y1 < 45.  Since DSIND and
              034C   ; DCOSD have a period of 360 it follows that
              034C   ;
              034C   ;       DSIND(x) = DSIND(I*45 + Y1) and
              034C   ;       DCOSD(x) = DCOSD(I*45 + Y1).
              034C   ;
              034C   ; Using the trigonometric identities for the sum and difference of two angles,
              034C   ; the following table can be generated:
              034C   ;
              034C   ;       If I =          then DSIND(x) =            and DCOSD(x) =
              034C   ;       ------          -------------              -------------
              034C   ;         0                DSIND(Y1)                  DCOSD(Y1)
              034C   ;         1                DCOSD(45-Y1)               DSIND(45-Y1)
              034C   ;         2                DCOSD(Y1)                 -DSIND(Y1)
              034C   ;         3                DSIND(45-Y1)              -DCOSD(45-Y1)
              034C   ;         4               -DSIND(Y1)                 -DCOSD(Y1)
              034C   ;         5               -DCOSD(45-Y1)              -DSIND(45-Y1)
              034C   ;         6               -DCOSD(Y1)                  DSIND(Y1)
              034C   ;         7               -DSIND(45-Y1)               DCOS(45-Y1)
              034C   ;
              034C   ; Let Y be defined as Y = Y1 if I is even and Y = 45 - Y1, if I is odd, then
              034C   ; each entry of the above table is of the for +/-DSIN(Y) or +/-DCOS(Y).  Based
              034C   ; on the above remarks, the DSIND, DCOSD and DSINCOSD routines process the input
              034C   ; argument x, to obtain I and Y, and based on I selects a suitable polynomial
              034C   ; approximation, p(Y), to evaluate the desired fuction.
              034C
              034C
    00000004  034C          LONG = 4
    00000008  034C          sind = 2*LONG
    0000000C  034C          cosd = 3*LONG
              034C
```

```
                       00FC  034C   447          .ENTRY  MTH$DSINCOSD     ^M<R2, R3, R4, R5, R6, R7>
                             034E   448
                             034E   449          MTH$FLAG_JACKET
                             034E
     6D    00000000'GF   9E  034E                MOVAB   G^MTH$$JACKET_HND, (FP)
                             0355                                                ; set handler address to jacket
                             0355                                                ; handler
                             0355
                             0355   450
       50     04 BC      70  0355   451          MOVD    @X(AP), R0
          000005A8'EF    16  0359   452          JSB     MTH$DSINCOSD_R7
        08 BC     50     7D  035F   453          MOVQ    R0, @sind(AP)
        0C BC     52     7D  0363   454          MOVQ    R2, @cosd(AP)
                             0367   455
                         04  0367   456          RET
                             0368   457
                             0368   458
                             0368   459
                       00FC  0368   460          .ENTRY  MTH$DSIND        ^M<R2, R3, R4, R5, R6, R7>
                             036A   461
                             036A   462          MTH$FLAG_JACKET
                             036A
     6D    00000000'GF   9E  036A                MOVAB   G^MTH$$JACKET_HND, (FP)
                             0371                                                ; set handler address to jacket
                             0371                                                ; handler
                             0371
                             0371   463
       50     04 BC      70  0371   464          MOVD    @X(AP), R0
          00000603'EF    16  0375   465          JSB     MTH$DSIND_R7
                             037B   466
                         04  037B   467          RET
                             037C   468
                             037C   469
                             037C   470
                       00FC  037C   471          .ENTRY  MTH$DCOSD        ^M<R2, R3, R4, R5, R6, R7>
                             037E   472
                             037E   473          MTH$FLAG_JACKET
                             037E
     6D    00000000'GF   9E  037E                MOVAB   G^MTH$$JACKET_HND, (FP)
                             0385                                                ; set handler address to jacket
                             0385                                                ; handler
                             0385
                             0385   474
       50     04 BC      70  0385   475          MOVD    @X(AP), R0
          00000661'EF    16  0389   476          JSB     MTH$DCOSD_R7
                             038F   477
                         04  038F   478          RET
                             0390   479
```

```
                          0390          .SBTTL MTH$DSINCOS_R7
                          0390
                          0390  ; This routine computes the DSIN and DCOS of the G-format value of R0/R1.  The
                          0390  ; computation is performed one of three ways depending on the size of the
                          0390  ; input argument, X:
                          0390  ;
                          0390  ;        1) If |X| < pi/4, then X is used directly in polynomial approximation
                          0390  ;           of DSIN and DCOS.
                          0390  ;        2) If pi/4 =< |X| < 9*pi/4, then the subroutine REDUCE_MEDIUM is called
                          0390  ;           to reduce the argument to an equivalent argument in radians, Y, and
                          0390  ;           the octant, I, containing the argument.  Y is then evaluated in two
                          0390  ;           polynomials chosen as a function of I, to compute DSIN(X) and DCOS(X).
                          0390  ;        3) If 9*pi/4 =< |X|, then the subroutine REDUCE_LARGE is called to
                          0390  ;           reduce the argument to an equivalent argument in cycles, Y, and the
                          0390  ;           octant, I, containining the argument.  Y is then evaluated in two
                          0390  ;           polynomials chosen as a function of I, to compute DSIN(X) and DCOS(X).
                          0390
                          0390  MTH$DSINCOS_R7::
          56     50    70 0390          MOVD    R0, R6                  ; R6 = X
                 0F    18 0393          BGEQ    POS_SINCOS              ;
          0000039F'EF  16 0395          JSB     SINCOS                  ; R0/R1 = DSIN(|X|), R2/R3 = DCOS(X)
          50     50    72 039B          MNEGD   R0, R0                  ; R0/R1 = DSIN(X)
                       05 039E          RSB
                          039F
                          039F  SINCOS:
     56   8000 8F      AA 039F          BICW    #^X8000, R6             ; R6/R7 = |X|
                          03A4  POS_SINCOS:
     56   FC58 CF      71 03A4          CMPD    D_PI_OV_4, R6           ; Compare pi/4 with |X|
                 34    14 03A9          BGTR    SMALL_SINCOS            ; No argument reduction is necessary
     56   FC59 CF      71 03AB          CMPD    D_9_PI_OV_4, R6         ; Compare 9*pi/4 with |X|
                 03    18 03B0          BGEQ    1$
               00B4    31 03B2          BRW     LARGE_SINCOS            ; Use special logic for |X| > 9*pi/4
                          03B5
                          03B5  ;
                          03B5  ;  pi/4 =< |X| < 9*pi/4
                          03B5  ;
       0000069A'EF     16 03B5  1$:     JSB     REDUCE_MEDIUM           ; Medium argument reduction routine
                          03BB                                          ; R4/R7 = Y = reduced argument
                          03BB                                          ; R2 = octant
          7E     54    7D 03BB          MOVQ    R4, -(SP)               ; Save reduced argument on stack
          7E     56    7D 03BE          MOVQ    R6, -(SP)               ;
                 52    DD 03C1          PUSHL   R2                      ; Save octant bits on stack
       00000535'EF     16 03C3          JSB     M_COS                   ; R0/R1 = DCOS(X)
          52     8E    D0 03C9          MOVL    (SP)+, R2               ; R2 = Octant bits
          56     8E    7D 03CC          MOVQ    (SP)+, R6               ;
          54     8E    7D 03CF          MOVQ    (SP)+, R4               ; R4/R7 = reduced argument
          7E     50    7D 03D2          MOVQ    R0, -(SP)               ; Save DCOS(X) on stack
       000004BA'EF     16 03D5          JSB     M_SIN                   ; R0/R1 = DSIN(X)
          52     8E    7D 03DB          MOVQ    (SP)+, R2               ; R2/R3 = DCOS(X)
                       05 03DE          RSB
                          03DF  ;
                          03DF  ; Logic for small arguments.  |X| < pi/4.
                          03DF  ;
                          03DF
                          03DF  SMALL_SINCOS:
     56   4000 8F      B1 03DF          CMPW    #^X4000, R6             ; Compare 1/2 with |X|
                 33    15 03E4          BLEQ    2$                      ; Sufficent overghang not available
```

```
         56     3280 8F   B1 03E6            CMPW    #^X3280, R6              ; Compare with 2^-28
                     23   18 03EB            BGEQ    1$                       ; No polynomial evaluation is needed
         54     56   56   65 03ED            MULD3   R6, R6, R4               ; R4/R5 = X*X
                7E   54   7D 03F1            MOVQ    R4, -(SP)                ; Put X*X on stack
    FCCE CF     07   54   75 03F4            POLYD   R4, #COSLENR1-1, COSTBR1; R0/R1 = DCOS(X)
                54   6E   7D 03FA            MOVQ    (SP), R4                 ; R4/R5 = X*X
                6E   50   7D 03FD            MOVQ    R0, (SP)                 ; Save DCOS(X) on stack
    FD42 CF     07   54   75 0400            POLYD   R4, #SINLENR-1, SINTBR   ; R0/R1 = g(X^2)
                50   56   64 0406            MULD    R6, R0                   ; R0/R1 = X*g(X^2)
                50   56   60 0409            ADDD    R6, R0                   ; R0/R1 = DSIN(X)
                52   8E   7D 040C            MOVQ    (SP)+, R2                ; R2/R3 = DCOS(X)
                          05 040F            RSB
                             0410
                52   08   70 0410    1$:     MOVD    #1.0, R2                 ; R2/R3 = 1.0 = DCOS(X)
         50     8000 8F   AA 0413            BICW    #^X8000, R0              ; R0/R1 = |X|
                          05 0418            RSB
                             0419
                             0419
         54     56   56   65 0419    2$:     MULD3   R6, R6, R4               ; R4/R5 = X^2
                7E   54   7D 041D            MOVQ    R4, -(SP)                ; Save X^2
    FCE2 CF     07   54   75 0420            POLYD   R4, #COSLENR2-1, COSTBR2; R0/R1 = Q(Y^2)
                54   56   7D 0426            MOVQ    R6, R4                   ; R4/R5 = X
       55   FFFF1FFF 8F   CA 0429            BICL    #^XFFFF1FFF, R5          ; R4/R5 = XHI
         7E     56   54   63 0430            SUBD3   R4, R6, -(SP)            ; (SP) = XLO
         52     54   56   61 0434            ADDD3   R6, R4, R2               ; R2/R3 = X + XHI
                52   8E   64 0438            MULD    (SP)+, R2                ; R2/R3 = XLO*(X + XHI) = A2
                     08   13 043B            BEQL    3$                       ; Check for A2 = 0
         52     0080 8F   A2 043D            SUBW    #^X80, R2                ; R2/R3 = A2/2
                50   52   62 0442            SUBD    R2, R0                   ; R0/R1 = Q(Y^2) - A2/2
                54   54   64 0445    3$:     MULD    R4, R4                   ; R4/R5 = XHI^2
         54     0080 8F   A2 0448            SUBW    #^X80, R4                ; R4/R5 = XHI^2/2
                54   08   62 044D            SUBD    #1, R4                   ; R4/R5 = XHI^2/2 - 1
                50   54   62 0450            SUBD    R4, R0                   ; R0/R1 = DCOS(X)
                52   6E   7D 0453            MOVQ    (SP), R2                 ; R2/R3 = X^2
                6E   50   7D 0456            MOVQ    R0, (SP)                 ; Save DCOS(X)
    FCE9 CF     07   52   75 0459            POLYD   R2, #SINLENR-1, SINTBR   ; R0/R1 = Q(X^2)
                50   56   64 045F            MULD    R6, R0                   ; R0/R1 = X*Q(X^2)
                50   56   60 0462            ADDD    R6, R0                   ; R0/R1 = DSIN(X)
                52   8E   7D 0465            MOVQ    (SP)+, R2                ; R2/R3 = DCOS(X)
                          05 0468            RSB
                             0469
                             0469
                             0469    LARGE_SINCOS:
       00000718'EF        16 0469            JSB     REDUCE_LARGE             ; R4/R7 = reduced argument (in cycles)
                             046F                                             ; R2 = octant bits
                     52   DD 046F            PUSHL   R2                       ; Save octant bits on stack
                7E   56   7D 0471            MOVQ    R6, -(SP)                ; Save reduced
                7E   54   7D 0474            MOVQ    R4, -(SP)                ;     argument on stack
       0000057C'EF        16 0477            JSB     L_COS                    ; R0/R1 = DCOS(X)
                54   8E   7D 047D            MOVQ    (SP)+, R4                ; Reduced argument
                56   8E   7D 0480            MOVQ    (SP)+, R6                ;     in R4/R7
                52   8E   D0 0483            MOVL    (SP)+, R2                ; R2 = octant bits
                7E   50   7D 0486            MOVQ    R0, -(SP)                ; R2/R3 = DCOS(X)
       000004EE'EF        16 0489            JSB     L_SIN                    ; R0/R1 = DSIN(X)
                52   8E   7D 048F            MOVQ    (SP)+, R2                ; R2/R3 = DCOS(X)
                          05 0492            RSB
```

```
                          0493        .SBTTL MTH$DSIN_R7
                          0493
                          0493  ; This routine computes the DSIN of the G-format value of R0/R1.  The
                          0493  ; computation is performed one of three ways depending on the size of the
                          0493  ; input argument, X:
                          0493  ;
                          0493  ;        1) If |X| < pi/4, then X is used directly in a polynomial approximation
                          0493  ;           of DSIN.
                          0493  ;        2) If pi/4 =< |x| < 9*pi/4, then the subroutine REDUCE_MEDIUM is called
                          0493  ;           to reduce the argument to an equivalent argument in radians, Y, and
                          0493  ;           the octant, I, containing the argument.  Y is then evaluated in a
                          0493  ;           polynomial chosen as a function of I to compute DSIN(X).
                          0493  ;        3) If 9*pi/4 =< |X|, then the subroutine REDUCE_LARGE is called to
                          0493  ;           reduce the argument to an equivalent argument in cycles, Y, and the
                          0493  ;           octant, I, containing the argument.  Y is then evaluated in a
                          0493  ;           polynomial chosen as a function of I to compute DSIN(X).
                          0493
                          0493  MTH$DSIN_R7::
               50     73  0493          TSTD    R0                      ; Check the sign of R0
               0F     18  0495          BGEQ    POS_SIN
       000004A1'EF    16  0497          JSB     SIN                     ; R0/R1 = DSIN(|X|)
          50   50     72  049D          MNEGD   R0, R0                  ; R0/R1 = DSIN(X)
                      05  04A0          RSB
                          04A1
                          04A1  SIN:
  50    8000 8F       AA  04A1          BICW    #^X8000, R0             ; R0/R1 = |X|
                          04A6  POS_SIN:
  50    FB56 CF       71  04A6          CMPD    D_PI_OV_4, R0           ; Compare pi/4 with |X|
                 21   14  04AB          BGTR    SMALL_SIN               ; No argument reduction is necessary
  50    FB57 CF       71  04AD          CMPD    D_9_PI_OV_4, R0         ; Compare 9*pi/4 with |X|
                 34   19  04B2          BLSS    LARGE_SIN               ; Use special logic for |X| > 9*pi/4
                          04B4
                          04B4  ;
                          04B4  ;  pi/4 =< |X| < 9*pi/4
                          04B4  ;
       0000069A'EF    16  04B4          JSB     REDUCE_MEDIUM           ; Medium argument reduction routine
                          04BA                                          ; R4/R7 = Y = reduced argument
                          04BA                                          ; R2 = octant
  07    01    52      8F  04BA  M_SIN:  CASEB   R2, #1, #7              ; Branch to one of four polynomial
                          04BE                                          ;     evaluations depending on the
                   05B7'  04BE  1$:     .WORD   P_COS_R-1$
                   05B7'  04C0          .WORD   P_COS_R-1$
                   0653'  04C2          .WORD   N_SIN_R-1$
                   0653'  04C4          .WORD   N_SIN_R-1$
                   0602'  04C6          .WORD   N_COS_R-1$
                   0602'  04C8          .WORD   N_COS_R-1$
                   065D'  04CA          .WORD   P_SIN_R-1$
                   065D'  04CC          .WORD   P_SIN_R-1$              ;     octant bits.
                          04CE
                          04CE  ;
                          04CE  ; Logic for small arguments.  |X| < pi/4.
                          04CE  ;
                          04CE
                          04CE  SMALL_SIN:
  50    3280 8F       B1  04CE          CMPW    #^X3280, R0             ; Compare with 2^-28
                 12   18  04D3          BGEQ    1$                      ; No polynomial evaluation is needed
          56   50     7D  04D5          MOVQ    R0, R6                  ; R6/R7 = X
```

```
             50    50    64  04D8         MULD    R0,R0                      ; R0/R1 = X*X
     FC67 CF  07   50    75  04DB         POLYD   R0, #SINLENR-1, SINTBR     ; R0/R1 = g(x^2)
             50    56    64  04E1         MULD    R6, R0                     ; R0/R1 = X*g(x^2)
             50    56    60  04E4         ADDD    R6, R0                     ; R0/R1 = DSIN(X)
                         05  04E7  1$:    RSB
                             04E8
                             04E8
                             04E8  LARGE_SIN:
        00000718'EF      16  04E8         JSB     REDUCE_LARGE               ; R4/R7 = reduced argument (in cycles)
                             04EE                                            ; R2 = octant bits
                   54    D5  04EE  L_SIN: TSTL    R4                         ; Check for degenerate case
                   14    13  04F0         BEQL    DEGENERATE_CASE_SIN
                             04F2
          07   00  52    8F  04F2         CASEB   R2, #0, #7
                             04F6
                     06E4'   04F6  1$:    .WORD   P_SIN_C-1$
                     0640'   04F8         .WORD   P_COS_C-1$
                     0640'   04FA         .WORD   P_COS_C-1$
                     06E4'   04FC         .WORD   P_SIN_C-1$
                     06DA'   04FE         .WORD   N_SIN_C-1$
                     0688'   0500         .WORD   N_COS_C-1$
                     0688'   0502         .WORD   N_COS_C-1$
                     06DA'   0504         .WORD   N_SIN_C-1$
                             0506
                             0506
                             0506  DEGENERATE_CASE_SIN:
                             0506
             52    01    8A  0506         BICB    #1, R2                     ; Compute index as (R2 - 1)/2
      52  52  FF   8F    9C  0509         ROTL    #-1, R2, R2
          03   00  52    8F  050E         CASEB   R2, #0, #3
                             0512
                     07BB'   0512  1$:    .WORD   P_ONE-1$
                     07CB'   0514         .WORD   UNFL -1$
                     07BF'   0516         .WORD   N_ONE-1$
                     07CB'   0518         .WORD   UNFL -1$
                             051A
```

```
                           051A
                           051A
                           051A                .SBTTL MTH$DCOS_R7
                           051A
                           051A   ; This routine computes the DCOS of the G-format value of R0/R1.  The
                           051A   ; computation is performed one of three ways depending on the size of the
                           051A   ; input argument, X.  The processing is the same as described for MTH$DSIN_R4.
                           051A   ;
                           051A
                           051A   MTH$DCOS_R7::
                  50   73  051A                TSTD    R0                        ; Check for reserved operand
      50   8000 8F     AA  051C                BICW    #^X8000, R0               ; R0/R1 = |X|
      50   FADB CF     71  0521                CMPD    D_PI_OV_4, R0             ; Compare pi/4 with |X|
                21     14  0526                BGTR    SMALL_COS                 ; No argument reduction is necessary
      50   FADC CF     71  0528                CMPD    D_9_PI_OV_4, R0           ; Compare 9*pi/4 with |X|
                47     19  052D                BLSS    LARGE_COS                 ; Use special logic for |X| > 9*pi/4
                           052F
                           052F   ;
                           052F   ;  pi/4 =< |X| < 9*pi/4
                           052F   ;
       0000069A'EF     16  052F                JSB     REDUCE_MEDIUM             ; Medium argument reduction routine
                           0535                                                  ; R4/R7 = Y = reduced argument
                           0535                                                  ; R2 = octant
      07   01   52     8F  0535   M_COS:       CASEB   R2, #1, #7                ; Branch to one of four polynomial
                           0539                                                  ;     evaluations depending on the
                    05D8'  0539   1$:          .WORD   N_SIN_R-1$
                    05D8'  053B                .WORD   N_SIN_R-1$
                    0587'  053D                .WORD   N_COS_R-1$
                    0587'  053F                .WORD   N_COS_R-1$
                    05E2'  0541                .WORD   P_SIN_R-1$
                    05E2'  0543                .WORD   P_SIN_R-1$
                    053C'  0545                .WORD   P_COS_R-1$
                    053C'  0547                .WORD   P_COS_R-1$                ;     octant bits.
                           0549
                           0549   ;
                           0549   ; Logic for small arguments.  |X| < pi/4.
                           0549   ;
                           0549
                           0549   SMALL_COS:
      50   4000 8F     B1  0549                CMPW    #^X4000, R0               ; Compare 1/2 with |X|
                11     14  054E                BGTR    1$                        ; Sufficent overghang is available
           56   50     7D  0550                MOVQ    R0, R6                    ; R6/R7 = X
   57  FFFF1FFF 8F     CA  0553                BICL    #^XFFFF1FFF, R7           ; R6/R7 = XHI
      54   50   56     63  055A                SUBD3   R6, R0, R4                ; R4/R5 = XLO
              0521     31  055E                BRW     NEEDS_DOUBLE              ; Use special logic to obtain overhang
      50   3280 8F     B1  0561   1$:          CMPW    #^X3280, R0               ; Compare with 2^-28
                0A     18  0566                BGEQ    2$                        ; No polynomial evaluation is needed
           50   50     64  0568                MULD    R0,R0                     ; R0/R1 = X*X
 FB57 CF   07   50     75  056B                POLYD   R0, #COSLENR1-1, COSTBR1; R0/R1 = DCOS(X)
                       05  0571                RSB
                           0572
           50   08     70  0572   2$:          MOVD    #1.0, R0                  ; R0/R1 = 1.0 = DCOS(X)
                       05  0575                RSB
                           0576
                           0576
                           0576   LARGE_COS:
       00000718'EF     16  0576                JSB     REDUCE_LARGE              ; R4/R7 = reduced argument (in cycles)
```

```
                       057C                                          ; R2 = octant bits
            54    D5   057C  L_COS:   TSTL    R4                     ; Check for degenerate case
            14    13   057E           BEQL    DEGENERATE_CASE_COS
                       0580
   07   00  52    8F   0580           CASEB   R2, #0, #7
             05B2'     0584  1$:      .WORD   P_COS_C-1$
             0656'     0586           .WORD   P_SIN_C-1$
             064C'     0588           .WORD   N_SIN_C-1$
             05FA'     058A           .WORD   N_COS_C-1$
             05FA'     058C           .WORD   N_COS_C-1$
             064C'     058E           .WORD   N_SIN_C-1$
             0656'     0590           .WORD   P_SIN_C-1$
             05B2'     0592           .WORD   P_COS_C-1$
                       0594
                       0594
                       0594  DEGENERATE_CASE_COS:
                       0594
        52  01    8A   0594           BICB    #1, R2                 ; Compute index as (R2 - 1)/2
52   52  FF 8F    9C   0597           ROTL    #-1, R2, R2
   03   00  52    8F   059C           CASEB   R2, #0, #3
                       05A0
             073D'     05A0  1$:      .WORD   UNFL -1$
             0731'     05A2           .WORD   N_ONE-1$
             073D'     05A4           .WORD   UNFL -1$
             072D'     05A6           .WORD   P_ONE-1$
                       05A8
```

```
                        05A8        .SBTTL  MTH$DSINCOSD_R7
                        05A8
                        05A8  ; This routine computes the DSIND and DCOSD of the D-format value of R0/R1.
                        05A8  ; The computation is performed one of two ways depending on the size of the
                        05A8  ; input argument, X:
                        05A8  ;
                        05A8  ;         1) If |X| < 45, then X is used directly in polynomial approximation
                        05A8  ;            of DSIND and DCOSD.
                        05A8  ;         2) If 45 =< |x|, then the subroutine REDUCE_DEGREES is called to reduce
                        05A8  ;            the argument to an equivalent argument in degrees, Y, and the
                        05A8  ;            octant, I, containing the argument.  Y is then evaluated in two
                        05A8  ;            polynomials chosen as a function of I, to compute DSIND(X) and
                        05A8  ;            DCOSD(X).
                        05A8
                        05A8  MTH$DSINCOSD_R7::
            50    73    05A8        TSTD    R0
            0F    18    05AA        BGEQ    SINCOSD
   50  8000 8F    AA    05AC        BICW    #^X8000, R0             ; R0/R1 = |X|
  000005BB'EF     16    05B1        JSB     SINCOSD                 ; R0/R1 = DSIND(|X|)
                        05B7                                        ; R2/R3 = DCOSD(|X|)
       50   50    72    05B7        MNEGD   R0, R0                  ; R0/R1= -DSIND(|X|)
                  05    05BA        RSB
                        05BB
                        05BB  SINCOSD:
   50  FA71 CF    71    05BB        CMPD    D_45, R0                ; Compare 45 to |X|
            24    14    05C0        BGTR    SMALL_SINCOSD           ; special processing for small arg
  000009CB'EF     16    05C2        JSB     REDUCE_DEGREES          ; R6/R7 = reduced argument
                        05C8                                        ; R3 = octant
       7E   56    7D    05C8        MOVQ    R6, -(SP)               ; Save reduced arg
            53    DD    05CB        PUSHL   R3                      ; Save octant bits
  00000675'EF     16    05CD        JSB     EVAL_COSD               ; R0/R1 = DCOSD(Y)
       53   8E    D0    05D3        MOVL    (SP)+, R3               ; R3 = octant bits
       56   6E    7D    05D6        MOVQ    (SP), R6                ; R6/R7 = reduced argument
       6E   50    7D    05D9        MOVQ    R0, (SP)                ; Save DCOSD(Y)
  00000623'EF     16    05DC        JSB     EVAL_SIND               ; R0/R1 = DSIND(Y)
       52   8E    7D    05E2        MOVQ    (SP)+, R2               ; R2/R3 = DCOSD(Y)
                  05    05E5        RSB
                        05E6
                        05E6
                        05E6  SMALL_SINCOSD:
       5E   10    C2    05E6        SUBL    #16, SP                 ; Allocate 4 longwords on stack
       6E   50    7D    05E9        MOVQ    R0, (SP)                ; Save argument
  00000689'EF     16    05EC        JSB     SMALL_COSD              ; R0/R1 = DCOSD(|X|)
    08 AE   50    7D    05F2        MOVQ    R0, 8(SP)               ; Save DCOSD(|X|)
       50   8E    7D    05F6        MOVQ    (SP)+, R0               ; R0/R1 = argument
  00000637'EF     16    05F9        JSB     SMALL_SIND              ; R0/R1 = DSIND(X)
       52   8E    7D    05FF        MOVQ    (SP)+, R2               ; R2/R3 = DCOSD(|X|)
                  05    0602        RSB
```

```
                              0603            .SBTTL  MTH$DSIND_R7
                              0603
                              0603    ; This routine computes the DSIND of the D-format value of R0/R1.  The
                              0603    ; computation is performed one of two ways depending on the size of the input
                              0603    ; argument, X:
                              0603    ;
                              0603    ;       1) If |X| < 45, then X is used directly in polynomial approximation
                              0603    ;          of DSIND.
                              0603    ;       2) If 45 =< |x|, then the subroutine REDUCE_DEGREES is called to reduce
                              0603    ;          the argument to an equivalent argument in degrees, Y, and the
                              0603    ;          octant, I, containing the argument.  Y is then evaluated in two
                              0603    ;          polynomials chosen as a function of I, to compute DSIND(X).
                              0603
                              0603    MTH$DSIND_R7::
                  50    73    0603            TSTD    R0                      ; R0/R1 = X
                  0F    18    0605            BGEQ    POS_SIND                ;
          00000611'EF   16    0607            JSB     NEG_SIND                ; R0/R1 = DSIND(|X|)
             50   50    72    060D            MNEGD   R0, R0                  ; R0/R1 = -DSIND(|X|)
                        05    0610            RSB
                              0611
                              0611    NEG_SIND:
     50     8000 8F     AA    0611            BICW    #^X8000, R0             ; R0/R1 = |X|
                              0616    POS_SIND:
     50     FA16 CF     71    0616            CMPD    D_45, R0                ; Compare 45 to |X|
                  1A    14    061B            BGTR    SMALL_SIND              ; special processing for small arg
          000009CB'EF   16    061D            JSB     REDUCE_DEGREES          ; R6/R7 = reduced argument
                              0623                                            ; R3 = octant
                              0623
                              0623    EVAL_SIND:
     07     00    53    8F    0623            CASEB   R3, #0, #7
                     068E'    0627    1$:     .WORD   P_SIN_D-1$
                     05E1'    0629            .WORD   P_COS_D-1$
                     05E1'    062B            .WORD   P_COS_D-1$
                     068E'    062D            .WORD   P_SIN_D-1$
                     068B'    062F            .WORD   N_SIN_D-1$
                     062F'    0631            .WORD   N_COS_D-1$
                     062F'    0633            .WORD   N_COS_D-1$
                     068B'    0635            .WORD   N_SIN_D-1$
                              0637
                              0637
                              0637    SMALL_SIND:
     50     FA05 CF     71    0637            CMPD    D_SMALLD, R0            ; Compare 180/pi*2^-27 with |x|
                  06    14    063C            BGTR    1$                      ; No polynomial evaluation is
             56   50    7D    063E            MOVQ    R0, R6                  ;   necessary
                0671    31    0641            BRW     P_SIN_D                 ;
                  50    73    0644    1$:     TSTD    R0                      ; Check for zero
                  18    13    0646            BEQL    3$                      ; Return if R0 = 0
     50     FA14 CF     71    0648            CMPD    D_SMALLEST_DEG, R0      ; Check for possible underflow on
                  03    15    064D            BLEQ    2$                      ;   conversion to radians
                068B    31    064F            BRW     UNFL                    ; Underflow will occur on conversion
52   50     F9FA CF     65    0652    2$:     MULD3   D_CONVERT, R0, R2       ; R2/R3 = (pi/180 - 2^-6)*|x|
     50     0300 8F     A2    0658            SUBW    #^X300, R0              ; R0/R1 = |X|*2^-6
             50   52    60    065D            ADDD    R2, R0                  ; R0/R1 = DSIND(|X|) = (pi/180)|X|
                        05    0660    3$:     RSB
```

```
                          0661          .SBTTL  MTH$DCOSD_R7
                          0661
                          0661  ; This routine computes the DCOSD of the D-format value of R0.  The computation
                          0661  ; is performed one of two ways depending on the size of the input argument, X:
                          0661  ; Details are given in the discussion on MTH$DCOSD_R4.
                          0661
                          0661  MTH$DCOSD_R7::
               50     73  0661          TSTD    R0                              ; Check for reserved operand
    50    8000 8F     AA  0663          BICW    #^X8000, R0                     ; R0/R1 = |X|
    50    F9C4 CF     71  0668          CMPD    D_45, R0                        ; Compare 45 to |X|
               1A     14  066D          BGTR    SMALL_COSD                      ;
     000009CB'EF      16  066F          JSB     REDUCE_DEGREES                  ; R6/R7 = reduced argument
                          0675                                                  ; R3 = octant
                          0675
                          0675  EVAL_COSD:
    07     00  53     8F  0675          CASEB   R3, #0, #7
                   058F'  0679  1$:     .WORD   P_COS_D-1$
                   063C'  067B          .WORD   P_SIN_D-1$
                   0639'  067D          .WORD   N_SIN_D-1$
                   05DD'  067F          .WORD   N_COS_D-1$
                   05DD'  0681          .WORD   N_COS_D-1$
                   0639'  0683          .WORD   N_SIN_D-1$
                   063C'  0685          .WORD   P_SIN_D-1$
                   058F'  0687          .WORD   P_COS_D-1$
                          0689
                          0689
                          0689  SMALL_COSD:
    50    F9B3 CF     71  0689          CMPD    D_SMALLD, R0                    ; Compare 180/pi*2^-27  with |X|
               06     14  068E          BGTR    1$                              ; Check if polyinomial evaluation is
         56    50     7D  0690          MOVQ    R0, R6                          ;    necessary.
             0572     31  0693          BRW     P_COS_D                         ; POLY needed
         50    08     70  0696  1$:     MOVD    #1, R0                          ; R0 = 1. = DCOSD(|X|)
                      05  0699          RSB
                          069A
```

```
                          069A
                          069A
                          069A                .SBTTL  REDUCE_MEDIUM
                          069A
                          069A   ;
                          069A   ; This routine assumes that the absolute value of the argument, X,  is in R0/R1
                          069A   ; and that pi/4 =< |X| < 9*pi/4.  It returns a pair of d-format values for the
                          069A   ; reduced argument:  YHI in R6/R7, and YLO in R4/R5.  The octant bits in are
                          069A   ;  returned in R2.
                          069A   ;
                          069A   ; The reduced argument is obtained by locating the octant that X is in through
                          069A   ; a binary search and then subtracting off a suitable multiple of pi/2
                          069A   ;
                          069A
                          069A   REDUCE_MEDIUM:
         50   8000 8F  AA 069A           BICW    #^X8000, R0             ; R0/R1 = |X|
         50   F975 CF  71 069F           CMPD    D_5_PI_OV_4, R0         ;
                   11  15 06A4           BLEQ    5$                      ; |X| >= 5*pi/4
         50   F966 CF  71 06A6           CMPD    D_3_PI_OV_4, R0         ;
                   05  15 06AB           BLEQ    3$                      ; |X| >= 3*pi/4
         52        01  D0 06AD           MOVL    #1, R2                  ; First quadrant
                   16  11 06B0           BRB     SUBTRACT
                          06B2
         52        03  D0 06B2   3$:     MOVL    #3, R2                  ; Second quadrant
                   11  11 06B5           BRB     SUBTRACT
                          06B7
         50   F965 CF  71 06B7   5$:     CMPD    D_7_PI_OV_4, R0 ;
                   05  15 06BC           BLEQ    7$                      ; |X| >= 7*pi/4
         52        05  D0 06BE           MOVL    #5, R2                  ; Third quadrant
                   05  11 06C1           BRB     SUBTRACT
                          06C3
         52        07  D0 06C3   7$:     MOVL    #7, R2                  ; Fourth quadrant
                   00  11 06C6           BRB     SUBTRACT
                          06C8
                          06C8
                          06C8   SUBTRACT:
         53   FD A242  3E 06C8           MOVAW   -3(R2)[R2], R3          ; R3 = index into PI_OV_2 table
      53   F996 CF43   DE 06CD           MOVAL   PI_OV_2[R3], R3         ; R3 = pointer into PI_OV_2 table
      56     50    83  63 06D3           SUBD3   (R3)+, R0, R6           ; R6/R7 = 1st approximation to YHI
                   02  15 06D7           BLEQ    1$                      ;       = YHI'
                   52  D6 06D9           INCL    R2                      ; Adjust octant bits
   54    56   8000 8F  AB 06DB   1$:     BICW3   #^X8000, R6, R4         ; R4 = high 16 bits of |YHI'|
         54   2700 8F  B1 06E1           CMPW    #^X2700, R4             ; Check for at least 6 significant bits
                   0C  14 06E6           BGTR    NOT_ENOUGH_BITS         ;
                          06E8
         54        56  7D 06E8           MOVQ    R6, R4                  ; R4/R5 = YHI'
                   57  D4 06EB           CLRL    R7                      ; R6/R7 = high 24 bits of YHI' = YHI
         54        56  62 06ED           SUBD    R6, R4                  ; R4/R5 = low bits of YHI'
         54        63  62 06F0           SUBD    (R3), R4                ; R4/R5 = YLO
                   05     06F3           RSB
                          06F4
                          06F4   NOT_ENOUGH_BITS:
         50        63  7D 06F4           MOVQ    (R3), R0                ;
   50    003F0000  8F  CA 06F7           BICL    #^X003F0000, R0         ;
                   51  D4 06FE           CLRL    R1                      ;
         54   83   50  63 0700           SUBD3   R0, (R3)+, R4           ;
         56        50  62 0704           SUBD    R0, R6                  ; R6/R7 = YHI
```

```
         54   63   60 0707   969          ADDD    (R3), R4                   ; R4/R5 = -YLO
  54   8000   8F   AC 070A   970          XORW    #^X8000, R4                ; R4/R5 = YLO
                   05 070F   971          RSB
                      0710   972
                      0710   973
```

```
                                0710         .SBTTL  REDUCE_LARGE
                                0710
                                0710  ;
                                0710  ; This routine is used to reduce large arguments (|X| >= 9*pi/4) modulo pi/4.
                                0710  ; It returns the reduced argument, Y, in R4/R7 in units of cycles, and returns
                                0710  ; the octant bits, I, in R2.
                                0710  ;
                                0710  ; The method of reduction is as follows:
                                0710  ;
                                0710  ;        x*(4/pi) = 2^n*f*(4/pi) where n is an integer and 1/2 =< f < 1
                                0710  ;                 = 2^(n-56)*(2^56*f)*(4/pi)
                                0710  ;                 = (2^56*f)*(2^(n-56)*4/pi)
                                0710  ;                 = K*C, where K = 2^56*f is an integer and C = 2*(n-56)*4/pi
                                0710  ;        Let L = K*C modulo 8, where 0 =< L < 8, and let I = the integer(L) and
                                0710  ;        h = fract(L), then if I is even Y = h, otherwise Y = 1-h
                                0710  ;
                                0710  ; CONSTANTS:
                                0710
                       00001E80 0710         L_INT_WEIGHT  = ^X1E80              ; weights exponent by 61
                       00001000 0710         W_TERM_WEIGHT = ^X1000              ; weights exponent by 32
                       00004000 0710         W_MAX_WEIGHT  = ^X4000              ; maximum unbiased exponent
                       00000039 0710         W_ADJUST      = ^X39                ; Used to locate binary point in
                                0710                                             ;   MTH$AL_4_OV_PI table
                                0710  D_2_TO_32:
              00000000 00005080 0710         .QUAD   ^X5080                      ; 2^32
                                0718
                                0718
                                0718
                                0718  REDUCE_LARGE:
                                0718  ;
                                0718  ; The first step is to obtain the location of the binary point in the represen-
                                0718  ; tation of C = 2^(n-56)*(4/pi) in two parts - the number of longwords from
                                0718  ; the start and the number of bits from the most significant bit of the next
                                0718  ; longword. Also K = 2^56*f must be obtained.
                                0718  ;
           50    8000 8F     AA 0718         BICW    #^X8000, R0                 ; R0/R1 = |X|
       53  50      F9 8F     9C 071D         ROTL    #-7, R0, R3                 ; Shift exponext field 7 bits right
              53      39     A2 0722         SUBW    #W_ADJUST, R3               ; Unbias exp and adjust for leading
                                0725                                             ;   zeroes.  R3 = location of binary
                                0725                                             ;   point
       54  53      FD 8F     9C 0725         ROTL    #-3, R3, R4                 ; Divide R3 by 32 and mull by 4 to get
       54   FFFFFFE3 8F      CA 072A         BICL    #^XFFFFFFE3, R4             ;   R4 = # of longwords (in bytes) to
                                0731                                             ;   binary point.
       52     00000000'EF    DE 0731         MOVAL   MTH$AL_4_OV_PI, R2          ; Get base address of MTH$AL_4_OV_PI
                                0738                                             ;   table
              52      54     C2 0738         SUBL    R4, R2                      ; R2 points to 1st quadword of interest
           53      E0 8F     8A 073B         BICB    #^XE0, R3                   ; R3(7:0) = # of bits within longword
                                073F
           50    7F80 8F     AA 073F         BICW    #^X7F80, R0                 ; Clear exponent field
           50    4C00 8F     A8 0744         BISW    #^X4C00, R0                 ; R0 = 2^24*f
              50      50     6A 0749         CVTDL   R0, R0                      ; R0 = High 24 bits of K
       51     51      10     9C 074C         ROTL    #16, R1, R1                 ; R1 = Low 32 bits of K
                      02     18 0750         BGEQ    1$                          ; Check for high bit of R1 set
                      50     D6 0752         INCL    R0                          ; Adjust R0 if R1 is negative
                                0754
                                0754  ;
                                0754  ; The next step is to generate an approximation to C, call it C'' to be used
```

```
                          0754  ; in computing x*4/pi.  C'' will consist of the first three integer bits of
                          0754  ; C and the first 61 fraction bits of C.  These bits will be obtained from a
                          0754  ; constant stored in the interger array MTH$AL_4_OV_PI.
                          0754  ;
                          0754  ; NOTE:  The ASHQ, ADDL, and MULL instructions in the follow sections may
                          0754  ; result in an integer overflow trap.  The overflow incurred is intentional,
                          0754  ; so that the IV bit must be turned off.  The IV bit is not restored until
                          0754  ; after all of the necessary fraction bits hav been generated.
                          0754  ;
                  7E  DC  0754  1$:     MOVPSL  -(SP)                   ; Put current PSL on stack
  6E    FFFFFFDF  8F  CA  0756          BICL    #^C<PSL$M_IV>, (SP)     ; (SP) = current IV bit
                  20  B9  075D          BICPSW  #PSL$M_IV               ; Clear integer overflow bit
                          075F
                          075F  ; The necessary calculation to produce the reduced argument can require up to
                          075F  ; nine longwords of temporary work space.  This work space will be allocated
                          075F  ; on the stack.  The work space will be accessed though the use of three
                          075F  ; registers: R4, R5, and SP.  For the purposes of comments the temporary work
                          075F  ; space will be referred to as locations T0 though T8.  The stack and its
                          075F  ; pointers will look something like this:
                          075F  ;
                          075F  ;                               |                     |
                          075F  ;                               |---------------------|
                          075F  ;                           T0: |                     |   <---- SP
                          075F  ;                               |---------------------|
                          075F  ;                           T1: |                     |
                          075F  ;                               |---------------------|
                          075F  ;                           T2: |                     |
                          075F  ;                               |---------------------|
                          075F  ;                           T3: |                     |
                          075F  ;                               |---------------------|
                          075F  ;                           T4: |                     |   <---- R4
                          075F  ;                               |---------------------|
                          075F  ;                           T5: |                     |
                          075F  ;                               |---------------------|
                          075F  ;                           T6: |                     |
                          075F  ;                               |---------------------|
                          075F  ;                           T7: |                     |
                          075F  ;                               |---------------------|
                          075F  ;                           T8: |                     |   <---- R5
                          075F  ;                               |---------------------|
                          075F  ;                               |                     |
                          075F  ;
                          075F  ; The following code allocates the storage and sets up the pointers.
                          075F
           5E     24  C2  075F          SUBL    #36, SP                 ; Allocate 9 longwords on the stack
      54   5E     10  C1  0762          ADDL3   #16, SP, R4             ; R4 points to T4
      55   5E     20  C1  0766          ADDL3   #32, SP, R5             ; R5 points to T8
                          076A
                          076A  ;
                          076A  ; Get C'' = C(0):C(1):C(2):C(3) in T5/T8.  C(0) though  C(3) are unsigned
                          076A  ; integers generated from the binary representation of C.  The high three bits
                          076A  ; of C(0) are the the first three bits to the left of the binary point of C.
                          076A  ; The remaining bits C(0) and C(1) though C(3) are the first 125 bits to the
                          076A  ; right of the binary point of C. Note that the C(i)'s are adjusted to
                          076A  ; compensate for their signed (rather than unsigned) interpretation in the EMUL
                          076A  ; instruction.  Note also that the representation of C has no more than 15
                          076A  ; consequtive ones, so that no carry is possible from the adjustment.
```

```
                                076A  ;
                                076A
           57   54   0C   C1    076A          ADDL3   #12, R4, R7            ; Initailize loop counter.  R7 points
                                076E                                         ;   to T7
           67   62   53   79    076E          ASHQ    R3, (R2), (R7)         ; Shift the proper quadword so that
                                0772                                         ;   T8 has C(0)in it
                57   04   C2    0772          SUBL    #4, R7                 ; R7 points to T6
                52   04   C2    0775  2$:     SUBL    #4, R2                 ; R2 points to next quadword in
                                0778                                         ;   MTH$AL_4_OV_PI table
           67   62   53   79    0778          ASHQ    R3, (R2), (R7)         ; Shift quadword so that C(n) is in
                                077C                                         ;   T(8-n) , n = 0,1,2,3
                     03   18    077C          BGEQ    3$                     ; Check for high bit of C(n) set
                  08 A7   D6    077E          INCL    8(R7)                  ; Bit set.  Adjust C(n-1)
 FFEA 57   FFFFFFFC 8F   54   F1 0781  3$:     ACBL    R4, #-4, R7, 2$        ; Loop until C(0) though C(3) are in
                                078B                                         ;   T5 though T8
                                078B
                                078B  ;
                                078B  ; Generate the low 128 bits of the product K*C'' = L.  This product is
                                078B  ; equivalent to multiplying K times C'' modulo 8.  The result of the
                                078B  ; product is in T4/T7 with bits 31:29 of T4 the octant bits, and the remaining
                                078B  ; 125 bits the faction bits of the product.  The last 53 fraction bits (bits
                                078B  ; 20:0 of T6 and 31:0 of T5) are non-valid fraction bits that will be used
                                078B  ; later if more fraction bits need to be generated.
                                078B  ;
                                078B  ;
                                078B  ; Multiply the high order bits of K (R0) times C'' and store the result in
                                078B  ; T0/T2.
                                078B
       6E   00   04 A4   50   7A 078B          EMUL    R0, 4(R4), #0, (SP)    ; T0/T1 = KHI*C(3)
 04 AE   04 AE   08 A4   50   7A 0791          EMUL    R0, 8(R4), 4(SP), 4(SP) ; T0/T2 = KHI*[C(2):C(3)]
           64    0C A4   50   C5 0799          MULL3   R0, 12(R4), (R4)       ; T4 = Low 32 bits of KHI*C(1)
                 08 AE   64   C0 079E          ADDL    (R4), 8(SP)            ; T0/T2 = KHI*C'' modulo 8
                                07A2
                                07A2  ; Multiply the low order bits of K (R1) times C'' and store the result in
                                07A2  ; T4/T8.
                                07A2
       64   00   04 A4   51   7A 07A2          EMUL    R1, 4(R4), #0, (R4)    ; T4/T5 = KLO*C(3)
 04 A4   04 A4   08 A4   51   7A 07A8          EMUL    R1, 8(R4), 4(R4), 4(R4) ; T4/T6 = KLO*[C(2):C(3)]
 08 A4   08 A4   0C A4   51   7A 07B0          EMUL    R1, 12(R4), 8(R4), 8(R4); T4/T7 = KLO*[C(1):C(2):C(3)]
                    65   51   C4 07B8          MULL    R1, (R5)               ; T8 = KLO*C(0)
                 0C A4   65   C0 07BB          ADDL    (R5), 12(R4)           ; T4/T7 = KLO*C'' modulo 8
                                07BF
                                07BF  ; Add KHI*C'' to KLO*C'' to get K*C''.  Store the result in T4/T7.
                                07BF
            04 A4   6E   C0     07BF          ADDL    (SP), 4(R4)            ;
         08 A4   04 AE   D8     07C3          ADWC    4(SP), 8(R4)           ;
         0C A4   08 AE   D8     07C8          ADWC    8(SP), 12(R4)          ; T4/T7 = K*C'' modulo 8
                                07CD
                                07CD  ;
                                07CD  ; At this point there may or may not be enough valid bits in R3/R4 to generate
                                07CD  ; Y.  If the first 12 fratction bits are all 1's or 0's, there a possibility of
                                07CD  ; loss of significance when computing Y.  Consequently, we must check for loss
                                07CD  ; of significance before converting T4/T7 to Y and I.
                                07CD  ;
                                07CD
   65   FC A5   00200000 8F   C1 07CD          ADDL3   #^X200000, -4(R5), (R5) ; If the first 12 fraction bits are 1's
           65   3FC00000 8F   D3 07D6          BITL    #^X3FC00000, (R5)      ;   and the reduced arg = 1-f or the
```

```
             37    12  07DD              BNEQ    CONVERT                    ;   first 7 bit are 0 and the reduced
                       07DF                                                 ;   arg = f, then (and only then) bits
                       07DF                                                 ;   29:22 are 0 and significance will
                       07DF                                                 ;   be lost.
                       07DF
                       07DF      ;
                       07DF      ; More bits need to be generated to cover the loss of significance.  There are
                       07DF      ; not enough registers to hold all the potential entra bits, so that the bits
                       07DF      ; already generated must be put on the stack.
                       07DF      ;
                       07DF
        0000090B'EF    16  07DF              JSB     GEN_MORE_BITS              ; Generate 85 additional bits and add
                       07E5                                                 ;   them to existing bits.  Results are
                       07E5                                                 ;   stored in T3/T7
          54    04     C2  07E5              SUBL    #4, R4                     ; Adjust R4 to reflect the addition of
                       07E8                                                 ;   another longword of K*C''
       15 FC A5    1D  E0  07E8              BBS     #29, -4(R5), 4$            ; Check if loss of significance is due
                       07ED                                                 ;   to leading ones or zeros
                       07ED
                       07ED      ; Lost significance due to leading zeros
                       07ED
 65    10 A4   15   00 EA  07ED              FFS     #0, #21, 16(R4), (R5)      ; If at least one bit is set.  This
                    21 12  07F3              BNEQ    CONVERT                    ;   means lost significance was minor.
  0C A4   1FFFFFFF 8F D1   07F5              CMPL    #^X1FFFFFFF, 12(R4)        ; If one of the three high bits is set,
                    17 15  07FD              BLEQ    CONVERT                    ;   lost significance was minor.
                  00B2 31  07FF              BRW     LEADING_ZEROS
                       0802
                       0802      ; Lost significance due to leading ones
                       0802
 65    10 A4   15   00 EB  0802      4$:     FFC     #0, #21, 16(R4), (R5)      ; If at least one bit is clear. This
                    0C 12  0808              BNEQ    CONVERT                    ;   means lost significance was minor.
  0C A4   E0000000 8F D1   080A              CMPL    #^XE0000000, 12(R4)        ; If one of the three high bits is
                    02 1E  0812              BGEQU   CONVERT                    ;   clear, lost significance was minor.
                    3B 11  0814              BRB     LEADING_ONES
                       0816
                       0816
                       0816      CONVERT:
                       0816
                       0816      ;       Isolate octant bits and convert fraction bits to a pair of D-format
                       0816      ;       quantities YHI and YLO
                       0816
 65   FC A5   03    1D EF  0816              EXTZV   #29, #3, -4(R5), (R5)      ; T8 = octant bits
   FC A5   E0000000 8F CA  081C              BICL    #^XE0000000, -4(R5)        ; Clear octant bits
          54    55  0C C3  0824              SUBL3   #12, R5, R4                ; R4 points to low order bits of h
        00000951'EF    16  0828              JSB     CVT_TO_DOUBLE              ; R0/R1 = 2^29*h_lo
                       082E                                                 ; R6/r7 = 2^29*h_hi
        56    0E80 8F  A2  082E              SUBW    #^XE80, R6                 ; R6/R7 = h_hi
                    02 14  0833              BGTR    3$                         ; Check for h_hi = 0
                    56 D4  0835              CLRL    R6                         ; Restore h_hi to 0
                    50 B5  0837      3$:     TSTW    R0                         ; Check for h_lo = 0
                    05 13  0839              BEQL    1$
        50    0E80 8F  A2  083B              SUBW    #^XE80, R0                 ; R0/R1 = h_lo
                 07 65 E9  0840      1$:     BLBC    (R5), 2$                   ; Check for odd or even octant bits
                       0843
                       0843      ;       Octant bits are odd.  Reduced argument equals 1 - h.
                       0843
 56    08    56     63     0843              SUBD3   R6, #1, R6                 ; R6/R7 = Y = 1 - h_hi
```

```
        50    50      72  0847         MNEGD   R0, R0                  ; R0/R1 = -h_lo
                          084A
                          084A  ;      Get octant bits
                          084A
        52    20 AE   D0  084A  2$:    MOVL    32(SP), R2              ; R2 = octant bits
              00A8    31  084E         BRW     GET_YHI_YLO
                          0851
                          0851
                          0851  ;
                          0851  ; At this point it has been determined that there is a major loss of
                          0851  ; significance and the processing begins a looping phase.  Each iteration of
                          0851  ; the loop will generate additional extra bits of K*C' until enough significant
                          0851  ; bits to compute Y are available.  During this time the nine longwords
                          0851  ; allocated on the stack will be used as follows:
                          0851  ;
                          0851  ;        T0/T2   Temporary storage used when generating extra bits.
                          0851  ;
                          0851  ;        T3/T7   Contains all significant bits generated so far.
                          0851  ;
                          0851  ;        T8      Contains a counter, W, indicating the appropriate exponent
                          0851  ;                  of the last longword of fraction bits used in converting
                          0851  ;                  to Y.
                          0851
                          0851  LEADING_ONES:
                          0851
                          0851  ; If processing continues here it is known that the loss of significance is due
                          0851  ; to a string of leading ones.
                          0851
     65   00001E80 8F     D0  0851         MOVL    #L_INT_WEIGHT, (R5)     ; T8 = exp bias for last longword
                          0858                                         ;      of the product K*C'
                          0858
  0C A4   FFE00000 8F     D1  0858  LOOP_1: CMPL   #^XFFE00000, 12(R4)     ; Check for enough significant bits
                2E        1A  0860         BGTRU   CONVERT_1               ; Enough bits.  Convert to floating.
          0000090B'EF     16  0862         JSB     GEN_MORE_BITS           ; T2/T7 contains K*C''
  0C A4   FFFFFFFF 8F     D1  0868         CMPL    #-1, 12(R4)             ; Check for all 1's
                1E        1A  0870         BGTRU   CONVERT_1               ; Not all 1's.  Enough precision bits
                          0872                                         ;      to compute Y
        08 A4    04 A4    7D  0872         MOVQ    4(R4), 8(R4)            ; Compress representation
           64    FC A4    7D  0877         MOVQ    -4(R4), (R4)            ;      of K*C''
FFD3 65   1000 8F   4000 8F   3D  087B     ACBW    #W_MAX_WEIGHT, #W_TERM_WEIGHT, (R5), LOOP_1
                          0885                                         ; Increment weighting factor.  If
                          0885                                         ;   weighting factor is greater than
                          0885                                         ;   1024 then no more bits need to be
                          0885                                         ;   generated.
                          0885
                          0885  ;
                          0885  ; The weighting factor is greater than 1024.  This means that the reduced
                          0885  ; argument is either not distinguishable from 1 or too small to be represented
                          0885  ; in F-format (i.e. underflow.)  Zero is returned in R4 for the reduced
                          0885  ; argument to signal this occurance.  Note that under these conditions the
                          0885  ; correct function value is one of the values 0, +/-1.  The
                          0885  ; correct choice is determined by the calling program based on the octant bits
                          0885  ; returned in R1.
                          0885  ;
                53        D4  0885         CLRL    R3                      ; Reduced argument is zero
  52    08 AE    03  1D   EF  0887         EXTZV   #29, #3, 8(SP), R2      ; R2 = octant bits
              0074        31  088D         BRW     RESTORE
```

```
                        0890
                        0890
                        0890     CONVERT_1:
          54   04   C0  0890             ADDL    #4, R4                  ; R4 points to low bits of h
     00000951'EF    16  0893             JSB     CVT_TO_DOUBLE           ; R0/R1 = 2^W*h_lo
                        0899                                             ; R6/R7 = 2^W*h_hi
  56   FE72 CF  56  63  0899             SUBD3   R6, D_2_TO_32, R6       ; R6/R7 = 2^W*(T - h_hi)
          50   50   72  089F             MNEGD   R0, R0                  ; R0/R1 = -2^W*h_lo
  52   FC A5  03  1D EF  08A2            EXTZV   #29, #3, -4(R5), R2     ; R2 = octant bits
          56   65   C2  08A8             SUBL    (R5), R6                ; R6/R7 = 1 - h_hi
               50   B5  08AB             TSTW    R0                      ; Check for h_lo = 0
               4A   13  08AD             BEQL    GET_YHI_YLO             ;
          50   65   C2  08AF             SUBL    (R5), R0                ; R0/R1 = - h_lo
               45   11  08B2             BRB     GET_YHI_YLO
                        08B4
                        08B4
                        08B4     LEADING_ZEROS:
                        08B4
                        08B4     ; If processing continues here it is known that the loss of significance is due
                        08B4     ; to a string of leading zeros.  Note that it is known that the loop for
                        08B4     ; leading zeros will terminate before an underflow condition occurs so that the
                        08B4     ; loop does not include a test for underflow.
                        08B4
      65   00001E80 8F  D0  08B4         MOVL    #L_INT_WEIGHT, (R5)     ; T8 = exp bias for last longword
                        08BB                                             ;     of the product K*C'
                        08BB
   0C A4   001FFFFF 8F  D1  08BB LOOP_0: CMPL    #^X001FFFFF, 12(R4)     ; Check enough fraction bits
               1B   19  08C3             BLSS    CONVERT_0               ; Enough bits.  Convert to floating
     0000090B'EF    16  08C5             JSB     GEN_MORE_BITS           ; T2/R7 contain K*C''
            0C A4   D5  08CB 1$:         TSTL    12(R4)                  ; Check for all 0's
               10   12  08CE             BNEQ    CONVERT_0               ; Not all 0's.  Enough precision bits.
   08 A4   04 A4    7D  08D0             MOVQ    4(R4), 8(R4)            ; Compress representation
      64   FC A4    7D  08D5             MOVQ    -4(R4), (R4)            ;     of K*C''
      65   1000 8F  A0  08D9             ADDW    #W_TERM_WEIGHT, (R5)    ; Increment weighting factor.
               DB   11  08DE             BRB     LOOP_0
                        08E0
                        08E0     CONVERT_0:
          54   04   C0  08E0             ADDL    #4, R4                  ; R4 points to low bits of h
     00000951'EF    16  08E3             JSB     CVT_TO_DOUBLE           ; R0/R1 = 2^W*h_lo
                        08E9                                             ; R6/R7 = 2^W*h_hi
  52   FC A5  03  1D EF  08E9            EXTZV   #29, #3, -4(R5), R2     ; R2 = octant bits
          56   65   C2  08EF             SUBL    (R5), R6                ; R6/R7 = h_hi
               50   B5  08F2             TSTW    R0                      ; Check for h_lo = 0
               03   13  08F4             BEQL    GET_YHI_YLO             ;
          50   65   C2  08F6             SUBL    (R5), R0                ; R0/R1 = h_lo
                        08F9
                        08F9     GET_YHI_YLO:
          54   56   7D  08F9             MOVQ    R6, R4                  ; R4/R5 = high bits of Y
               57   D4  08FC             CLRL    R7                      ; R6/R7 = high 24 bits Y = YHI
          54   56   62  08FE             SUBD    R6, R4                  ;
          54   50   60  0901             ADDD    R0, R4                  ; R4/R5 = YLO
                        0904     RESTORE:
            24 AE   B8  0904             BISPSW  36(SP)                  ; Restore IV bit and exit
          5E   28   C0  0907             ADDL    #40, SP                 ; Remove mask and temporary storage
                    05  090A             RSB
                        090B
                        090B
```

```
                                  090B  GEN_MORE_BITS:
                                  090B
                                  090B  ;
                                  090B  ; This subroutine generates 85 extra fraction bits and puts them to the
                                  090B  ; existing bits.  NOTE:  This routine is always entered via a JSB instruction.
                                  090B  ; Consequently, SP points to the first longword BEFORE T0, rather than T0
                                  090B  ; itself.
                                  090B  ;
                                  090B
                  52    04    C2  090B          SUBL    #4, R2                  ; Adjust pointer to get next quadword
                                  090E                                          ;    from MTH$AL_4_OV_PI
            56    62    53    79  090E          ASHQ    R3, (R2), R6            ; R7 = C(n)
                        17    18  0912          BGEQ    1$                      ; Branch if high bit is clear
                                  0914
                                  0914  ; Logic to process unsigned values greater than 2^31 - 1
                                  0914
     04 AE    00    57    51  7A  0914          EMUL    R1, R7, #0, 4(SP)       ;
                 08 AE    51  C0  091A          ADDL    R1, 8(SP)               ; T0/T1 = KLO*C(n)
  08 AE    08 AE    57    50  7A  091E          EMUL    R0, R7, 8(SP), 8(SP)    ;
                 0C AE    50  C0  0925          ADDL    R0, 12(SP)              ; T0/T2 = K*C(n)
                          0D  11  0929          BRB     2$
                                  092B
                                  092B  ; Logic to process unsigned values less than 2^31
                                  092B
     04 AE    00    57    51  7A  092B  1$:     EMUL    R1, R7, #0, 4(SP)       ; T0/T1 = KLO*C(n)
  08 AE    08 AE    57    50  7A  0931          EMUL    R0, R7, 8(SP), 8(SP)    ; T0/T2 = K*C(n)
                                  0938
                                  0938  ; Add new bits to old
                                  0938
            64    08 AE       C0  0938  2$:     ADDL    8(SP), (R4)             ;
         04 A4    0C AE       D8  093C          ADWC    12(SP), 4(R4)           ;
                        08    1E  0941          BCC     3$                      ; Check for carry from previous add
                     08 A4    D6  0943          INCL    8(R4)                   ; Propagate carry
                        03    1E  0946          BCC     3$                      ; Check for carry from previous add
                     0C A4    D6  0948          INCL    12(R4)                  ; Propagate carry
         FC A4    04 AE       D0  094B  3$:     MOVL    4(SP), -4(R4)           ; Move new low order bits to end of
                                  0950                                          ;    of old low order bits
                              05  0950          RSB                             ;
                                  0951
                                  0951
                                  0951
                                  0951
                                  0951
                                  0951  CVT_TO_DOUBLE:
                                  0951
                                  0951  ;
                                  0951  ; This routine converts an array of three longword pointed to by R4 to a pair
                                  0951  ; of D-format values.  The results are returned in R0/R1 (low 48 bits) and
                                  0951  ; R6/R7 (high 48 bits).  ;
                                  0951
                  50    84    6E  0951          CVTLD   (R4)+, R0               ; R0/R1 = Low 32 bits of h
                        0E    13  0954          BEQL    2$                      ;
                        07    14  0956          BGTR    1$                      ; Adjust for signed
                        64    D6  0958          INCL    (R4)                    ;    conversion error
                        03    1E  095A          BCC     1$                      ; If necessary,
                     04 A4    D6  095C          INCL    4(R4)                   ;    propagate carry
            50    1000 8F     A2  095F  1$:     SUBW    #W_TERM_WEIGHT, R0      ; R0/R1 = (low 32 bits of h)/2^32
```

```
FC A4   64   FFFF0000 8F   CB 0964  1374 2$:   BICL3  #^XFFFF0000, (R4), -4(R4)
             64    FC A4   C2 096D  1375       SUBL   -4(R4), (R4)                ;
             52    FC A4   6E 0971  1376       CVTLD  -4(R4), R2                  ;
                50    52   60 0975  1377       ADDD   R2, R0                      ; R0/R1 = (low 48 bits of h)/2^32
                      05   13 0978  1378       BEQL   3$                          ;
             50  1000 8F   A2 097A  1379       SUBW   #W_TERM_WEIGHT, R0          ; R0/R1 = (low 48 bits of h)/2^64
                52    84   6E 097F  1380 3$:   CVTLD  (R4)+, R2                   ; R2/R3 = next 16 bits of h
                      09   13 0982  1381       BEQL   5$                          ;
                      02   14 0984  1382       BGTR   4$                          ; Adjust for signed conversion error.
                      64   D6 0986  1383       INCL   (R4)                        ;   Note that no carry is possible
             52  1000 8F   A2 0988  1384 4$:   SUBW   #W_TERM_WEIGHT, R2          ; R2/R3 = (next 16 bits of h)/2^32
                56    64   6E 098D  1385 5$:   CVTLD  (R4), R6                    ; R6/R7 = high 32 bits of h
                      05   18 0990  1386       BGEQ   6$                          ; Adjust for signed conversion
             56  FD7A CF   60 0992  1387       ADDD   D_2_TO_32, R6               ;   error
                56    52   60 0997  1388 6$:   ADDD   R2, R6                      ; R6/R7 = (high 48 bits of h)/2^32
                           05 099A  1389       RSB
                              099B  1390
```

```
                            099B          .SBTTL  REDUCE_DEGREES
                            099B
                            099B  ; This routine assumes that the absolute value of the argument is in R0/R1.
                            099B  ; The reduction process is performed in two stages.  The first stage of
                            099B  ; the reduction reduces the argument modulo 360 to a value less that 2^55,
                            099B  ; and the second stage reduces the argument modulo 45 to a value less than 45.
                            099B
                            099B  ; Constants used in this reduction:
                            099B  ;
                            099B
                            099B  POWER_MOD_360_0:        ; Powers of 2 modulo 360 for t1 = 0
    0008 0004 0002 0001     099B          .WORD      1,       2,       4,      8
    0080 0040 0020 0010     09A3          .WORD     16,      32,      64,    128
    00F8 0130 0098 0100     09AB          .WORD    256,     152,     304,    248
                            09B3
                            09B3  POWER_MOD_360_1:        ; Powers of 2 modulo 360 for t1 <> 0
    0008 00B8 0110 0088     09B3          .WORD    136,     272,     184,      8
    0080 0040 0020 0010     09BB          .WORD     16,      32,      64,    128
    00F8 0130 0098 0100     09C3          .WORD    256,     152,     304,    248
                            09CB
                            09CB
                            09CB
                            09CB  REDUCE_DEGREES:
      50    5C00 8F   B1    09CB          CMPW    #^X5C00, R0                 ; Compare |x| with 2^55
                 49   14    09D0          BGTR    LAST_STEP                   ; Branch to special logic for med arg
                            09D2
                            09D2  ;
                            09D2  ; It is assumed here that the argument is greater than 2^55.
                            09D2  ;
                            09D2  ; The argument is reduced as follows:
                            09D2  ;     Let x = 2^t*f, where t > 56 and 1/2 =< f < 1.  And let J = 2^56*f =
                            09D2  ;     2^30*J1 + J2 and K = 2^(t-56).  Since 2^30 = 64 modulo 360, we have that
                            09D2  ;     J = 64*J1 + J2 modulo 360.  Now let t' = t - 56 = 12*t1 + t2.  Note that
                            09D2  ;     (2^12)^2 = (2^9)*(2^15) = (2^9)*(2^3) = 2^12 modulo 360.  Hence, if t1 is
                            09D2  ;     not zero, K = 2^t' = 2^(12*t1+t2) = (2^12)*(2^t2) = 136*2^t2 modulo 360.
                            09D2  ;     For t1 = 0 K = 2^t2.  Consequently, define K' congruent to 2^t2 if t1 = 0
                            09D2  ;     and congruent to 136*2^t2 otherwise, where 0 =< K' < 360.  Then x' =
                            09D2  ;     K'*(64*J1 + J2) is congruent to s modulo 360 and x' < 2^56.
                            09D2
            52    50   D0   09D2          MOVL    R0, R2                      ; R2 = high longword of X
   50   00007F80 8F    CA   09D5          BICL    #^X7F80, R0                 ; Clear exp bits of X
        50    5C00 8F  A8   09DC          BISW    #^X5C00, R0                 ; R0/R1 = J
            52    50   C2   09E1          SUBL    R0, R2                      ; R2 = t'*2^7
                            09E4
            53    50   7D   09E4          MOVQ    R0, R3                      ; R3/R4 = J
   51   FFFF3FFF 8F    CA   09E7          BICL    #^XFFFF3FFF, R1             ; R0/R1 = J1*2^30
            53    50   62   09EE          SUBD    R0, R3                      ; R3/R4 = J2
        50    0C00 8F  A2   09F1          SUBW    #^XC00, R0                  ; R0/R1 = 64*J1
            50    53   60   09F6          ADDD    R3, R0                      ; R0/R1 = 64*J1 + J2 = J modulo 45
                            09F9
    52    52    F9 8F  9C   09F9          ROTL    #-7, R2, R2                 ; R2 = t'
       53    52   0C   A7   09FE          DIVW3   #12, R2, R3                 ; R3 = t1
             53   0C   A4   0A02          MULW    #12, R3                     ; R3 = 12*t1
             52   53   A2   0A05          SUBW    R3, R2                      ; R2 = t2
                            0A08
                  53   B5   0A08          TSTW    R3                          ; Check for t1 = 0 and choose K'
                  07   12   0A0A          BNEQ    1$                          ;   accordingly
```

```
              52    8B AF42   6D  0A0C              CVTWD   POWER_MOD_360_0[R2], R2 ; R2/R3 = K'
                         05   11  0A11              BRB     2$
              52    9C AF42   6D  0A13        1$:   CVTWD   POWER_MOD_360_1[R2], R2 ; R2/R3 = K'
                    50   52   64  0A18        2$:   MULD    R2, R0                  ; R0/R1 = X' (mod 45) 0 =< R0 < 2^55
                                  0A1B
                                  0A1B
                                  0A1B  LAST_STEP:
                                  0A1B  ;
                                  0A1B  ; Argument reduction scheme for arguments with absolute value less than 2^55
                                  0A1B  ;
                                  0A1B  ; The reduced argument Y is computed as follows:
                                  0A1B  ;     Let I = int(X/45)
                                  0A1B  ;         if I is even
                                  0A1B  ;             then Y = X - 45*I
                                  0A1B  ;             else Y = (I+1)*45 - x
                                  0A1B
                                  0A1B
              50    5200 8F   B1  0A1B              CMPW    #^X5200, R0                 ; Compare 2^36 with |X|
                         24   18  0A20              BGEQ    NO_OVERFLOW                 ;
         56   50    F622 CF   65  0A22              MULD3   D_T_OV_45, R0, R6           ; R6/R7 = |X|/45
                                  0A28
                                  0A28  ;
                                  0A28  ; Turn off IV to avoid an exception in EMODD
                                  0A28  ;
                         52   DC  0A28              MOVPSL  R2                          ; Move PSL to R2
           52   FFFFFFDF 8F   CA  0A2A              BICL    #^C<PSL$M_IV>, R2           ; Save current IV bit
                         20   B9  0A31              BICPSW  #PSL$M_IV                   ; Turn off integer overflow trap
                                  0A33
     54   53   56   00   08   74  0A33              EMODD   #1, #0, R6, R3, R4          ; R3 = low 32 integer bits of |X|/45
                                  0A39                                                  ; R4/R5 = fractional part of |X|/45
                                  0A39
                         52   B8  0A39              BISPSW  R2                          ; Restore IV bit
                                  0A3B
                    56   54   62  0A3B              SUBD2   R4, R6                      ; R6/R7 = Integer part of |X|/45 = I
                    27   53   E9  0A3E              BLBC    R3, EVEN                    ;
                    56   08   60  0A41              ADDD2   #1, R6                      ; R6/R7 = I + 1
                         12   11  0A44              BRB     ODD
                                  0A46
                                  0A46
                                  0A46  NO_OVERFLOW:
54   53   50   0B   F5FE CF   74  0A46              EMODD   D_1_OV_45, #X_1_OV_45, R0, R3, R4
                                  0A4E                                                  ; R3 = I = integer part of |X|/45
                    14   53   E9  0A4E              BLBC    R3, CVT                     ; Branch if octant bits are even
              56    53   01   C1  0A51              ADDL3   #1, R3, R6                  ; R6 = I + 1
                    56   56   6E  0A55              CVTLD   R6, R6                      ; R6/R7 = I + 1
              56    F5D4 CF   64  0A58  ODD:        MULD2   D_45, R6                    ; R6/R7 = 45*(I+1)
                    56   50   62  0A5D              SUBD2   R0, R6                      ; R6/R7 = Y
              53    F8   8F   8A  0A60              BICB    #^XF8, R3                   ; Save only last three octant bits
                         05       0A64              RSB
                                  0A65
                    56   53   6E  0A65  CVT:        CVTLD   R3, R6                      ; R6/R7 = I
              56    F5CC CF   64  0A68  EVEN:       MULD2   D_M45, R6                   ; R6/R7 = -45*I
                    56   50   60  0A6D              ADDD2   R0, R6                      ; R6/R7 = Y
              53    F8   8F   8A  0A70              BICB    #^XF8, R3                   ; Save only last three octant bits
                              05  0A74              RSB
                                  0A75
                                  0A75
```

```
                          0A75
                          0A75             .SBTTL  RADIAN_POLYNOMIALS       ; Polynomials for arguments in radians
                          0A75
                          0A75
                          0A75
                          0A75      ;
                          0A75      ; Polynomial evaluation for DCOS(Y) for Y in radians
                          0A75      ;
                          0A75
                          0A75      P_COS_R:
53    56   8000 8F   AB   0A75             BICW3   #^X8000, R6, R3           ;
      53   4000 8F   B1   0A7B             CMPW    #^X4000, R3               ; Compare 1/2 with |YHI|
                31   14   0A80             BGTR    LEQL_HALF                 ; Sufficent overhang is available
                          0A82      NEEDS_DOUBLE:
           7E   54   7D   0A82             MOVQ    R4, -(SP)                 ; Save YLO
      7E   54   56   61   0A85             ADDD3   R6, R4, -(SP)             ; Save Y
      50   6E   6E   65   0A89             MULD3   (SP), (SP), R0            ; R0/R1 = Y^2
F675  CF   07   50   75   0A8D             POLYD   R0, #COSLENR2-1, COSTBR2; R0/R1 = Q(Y^2)
      54   56   8E   61   0A93             ADDD3   (SP)+, R6, R4             ; R4/R5 = Y + YHI
           54   8E   64   0A97             MULD    (SP)+, R4                 ; R4/R5 = YLO*(Y + YHI) = A2
                05   13   0A9A             BEQL    1$                        ; Check for A2 = 0
      54   0080 8F   A2   0A9C             SUBW    #^X80, R4                 ; R4/R5 = A2/2
           50   54   62   0AA1      1$:    SUBD    R4, R0                    ; R0/R1 = Q(Y^2) - A2/2
           56   56   64   0AA4             MULD    R6, R6                    ; R6/R7 = YHI^2
      56   0080 8F   A2   0AA7             SUBW    #^X80, R6                 ; R6/R7 = YHI^2/2
           56   08   62   0AAC             SUBD    #1, R6                    ; R6/R7 = -(1 - YHI^2/2)
           50   56   62   0AAF             SUBD    R6, R0                    ; R0/R1 = DCOS(Y)
                     05   0AB2             RSB
                          0AB3
                          0AB3      LEQL_HALF:
           56   54   60   0AB3             ADDD    R4, R6                    ; R6/R7 = Y
           56   56   64   0AB6             MULD    R6, R6                    ; R6/R7 = Y^2
F609  CF   07   56   75   0AB9             POLYD   R6, #COSLENR1-1, COSTBR1; R0/R1 = DCOS(Y)
                     05   0ABF             RSB
                          0AC0
                          0AC0
                          0AC0      ;
                          0AC0      ; Polynomial evaluation for -DCOS(Y)
                          0AC0      ;
                          0AC0
                          0AC0      N_COS_R:
53    56   8000 8F   AB   0AC0             BICW3   #^X8000, R6, R3           ;
      53   4000 8F   B1   0AC6             CMPW    #^X4000, R3               ; Compare 1/2 with |YHI|
                32   14   0ACB             BGTR    2$                        ; Sufficent overhang is available
           7E   54   7D   0ACD             MOVQ    R4, -(SP)                 ; Save YLO
      7E   54   56   61   0AD0             ADDD3   R6, R4, -(SP)             ; Save Y
      50   6E   6E   65   0AD4             MULD3   (SP), (SP), R0            ; R0/R1 = Y^2
F62A  CF   07   50   75   0AD8             POLYD   R0, #COSLENR2-1, COSTBR2; R0/R1 = Q(Y^2)
      54   56   8E   61   0ADE             ADDD3   (SP)+, R6, R4             ; R4/R5 = Y + YHI
           54   8E   64   0AE2             MULD    (SP)+, R4                 ; R4/R5 = YLO*(Y + YHI) = A2
                05   13   0AE5             BEQL    1$                        ; Check for A2 = 0
      54   0080 8F   A2   0AE7             SUBW    #^X80, R4                 ; R4/R5 = A2/2
           50   54   62   0AEC      1$:    SUBD    R4, R0                    ; R0/R1 = Q(Y^2) - A2/2
           56   56   64   0AEF             MULD    R6, R6                    ; R6/R7 = YHI^2
      56   0080 8F   A2   0AF2             SUBW    #^X80, R6                 ; R6/R7 = YHI^2/2
           56   08   62   0AF7             SUBD    #1, R6                    ; R6/R7 = -(1 - YHI^2/2)
      50   56   50   63   0AFA             SUBD3   R0, R6, R0                ; R0/R1 = -DCOS(Y)
```

```
                      05  0AFE  1564          RSB
                          0AFF  1565
            56   54   60  0AFF  1566 2$:      ADDD    R4, R6                  ; R6/R7 = Y
            56   56   64  0B02  1567          MULD    R6, R6                  ; R6/R7 = Y^2
   F5BD CF  07   56   75  0B05  1568          POLYD   R6, #COSLENR1-1, COSTBR1; R0/R1 = DCOS(Y)
        50  8000 8F   AC  0B0B  1569          XORW    #^X8000, R0             ; R0/R1 = -DCOS(Y)
                      05  0B10  1570          RSB
                          0B11  1571
                          0B11  1572 ;
                          0B11  1573 ; Polynomial evaluation for -DSIN(Y)
                          0B11  1574 ;
                          0B11  1575
                          0B11  1576 N_SIN_R:
        54  8000 8F   AC  0B11  1577          XORW    #^X8000, R4             ;
        56  8000 8F   AC  0B16  1578          XORW    #^X8000, R6             ; R4/R7 = -Y
                          0B1B  1579
                          0B1B  1580 ;
                          0B1B  1581 ; Polynomial evaluation for DSIN(Y)
                          0B1B  1582 ;
                          0B1B  1583
                          0B1B  1584 P_SIN_R:
            7E   54   7D  0B1B  1585          MOVQ    R4, -(SP)               ; Save YLO
        7E  56   54   61  0B1E  1586          ADDD3   R4, R6, -(SP)           ; Save Y
        54  6E   6E   65  0B22  1587          MULD3   (SP), (SP), R4          ; R4 = Y^2
   F61C CF  07   54   75  0B26  1588          POLYD   R4, #SINLENR-1, SINTBR  ; R0/R1 = P(Y^2)
            50   8E   64  0B2C  1589          MULD    (SP)+, R0               ; R0/R1 = Y*P(Y^2)
            50   8E   60  0B2F  1590          ADDD    (SP)+, R0               ; R0/R1 = YLO + Y*P(Y^2)
            50   56   60  0B32  1591          ADDD    R6, R0                  ; R0/R1 = Y + Y*P(Y^2) = DSIN(Y)
                      05  0B35  1592          RSB
                          0B36  1593
                          0B36  1594
                          0B36  1595
```

```
                        0B36          .SBTTL  CYCLE_POLYNOMIALS        ; Polynomials for arguments in cycles
                        0B36
                        0B36
                        0B36
                        0B36     ;
                        0B36     ; Polynomial evaluation for DCOS(Y) for Y in cycles
                        0B36     ;
                        0B36
                        0B36     P_COS_C:
     56  F4EE CF    71  0B36          CMPD    D_2_OV_PI, R6            ; Compare 2/pi with |YHI|
              31    18  0B3B          BGEQ    2$                       ; Sufficent overhang is available
          7E  54    7D  0B3D          MOVQ    R4, -(SP)                ; Save YLO
     7E   54  56    61  0B40          ADDD3   R6, R4, -(SP)            ; Save Y
     50   6E  6E    65  0B44          MULD3   (SP), (SP), R0           ; R0/R1 = Y^2
F67A CF   07  50    75  0B48          POLYD   R0, #COSLENC2-1, COSTBC2; R0/R1 = Q(Y^2)
     54   56  8E    61  0B4E          ADDD3   (SP)+, R6, R4            ; R4/R5 = Y + YHI
          54  8E    64  0B52          MULD    (SP)+, R4                ; R4/R5 = YLO*(Y + YHI) = A2
              05    13  0B55          BEQL    1$                       ; Check for A2 = 0
     54  0100 8F    A2  0B57          SUBW    #^X100, R4               ; R4/R5 = A2/4
          50  54    62  0B5C     1$:  SUBD    R4, R0                   ; R0/R1 = Q(Y^2) - A2/4
          56  56    64  0B5F          MULD    R6, R6                   ; R6/R7 = YHI^2
     56  0100 8F    A2  0B62          SUBW    #^X100, R6               ; R6/R7 = YHI^2/4
          56  08    62  0B67          SUBD    #1, R6                   ; R6/R7 = -(1 - YHI^2/4)
          50  56    62  0B6A          SUBD    R6, R0                   ; R0/R1 = DCOS(Y)
                    05  0B6D          RSB
                        0B6E
          56  54    60  0B6E     2$:  ADDD    R4, R6                   ; R6/R7 = Y
          56  56    64  0B71          MULD    R6, R6                   ; R6/R7 = Y^2
F60E CF   07  56    75  0B74          POLYD   R6, #COSLENC1-1, COSTBC1; R0/R1 = DCOS(Y) - 1
          50  08    60  0B7A          ADDD    #1, R0                   ; R0/R1 = DCOS(Y)
                    05  0B7D          RSB
                        0B7E
                        0B7E
                        0B7E     ;
                        0B7E     ; Polynomial evaluation for -DCOS(Y)
                        0B7E     ;
                        0B7E
                        0B7E     N_COS_C:
     56  F4A6 CF    71  0B7E          CMPD    D_2_OV_PI, R6            ; Compare 2/pi with |YHI|
              32    18  0B83          BGEQ    2$                       ; Sufficent overhang is available
          7E  54    7D  0B85          MOVQ    R4, -(SP)                ; Save YLO
     7E   54  56    61  0B88          ADDD3   R6, R4, -(SP)            ; Save Y
     50   6E  6E    65  0B8C          MULD3   (SP), (SP), R0           ; R0/R1 = Y^2
F632 CF   07  50    75  0B90          POLYD   R0, #COSLENC2-1, COSTBC2; R0/R1 = Q(Y^2)
     54   56  8E    61  0B96          ADDD3   (SP)+, R6, R4            ; R4/R5 = Y + YHI
          54  8E    64  0B9A          MULD    (SP)+, R4                ; R4/R5 = YLO*(Y + YHI) = A2
              05    13  0B9D          BEQL    1$                       ; Check for A2 = 0
     54  0100 8F    A2  0B9F          SUBW    #^X100, R4               ; R4/R5 = A2/4
          50  54    62  0BA4     1$:  SUBD    R4, R0                   ; R0/R1 = Q(Y^2) - A2/4
          56  56    64  0BA7          MULD    R6, R6                   ; R6/R7 = YHI^2
     56  0100 8F    A2  0BAA          SUBW    #^X100, R6               ; R6/R7 = YHI^2/4
          56  08    62  0BAF          SUBD    #1, R6                   ; R6/R7 = -(1 - YHI^2/4)
     50   56  50    63  0BB2          SUBD3   R0, R6, R0               ; R0/R1 = -DCOS(Y)
                    05  0BB6          RSB
                        0BB7
          56  54    60  0BB7     2$:  ADDD    R4, R6                   ; R6/R7 = Y
          56  56    64  0BBA          MULD    R6, R6                   ; R6/R7 = Y^2
```

```
          F5C5 CF   07   56   75  0BBD  1654          POLYD   R6, #COSLENC1-1, COSTBC1; R0/R1 = DCOS(Y) - 1
50  00000000 0000C080 8F   50   63  0BC3  1655          SUBD3   R0, #-1, R0             ; R0/R1 = -DCOS(Y)
                                05  0BCF  1656          RSB
                                    0BD0  1657
                                    0BD0  1658 ;
                                    0BD0  1659 ; Polynomial evaluation for -SIN(Y)
                                    0BD0  1660 ;
                                    0BD0  1661
                                    0BD0  1662 N_SIN_C:
               54   8000 8F   AC  0BD0  1663          XORW    #^X8000, R4             ;
               56   8000 8F   AC  0BD5  1664          XORW    #^X8000, R6             ; R4/R7 = - Y
                                    0BDA  1665
                                    0BDA  1666 ;
                                    0BDA  1667 ; Polynomial evaluation for DSIN(Y)
                                    0BDA  1668 ;
                                    0BDA  1669
                                    0BDA  1670 P_SIN_C:
                    7E   54   7D  0BDA  1671          MOVQ    R4, -(SP)               ; Save YLO
               7E   56   54   61  0BDD  1672          ADDD3   R4, R6, -(SP)           ; Save Y
               54   6E   6E   65  0BE1  1673          MULD3   (SP), (SP), R4          ; R4 = Y^2
          F61D CF   07   54   75  0BE5  1674          POLYD   R4, #SINLENC-1, SINTBC  ; R0/R1 = P(Y^2)
                    50   8E   64  0BEB  1675          MULD    (SP)+, R0               ; R0/R1 = Y*P(Y^2)
                    50   6E   60  0BEE  1676          ADDD    (SP), R0                ; R0/R1 = Y*P(Y^2) + YLO
               6E   0100 8F   A2  0BF1  1677          SUBW    #^X100, (SP)            ; (SP) = YLO/4
                    50   8E   62  0BF6  1678          SUBD    (SP)+, R0               ; R0/R1 = Y*P(Y^2) + 3/4*YLO
                    54   56   7D  0BF9  1679          MOVQ    R6, R4                  ; R4/R5 = YHI
               54   0100 8F   A2  0BFC  1680          SUBW    #^X100, R4              ; R4/R5 = YHI/4
                    56   54   62  0C01  1681          SUBD    R4, R6                  ; R6/R7 = 3/4*YHI
                    50   56   60  0C04  1682          ADDD    R6, R0                  ; R0/R1 = DSIN(Y)
                                05  0C07  1683          RSB
                                    0C08  1684
```

```
                              0C08          .SBTTL  DEGREE_POLYNOMIALS
                              0C08
                              0C08
                              0C08  P_COS_D:
         56   F44C CF    71   0C08          CMPD    D_90_OV_PI, R6           ; Compare 90/pi with Y
                    37   18   0C0D          BGEQ    2$                       ; Double precision isn't needed
         54    56   56   65   0C0F          MULD3   R6, R6, R4               ; R0/R1 = Y^2
    F66F CF    07   54   75   0C13          POLYD   R4, #COSDLN1, COSDTB1    ; R0/R1 = Q(Y^2)
  54   56   00070000 8F  CB   0C19          BICL3   #^X70000, R6, R4         ;
                    55   D4   0C21          CLRL    R5                       ; R4/R5 = YHI
         52    56   54   63   0C23          SUBD3   R4, R6, R2               ; R2 = YLO
               56   54   60   0C27          ADDD    R4, R6                   ; R6/R7 = Y + YHI
               56   52   64   0C2A          MULD    R2, R6                   ; R6/R7 = YLO*(Y + YHI) = A2
                    05   13   0C2D          BEQL    1$                       ; Check for A2 = 0
         56   0680 8F    A2   0C2F          SUBW    #^X680, R6               ; R6/R7 = A2/2^13
               50   56   62   0C34  1$:     SUBD    R6, R0                   ; R0/R1 = Q(Y^2) - A2/2^13
               54   54   64   0C37          MULD    R4, R4                   ; R4/R5 = YHI^2
         54   0680 8F    A2   0C3A          SUBW    #^X680, R4               ; R4/R5 = YHI^2/2^13
               54   08   62   0C3F          SUBD    #1, R4                   ; R4/R5 = -(1 - YHI^2/2^13)
               50   54   62   0C42          SUBD    R4, R0                   ; R0/R1 = DCOS(Y)
                         05   0C45          RSB
                              0C46
         56    56        64   0C46  2$:     MULD    R6, R6                   ; R6/R7 = Y^2
               07        13   0C49          BEQL    3$                       ; Check for Y = 0
    F5F7 CF    07   56   75   0C4B          POLYD   R6, #COSDLN2, COSDTB2    ; R0/R1 = Q(Y^2)
                         05   0C51          RSB
                              0C52
         50    08        70   0C52  3$:     MOVD    #1, R0                   ; R0/R1 = DCOS(Y)
                         05   0C55          RSB
                              0C56
                              0C56
                              0C56  N_COS_D:
         56   F3FE CF    71   0C56          CMPD    D_90_OV_PI, R6           ; Compare 90/pi with Y
                    38   18   0C5B          BGEQ    2$                       ; Double precision isn't needed
         54    56   56   65   0C5D          MULD3   R6, R6, R4               ; R0/R1 = Y^2
    F621 CF    07   54   75   0C61          POLYD   R4, #COSDLN1, COSDTB1    ; R0/R1 = Q(Y^2)
  54   56   00070000 8F  CB   0C67          BICL3   #^X70000, R6, R4         ;
                    55   D4   0C6F          CLRL    R5                       ; R4/R5 = YHI
         52    56   54   63   0C71          SUBD3   R4, R6, R2               ; R2 = YLO
               56   54   60   0C75          ADDD    R4, R6                   ; R6/R7 = Y + YHI
               56   52   64   0C78          MULD    R2, R6                   ; R6/R7 = YLO*(Y + YHI) = A2
                    05   13   0C7B          BEQL    1$                       ; Check for A2 = 0
         56   0680 8F    A2   0C7D          SUBW    #^X680, R6               ; R6/R7 = A2/2^13
               50   56   62   0C82  1$:     SUBD    R6, R0                   ; R0/R1 = Q(Y^2) - A2/2^13
               54   54   64   0C85          MULD    R4, R4                   ; R4/R5 = YHI^2
         54   0680 8F    A2   0C88          SUBW    #^X680, R4               ; R4/R5 = YHI^2/2^13
               54   08   62   0C8D          SUBD    #1, R4                   ; R4/R5 = -(1 - YHI^2/2^13)
         50    54   50   63   0C90          SUBD3   R0, R4, R0               ; R0/R1 = -DCOS(Y)
                         05   0C94          RSB
                              0C95
         56    56        64   0C95  2$:     MULD    R6, R6                   ; R6/R7 = Y^2
               0C        13   0C98          BEQL    3$                       ; Check for Y = 0
    F5A8 CF    07   56   75   0C9A          POLYD   R6, #COSDLN2, COSDTB2    ; R0/R1 = DCOSD(Y)
         50   8000 8F    AC   0CA0          XORW    #^X8000, R0              ; R0/R1 = -DCOSD(Y)
                         05   0CA5          RSB
                              0CA6
  50   00000000 0000C080 8F  70   0CA6  3$:     MOVD    #-1, R0                  ; R0/R1 = DCOS(Y)
```

```
                        05  0CB1  1743          RSB
                            0CB2  1744
                            0CB2  1745 N_SIN_D:
              56    56  72  0CB2  1746          MNEGD   R6, R6                          ; R6/R7 = -Y
                            0CB5  1747 P_SIN_D:
        50    56    56  65  0CB5  1748          MULD3   R6, R6, R0                      ; R0/R1 = Y^2
                    11  13  0CB9  1749          BEQL    RETURN
  F607 CF     07    50  75  0CBB  1750          POLYD   R0, #SINDLN, SINDTB             ; R0/R1 = P(Y^2)
              50    56  64  0CC1  1751          MULD    R6, R0                          ; R0/R1 = Y*P(Y^2)
        56  0300    8F  A2  0CC4  1752          SUBW    #^X300, R6                      ; R6/R7 = Y/2^6
              50    56  60  0CC9  1753          ADDD    R6, R0                          ; R0/R1 = DSIN(Y)
                        05  0CCC  1754 RETURN:  RSB
                            0CCD  1755
```

```
                                   0CCD
                                   0CCD
                                   0CCD               .SBTTL  DEGENERATE_SOLUTIONS
                                   0CCD
                                   0CCD      P_ONE:
               50      08      70  0CCD               MOVD    #1, R0                   ; Answer is 1
                               05  0CD0               RSB
                                   0CD1
                                   0CD1
                                   0CD1      N_ONE:
  50   00000000 0000C080 8F    70  0CD1               MOVD    #-1, R0                  ; Answer is -1
                               05  0CDC               RSB
                                   0CDD
                                   0CDD
                                   0CDD      UNFL:
                                   0CDD      ;
                                   0CDD      ; Underflow; if user has FU set, signal error.  Always return 0.0
                                   0CDD      ;
                       52      DC  0CDD               MOVPSL  R2                       ; R2 = user's or jacket routine's PSL
          00000000'GF  00      FB  0CDF               CALLS   #0, G^MTH$$JACKET_TST    ; R0 = TRUE if JSB from jacket routine
                    04 50      E9  0CE6               BLBC    R0, 10$                  ; branch if user did JSB
            52      04 AD      3C  0CE9               MOVZWL  SF$W_SAVE_PSW(FP), R2    ; get user PSL saved by CALL
                       50      D4  0CED      10$:     CLRL    R0                       ; R0 = result. LIB$SIGNAL will save in
                                   0CEF                                                ; CHF$L_MCH_R0/R1 so any handler can
                                   0CEF                                                ;   fixup
                 0D 52 06      E1  0CEF               BBC     #6, R2, 20$              ; has user enabled floating underflow?
                       6E      DD  0CF3               PUSHL   (SP)                     ; yes, return PC from special routine
               7E   00'8F      9A  0CF5               MOVZBL  #MTH$K_FLOUNDMAT, -(SP)  ; trap code for hardware floating
                                   0CF9                                                ;   underflow convert to MTH$_FLOUNDMAT
                                   0CF9                                                ;   (32-bit VAX-11 exception code)
          00000000'GF  02      FB  0CF9               CALLS   #2, G^MTH$$SIGNAL        ; signal (condition, PC)
                               05  0D00      20$:     RSB                              ; return
                                   0D01
                                   0D01               .END
```

```
CONVERT                  00000816 R     02   MTH$DCOS                 00000338 RG    02
CONVERT_0                000008E0 R     02   MTH$DCOSD                0000037C RG    02
CONVERT_1                00000890 R     02   MTH$DCOSD_R7             00000661 RG    02
COSD                   = 0000000C            MTH$DCOS_R7              0000051A RG    02
COSDLN1                = 00000007            MTH$DSIN                 00000324 RG    02
COSDLN2                = 00000007            MTH$DSINCOS              00000308 RG    02
COSDTB1                  00000288 R     02   MTH$DSINCOSD             0000034C RG    02
COSDTB2                  00000248 R     02   MTH$DSINCOSD_R7          000005A8 RG    02
COSINE                 = 0000000C            MTH$DSINCOS_R7           00000390 RG    02
COSLENC1               = 00000008            MTH$DSIND                00000368 RG    02
COSLENC2               = 00000008            MTH$DSIND_R7             00000603 RG    02
COSLENR1               = 00000008            MTH$DSIN_R7              00000493 RG    02
COSLENR2               = 00000008            MTH$K_FLOUNDMAT          ********    X  00
COSTBC1                  00000188 R     02   M_COS                    00000535 R     02
COSTBC2                  000001C8 R     02   M_SIN                    000004BA R     02
COSTBR1                  000000C8 R     02   NEEDS_DOUBLE             00000A82 R     02
COSTBR2                  00000108 R     02   NEG_SIND                 00000611 R     02
CVT                      00000A65 R     02   NOT_ENOUGH_BITS          000006F4 R     02
CVT_TO_DOUBLE            00000951 R     02   NO_OVERFLOW              00000A46 R     02
DEGENERATE_CASE_COS      00000594 R     02   N_COS_C                  00000B7E R     02
DEGENERATE_CASE_SIN      00000506 R     02   N_COS_D                  00000C56 R     02
D_1_OV_45                00000048 R     02   N_COS_R                  00000AC0 R     02
D_2_OV_PI                00000028 R     02   N_ONE                    00000CD1 R     02
D_2_TO_32                00000710 R     02   N_SIN_C                  00000BD0 R     02
D_3_PI_OV_4              00000010 R     02   N_SIN_D                  00000CB2 R     02
D_45                     00000030 R     02   N_SIN_R                  00000B11 R     02
D_5_PI_OV_4              00000018 R     02   ODD                      00000A58 R     02
D_7_PI_OV_4              00000020 R     02   PI_OV_2                  00000068 R     02
D_90_OV_PI               00000058 R     02   POS_SIN                  000004A6 R     02
D_9_PI_OV_4              00000008 R     02   POS_SINCOS               000003A4 R     02
D_CONVERT                00000050 R     02   POS_SIND                 00000616 R     02
D_M45                    00000038 R     02   POWER_MOD_360_0          0000099B R     02
D_PI_OV_4                00000000 R     02   POWER_MOD_360_1          000009B3 R     02
D_SMALLD                 00000040 R     02   PSL$M_IV               = 00000020
D_SMALLEST_DEG           00000060 R     02   P_COS_C                  00000B36 R     02
EVAL_COSD                00000675 R     02   P_COS_D                  00000C08 R     02
EVAL_SIND                00000623 R     02   P_COS_R                  00000A75 R     02
EVEN                     00000A68 R     02   P_ONE                    00000CCD R     02
GEN_MORE_BITS            0000090B R     02   P_SIN_C                  00000BDA R     02
GET_YHI_YLO              000008F9 R     02   P_SIN_D                  00000CB5 R     02
LARGE_COS                00000576 R     02   P_SIN_R                  00000B1B R     02
LARGE_SIN                000004E8 R     02   REDUCE_DEGREES           000009CB R     02
LARGE_SINCOS             00000469 R     02   REDUCE_LARGE             00000718 R     02
LAST_STEP                00000A1B R     02   REDUCE_MEDIUM            0000069A R     02
LEADING_ONES             00000851 R     02   RESTORE                  00000904 R     02
LEADING_ZEROS            000008B4 R     02   RETURN                   00000CCC R     02
LEQL_HALF                00000AB3 R     02   SF$W_SAVE_PSW          = 00000004
LONG                   = 00000004            SIN                      000004A1 R     02
LOOP_0                   000008BB R     02   SINCOS                   0000039F R     02
LOOP_1                   00000858 R     02   SINCOSD                  000005BB R     02
L_COS                    0000057C R     02   SIND                   = 00000008
L_INT_WEIGHT           = 00001E80            SINDLN                 = 00000007
L_SIN                    000004EE R     02   SINDTB                   000002C8 R     02
MTH$$JACKET_HND          ********    X  02   SINE                   = 00000008
MTH$$JACKET_TST          ********    X  00   SINLENC                = 00000008
MTH$$SIGNAL              ********    X  00   SINLENR                = 00000008
MTH$AL_4_OV_PI           ********    X  00   SINTBC                   00000208 R     02
```

```
SINTBR                          00000148 R      02
SMALL_COS                       00000549 R      02
SMALL_COSD                      00000689 R      02
SMALL_SIN                       000004CE R      02
SMALL_SINCOS                    000003DF R      02
SMALL_SINCOSD                   000005E6 R      02
SMALL_SIND                      00000637 R      02
SUBTRACT                        000006C8 R      02
UNFL                            00000CDD R      02
W_ADJUST                      = 00000039
W_MAX_WEIGHT                  = 00004000
W_TERM_WEIGHT                 = 00001000
X                             = 00000004
X_1_OV_45                     = 0000000B

                                            +----------------+
                                            ! 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                       00000D01  (   3329.)  02 (  2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC LONG

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

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                      35    00:00:00.08    00:00:00.59
Command processing                 116    00:00:00.68    00:00:03.34
Pass 1                             206    00:00:05.92    00:00:18.17
Symbol table sort                    0    00:00:00.27    00:00:00.42
Pass 2                             320    00:00:03.90    00:00:13.87
Symbol table output                 15    00:00:00.14    00:00:00.60
Psect synopsis output                3    00:00:00.02    00:00:00.02
Cross-reference output               0    00:00:00.00    00:00:00.00
Assembler run totals               697    00:00:11.02    00:00:37.02

The working set limit was 1650 pages.
38842 bytes (76 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 194 non-local and 57 local symbols.
1850 source lines were read in Pass 1, producing 35 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$:MTHDSINCO/OBJ=OBJ$:MTHDSINCO MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MS
```

MTHDCOSH
LIS

MTHDMIN1
LIS

MTHDLOG
LIS

MTHDSINCO
LIS

MTHDATANH
LIS

MTHDNINT
LIS

MTHDSQRT
LIS

MTHDCONJG
LIS

MTHDINT
LIS

MTHDMAX1
LIS

MTHDSIGN
LIS

MTHDIM
LIS

MTHDMOD
LIS

MTHDSINH
LIS

MTHDEXP
LIS

MTHDFLOOR
LIS

MTHDPROD
LIS