```
MMM          MMM  TTTTTTTTTTTTTTT   HHH          HHH  RRRRRRRRRRR     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         HHHHHHHHHHHHHHHH  RRRRRRRRRRR           TTT         LLL
MMM         MMM         TTT         HHHHHHHHHHHHHHHH  RRRRRRRRRRRR          TTT         LLL
MMM         MMM         TTT         HHHHHHHHHHHHHHHH  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         LLLLLLLLLLLLLLLL
MMM         MMM         TTT         HHH          HHH  RRR         RRR       TTT         LLLLLLLLLLLLLLLL
MMM         MMM         TTT         HHH          HHH  RRR         RRR       TTT         LLLLLLLLLLLLLLLL
```

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

```
0000     1         .TITLE  MTH$GSINCOS     ; Floating Point Sine, Cosine and Sincos
0000     2                                 ;    Functions
0000     3         .IDENT /2-003/          ; File: MTHGSINCOS.MAR  EDIT: RNH2003
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ;
0000    29 ; FACILITY:     MATH LIBRARY
0000    30 ;++
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ; MTH$GSIN and MTH$GCOS are functions which return the floating point
0000    34 ; sine or cosine value of their single precision floating point argu-
0000    35 ; ment (radians). The call is standard call-by-reference.
0000    36 ; MTH$GSIN_R7 and MTH$GCOS_R7 are special routines which are the same
0000    37 ; as MTH$GSIN and MTH$GCOS  except  a faster non-standard JSB call is
0000    38 ; used with the argument in R0 and no registers are saved.
0000    39 ;
0000    40 ; MTH$GSINCOS  is  a routine which returns the floating point sine and
0000    41 ; cosine value of its single precision floating point radian argument.
0000    42 ; The call is standard call-by-reference.  MTH$GSINCOS_R7 is a special
0000    43 ; routine  which  is  the  same  as  MTH$GSINCOS, except a faster non-
0000    44 ; standard JSB call is used with the argument in R0 and no  registers
0000    45 ; are saved.
0000    46 ;
0000    47 ; MTH$GSIND and MTH$GCOSD are functions which return the floating point
0000    48 ; sine  or  cosine value of their single precision floating point argu-
0000    49 ; ment (degrees). The call is standard call-by-reference.
0000    50 ; MTH$GSIND_R7 and MTH$GCOSD_R7 are special routines which are the same
0000    51 ; as MTH$GSIND and MTH$GCOSD  except  a faster non-standard JSB call is
0000    52 ; used with the argument in R0 and no registers are saved.
0000    53 ;
0000    54 ; MTH$GSINCOSD is  a routine which returns the floating point sine and
0000    55 ; cosine value of its single precision floating point degree argument.
0000    56 ; The call is standard call-by-reference. MTH$GSINCOSD_R7 is a special
0000    57 ; routine  which  is  the  same  as MTH$GSINCOSD, except a faster non-
```

```
0000     58 ; standard JSB c ll is used with the argument in R0 and no  registers
0000     59 ; are saved.
0000     60 ;
0000     61 ;--
0000     62 ;
0000     63 ; VERSION:      1
0000     64 ;
0000     65 ; HISTORY:
0000     66 ; AUTHOR:
0000     67 ;       MARY PAYNE & JUD LEONARD, 25-MAY-78:    Version 0
0000     68 ;
0000     69 ; MODIFIED BY:
0000     70 ;
0000     71 ; 1-1   Tryggve Fossum, 28-May-78
0000     72 ;
0000     73 ;
0000     74 ; VERSION:      2
0000     75 ;
0000     76 ; HISTORY:
0000     77 ; AUTHOR:
0000     78 ;       BOB HANEK, 25-MAY-78:   Version 2
0000     79 ;
0000     80 ;
0000     81 ; Edit history for Version 2:
0000     82 ;
0000     83 ; 2-001 - Original
0000     84 ; 2-002 - Change MTH$AL_4_OV_PI to MTH$AL_4_OV_PI_V.  RNH 29-Sep-81
0000     85 ; 2-003 - Modified REDUCE_LARGE to correct bug reported in QAR 896.
0000     86 ;         RNH 14-Jan-82
```

```
                                        .SBTTL  DECLARATIONS    -       Declarative Part of Module

                                ;
                                ; INCLUDE FILES:          MTH$JACKET.MAR
                                ;
                                ; EXTERNAL SYMBOLS:
                                ;
                                        .DSABL  GBL
                                        .EXTRN  MTH$AL_4_OV_PI_V
                                        .EXTRN  MTH$$SIGNAL
                                        .EXTRN  MTH$K_FLOUNDMAT
                                        .EXTRN  MTH$$JACKET_TST
                                ;
                                ; EQUATED SYMBOLS:
                                ;
                                        X_1_OV_45       = ^XC16C
                                ;
                                ; MACROS:
                                ;
                                        $SFDEF                  ; Define SF$ (stack frame) symbols
                                        $PSLDEF                 ; Define PSL$ symbols
                                ;
                                ; PSECT DECLARATIONS:
                                ;
                                        .PSECT  _MTH$CODE       PIC,SHR,LONG,EXE,NOWRT
                                                                ; program section for math routines
                                ;
                                ; OWN STORAGE:  none
                                ;
                                ; CONSTANTS:
                                ;
                                G_M1:
00000000 0000C010               .QUAD   ^X000000000000C010      ; -1
                                G_PI_OV_4:
2D185444 21FB4009               .QUAD   ^X2D18544421FB4009      ; 0.7853981633974483E+00
                                G_9_PI_OV_4:
B2BBBECC 463A403C               .QUAD   ^XB2BBBECC463A403C      ; 0.7068583470577035E+01
                                G_3_PI_OV_4:
21D27F33 D97C4022               .QUAD   ^X21D27F33D97C4022      ; 0.2356194490192345E+01
                                G_5_PI_OV_4:
385E2955 6A7A402F               .QUAD   ^X385E29556A7A402F      ; 0.3926990816987241E+01
                                G_7_PI_OV_4:
A775E9BB FDBB4035               .QUAD   ^XA775E9BBFDBB4035      ; 0.5497787143782138E+01

                                G_45:
00000000 80004066               .QUAD   ^X0000000080004066      ; 0.4500000000000000E+02
                                G_M45:
00000000 8000C066               .QUAD   ^X000000008000C066      ; -.4500000000000000E+02
                                G_SMALLD:
C1F81A63 A5DC3EBC               .QUAD   ^XC1F81A63A5DC3EBC      ; 0.4268868231257969E-06
                                G_1_OV_45:
6C1716C1 C16C3FB6               .QUAD   ^X6C1716C1C16C3FB6      ; 0.2222222222222222E-01
                                G_CONVERT:
D3912529 F46A3F7D               .QUAD   ^XD3912529F46A3F7D      ; 0.1828292519943295E-02
                                G_90_OV_PI:
C1F81A63 A5DC405C               .QUAD   ^XC1F81A63A5DC405C      ; 0.2864788975654116E+02
```

```
                                0060   145 G_SMALLEST_DEG:
                C1F81A63 A5DC006C 0060   146         .QUAD   ^XC1F81A63A5DC006C      ; 0.3187183529933799-306
                                0068   147
                                0068   148 H_PI_OV_2:                              ; 1.57079632679489661923132169163975
01B8C517 898C8469 42D1B544 921F4001 0068   149         .OCTA   ^X01B8C517898C846942D1B544921F4001
                                0078   150 H_PI:                                   ; 3.14159265358979323846264338327950
01B8C517 898C8469 42D1B544 921F4002 0078   151         .OCTA   ^X01B8C517898C846942D1B544921F4002
                                0088   152 H_3_PI_OV_2:                           ; 4.71238898038468985769396507491925
414A93D1 2729234F 321DC7F3 2D974003 0088   153         .OCTA   ^X414A93D12729234F321DC7F32D974003
                                0098   154 H_2_PI:                                ; 6.28318530717958647692528676655901
01B8C517 898C8469 42D1B544 921F4003 0098   155         .OCTA   ^X01B8C517898C846942D1B544921F4003
                                00A8   156 H_2_OV_PI:                             ; .63661977236758134307535053490057
EA6AAFA3 F84E2A53 9C8806DC 45F34000 00A8   157         .OCTA   ^XEA6AAFA3F84E2A539C8806DC45F34000
```

```
                        00B8   159
                        00B8   160          .SBTTL  COEFFICIENT TABLES      -       Series Coefficients
                        00B8   161
                        00B8   162
                        00B8   163
                        00B8   164
                        00B8   165 ;
                        00B8   166 ; Polynomial Coefficient tables for arguments in radians
                        00B8   167 ;
                        00B8   168
                        00B8   169 COSTBR1:             ; GCOS coefficients for arguments less than 1/2
CD345EF0 F6ADBDC8       00B8   170          .QUAD   ^XCD345EF0F6ADBDC8      ; C7 = -.1135212320578394E-10
C93D4799 EE953E41       00C0   171          .QUAD   ^XC93D4799EE953E41      ; C6 = 0.2087555514567788E-08
19916FE2 7E4FBEB2       00C8   172          .QUAD   ^X19916FE27E4FBEB2      ; C5 = -.2755731286569608E-06
767E19AD 01A03F1A       00D0   173          .QUAD   ^X767E19AD01A03F1A      ; C4 = 0.2480158728289946E-04
362D16C1 C16CBF76       00D8   174          .QUAD   ^X362D16C1C16CBF76      ; C3 = -.1388888888885896E-02
55335555 55553FC5       00E0   175          .QUAD   ^X5533555555553FC5      ; C2 = 0.4166666666666643E-01
00000000 0000C000       00E8   176          .QUAD   ^X000000000000C000      ; C1 = -.5000000000000000E+00
00000000 00004010       00F0   177          .QUAD   ^X0000000000004010      ; C0 = 0.1000000000000000E+01
          00000008      00F8   178 COSLENR1 = .-COSTBR1/8
                        00F8   179
                        00F8   180 COSTBR2:             ; GCOS coefficients for arguments greater than 1/2
CD345EF0 F6ADBDC8       00F8   181          .QUAD   ^XCD345EF0F6ADBDC8      ; C7 = -.1135212320578394E-10
C93D4799 EE953E41       0100   182          .QUAD   ^XC93D4799EE953E41      ; C6 = 0.2087555514567788E-08
19916FE2 7E4FBEB2       0108   183          .QUAD   ^X19916FE27E4FBEB2      ; C5 = -.2755731286569608E-06
767E19AD 01A03F1A       0110   184          .QUAD   ^X767E19AD01A03F1A      ; C4 = 0.2480158728289946E-04
362D16C1 C16CBF76       0118   185          .QUAD   ^X362D16C1C16CBF76      ; C3 = -.1388888888885896E-02
55335555 55553FC5       0120   186          .QUAD   ^X5533555555553FC5      ; C2 = 0.4166666666666643E-01
5EE62C40 80673C80       0128   187          .QUAD   ^X5EE62C4080673C80      ; C1 = 0.7156417079102195E-17
5D30B344 297FBC05       0130   188          .QUAD   ^X5D30B344297FBC05      ; C0 = -.3584999999999999E-19
          00000008      0138   189 COSLENR2 = .-COSTBR2/8
                        0138   190
                        0138   191 SINTBR:             ; GSIN coefficients
B5E36E13 D8403E05       0138   192          .QUAD   ^XB5E36E13D8403E05      ; C6 = 0.1589413523004633E-09
3319B5B3 E5E2BE7A       0140   193          .QUAD   ^X3319B5B3E5E2BE7A      ; C5 = -.2505070582636817E-07
5AD851F3 1DE33EE7       0148   194          .QUAD   ^X5AD851F31DE33EE7      ; C4 = 0.2755731329888568E-05
2F3C19B9 01A0BF4A       0150   195          .QUAD   ^X2F3C19B901A0BF4A      ; C3 = -.1984126982840175E-03
F30C1110 11113FA1       0158   196          .QUAD   ^XF30C111011113FA1      ; C2 = 0.8333333333320002E-02
55435555 5555BFE5       0160   197          .QUAD   ^X554355555555BFE5      ; C1 = -.1666666666666662E+00
033809DB 8D82BC6E       0168   198          .QUAD   ^X 33809DB8D82BC6E      ; C0 = -.3312537470886997E-17
          00000007      0170   199 SINLENR = .-SINTBR/8
                        0170   200
                        0170   201
                        0170   202
                        0170   203
                        0170   204
                        0170   205 ;
                        0170   206 ; Polynomial coefficients for arguments in cycles
                        0170   207 ;
                        0170   208
                        0170   209 COSTBC1:             ; GCOS coefficients for arguments less than 2/pi
815BB995 2586BD7B       0170   210          .QUAD   ^X815BB99525 86BD7B     ; C7 = -.3857762037200000E-12
4DE15D51 9CC13DFF       0178   211          .QUAD   ^X4DE15D519CC13DFF      ; C6 = 0.1150049702426300E-09
8016C3D4 6D1EBE7A       0180   212          .QUAD   ^X8016C3D46D1EBE7A      ; C5 = -.2461136382637005E-07
3F246830 1F503EEE       0188   213          .QUAD   ^X3F2468301F503EEE      ; C4 = 0.3590860445885820E-05
8D5C7E3C 5D3CBF55       0190   214          .QUAD   ^X8D5C7E3C5D3CBF55      ; C3 = -.3259918869266876E-03
5AAA081B 3C1F3FB0       0198   215          .QUAD   ^X5AAA081B3C1F3FB0      ; C2 = 0.1585434424381541E-01
```

```
45DEC9BE BD3CBFF3  01A0   216          .QUAD   ^X45DEC9BEBD3CBFF3      ; C1 = -.3084251375340424E+00
5D30B344 297FBC05  01A8   217          .QUAD   ^X5D30B344297FBC05      ; C0 = -.3584999999999999E-19
         00000008  01B0   218 COSLENC1 = .-COSTBC1/8
                   01B0   219
                   01B0   220 COSTBC2:          ; GCOS coefficients for arguments greater than 2/pi
815BB995 2586BD7B  01B0   221          .QUAD   ^X815BB9952586BD7B      ; C7 = -.3857762037200000E-12
4DE15D51 9CC13DFF  01B8   222          .QUAD   ^X4DE15D519CC13DFF      ; C6 = 0.1150049702426300E-09
8016C3D4 6D1EBE7A  01C0   223          .QUAD   ^X8016C3D46D1EBE7A      ; C5 = -.2461136382637005E-07
3F246830 1F503EEE  01C8   224          .QUAD   ^X3F2468301F503EEE      ; C4 = 0.3590860445885820E-05
8D5C7E3C 5D3CBF55  01D0   225          .QUAD   ^X8D5C7E3C5D3CBF55      ; C3 = -.3259978869266876E-03
5AAA081B 3C1F3FB0  01D8   226          .QUAD   ^X5AAA081B3C1F3FB0      ; C2 = 0.1585434424381541E-01
2EF24DF2 E9E6BFCD  01E0   227          .QUAD   ^X2EF24DF2E9E6BFCD      ; C1 = -.5842513753404245E-01
5D30B344 297FBC05  01E8   228          .QUAD   ^X5D30B344297FBC05      ; C0 = -.3584999999999999E-19
         00000008  01F0   229 COSLENC2 = .-COSTBC2/8
                   01F0   230
                   01F0   231 SINTBC:           ; GSIN coef for arg in cycles
CB82386D 3EED3DBE  01F0   232          .QUAD   ^XCB82386D3EED3DBE      ; C6 = 0.6877100349000000E-11
0141399B 3006BE3E  01F8   233          .QUAD   ^X 141399B3006BE3E      ; C5 = -.1757149292755000E-08
1EF8FCA3 07823EB5  0200   234          .QUAD   ^X1EF8FCA307823EB5      ; C4 = 0.3133616216619040E-06
52FACE2D 2D2CBF23  0208   235          .QUAD   ^X52FACE2D2D2CBF23      ; C3 = -.3657620415845570E-04
86FF6775 66BC3F84  0210   236          .QUAD   ^X86FF677566BC3F84      ; C2 = 0.2490394570188736E-02
BE41E625 ABBCBFD4  0218   237          .QUAD   ^XBE41E625ABBCBFD4      ; C1 = -.8074551218828054E-01
D1844442 1FB53FC2  0220   238          .QUAD   ^XD18444421FB53FC2      ; C0 = 0.3539816339744831E-01
         00000007  0228   239 SINLENC = .-SINTBC/8
                   0228   240
                   0228   241
                   0228   242
                   0228   243
                   0228   244
                   0228   245 ;
                   0228   246 ; Polynomial coefficients for arguments in degrees
                   0228   247 ;
                   0228   248
                   0228   249 COSDTB2:          ; GCOS coefficients for arguments less than 90/pi
20197263 613EB8AD  0228   250          .QUAD   ^X20197263613EB8AD      ; C7 = -.2762868673216389E-35
ADD69711 EA1439E0  0230   251          .QUAD   ^XADD69711EA1439E0      ; C6 = 0.1667886312398853E-29
B183A296 F623BB0B  0238   252          .QUAD   ^XB183A296F623BB0B      ; C5 = -.7227873495985314E-24
D1DD5C04 83AB3C2F  0240   253          .QUAD   ^XD1DD5C0483AB3C2F      ; C4 = 0.2135494301985904E-18
78AF5BBC 19B8BD46  0248   254          .QUAD   ^X78AF5BBC19B8BD46      ; C3 = -.3925831985734635E-13
DC736A83 9B113E50  0250   255          .QUAD   ^XDC736A839B113E50      ; C2 = 0.3866323851562971E-08
1FB9DB14 F6A1BF43  0258   256          .QUAD   ^X1FB9DB14F6A1BF43      ; C1 = -.1523087098933543E-03
00000000 00004010  0260   257          .QUAD   ^X0000000000004010      ; C0 = 0.1000000000000000E+01
         00000007  0268   258 COSDLN2 = .-COSDTB2/8 - 1
                   0268   259
                   0268   260 COSDTB1:          ; GCOS coefficients for arguments greater than 90/pi
20197263 613EB8AD  0268   261          .QUAD   ^X20197263613EB8AD      ; C7 = -.2762868673216389E-35
ADD69711 EA1439E0  0270   262          .QUAD   ^XADD69711EA1439E0      ; C6 = 0.1667886312398853E-29
B183A296 F623BB0B  0278   263          .QUAD   ^XB183A296F623BB0B      ; C5 = -.7227873495985314E-24
D1DD5C04 83AB3C2F  0280   264          .QUAD   ^XD1DD5C0483AB3C2F      ; C4 = 0.2135494301985904E-18
78AF5BBC 19B8BD46  0288   265          .QUAD   ^X78AF5BBC19B8BD46      ; C3 = -.3925831985734635E-13
DC736A83 9B113E50  0290   266          .QUAD   ^XDC736A839B113E50      ; C2 = 0.3866323851562971E-08
FDCCD8A0 B50EBF1F  0298   267          .QUAD   ^XFDCCD8A0B50EBF1F      ; C1 = -.3023839739335430E-04
5D30B344 297FBC05  02A0   268          .QUAD   ^X5D30B344297FBC05      ; C0 = -.3584999999999999E-19
         00000007  02A8   269 COSDLN1 = .-COSDTB1/8 - 1
                   02A8   270
                   02A8   271 SINDTB:           ; GSIN coefficients
17E0B735 04203947  02A8   272          .QUAD   ^X17E0B73504203947      ; C6   0.2216372140147286E-32
```

```
C8145B97 B6BEBA76  02B0   273          .QUAD   ^XC8145B97B6BEBA76      ; C5 = -.1146755972041862E-26
85EFDBD8 4A5F3B9F  02B8   274          .QUAD   ^X85EFDBD84A5F3B9F      ; C4 = 0.4141266526843263E-21
BA169F5A 368DBCBC  02C0   275          .QUAD   ^XBA169F5A368DBCBC      ; C3 = -.9788384855269454E-16
D93DEAE0 AD943DCD  02C8   276          .QUAD   ^XD93DEAE0AD943DCD      ; C2 = 0.1349601623161096E-10
2F5E0D94 BB82BECD  02D0   277          .QUAD   ^X2F5E0D94BB82BECD      ; C1 = -.8860961557012952E-06
D3912529 F46A3F7D  02D8   278          .QUAD   ^XD3912529F46A3F7D      ; C0 = 0.1828292519943296E-02
         00000006  02E0   279 SINDLN = .-SINDTB/8 - 1
                   02E0   280
```

```
                     02E0   282          .SBTTL  MTH$GSINCOS     -       Radian arguments
                     02E0   283
                     02E0   284
                     02E0   285 ;
                     02E0   286 ; FUNCTIONAL DESCRIPTION:
                     02E0   287 ;
                     02E0   288 ; The GSIN, GCOS and GSINCOS routines are based on octant reduction.  Given an
                     02E0   289 ; argument, x, it is written in the form
                     02E0   290 ;
                     02E0   291 ;           x = I1*(2*pi) + I*(pi/4) + Y1,
                     02E0   292 ;
                     02E0   293 ; where I1 and I are integers, 0 =< I < 8 and 0 =< Y1 < pi/4.  Since GSIN and
                     02E0   294 ; GCOS have a period of 2*pi it follows that
                     02E0   295 ;
                     02E0   296 ;           GSIN(x) = GSIN(I*(pi/4) + Y1)) and
                     02E0   297 ;           GCOS(x) = GCOS(I*(pi/4) + Y1)).
                     02E0   298 ;
                     02E0   299 ; Using the trigonometric identities for the sum and difference of two angles,
                     02E0   300 ; the following table can be generated:
                     02E0   301 ;
                     02E0   302 ;       If I =            then GSIN(x) =              and GCOS(x) =
                     02E0   303 ;       ------            -------------               ------------
                     02E0   304 ;         0                 GSIN(Y1)                   GCOS(Y1)
                     02E0   305 ;         1                 GCOS(pi/4-Y1)              GSIN(pi/4-Y1)
                     02E0   306 ;         2                 GCOS(Y1)                  -GSIN(Y1)
                     02E0   307 ;         3                 GSIN(pi/4-Y1)             -GCOS(pi/4-Y1)
                     02E0   308 ;         4                -GSIN(Y1)                  -GCOS(Y1)
                     02E0   309 ;         5                -GCOS(pi/4-Y1)             -GSIN(pi/4-Y1)
                     02E0   310 ;         6                -GCOS(Y1)                   GSIN(Y1)
                     02E0   311 ;         7                -GSIN(pi/4-Y1)              GCOS(pi/4-Y1)
                     02E0   312 ;
                     02E0   313 ; Let Y be defined as Y = Y1 if I is even and Y = pi/4 - Y1, if I is odd, then
                     02E0   314 ; each entry of the above table is of the for +/-GSIN(Y) or +/-GCOS(Y).  Based
                     02E0   315 ; on the above remarks, the GSIN, GCOS and GSINCOS routines process the input
                     02E0   316 ; argument x, to obtain I and Y, and based on I selects a suitable polynomial
                     02E0   317 ; approximation, p(Y), to evaluate the desired fuction.
                     02E0   318
                     02E0   319 ;
                     02E0   320 ; INPUT PARAMETERS:
                     02E0   321 ;
00000004             02E0   322          LONG    = 4
00000004             02E0   323          x       = 1*LONG            ; x is input angle in radians
00000008             02E0   324          sine    = 2*LONG            ; sine is GSIN(x)
0000000C             02E0   325          cosine  = 3*LONG            ; cosine is GCOS(x)
                     02E0   326
```

```
                                02E0    328
                                02E0    329
                                02E0    330 ;
                                02E0    331 ; Return sine and cosine of argument
                                02E0    332 ;
                                02E0    333
                                02E0    334
                        00FC    02E0    335         .ENTRY  MTH$GSINCOS, ^M<R2, R3, R4, R5, R6,R7>
                                02E2    336
                                02E2    337         MTH$FLAG_JACKET
                                02E2
 6D    00000000'GF      9E      02E2                MOVAB   G^MTH$$JACKET_HND, (FP)
                                02E9                                        ; set handler address to jacket
                                02E9                                        ; handler
                                02E9
                                02E9    338
  50    04 BC  50FD             02E9    339         MOVG    @x(AP), R0
 0000036E'EF    16              02EE    340         JSB     MTH$GSINCOS_R7
  08 BC    50   7D              02F4    341         MOVQ    R0, @sine(AP)
  0C BC    52   7D              02F8    342         MOVQ    R2, @cosine(AP)
               04               02FC    343         RET
                                02FD    344
                                02FD    345
                                02FD    346
                                02FD    347         .SBTTL  MTH$GSIN
                                02FD    348
                                02FD    349 ;
                                02FD    350 ; Return sine of argument
                                02FD    351 ;
                                02FD    352
                                02FD    353
                        00FC    02FD    354         .ENTRY  MTH$GSIN, ^M<R2, R3, R4, R5, R6, R7>
                                02FF    355
                                02FF    356         MTH$FLAG_JACKET
                                02FF
 6D    00000000'GF      9E      02FF                MOVAB   G^MTH$$JACKET_HND, (FP)
                                0306                                        ; set handler address to jacket
                                0306                                        ; handler
                                0306
                                0306    357
  50    04 BC  50FD             0306    358         MOVG    @x(AP), R0
 0000046B'EF    16              030B    359         JSB     MTH$GSIN_R7
               04               0311    360         RET
                                0312    361
                                0312    362
                                0312    363
                                0312    364         .SBTTL  MTH$GCOS
                                0312    365
                                0312    366 ;
                                0312    367 ; Return cosine of argument
                                0312    368 ;
                                0312    369
                                0312    370
                        00FC    0312    371         .ENTRY  MTH$GCOS, ^M<R2, R3, R4, R5, R6, R7>
                                0314    372
                                0314    373         MTH$FLAG_JACKET
                                0314
```

```
                6D   00000000'GF    9E  0314                MOVAB   G^MTH$$JACKET_HND, (FP)
                                         031B                                        ; set handler address to jacket
                                         031B                                        ; handler
                                         031B
                                         031B   374
                     50    04 BC  50FD   031B   375          MOVG    @x(AP), R0
                000004FA'EF     16   0320   376          JSB     MTH$GCOS_R7
                                04   0326   377          RET
                                     0327   378
```

```
0327   380          .SBTTL  MTH$GSINCOSD    -       Degrees
0327   381
0327   382
0327   383 ;
0327   384 ; FUNCTIONAL DESCRIPTION:
0327   385 ;
0327   386 ; The GSIND, GCOSD and GSINCOSD routines are based on octant reduction.  Given
0327   387 ; an argument, x, it is written in the form
0327   388 ;
0327   389 ;               x = I1*360 + I*45 + Y1,
0327   390 ;
0327   391 ; where I1 and I are integers, 0 =< I < 8 and 0 =< Y1 < 45.  Since GSIND and
0327   392 ; GCOSD have a period of 360 it follows that
0327   393 ;
0327   394 ;               GSIND(x) = GSIND(I*45 + Y1) and
0327   395 ;               GCOSD(x) = GCOSD(I*45 + Y1).
0327   396 ;
0327   397 ; Using the trigonometric identities for the sum and difference of two angles,
0327   398 ; the following table can be generated:
0327   399 ;
0327   400 ;         If I =              then GSIND(x) =                 and GCOSD(x) =
0327   401 ;         ------              --------------                  -------------
0327   402 ;           0                   GSIND(Y1)                       GCOSD(Y1)
0327   403 ;           1                   GCOSD(45-Y1)                    GSIND(45-Y1)
0327   404 ;           2                   GCOSD(Y1)                      -GSIND(Y1)
0327   405 ;           3                   GSIND(45-Y1)                   -GCOSD(45-Y1)
0327   406 ;           4                  -GSIND(Y1)                      -GCOSD(Y1)
0327   407 ;           5                  -GCOSD(45-Y1)                   -GSIND(45-Y1)
0327   408 ;           6                  -GCOSD(Y1)                      GSIND(Y1)
0327   409 ;           7                  -GSIND(45-Y1)                   GCOS(45-Y1)
0327   410 ;
0327   411 ; Let Y be defined as Y = Y1 if I is even and Y = 45 - Y1, if I is odd, then
0327   412 ; each entry of the above table is of the for +/-GSIN(Y) or +/-GCOS(Y).  Based
0327   413 ; on the above remarks, the GSIND, GCOSD and GSINCOSD routines process the input
0327   414 ; argument x, to obtain I and Y, and based on I selects a suitable polynomial
0327   415 ; approximation, p(Y), to evaluate the desired fuction.
0327   416
0327   417
00000004 0327   418          LONG = 4
00000008 0327   419          sind = 2*LONG
0000000C 0327   420          cosd = 3*LONG
0327   421
```

```
                     00FC  0327   423          .ENTRY  MTH$GSINCOSD    ^M<R2, R3, R4, R5, R6, R7>
                           0329   424
                           0329   425          MTH$FLAG_JACKET
                           0329
6D   00000000'GF      9E   032?                MOVAB   G^MTH$$JACKET_HND, (FP)
                           0330                                        ; set handler address to jacket
                           0330                                        ; handler
                           0330
                           0330   426
     50    04 BC  50FD     0330   427          MOVG    @X(AP), R0
     00000584'EF    16     0335   428          JSB     MTH$GSINCOSD_R7
      08 BC    50   7D     033B   429          MOVQ    R0, @sind(AP)
      0C BC    52   7D     033F   430          MOVQ    R2, @cosd(AP)
                           0343   431
                     04    0343   432          RET
                           0344   433
                           0344   434
                           0344   435
                     00FC  0344   436          .ENTRY  MTH$GSIND       ^M<R2, R3, R4, R5, R6, R7>
                           0346   437
                           0346   438          MTH$FLAG_JACKET
                           0346
6D   00000000'GF      9E   0346                MOVAB   G^MTH$$JACKET_HND, (FP)
                           034D                                        ; set handler address to jacket
                           034D                                        ; handler
                           034D
                           034D   439
     50    04 BC  50FD     034D   440          MOVG    @X(AP), R0
     000005E2'EF    16     0352   441          JSB     MTH$GSIND_R7
                           0358   442
                     04    0358   443          RET
                           0359   444
                           0359   445
                           0359   446
                     00FC  0359   447          .ENTRY  MTH$GCOSD       ^M<R2, R3, R4, R5, R6, R7>
                           035B   448
                           035B   449          MTH$FLAG_JACKET
                           035B
6D   00000000'GF      9E   035B                MOVAB   G^MTH$$JACKET_HND, (FP)
                           0362                                        ; set handler address to jacket
                           0362                                        ; handler
                           0362
                           0362   450
     50    04 BC  50FD     0362   451          MOVG    @X(AP), R0
     00000648'EF    16     0367   452          JSB     MTH$GCOSD_R7
                           036D   453
                     04    036D   454          RET
                           036E   455
```

```
                     036E    457          .SBTTL MTH$GSINCOS_R7
                     036E    458
                     036E    459 ; This routine computes the GSIN and GCOS of the G-format value of R0/R1.  The
                     036E    460 ; computation is performed one of three ways depending on the size of the
                     036E    461 ; input argument, X:
                     036E    462 ;
                     036E    463 ;       1) If |X| < pi/4, then X is used directly in polynomial approximation
                     036E    464 ;          of GSIN and GCOS.
                     036E    465 ;       2) If pi/4 =< |x| < 9*pi/4, then the subroutine REDUCE_MEDIUM is called
                     036E    466 ;          to reduce the argument to an equivalent argument in radians, Y, and
                     036E    467 ;          the octant, I, containing the argument.  Y is then evaluated in two
                     036E    468 ;          polynomials chosen as a function of I, to compute GSIN(X) and GCOS(X).
                     036E    469 ;       3) If 9*pi/4 =< |X|, then the subroutine REDUCE_LARGE is called to
                     036E    470 ;          reduce the argument to an equivalent argument in cycles, Y, and the
                     036E    471 ;          octant, I, contanining the argument.  Y is then evaluated in two
                     036E    472 ;          polynomials chosen as a function of I, to compute GSIN(X) and GCOS(X).
                     036E    473
                     036E    474 MTH$GSINCOS_R7::
          56   50 50FD  036E    475          MOVG    R0, R6                  ; R6 = X
              10   18   0372    476          BGEQ    POS_SINCOS              ;
 0000037F'EF       16   0374    477          JSB     SINCOS                  ; R0/R1 = GSIN(|X|), R2/R3 = GCOS(X)
          50   50 52FD  037A    478          MNEGG   R0, R0                  ; R0/R1 = GSIN(X)
                   05   037E    479          RSB
                        037F    480
                        037F    481 SINCOS:
50        8000 8F    AA  037F    482          BICW    #^X8000, R0             ; R0/R1 = |X|
                        0384    483 POS_SINCOS:
50        FC7F CF 51FD  0384    484          CMPG    G_PI_OV_4, R0           ; Compare pi/4 with |X|
              31   14   038A    485          BGTR    SMALL_SINCOS            ; No argument reduction is necessary
50        FC7F CF 51FD  038C    486          CMPG    G_9_PI_OV_4, R0         ; Compare 9*pi/4 with |X|
                03   18   0392    487          BGEQ    1$
                 00AE    31   0394    488          BRW     LARGE_SINCOS            ; Use special logic for |X| > 9*pi/4
                        0397    489
                        0397    490 ;
                        0397    491 ;  pi/4 =< |X| < 9*pi/4
                        0397    492 ;
 00000685'EF       16   0397    493 1$:      JSB     REDUCE_MEDIUM           ; Medium argument reduction routine
                        039D    494                                          ; R4/R7 = Y = reduced argument
                        039D    495                                          ; R2 = octant
          7E   54 7DFD  039D    496          MOVQ    R4, -(SP)               ; Save reduced argument on stack
              52   DD   03A1    497          PUSHL   R2                      ; Save octant bits on stack
 00000518'EF       16   03A3    498          JSB     M_COS                   ; R0/R1 = GCOS(X)
          52   8E   D0  03A9    499          MOVL    (SP)+, R2               ; R2 = Octant bits
          54   8E 7DFD  03AC    500          MOVQ    (SP)+, R4               ; R4/R7 = reduced argument
          7E   50   7D  03B0    501          MOVQ    R0, -(SP)               ; Save GCOS(X) on stack
 00000496'EF       16   03B3    502          JSB     M_SIN                   ; R0/R1 = GSIN(X)
          52   8E   7D  03B9    503          MOVQ    (SP)+, R2               ; R2/R3 = GCOS(X)
                   05   03BC    504          RSB
                        03BD    505 ;
                        03BD    506 ; Logic for small arguments.  |X| < pi/4.
                        03BD    507 ;
                        03BD    508
                        03BD    509 SMALL_SINCOS:
50        4000 8F    B1  03BD    510          CMPW    #^X4000, R0             ; Compare 1/2 with |X|
              37   19   03C2    511          BLSS    2$                      ; Sufficent overghang not available
50        3E60 8F    B1  03C4    512          CMPW    #^X3E60, R0             ; Compare with 2**-27
              2B   18   03C9    513          BGEQ    1$                      ; No polynomial evaluation is needed
```

```
                56   50    7D  03CB   514          MOVQ    R0, R6                  ; R6/R7 = |X|
          54    56   56 45FD  03CE   515          MULG3   R6, R6, R4              ; R4/R5 = X*X
                7E   54    7D  03D3   516          MOVQ    R4, -(SP)               ; Put X*X on stack
     FCDB CF    07   54 55FD  03D6   517          POLYG   R4, #COSLENR1-1, COSTBR1; R0/R1 = GCOS(X)
                54   6E    7D  03DD   518          MOVQ    (SP), R4                ; R4/R5 = X*X
                6E   50    7D  03E0   519          MOVQ    R0, (SP)                ; Save GCOS(X) on stack
     FD4E CF    06   54 55FD  03E3   520          POLYG   R4, #SINLENR-1, SINTBR  ; R0/R1 = g(X^2)
                50   56 44FD  03EA   521          MULG2   R6, R0                  ; R0/R1 = X*g(X^2)
                50   56 40FD  03EE   522          ADDG2   R6, R0                  ; R0/R1 = GSIN(X)
                52   8E    7D  03F2   523          MOVQ    (SP)+, R2               ; R2/R3 = GCOS(X)
                           05  03F5   524          RSB
                               03F6   525
                52   08 50FD  03F6   526 1$:      MOVG    #1.0, R2                ; R0/R1 = X, R2/R3 = 1.0 = GCOS(X)
                           05  03FA   527          RSB
                               03FB   528
                               03FB   529
                7E   50    7D  03FB   530 2$:      MOVQ    R0, -(SP)               ; Save |x| on stack
                54   50 56FD  03FE   531          CVTGH   R0, R4                  ; R4/R7 = |X|
                54   54 64FD  0402   532          MULH2   R4, R4                  ; R4/R7 = X^2
                52   54 76FD  0406   533          CVTHG   R4, R2                  ; R2/R3 = X^2
                7E   52    7D  040A   534          MOVQ    R2, -(SP)               ; Save X^2 on stack
                7E   54    7D  040D   535          MOVQ    R4, -(SP)               ; Save high half of X^2 on stack
     FCE1 CF    07   52 55FD  0410   536          POLYG   R2, #COSLENR2-1, COSTBR2; R0/R1 = Q(Y^2)
                54   8E    7D  0417   537          MOVQ    (SP)+, R4               ; R4/R7 = Y^2
                54   B7    041A   538          DECW    R4                      ; R4/R7 = Y^2/2
                54   08 62FD  041C   539          SUBH2   #1, R4                  ; R4/R7 = Y^2/2 - 1
                50   50 56FD  0420   540          CVTGH   R0, R0                  ; R0/R3 = Q(Y^2)
                50   54 62FD  0424   541          SUBH2   R4, R0                  ; R0/R3 = GCOS(Y)
                50   50 76FD  0428   542          CVTHG   R0, R0                  ; R0/R1 = GCOS(Y)
                52   6E    7D  042C   543          MOVQ    (SP), R2                ; R2/R3 = X^2
                6E   50    7D  042F   544          MOVQ    R0, (SP)                ; Save GCOS(X) in R5
     FCFF CF    06   52 55FD  0432   545          POLYG   R2, #SINLENR-1, SINTBR  ; R0/R1 = g(X^2)
                52   8E    7D  0439   546          MOVQ    (SP)+, R2               ; R2/R3 = GCOS(X)
                50   6E 44FD  043C   547          MULG2   (SP), R0                ; R0/R1 = X*g(X^2)
                50   8E 40FD  0440   548          ADDG2   (SP)+, R0               ; R0/R1 = GSIN(X)
                           05  0444   549          RSB
                               0445   550
                               0445   551
                               0445   552 LARGE_SINCOS:
     00000700'EF      16      0445   553          JSB     REDUCE_LARGE            ; R4/R7 = reduced argument (in cycles)
                               044B   554                                          ; R2 = octant bits
                52    DD      044B   555          PUSHL   R2                      ; Save octant bits on stack
          7E   54 7DFD  044D   556          MOVQ    R4, -(SP)               ; Save reduced argument on stack
     00000558'EF      16      0451   557          JSB     L_COS                   ; R0/R1 = GCOS(X)
          54   8E 7DFD  0457   558          MOVQ    (SP), R4                ; Reduced argument in R3/R6
          52   8E    D0  045B   559          MOVL    (SP)+, R2               ; R2 = octant bits
          7E   50    7D  045E   560          MOVQ    R0, -(SP)               ; R2/R3 = GCOS(X)
     000004CE'EF      16      0461   561          JSB     L_SIN                   ; R0/R1 = GSIN(X)
          52   8E    7D  0467   562          MOVQ    (SP)+, R2               ; R2/R3 = GCOS(X)
                           05  046A   563          RSB
```

```
                        046B   565          .SBTTL MTH$GSIN_R7
                        046B   566
                        046B   567 ; This routine computes the GSIN of the G-format value of R0/R1.  The
                        046B   568 ; computation is performed one of three ways depending on the size of the
                        046B   569 ; input argument, X:
                        046B   570 ;
                        046B   571 ;       1) If |X| < pi/4, then X is used directly in a polynomial approximation
                        046B   572 ;          of GSIN.
                        046B   573 ;       2) If pi/4 =< |x| < 9*pi/4, then the subroutine REDUCE_MEDIUM is called
                        046B   574 ;          to reduce the argument to an equivalent argument in radians, Y, and
                        046B   575 ;          the octant, I, containing the argument.  Y is then evaluated in a
                        046B   576 ;          polynomial chosen as a function of I to compute GSIN(X).
                        046B   577 ;       3) If 9*pi/4 =< |X|, then the subroutine REDUCE_LARGE is called to
                        046B   578 ;          reduce the argument to an equivalent argument in cycles, Y, and the
                        046B   579 ;          octant, I, contanining the argument.  Y is then evaluated in a
                        046B   580 ;          polynomial chosen as a function of I to compute GSIN(X).
                        046B   581
                        046B   582 MTH$GSIN_R7::
              50 53FD   046B   583          TSTG    R0                      ; Check the sign of R0
              10   18   046E   584          BGEQ    POS_SIN                 ;
 0000047B'EF    16   0470   585          JSB     SIN                     ; R0/R1 = GSIN(|X|)
      50   50 52FD   0476   586          MNEGG   R0, R0                  ; R0/R1 = GSIN(X)
                05   047A   587          RSB
                        047B   588
                        047B   589 SIN:
50    8000 8F    AA   047B   590          BICW    #^X8000, R0             ; R0/R1 = |X|
                        0480   591 POS_SIN:
50    FB83 CF 51FD   0480   592          CMPG    G_PI_OV_4, R0           ; Compare pi/4 with |X|
              22   14   0486   593          BGTR    SMALL_SIN               ; No argument reduction is necessary
50    FB83 CF 51FD   0488   594          CMPG    G_9_PI_OV_4, R0         ; Compare 9*pi/4 with |X|
              38   19   048E   595          BLSS    LARGE_SIN               ; Use special logic for |X| > 9*pi/4
                        0490   596
                        0490   597 ;
                        0490   598 ;  pi/4 =< |X| < 9*pi/4
                        0490   599 ;
 00000685'EF    16   0490   600          JSB     REDUCE_MEDIUM           ; Medium argument reduction routine
                        0496   601                                          ; R4/R7 = Y = reduced argument
                        0496   602                                          ; R2 = octant
07    00    52    8F   0496   603 M_SIN:  CASEB   R2, #0, #7              ; Branch to one of four polynomial
                        049A   604                                          ;    evaluations depending on the
                0628'   049A   605 1$:      .WORD   P_SIN_R-1$              ;    octant bits.
                0596'   049C   606          .WORD   P_COS_R-1$
                0596'   049E   607          .WORD   P_COS_R-1$
                0623'   04A0   608          .WORD   N_SIN_R-1$
                0623'   04A2   609          .WORD   N_SIN_R-1$
                05DA'   04A4   610          .WORD   N_COS_R-1$
                05DA'   04A6   611          .WORD   N_COS_R-1$
                0628'   04A8   612          .WORD   P_SIN_R-1$
                        04AA   613
                        04AA   614 ;
                        04AA   615 ; Logic for small arguments.  |X| < pi/4.
                        04AA   616 ;
                        04AA   617
                        04AA   618 SMALL_SIN:
50    3E60 8F    B1   04AA   619          CMPW    #^X3E60, R0             ; Compare with 2^-27
              16   18   04AF   620          BGEQ    1$                      ; No polynomial evaluation is needed
      56   50    7D   04B1   621          MOVQ    R0, R6                  ; R6/R7 = X
```

```
                50    50 44FD   04B4   622          MULG2   R0,R0                          ; R0/R1 = X*X
        FC79 CF 06    50 55FD   04B8   623          POLYG   R0, #SINLENR-1, SINTBR         ; R0/R1 = q(x^2)
                50    56 44FD   04BF   624          MULG2   R6, R0                         ; R0/R1 = X*q(x^2)
                50    56 40FD   04C3   625          ADDG2   R6, R0                         ; R0/R1 = GSIN(x)
                      05        04C7   626 1$:       RSB
                                04C8   627
                                04C8   628
                                04C8   629 LARGE_SIN:
         00000700'EF  16        04C8   630          JSB     REDUCE_LARGE                   ; R4/R7 = reduced argument (in cycles)
                                04CE   631                                                 ; R2 = octant bits
                54    D5        04CE   632 L_SIN:   TSTL    R4                             ; Check for degenerate case
                14    13        04D0   633          BEQL    DEGENERATE_CASE_SIN
                                04D2   634
      07    00    52    8F      04D2   635          CASEB   R2, #0, #7
                                04D6   636
                   06AA'        04D6   637 1$:       .WORD   P_SIN_C-1$
                   0613'        04D8   638          .WORD   P_COS_C-1$
                   0613'        04DA   639          .WORD   P_COS_C-1$
                   06AA'        04DC   640          .WORD   P_SIN_C-1$
                   06A5'        04DE   641          .WORD   N_SIN_C-1$
                   0658'        04E0   642          .WORD   N_COS_C-1$
                   0658'        04E2   643          .WORD   N_COS_C-1$
                   06A5'        04E4   644          .WORD   N_SIN_C-1$
                                04E6   645
                                04E6   646
                                04E6   647 DEGENERATE_CASE_SIN:
                                04E6   648
                52    01    8A  04E6   649          BICB    #1, R2                         ; Compute index as (R2 - 1)/2
   52   52      FF 8F    9C    04E9   650          ROTL    #-1, R2, R2
      03    00    52    8F      04EE   651          CASEB   R2, #0, #3
                                04F2   652
                   07AC'        04F2   653 1$:       .WORD   P_ONE-1$
                   07B8'        04F4   654          .WORD   UNFL -1$
                   07B1'        04F6   655          .WORD   N_ONE-1$
                   07B8'        04F8   656          .WORD   UNFL -1$
                                04FA   657
```

```
                                04FA   659
                                04FA   660
                                04FA   661          .SBTTL MTH$GCOS_R7
                                04FA   662
                                04FA   663 ; This routine computes the GCOS of the G-format value of R0/R1.  The
                                04FA   664 ; computation is performed one of three ways depending on the size of the
                                04FA   665 ; input argument, X.  The processing is the same as described for MTH$GSIN_R4.
                                04FA   666 ;
                                04FA   667
                                04FA   668 MTH$GCOS_R7::
                          50 53FD   04FA   669          TSTG    R0                      ; Check for reserved operand
50      8000 8F     AA    04FD   670          BICW    #^X8000, R0             ; R0/R1 = |X|
50      FB01 CF   51FD    0502   671          CMPG    G_PI_OV_4, R0           ; Compare pi/4 with |X|
                  22  14   0508   672          BGTR    SMALL_COS               ; No argument reduction is necessary
50      FB01 CF   51FD    050A   673          CMPG    G_9_PI_OV_4, R0         ; Compare 9*pi/4 with |X|
                  40  19   0510   674          BLSS    LARGE_COS               ; Use special logic for |X| > 9*pi/4
                           0512   675
                           0512   676 ;
                           0512   677 ;  pi/4 =< |X| < 9*pi/4
                           0512   678 ;
 00000685'EF      16       0512   679          JSB     REDUCE_MEDIUM           ; Medium argument reduction routine
                           0518   680                                          ; R4/R7 = Y = reduced argument
                           0518   681                                          ; R2 = octant
07      00      52   8F    0518   682 M_COS:   CASEB   R2, #0, #7              ; Branch to one of four polynomial
                           051C   683                                          ;    evaluations depending on the
                    0514' 051C   684 1$:      .WORD   P_COS_R-1$              ;    octant bits.
                    05A1' 051E   685          .WORD   N_SIN_R-1$
                    05A1' 0520   686          .WORD   N_SIN_R-1$
                    0558' 0522   687          .WORD   N_COS_R-1$
                    0558' 0524   688          .WORD   N_COS_R-1$
                    05A6' 0526   689          .WORD   P_SIN_R-1$
                    05A6' 0528   690          .WORD   P_SIN_R-1$
                    0514' 052A   691          .WORD   P_COS_R-1$
                           052C   692
                           052C   693 ;
                           052C   694 ; Logic for small arguments.  |X| < pi/4.
                           052C   695 ;
                           052C   696
                           052C   697 SMALL_COS:
      50    4000 8F     B1    052C   698          CMPW    #^X4000, R0             ; Compare 1/2 with |X|
                  07  18   0531   699          BGEQ    1$                      ; Sufficent overghang is available
            54    50 56FD   0533   700          CVTGH   R0, R4                  ; R4/R7 = |X|
                0502  31   0537   701          BRW     NEEDS_DOUBLE            ; Use special logic to obtain overhang
      50    3E60 8F     B1    053A   702 1$:      CMPW    #^X3E60, R0             ; Compare with 2**-27
                  0C  18   053F   703          BGEQ    2$                      ; No polynomial evaluation is needed
            50    50 44FD   0541   704          MULG2   R0,R0                   ; R0/R1 = X*X
FB6C CF     07    50 55FD   0545   705          POLYG   R0, #COSLENR1-1, COSTBR1; R0/R1 = GCOS(X)
                     05     054C   706          RSB
                           054D   707
            50    08 50FD   054D   708 2$:      MOVG    #1.0, R0                ; R0/R1 = 1.0 = GCOS(X)
                     05     0551   709          RSB
                           0552   710
                           0552   711
                           0552   712 LARGE_COS:
 00000700'EF      16       0552   713          JSB     REDUCE_LARGE            ; R4/R7 = reduced argument (in cycles)
                           0558   714                                          ; R2 = octant bits
                54    D5   0558   715 L_COS:   TSTL    R4                      ; Check for degenerate case
```

```
                     14   13  055A   716          BEQL    DEGENERATE_CASE_COS
                              055C   717
          07   00    52   8F  055C   718          CASEB   R2, #0, #7
                         0589' 0560   719 1$:      .WORD   P_COS_C-1$
                         0620' 0562   720          .WORD   P_SIN_C-1$
                         061B' 0564   721          .WORD   N_SIN_C-1$
                         05CE' 0566   722          .WORD   N_COS_C-1$
                         05CE' 0568   723          .WORD   N_COS_C-1$
                         061B' 056A   724          .WORD   N_SIN_C-1$
                         0620' 056C   725          .WORD   P_SIN_C-1$
                         0589' 056E   726          .WORD   P_COS_C-1$
                               0570   727
                               0570   728
                               0570   729 DEGENERATE_CASE_COS:
                               0570   730
                     52   01   8A   0570   731          BICB    #1, R2                  ; Compute index as (R2 - 1)/2
          52   52    FF   8F   9C   0573   732          ROTL    #-1, R2, R2
             03   00    52   8F   0578   733          CASEB   R2, #0, #3
                               057C   734
                         072E' 057C   735 1$:      .WORD   UNFL -1$
                         0727' 057E   736          .WORD   N_ONE-1$
                         072E' 0580   737          .WORD   UNFL -1$
                         0722' 0582   738          .WORD   P_ONE-1$
                               0584   739
```

```
                     0584    741          .SBTTL  MTH$GSINCOSD_R7
                     0584    742
                     0584    743 ; This routine computes the GSIND and GCOSD of the G-format value of R0/R1.
                     0584    744 ; The computation is performed one of two ways depending on the size of the
                     0584    745 ; input argument, X:
                     0584    746 ;
                     0584    747 ;       1) If  |X| < 45, then X is used directly in polynomial approximation
                     0584    748 ;          of GSIND and GCOSD.
                     0584    749 ;       2) If 45 =<  |X|, then the subroutine REDUCE_DEGREES is called to reduce
                     0584    750 ;          the argument to an equivalent argument in degrees, Y, and the
                     0584    751 ;          octant, I, containing the argument.  Y is then evaluated in two
                     0584    752 ;          polynomials chosen as a function of I, to compute GSIND(X) and
                     0584    753 ;          GCOSD(X).
                     0584    754
                     0584    755 MTH$GSINCOSD_R7::
           50 53FD   0584    756          TSTG    R0
           10   18   0587    757          BGEQ    SINCOSD
50   8000 8F   AA   0589    758          BICW    #^X8000, R0                     ; R0/R1 = |X|
 00000599'EF   16   058E    759          JSB     SINCOSD                         ; R0/R1 = GSIND(|X|)
                     0594    760                                                  ; R2/R3 = GCOSD(|X|)
     50   50 52FD   0594    761          MNEGG   R0, R0                          ; R2/R3= -GSIND(|X|)
                05   0598    762          RSB
                     0599    763
                     0599    764 SINCOSD:
50   FA92 CF 51FD   0599    765          CMPG    G_45, R0                        ; Compare 45 to |X|
           24   14   059F    766          BGTR    SMALL_SINCOSD                   ; special processing for small arg
 00000974'EF   16   05A1    767          JSB     REDUCE_DEGREES                  ; R6/R7 = reduced argument
                     05A7    768                                                  ; R3 = octant
     7E   56   7D   05A7    769          MOVQ    R6, -(SP)                       ; Save reduced arg
          53   DD   05AA    770          PUSHL   R3                              ; Save octant bits
 0000065E'EF   16   05AC    771          JSB     EVAL_COSD                       ; R0/R1 = GCOSD(Y)
     53   8E   D0   05B2    772          MOVL    (SP)+, R3                       ; R3 = octant bits
     56   6E   7D   05B5    773          MOVQ    (SP), R6                        ; R6/R7 = reduced argument
     6E   50   7D   05B8    774          MOVQ    R0, (SP)                        ; Save GCOSD(Y)
00000605'EF   16   05BB    775          JSB     EVAL_SIND                       ; R0/R1 = GSIND(Y)
     52   8E   7D   05C1    776          MOVQ    (SP)+, R2                       ; R2/R3 = GCOSD(Y)
                05   05C4    777          RSB
                     05C5    778
                     05C5    779
                     05C5    780 SMALL_SINCOSD:
     5E   10   C2   05C5    781          SUBL    #16, SP                         ; Allocate 4 longwords on stack
     6E   50   7D   05C8    782          MOVQ    R0, (SP)                        ; Save argument
00000672'EF   16   05CB    783          JSB     SMALL_COSD                      ; R0/R1 = GCOSD(.X|)
 08 AE   50   7D   05D1    784          MOVQ    R0, 8(SP)                       ; Save GCOSD(|X|)
     50   8E   7D   05D5    785          MOVQ    (SP)+, R0                       ; R0/R1 = argument
00000619'EF   16   05D8    786          JSB     SMALL_SIND                      ; R0/R1 = GSIND(X)
     52   8E   7D   05DE    787          MOVQ    (SP)+, R2                       ; R2/R3 = GCOSD(|X|)
                05   05E1    788          RSB
```

```
                        05E2    790         .SBTTL  MTH$GSIND_R7
                        05E2    791
                        05E2    792 ; This routine computes the GSIND of the G-format value of R0/R1.  The
                        05E2    793 ; computation is performed one of two ways depending on the size of the input
                        05E2    794 ; argument, X:
                        05E2    795 ;
                        05E2    796 ;       1) If |X| < 45, then X is used directly in polynomial approximation
                        05E2    797 ;          of GSIND.
                        05E2    798 ;       2) If 45 =< |x|, then the subroutine REDUCE_DEGREES is called to reduce
                        05E2    799 ;          the argument to an equivalent argument in degrees, Y, and the
                        05E2    800 ;          octant, I, containing the argument.  Y is then evaluated in two
                        05E2    801 ;          polynomials chosen as a function of I, to compute GSIND(X).
                        05E2    802
                        05E2    803 MTH$GSIND_R7::
                50 53FD 05E2    804         TSTG    R0                      ; R0/R1 = X
                10   18 05E5    805         BGEQ    POS_SIND                ;
 000005F2'EF    16 05E7    806         JSB     NEG_SIND                ; R0/R1 = GSIND(|X|)
     50 50 52FD 05ED    807         MNEGG   R0, R0                  ; R0/R1 = -GSIND(|X|)
                   05 05F1    808         RSB
                        05F2    809
                        05F2    810 NEG_SIND:
50   8000 8F   AA 05F2    811         BICW    #^X8000, R0             ; R0/R1 = |X|
                        05F7    812 POS_SIND:
50   FA34 CF 51FD 05F7    813         CMPG    G_45, R0                ; Compare 45 to |X|
                1A   14 05FD    814         BGTR    SMALL_SIND              ; special processing for small arg
 00000974'EF    16 05FF    815         JSB     REDUCE_DEGREES          ; R6/R7 = reduced argument
                        0605    816                                 ; R3 = octant
                        0605    817
                        0605    818 EVAL_SIND:
07   00   53   8F 0605    819         CASEB   R3, #0, #7
                0679' 0609    820 1$:     .WORD   P_SIN_D-1$
                05A6' 060B    821         .WORD   P_COS_D-1$
                05A6' 060D    822         .WORD   P_COS_D-1$
                0679' 060F    823         .WORD   P_SIN_D-1$
                0675' 0611    824         .WORD   N_SIN_D-1$
                060A' 0613    825         .WORD   N_COS_D-1$
                060A' 0615    826         .WORD   N_COS_D-1$
                0675' 0617    827         .WORD   N_SIN_D-1$
                        0619    828
                        0619    829
                        0619    830 SMALL_SIND:
     50 FA22 CF 51FD 0619    831         CMPG    G_SMALLD, R0            ; Compare 180/pi*2^-27 with |x|
                06   14 061F    832         BGTR    1$                      ; No polynomial evaluation is
     56   50   7D 0621    833         MOVQ    R0, R6                  ;   necessary
        065B   31 0624    834         BRW     P_SIN_D                 ;
                50 53FD 0627    835 1$:     TSTG    R0                      ; Check for zero
                1B   13 062A    836         BEQL    3$                      ; Return if R0 = 0
     50 FA2F CF 51FD 062C    837         CMPG    G_SMALLEST_DEG, R0      ; Check for possible underflow on
                03   15 0632    838         BLEQ    2$                      ;   conversion to radians
        0673   31 0634    839         BRW     UNFL                    ; Underflow will occur on conversion
52   50 FA14 CF 45FD 0637    840 2$:     MULG3   G_CONVERT, R0, R2       ; R2/R3 = (pi/180 - 2^-6)*|x|
50   0060 8F   A2 063E    841         SUBW    #^X60, R0               ; R0/R1 = |X|*2^-6
     50   52 40FD 0643    842         ADDG2   R2, R0                  ; R0/R1 = GSIND(|X|) = (pi/180)|X|
                   05 0647    843 3$:     RSB
```

```
                     0648   845          .SBTTL  MTH$GCOSD_R7
                     0648   846
                     0648   847 ; This routine computes the GCOSD of the G-format value of R0.  The computation
                     0648   848 ; is performed one of two ways depending on the size of the input argument, X:
                     0648   849 ; Details are given in the discussion on MTH$GCOSD_R4.
                     0648   850
                     0648   851 MTH$GCOSD_R7::
            50 53FD  0648   852          TSTG    R0                      ; Check for reserved operand
50   8000 8F   AA    064B   853          BICW    #^X8000, R0             ; R0/R1 = |X|
50   F9DB CF 51FD    0650   854          CMPG    G_45, R0                ; Compare 45 to |X|
         1A   14     0656   855          BGTR    SMALL_COSD              ;
 00000974'EF   16    0658   856          JSB     REDUCE_DEGREES          ; R6/R7 = reduced argument
                     065E   857                                          ; R3 = octant
                     065E   858
                     065E   859 EVAL_COSD:
07   00   53   8F    065E   860          CASEB   R3, #0, #7
                054D' 0662   861 1$:      .WORD   P_COS_D-1$
                0620' 0664   862          .WORD   P_SIN_D-1$
                061C' 0666   863          .WORD   N_SIN_D-1$
                05B1' 0668   864          .WORD   N_COS_D-1$
                05B1' 066A   865          .WORD   N_COS_D-1$
                061C' 066C   866          .WORD   N_SIN_D-1$
                0620' 066E   867          .WORD   P_SIN_D-1$
                054D' 0670   868          .WORD   P_COS_D-1$
                     0672   869
                     0672   870
                     0672   871 SMALL_COSD:
50   F9C9 CF 51FD    0672   872          CMPG    G_SMALLD, R0            ; Compare 180/pi*2^-27  with |X|
         06   14     0678   873          BGTR    1$                      ; Check if polyinomial evaluation is
     56   50   7D     067A   874          MOVQ    R0, R6                  ;    necessary.
        052F   31     067D   875          BRW     P_COS_D                 ; POLY needed
     50   08 50FD    0680   876 1$:      MOVG    #1, R0                  ; R0 = 1. = GCOSD(|X|)
                05   0684   877          RSB
                     0685   878
```

```
                                 06B5   880
                                 06B5   881
                                 06B5   882          .SBTTL  REDUCE_MEDIUM
                                 06B5   883
                                 06B5   884 ;
                                 06B5   885 ; This routine assumes that the absolute value of the argument, X,  is in R0/R1
                                 06B5   886 ; and that pi/4 =< !X! < 9*pi/4.  It returns an h- format reduced argument, Y,
                                 06B5   887 ; in R4/R7, and the octant bits in R2.
                                 06B5   888 ;
                                 06B5   889 ; The reduced argument is obtained by locating the octant that X is in through
                                 06B5   890 ; a binary search and then subtracting off a suitable multiple of pi/2
                                 06B5   891 ;
                                 06B5   892
                                 06B5   893 REDUCE_MEDIUM:
     50   F996 CF 51FD           06B5   894          CMPG    G_5_PI_OV_4, R0 ;
                  32   15        06BB   895          BLEQ    5$                      ; !X! >= 5*pi/4
     50   F986 CF 51FD           06BD   896          CMPG    G_3_PI_OV_4, R0
                  15   15        06C3   897          BLEQ    3$                      ; !X! >= 3*pi/4
          50   50 56FD           06C5   898          CVTGH   R0, R0
54   50   F9CA CF 63FD           06C9   899          SUBH3   H_PI_OV_2, R0, R4       ; R4/R7 = !X! - pi/2
                  04   18        06D0   900          BGEQ    2$                      ;
          52   01   D0           06D2   901          MOVL    #1, R2                  ; Octant bits = 001
                    05           06D5   902          RSB
                                 06D6   903
          52   02   D0           06D6   904 2$:      MOVL    #2, R2                  ; Octant bits = 010
                    05           06D9   905          RSB
                                 06DA   906
          50   50 56FD           06DA   907 3$:      CVTGH   R0, R0
54   50   F9C5 CF 63FD           06DE   908          SUBH3   H_PI, R0, R4            ; R4/R7 = !X! - pi
                  04   18        06E5   909          BGEQ    4$                      ;
          52   03   D0           06E7   910          MOVL    #3, R2                  ; Octant bits = 011
                    05           06EA   911          RSB
                                 06EB   912
          52   04   D0           06EB   913 4$:      MOVL    #4, R2                  ; Octant bits = 100
                    05           06EE   914          RSB
                                 06EF   915
     50   F964 CF 51FD           06EF   916 5$:      CMPG    G_7_PI_OV_4, R0 ;
                  15   15        06F5   917          BLEQ    7$                      ; !X! >= 7*pi/4
          50   50 56FD           06F7   918          CVTGH   R0, R0
54   50   F9B8 CF 63FD           06FB   919          SUBH3   H_3_PI_OV_2, R0, R4     ; R4/R7 = !X! - 3*pi/2
                  04   18        0702   920          BGEQ    6$                      ;
          52   05   D0           0704   921          MOVL    #5, R2                  ; Octant bits = 101
                    05           0707   922          RSB
                                 0708   923
          52   06   D0           0708   924 6$:      MOVL    #6, R2                  ; Octant bits = 110
                    05           070B   925          RSB
                                 070C   926
          50   50 56FD           070C   927 7$:      CVTGH   R0, R0
54   50   F9B3 CF 63FD           0710   928          SUBH3   H_2_PI, R0, R4          ; R4/R7 = !X! - 2*pi
                  04   18        0717   929          BGEQ    8$                      ;
          52   07   D0           0719   930          MOVL    #7, R2                  ; Octant bits = 111
                    05           071C   931          RSB
                                 071D   932
               52   D4           071D   933 8$:      CLRL    R2                      ; Octant bits = 000
                    05           071F   934          RSB
                                 0720   935
                                 0720   936
```

```
06F0    937
06F0    938
06F0    939
```

```
                                06F0   941          .SBTTL  REDUCE_LARGE
                                06F0   942
                                06F0   943 ;
                                06F0   944 ; This routine is used to reduce large arguments (|X| >= 9*pi/4) modulo pi/4.
                                06F0   945 ; It returns the reduced argument, Y, in R4/R5 in units of cycles, and returns
                                06F0   946 ; the octant bits, I, in R2.
                                06F0   947 ;
                                06F0   948 ; The method of reduction is as follows:
                                06F0   949 ;
                                06F0   950 ;     x*(4/pi) = 2^n*f*(4/pi) where n is an integer and 1/2 =< f < 1
                                06F0   951 ;              = 2^(n-53)*(2^53*f)*(4/pi)
                                06F0   952 ;              = (2^53*f)*(2^(n-53)*4/pi)
                                06F0   953 ;              = K*C, where K = 2^53*f is an integer and C = 2*(n-53)*4/pi
                                06F0   954 ;     Let L = K*C modulo 8, where 0 =< L < 8, and let I = the integer(L) and
                                06F0   955 ;     h = fract(L), then if I is even Y = h, otherwise Y = 1-h
                                06F0   956 ;
                                06F0   957 ; CONSTANTS:
                                06F0   958
                     0000003D   06F0   959          L_INT_WEIGHT  = ^X3D            ; weights exponent by 61
                     00000020   06F0   960          W_TERM_WEIGHT = ^X20            ; weights exponent by 32
                     00000400   06F0   961          W_MAX_WEIGHT  = ^X400           ; maximum unbiased exponent
                     000003B6   06F0   962          W_ADJUST      = ^X3B6           ; Used to locate binary point in
                                06F0   963                                         ;   MTH$AL_4_OV_PI_V table
                                06F0   964 H_2_TO_32:
00000000 00000000 00000000 00004021 06F0   965          .OCTA   ^X4021            ; 2^32
                                0700   966
                                0700   967
                                0700   968
                                0700   969 REDUCE_LARGE:
                                0700   970 ;
                                0700   971 ; The first step is to obtain the location of the binary point in the represen-
                                0700   972 ; tation of C = 2^(n-53)*(4/pi) in two parts - the number of longwords from
                                0700   973 ; the start and the number of bits from the most significant bit of the next
                                0700   974 ; longword. Also K = 2^53*f must be obtained.
                                0700   975 ;
 53    50    FC 8F    9C       0700   976          ROTL    #-4, R0, R3            ; Shift exponext field 4 bits right
    53    03B6 8F    A2       0705   977          SUBW    #W_ADJUST, R3          ; Unbias exp and adjust for leading
                                070A   978                                         ;   zeroes.  R3 = location of binary
                                070A   979                                         ;   point
 54    53    FD 8F    9C       070A   980          ROTL    #-3, R3, R4            ; Divide R3 by 32 and mull by 4 to get
54    FFFFFF03 8F    CA       070F   981          BICL    #^XFFFFFF03, R4        ;   R4 = # of longwords (in bytes) to
                                0716   982                                         ;   binary point.
                                0716   983
52    00000000'GF    DE       0716   984          MOVAL   G^MTH$AL_4_OV_PI_V, R2 ; R2 = address of RTL vector entry
52    00000000'GF    C0       071D   985          ADDL    G^MTH$AL_4_OV_PI_V, R2 ; R2 = address of MTH$AL_4_OV_PI table
      52    54       C2       0724   986          SUBL    R4, R2                 ; R2 points to 1st quadword of interest
      53    E0 8F    8A       0727   987          BICB    #^XE0, R3              ; R3(7:0) = # of bits within longword
                                072B   988
    50    7FF0 8F    AA       072B   989          BICW    #^X7FF0, R0            ; Clear exponent field
    50    4150 8F    A8       0730   990          BISW    #^X4150, R0            ; R0 = 2^21*f
          50    50 4AFD       0735   991          CVTGL   R0, R0                 ; R0 = High 21 bits of K
    51    51    10    9C       0739   992          ROTL    #16, R1, R1            ; R1 = Low 32 bits of K
                02    18       073D   993          BGEQ    1$                     ; Check for high bit of R1 set
                50    D6       073F   994          INCL    R0                     ; Adjust R0 if R1 is negative
                                0741   995
                                0741   996 ;
                                0741   997 ; The next step is to generate an approximation to C, call it C'' to be used
```

```
                                0741   998 ; in computing x*4/pi.  C'' will consist of the first three integer bits of
                                0741   999 ; C and the first 61 fraction bits of C.  These bits will be obtained from a
                                0741  1000 ; constant stored in the interger array MTH$AL_4_OV_PI_V.
                                0741  1001 ;
                                0741  1002 ; NOTE:  The ASHQ, ADDL, and MULL instructions in the follow sections may
                                0741  1003 ; result in an integer overflow trap.  The overflow incurred is intentional,
                                0741  1004 ; so that the IV bit must be turned off.  The IV bit is not restored until
                                0741  1005 ; after all of the necessary fraction bits hav been generated.
                                0741  1006 ;
                          7E DC 0741  1007 1$:     MOVPSL  -(SP)                   ; Put current PSL on stack
                6E FFFFFFDF 8F CA 0743  1008          BICL    #^C<PSL$M_IV>, (SP)     ; (SP) = current IV bit
                          20 B9 074A  1009          BICPSW  #PSL$M_IV               ; Clear integer overflow bit
                                074C  1010
                                074C  1011 ; The necessary calculation to produce the reduced argumen' can require up to
                                074C  1012 ; nine longwords of temporary work space.  This work space will be allocated
                                074C  1013 ; on the stack.  The work space will be accessed though the use of three
                                074C  1014 ; registers: R4, R5, and SP.  For the purposes of comments the temporary work
                                074C  1015 ; space will be referred to as locations T0 though T8.  The stack and its
                                074C  1016 ; pointers will look something like this:
                                074C  1017 ;
                                074C  1018 ;                          |                    |
                                074C  1019 ;                          |--------------------|
                                074C  1020 ;                     T0:  |                    |  <---- SP
                                074C  1021 ;                          |--------------------|
                                074C  1022 ;                     T1:  |                    |
                                074C  1023 ;                          |--------------------|
                                074C  1024 ;                     T2:  |                    |
                                074C  1025 ;                          |--------------------|
                                074C  1026 ;                     T3:  |                    |
                                074C  1027 ;                          |--------------------|
                                074C  1028 ;                     T4:  |                    |  <---- R4
                                074C  1029 ;                          |--------------------|
                                074C  1030 ;                     T5:  |                    |
                                074C  1031 ;                          |--------------------|
                                074C  1032 ;                     T6:  |                    |
                                074C  1033 ;                          |--------------------|
                                074C  1034 ;                     T7:  |                    |
                                074C  1035 ;                          |--------------------|
                                074C  1036 ;                     T8:  |                    |  <---- R5
                                074C  1037 ;                          |--------------------|
                                074C  1038 ;                          |                    |
                                074C  1039 ;
                                074C  1040 ; The following code allocates the storage and sets up the pointers.
                                074C  1041
                          5E 24 C2 074C  1042          SUBL    #36, SP                 ; Allocate 9 longwords on the stack
                54        5E 10 C1 074F  1043          ADDL3   #16, SP, R4             ; R4 points to T4
                55        5E 20 C1 0753  1044          ADDL3   #32, SP, R5             ; R5 points to T8
                                0757  1045
                                0757  1046 ;
                                0757  1047 ; Get C'' = C(0):C(1):C(2):C(3) in T5/T8.  C(0) though  C(3) are unsigned
                                0757  1048 ; integers generated from the binary representation of C.  The high three bits
                                0757  1049 ; of C(0) are the first three bits to the left of the binary point of C.
                                0757  1050 ; The remaining bits C(0) and C(1) though C(3) are the first 125 bits to the
                                0757  1051 ; right of the binary point of C. Note that the C(i)'s are adjusted to
                                0757  1052 ; compensate for their signed (rather than unsigned) interpretation in the EMUL
                                0757  1053 ; instruction.  Note also that the representation of C has no more than 15
                                0757  1054 ; consequtive ones, so that no carry is possible from the adjustment.
```

```
                                0757  1055 ;
                                0757  1056
                     57  54  0C  C1  0757  1057          ADDL3   #12, R4, R7              ; Initailize loop counter.  R7 points
                                0758  1058                                               ;   to T7
                     67  62  53  79  0758  1059          ASHQ    R3, (R2), (R7)          ; Shift the proper quadword so that
                                075F  1060                                               ;   T8 has C(0)in it
                         57  04  C2  075F  1061          SUBL    #4, R7                  ; R7 points to T6
                         52  04  C2  0762  1062 2$:      SUBL    #4, R2                  ; R2 points to next quadword in
                                0765  1063                                               ;   MTH$AL_4_OV_PI_V table
                     67  62  53  79  0765  1064          ASHQ    R3, (R2), (R7)          ; Shift quadword so that C(n) is in
                                0769  1065                                               ;   T(8-n) , n = 0,1,2,3
                             03  18  0769  1066          BGEQ    3$                      ; Check for high bit of C(n) set
                          08 A7  D6  076B  1067          INCL    8(R7)                   ; Bit set.  Adjust C(n-1)
FFEA 57   FFFFFFFC 8F   54  F1  076E  1068 3$:      ACBL    R4, #-4, R7, 2$         ; Loop until C(0) though C(3) are in
                                0778  1069                                               ;   T5 though T8
                                0778  1070
                                0778  1071 ;
                                0778  1072 ; Generate the low 128 bits of the product K*C'' = L.  This product is
                                0778  1073 ; equivalent to multiplying K times C'' modulo 8.  The result of the
                                0778  1074 ; product is in T4/T7 with bits 31:29 of T4 the octant bits, and the remaining
                                0778  1075 ; 125 bits the faction bits of the product.  The last 53 fraction bits (bits
                                0778  1076 ; 20:0 of T6 and 31:0 of T5) are non-valid fraction bits that will be used
                                0778  1077 ; later if more fraction bits need to be generated.
                                0778  1078 ;
                                0778  1079 ;
                                0778  1080 ; Multiply the high order bits of K (R0) times C'' and store the result in
                                0778  1081 ; T0/T2.
                                0778  1082
      6E   00   04 A4   50  7A  0778  1083          EMUL    R0, 4(R4), #0, (SP)     ; T0/T1 = KHI*C(3)
04 AE   04 AE   08 A4   50  7A  077E  1084          EMUL    R0, 8(R4), 4(SP), 4(SP) ; T0/T2 = KHI*[C(2):C(3)]
           64   0C A4   50  C5  0786  1085          MULL3   R0, 12(R4), (R4)        ; T4 = Low 32 bits of KHI*C(1)
                08 AE   64  C0  078B  1086          ADDL    (R4), 8(SP)             ; T0/T2 = KHI*C'' modulo 8
                                078F  1087
                                078F  1088 ; Multiply the low order bits of K (R1) times C'' and store the result in
                                078F  1089 ; T4/T8.
                                078F  1090
      64   00   04 A4   51  7A  078F  1091          EMUL    R1, 4(R4), #0, (R4)     ; T4/T5 = KLO*C(3)
04 A4   04 A4   08 A4   51  7A  0795  1092          EMUL    R1, 8(R4), 4(R4), 4(R4) ; T4/T6 = KLO*[C(2):C(3)]
08 A4   08 A4   0C A4   51  7A  079D  1093          EMUL    R1, 12(R4), 8(R4), 8(R4); T4/T7 = KLO*[C(1):C(2):C(3)]
                    65  51  C4  07A5  1094          MULL    R1, (R5)                ; T8 = KLO*C(0)
                0C A4   65  C0  07A8  1095          ADDL    (R5), 12(R4)            ; T4/T7 = KLO*C'' modulo 8
                                07AC  1096
                                07AC  1097 ; Add KHI*C'' to KLO*C'' to get K*C''.  Store the result in T4/T7.
                                07AC  1098
                04 A4   6E  C0  07AC  1099          ADDL    (SP), 4(R4)             ;
             08 A4   04 AE  D8  07B0  1100          ADWC    4(SP), 8(R4)            ;
             0C A4   08 AE  D8  07B5  1101          ADWC    8(SP), 12(R4)           ; T4/T7 = K*C'' modulo 8
                                07BA  1102
                                07BA  1103 ;
                                07BA  1104 ; At this point there may or may not be enough valid bits in R3/R4 to generate
                                07BA  1105 ; Y.  If the first 12 fratction bits are all 1's or 0's, there a possibility of
                                07BA  1106 ; loss of significance when computing Y.  Consequently, we must check for loss
                                07BA  1107 ; of significance before converting T4/T7 to Y and I.
                                07BA  1108 ;
                                07BA  1109
  65   FC A5   00008000 8F  C1  07BA  1110          ADDL3   #^X8000, -4(R5), (R5)   ; If the first 14 fraction bits are 1's
          65   3FFF0000 8F  D3  07C3  1111          BITL    #^X3FFF0000, (R5)       ;   and the reduced arg = 1-f or the
```

```
                     37   12  07CA  1112          BNEQ    CONVERT                 ;   first 14 bits are 0 and the reduced
                                 07CC  1113                                          ;   arg = f, then (and only then) bits
                                 07CC  1114                                          ;   29:16 are 0 and significance will
                                 07CC  1115                                          ;   be lost.
                                 07CC  1116
                                 07CC  1117 ;
                                 07CC  1118 ; More bits need to be generated to cover the loss of significance.  There are
                                 07CC  1119 ; not enough registers to hold all the potential extra bits, so that the bits
                                 07CC  1120 ; already generated must be put on the stack.
                                 07CC  1121 ;
                                 07CC  1122
          000008CA'EF    16      07CC  1123          JSB     GEN_MORE_BITS           ; Generate 85 additional bits and add
                                 07D2  1124                                          ;   them to existing bits.  Results are
                                 07D2  1125                                          ;   stored in T3/T7
               54   04   C2      07D2  1126          SUBL    #4, R4                  ; Adjust R4 to reflect the addition of
                                 07D5  1127                                          ;   another longword of K*C''
        15 FC A5    1D   E0      07D5  1128          BBS     #29, -4(R5), 4$         ; Check if loss of significance is due
                                 07DA  1129                                          ;   to leading ones or zeros
                                 07DA  1130
                                 07DA  1131 ; Lost significance due to leading zeros
                                 07DA  1132
    65   10 A4   0F   00   EA    07DA  1133          FFS     #0, #15, 16(R4), (R5)   ; If at least one bit is set.  This
                     21   12      07E0  1134          BNEQ    CONVERT                 ;   means lost significance was minor.
     0C A4   0EFFFFFF 8F   D1      07E2  1135          CMPL    #^XEFFFFFF, 12(R4)      ; If one of the five high bits is set,
                     17   15      07EA  1136          BLEQ    CONVERT                 ;   lost significance was minor.
                   009A   31      07EC  1137          BRW     LEADING_ZEROS
                                 07EF  1138
                                 07EF  1139 ; Lost significance due to leading ones
                                 07EF  1140
    65   10 A4   0F   00   EB    07EF  1141 4$:      FFC     #0, #15, 16(R4), (R5)   ; If at least one bit is clear. This
                     0C   12      07F5  1142          BNEQ    CONVERT                 ;   meanslost significance was minor.
     0C A4   F8000000 8F   D1      07F7  1143          CMPL    #^XF8000000, 12(R4)     ; If one of the five high bits is
                     02   1E      07FF  1144          BGEQU   CONVERT                 ;   clear, lost significance was minor.
                     35   11      0801  1145          BRB     LEADING_ONES
                                 0803  1146
                                 0803  1147 ;
                                 0803  1148 ; Convert bit string to double precion reduced argument.
                                 0803  1149 ;
                                 0803  1150
                                 0803  1151 CONVERT:
                                 0803  1152
                                 0803  1153 ;    Isolate octant bits and convert low order bits to H-format.
                                 0803  1154
    65   FC A5   03   1D   EF    0803  1155          EXTZV   #29, #3, -4(R5), (R5)   ; T8 = octant bits
     FC A5   E0000000 8F   CA    0809  1156          BICL    #^XE0000000, -4(R5)     ; Clear octant bits
           54   55   0C   C3      0811  1157          SUBL3   #12, R5, R4             ; R4 points to low order bits of h
            00000910'EF   16      0815  1158          JSB     CVT_LONG_TO_H           ; R0/R3 = 2^29*h
                 50   1D   A2      081B  1159          SUBW    #^XTD, R0               ; R0/R3 = h
                   0C 65   E9      081E  1160          BLBC    (R5), 1$                ; Check for odd or even octant bits
                                 0821  1161
                                 0821  1162 ;    Octant bits are odd.  Reduced argument equals 1 - h.
                                 0821  1163
           54   08   50 63FD      0821  1164          SUBH3   R0, #1, R4              ; R4/R7 = Y = 1 - h
             52   20 AE   D0      0826  1165          MOVL    32(SP), R2              ; R2 = octant bits
                   0096   31      082A  1166          BRW     RESTORE
                                 082D  1167
                                 082D  1168 ;    Octant bits are even.  Reduced argument equals h
```

```
                                082D  1169
                     54    50 7DFD  082D  1170 1$:     MOVO    R0, R4                  ; R4/R7 = Y
                 52    20 AE   D0  0831  1171         MOVL    32(SP), R2              ; R2 = octant bits
                       008B   31  0835  1172         BRW     RESTORE
                                0838  1173
                                0838  1174
                                0838  1175
                                0838  1176 ;
                                0838  1177 ; At this point it has been determined that there is a major loss of
                                0838  1178 ; significance and the processing begins a looping phase.  Each iteration of
                                0838  1179 ; the loop will generate additional extra bits of K*C' until enough significant
                                0838  1180 ; bits to compute Y are available.  During this time the nine longwords
                                0838  1181 ; allocated on the stack will be used as follows:
                                0838  1182 ;
                                0838  1183 ;       T0/T2   Temporary storage used when generating extra bits.
                                0838  1184 ;
                                0838  1185 ;       T3/T7   Contains all significant bits generated so far.
                                0838  1186 ;
                                0838  1187 ;       T8      Contains a counter, W, indicating the appropriate exponent
                                0838  1188 ;                 of the last longword of fraction bits used in converting
                                0838  1189 ;                 to Y.
                                0838  1190
                                0838  1191 LEADING_ONES:
                                0838  1192
                                0838  1193 ; If processing continues here it is known that the loss of significance is due
                                0838  1194 ; to a string of leading ones.
                                0838  1195
                          65    3D   D0  0838  1196         MOVL    #L_INT_WEIGHT, (R5)     ; T8 = exp bias for last longword
                                083B  1197                                                ;    of the product K*C'
                                083B  1198
          0C A4    FFFF0000 8F   D1  083B  1199 LOOP_1: CMPL    #^XFFFF0000, 12(R4)     ; Check for enough significant bits
                             28   1A  0843  1200         BGTRU   CONVERT_1               ; Enough bits.  Convert to floating.
                   000008CA'EF   16  0845  1201         JSB     GEN_MORE_BITS           ; T2/T7 contains K*C''
          0C A4    FFFFFFFF 8F   D1  084B  1202         CMPL    #-1, 12(R4)             ; Check for all 1's
                             18   1A  0853  1203         BGTRU   CONVERT_1               ; Not all 1's.  Enough precision bits
                                0855  1204                                                ;    to compute Y
                       64    FC A4 7DFD  0855  1205         MOVO    -4(R4), (R4)            ; Compress representation of K*C''
      FFD9 65    20    0400 8F   3D  085A  1206         ACBW    #W_MAX_WEIGHT, #W_TERM_WEIGHT, (R5), LOOP_1
                                0862  1207                                                ; Increment weighting factor.  If
                                0862  1208                                                ;   weighting factor is greater than
                                0862  1209                                                ;   1024 then no more bits need to be
                                0862  1210                                                ;   generated.
                                0862  1211
                                0862  1212 ;
                                0862  1213 ; The weighting factor is greater than 1024.  This means that the reduced
                                0862  1214 ; argument is either not distinguishable from 1 or too small to be represented
                                0862  1215 ; in F-format (i.e. underflow.)  Zero is returned in R4 for the reduced
                                0862  1216 ; argument to signal this occurance.  Note that under these conditions the
                                0862  1217 ; correct function value is one of the values 0, +/-1.  The
                                0862  1218 ; correct choice is determined by the calling program based on the octant bits
                                0862  1219 ; returned in R1.
                                0862  1220 ;
                             53   D4  0862  1221         CLRL    R3                      ; Reduced argument is zero
                52    08 AE    03   1D  0864  1222         EXTZV   #29, #3, 8(SP), R2      ; R2 = octant bits
                             0056   31  086A  1223         BRW     RESTORE
                                086D  1224
                                086D  1225
```

```
                                086D  1226 CONVERT_1:
                          54    04   C0  086D  1227          ADDL    #4, R4                  ; R4 points to low bits of h
                00000910'EF     16   08  0870  1228          JSB     CVT_LONG_TO_H           ; R0/R3 = 2^W*h
                          54    50 63FD  0876  1229          SUBH3   R0, H_2_TO_32, R4       ; R4/R7 = 2^W*(1 - h)
          52  1C AE    03     1D   EF  087D  1230          EXTZV   #29, #3, 28(SP), R2     ; R2 = octant bits
                       54  20 AE   C2  0883  1231          SUBL    32(SP), R4              ; R4/R7 = Y = 1 - h
                           3A   11     0887  1232          BRB     RESTORE
                                       0889  1233
                                       0889  1234
                                       0889  1235 LEADING_ZEROS:
                                       0889  1236
                                       0889  1237
                                       0889  1238 ; If processing continues here it is known that the loss of significance is due
                                       0889  1239 ; to a string of leading zeros.  Note that it is known that the loop for
                                       0889  1240 ; leading zeros will terminate before an underflow condition occurs so that the
                                       0889  1241 ; loop does not include a test for underflow.
                                       0889  1242
                             65   3D   D0  0889  1243          MOVL    #L_INT_WEIGHT, (R5)     ; T8 = exp bias for last longword
                                       088C  1244                                              ;    of the product K*C'
                                       088C  1245
          0C A4          0000FFFF 8F   D1  088C  1246 LOOP_0: CMPL    #^X0000FFFF, 12(R4)     ; Check enough fraction bits
                                   14   19  0894  1247          BLSS    CONVERT_0               ; Enough bits.  Convert to floating
                00000BCA'EF        16   08  0896  1248          JSB     GEN_MORE_BITS           ; T2/R7 contain K*C''
                             0C A4    D5  089C  1249 1$:        TSTL    12(R4)                  ; Check for all 0's
                                   09   12  089F  1250          BNEQ    CONVERT_0               ; Not all 0's.  Enough precision bits.
                          64   FC A4   7D  08A1  1251          MOVQ    -4(R4), (R4)            ; Compress representation of K*C''
                             65   20   A0  08A5  1252          ADDW    #W_TERM_WEIGHT, (R5)    ; Increment weighting factor.
                                   E2   11  08A8  1253          BRB     LOOP_0
                                       08AA  1254
                                       08AA  1255 CONVERT_0:
                          54    04   C0  08AA  1256          ADDL    #4, R4                  ; R4 points to low bits of h
                00000910'EF     16   08  08AD  1257          JSB     CVT_LONG_TO_H           ; R0/R3 = 2^W*h
                          54    50 7DFD  08B3  1258          MOVQ    R0, R4                  ; R4/R7 = 2^W*h
          52  1C AE    03     1D   EF  08B7  1259          EXTZV   #29, #3, 28(SP), R2     ; R2 = octant bits
                       54  20 AE   C2  08BD  1260          SUBL    32(SP), R4              ; R4/R7 = Y = h
                           00   11     08C1  1261          BRB     RESTORE
                                       08C3  1262
                                       08C3  1263
                                       08C3  1264 RESTORE:
                          24 AE    B8  08C3  1265          BISPSW  36(SP)                  ; Restore IV bit and exit
                             5E   28   C0  08C6  1266          ADDL    #40, SP                 ; Remove mask and temporary storage
                                       08C9  1267                                              ;   from stack
                                   05  08C9  1268          RSB
                                       08CA  1269
                                       08CA  1270 GEN_MORE_BITS:
                                       08CA  1271
                                       08CA  1272 ;
                                       08CA  1273 ; This subroutine generates 85 extra fraction bits and puts them to the
                                       08CA  1274 ; existing bits.  NOTE:  This routine is always entered via a JSB instruction.
                                       08CA  1275 ; Consequently, SP points to the first longword BEFORE T0, rather than T0
                                       08CA  1276 ; itself.
                                       08CA  1277 ;
                                       08CA  1278
                             52   04   C2  08CA  1279          SUBL    #4, R2                  ; Adjust pointer to get next quadword
                                       08CD  1280                                              ;   from MTH$AL_4_OV_PI_V
                    56       62   53   79  08CD  1281          ASHQ    R3, (R2), R6            ; R7 = C(n)
                                   17   18  08D1  1282          BGEQ    1$                      ; Branch if high bit is clear
```

```
                                08D3  1283
                                08D3  1284 ; Logic to process unsigned values greater than 2^31 - 1
                                08D3  1285
    04 AE    00    57    51    7A  08D3  1286          EMUL    R1, R7, #0, 4(SP)           ;
             08 AE   51    C0    08D9  1287          ADDL    R1, 8(SP)                   ; T0/T1 = KLO*C(n)
08 AE    08 AE    57    50    7A  08DD  1288          EMUL    R0, R7, 8(SP), 8(SP)        ;
             0C AE   50    C0    08E4  1289          ADDL    R0, 12(SP)                  ; T0/T2 = K*C(n)
                     0D    11    08E8  1290          BRB     2$
                                08EA  1291
                                08EA  1292 ; Logic to process unsigned values less than 2^31
                                08EA  1293
    04 AE    00    57    51    7A  08EA  1294 1$:      EMUL    R1, R7, #0, 4(SP)           ; T0/T1 = KLO*C(n)
08 AE    08 AE    57    50    7A  08F0  1295          EMUL    R0, R7, 8(SP), 8(SP)        ; T0/T2 = K*C(n)
                                08F7  1296
                                08F7  1297 ; Add new bits to old
                                08F7  1298
          64    08 AE   C0    08F7  1299 2$:      ADDL    8(SP), (R4)                 ;
       04 A4    0C AE   D8    08FB  1300          ADWC    12(SP), 4(R4)               ;
                     08    1E    0900  1301          BCC     3$                          ; Check for carry from previous add
                08 A4    D6    0902  1302          INCL    8(R4)                       ; Propagate carry
                     03    1E    0905  1303          BCC     3$                          ; Check for carry from previous add
                0C A4    D6    0907  1304          INCL    12(R4)                      ; Propagate carry
          FC A4    04 AE   D0    090A  1305 3$:      MOVL    4(SP), -4(R4)               ; Move new low order bits to end of
                                090F  1306                                              ;   of old low order bits
                           05   090F  1307          RSB                                 ;
                                0910  1308
                                0910  1309
                                0910  1310
                                0910  1311
                                0910  1312
                                0910  1313 CVT_LONG_TO_H:
                                0910  1314
                                0910  1315 ;
                                0910  1316 ; This routine converts an array of three longword pointed to by R4 H format.
                                0910  1317 ; The result is returned in R0/R3.  NOTE:  This routine is always entered via
                                0910  1318 ; a JSB instruction.  Consequently, SP points to the first longword BEFORE T0,
                                0910  1319 ;  rather than T0 itself.
                                0910  1320 ;
56    5E    04    C1    0910  1321          ADDL3   #4, SP, R6                  ; R6 points to T0
      50    84 6EFD   0914  1322          CVTLH   (R4)+, R0                   ; R0/R3 = low 32 bits of h
            07    18    0918  1323          BGEQ    1$                          ; Check for signed conversion error
            64    D6    091A  1324          INCL    (R4)                        ;   and adjust  accordingly
            03    1E    091C  1325          BCC     1$                          ; If necessary,
         04 A4    D6    091E  1326          INCL    4(R4)                       ;   propogate carry
      50    20    A2    0921  1327 1$:      SUBW    #W_TERM_WEIGHT, R0          ; R0/R3 = (low 32 bits of h)/2^32
      66    84 6EFD   0924  1328          CVTLH   (R4)+, (R6)                 ; T0/T3 = 2nd lowest 32 bits of h
            02    18    0928  1329          BGEQ    2$                          ; Check for signed conversion error
            64    D6    092A  1330          INCL    (R4)                        ;   and adjust  accordingly
      50    66 60FD   092C  1331 2$:      ADDH2   (R6), R0                    ; R0/R3 = (low 64 bits of h)/2^32
      50    20    A2    0930  1332          SUBW    #W_TERM_WEIGHT, R0          ; R0/R3 = (low 64 bits of h)/2^64
      66    84 6EFD   0933  1333          CVTLH   (R4)+, (R6)                 ; T0/T3 = 3rd lowest 32 bits of h
            06    18    0937  1334          BGEQ    3$                          ; Check for signed conversion error
66    FDB2  CF 60FD   0939  1335          ADDH2   H_2_TO_32, (R6)             ;   and adjust  accordingly
      50    66 60FD   093F  1336 3$:      ADDH2   (R6), R0                    ; R0/R3 = (low 96 bits of h)/2^64
                  05  0943  1337          RSB
                      0944  1338
```

```
                     0944  1340          .SBTTL  REDUCE_DEGREES
                     0944  1341
                     0944  1342 ; This routine assumes that the absolute value of the argument is in R0/R1.
                     0944  1343 ; The reduction process is performed in two stages.  The first stage of
                     0944  1344 ; the reduction reduces the argument modulo 360 to a value less that 2^53,
                     0944  1345 ; and the second stage reduces the argument modulo 45 to a value less than 45.
                     0944  1346
                     0944  1347 ; Constants used in this reduction:
                     0944  1348 ;
                     0944  1349
                     0944  1350 POWER_MOD_360_0:          ; Powers of 2 modulo 360 for t1 = 0
0008 0004 0002 0001  0944  1351          .WORD     1,      2,      4,      8
0080 0040 0020 0010  094C  1352          .WORD    16,     32,     64,    128
00F8 0130 0098 0100  0954  1353          .WORD   256,    152,    304,    248
                     095C  1354
                     095C  1355 POWER_MOD_360_1:          ; Powers of 2 modulo 360 for t1 <> 0
0008 00B8 0110 0088  095C  1356          .WORD   136,    272,    184,      8
0080 0040 0020 0010  0964  1357          .WORD    16,     32,     64,    128
00F8 0130 0098 0100  096C  1358          .WORD   256,    152,    304,    248
                     0974  1359
                     0974  1360
                     0974  1361
                     0974  1362 REDUCE_DEGREES:
  50   4360 8F    B1  0974  1363          CMPW    #^X4360, R0             ; Compare |x| with 2^53
               4E    14  0979  1364          BGTR    LAST_STEP               ; Branch to special logic for med arg
                     097B  1365
                     097B  1366 ;
                     097B  1367 ; It is assumed here that the argument is greater than 2^53.
                     097B  1368 ;
                     097B  1369 ; The argument is reduced as follows:
                     097B  1370 ;     Let x = 2^t*f, where t > 53 and 1/2 =< f < 1.  And let J = 2^53*f =
                     097B  1371 ;     2^30*J1 + J2 and K = 2^(t-53).  Since 2^30 = 64 modulo 360, we have that
                     097B  1372 ;     J = 64*J1 + J2 modulo 360.  Now let t' = t - 53 = 12*t1 + t2.  Note that
                     097B  1373 ;     (2^12)^2 = (2^9)*(2^15) = (2^9)*(2^3) = 2^12 modulo 360.  Hence, if t1 is
                     097B  1374 ;     not zero  K = 2^t' = 2^(12*t1+t2) = (2^12)*(2^t2) = 136*2^t2 modulo 360.
                     097B  1375 ;     For t1 = 0 K = 2^t2.  Consequently, define K' congruent to 2^t2 if t1 = 0
                     097B  1376 ;     and congruent to 136*2^t2 otherwise, where 0 =< K' < 360.  Then x' =
                     097B  1377 ;     K'*(8*J1 + J2) is congruent to s modulo 360 and x' < 2^53.
                     097B  1378
        52   50    D0  097B  1379          MOVL    R0, R2                  ; R2 = high longword of X
50   00007FF0 8F    CA  097E  1380          BICL    #^X7FF0, R0             ; Clear exp bits of X
     50   4350 8F    A8  0985  1381          BISW    #^X4350, R0             ; R0/R1 = J
        52   50    C2  098A  1382          SUBL    R0, R2                  ; R2 = t'*2^7
                     098D  1383
        53   50    7D  098D  1384          MOVQ    R0, R3                  ; R3/R4 = J
51    FFFF3FFF 8F    CA  0990  1385          BICL    #^XFFFF3FFF, R1         ; R0/R1 = J1*2^30
        53   50 42FD  0997  1386          SUBG2   R0, R3                  ; R3/R4 = J2
     50   0180 8F    A2  099B  1387          SUBW    #^X180, R0              ; R0/R1 = 64*J1
        50   53 40FD  09A0  1388          ADDG2   R3, R0                  ; R0/R1 = 64*J1 + J2 = J modulo 45
                     09A4  1389
 52   52   FC 8F    9C  09A4  1390          ROTL    #-4, R2, R2             ; R2 = t'
     53   52   0C    A7  09A9  1391          DIVW3   #12, R2, R3             ; R3 = t1
        53   0C    A4  09AD  1392          MULW    #12, R3                 ; R3 = 12*t1
        52   53    A2  09B0  1393          SUBW    R3, R2                  ; R2 = t2
                     09B3  1394
             53    B5  09B3  1395          TSTW    R3                      ; Check for t1 = 0 and choose K'
             08    12  09B5  1396          BNEQ    1$                      ;   accordingly
```

```
                52   88 AF42 4DFD  09B7  1397          CVTWG   POWER_MOD_360_0[R2], R2 ; R2/R3 = K'
                          06   11   09BD  1398          BRB     2$                      ;
                52   98 AF42 4DFD  09BF  1399 1$:       CVTWG   POWER_MOD_360_1[R2], R2 ; R2/R3 = K'
                50   52 44FD       09C5  1400 2$:       MULG2   R2, R0                  ; R0/R1 = X' (mod 45) 0 =< R0 < 2^53
                                    09C9  1401
                                    09C9  1402
                                    09C9  1403 LAST_STEP:
                                    09C9  1404 ;
                                    09C9  1405 ; Argument reduction scheme for arguments with absolute value less than 2^53
                                    09C9  1406 ;
                                    09C9  1407 ; The reduced argument Y is computed as follows:
                                    09C9  1408 ;    Let I = int(X/45)
                                    09C9  1409 ;        if I is even
                                    09C9  1410 ;           then Y = X - 45*I
                                    09C9  1411 ;           else Y = (I+1)*45 - x
                                    09C9  1412
                50   4240 8F    B1   09C9  1413          CMPW    #^X4240, R0             ; Compare 2^36 with |X|
                          28   18   09CE  1414          BGEQ    NO_OVERFLOW             ;
          56    50   F673 CF 45FD  09D0  1415          MULG3   G_T_OV_45, R0, R6       ; R6/R7 = |X|/45
                                    09D7  1416
                                    09D7  1417 ;
                                    09D7  1418 ; Turn off IV to avoid an exception in EMODG
                                    09D7  1419 ;
                          52   DC   09D7  1420          MOVPSL  R2                      ; Move PSL to R2
           52     FFFFFFDF 8F   CA   09D9  1421          BICL    #^C<PSL$M_IV>, R2       ; Save current IV bit
                          20   B9   09E0  1422          BICPSW  #PSL$M_IV               ; Turn off integer overflow trap
                                    09E2  1423
     54   53    56   00   08 54FD  09E2  1424          EMODG   #1, #0, R6, R3, R4      ; R3 = low 32 integer bits of |X|/45
                                    09E9  1425                                          ; R4/R5 = fractional part of |X|/45
                                    09E9  1426
                          52   B8   09E9  1427          BISPSW  R2                      ; Restore IV bit
                                    09EB  1428
                     56   54 42FD  09EB  1429          SUBG2   R4, R6                  ; R6/R7 = Integer part of |X|/45 = I
                       2F 53   E9   09EF  1430          BLBC    R3, EVEN                ;
                     56   08 40FD  09F2  1431          ADDG2   #1, R6                  ; R6/R7 = I + 1
                          16   11   09F6  1432          BRB     ODD
                                    09F8  1433
                                    09F8  1434
                                    09F8  1435 NO_OVERFLOW:
53   50   C16C 8F    F64B CF 54FD  09F8  1436          EMODG   G_1_OV_45, #X_1_OV_45, R0, R3, R4
                          54        0A02
                                    0A03  1437                                          ; R3 = I = integer part of |X|/45
                    17 53   E9      0A03  1438          BLBC    R3, CVT                 ; Branch if octant bits are even
                56  53   01   C1    0A06  1439          ADDL3   #1, R3, R6              ; R6 = I + 1
                    56   56 4EFD     0A0A  1440          CVTLG   R6, R6                  ; R6/R7 = I + 1
                56  F61D CF 44FD     0A0E  1441 ODD:     MULG2   G_45, R6                ; R6/R7 = 45*(I+1)
                    56   50 42FD     0A14  1442          SUBG2   R0, R6                  ; R6/R7 = Y
                  53   F8 8F    8A    0A18  1443          BICB    #^XF8, R3               ; Save only last three octant bits
                                05    0A1C  1444          RSB
                                    0A1D  1445
                    56   53 4EFD     0A1D  1446 CVT:     CVTLG   R3, R6                  ; R6/R7 = I
                56  F612 CF 44FD     0A21  1447 EVEN:    MULG2   G_M45, R6               ; R6/R7 = -45*I
                    56   50 40FD     0A27  1448          ADDG2   R0, R6                  ; R6/R7 = Y
                  53   F8 8F    8A    0A2B  1449          BICB    #^XF8, R3               ; Save only last three octant bits
                                05    0A2F  1450          RSB
                                    0A30  1451
```

```
                                0A30  1453
                                0A30  1454          .SBTTL  RADIAN_POLYNOMIALS      ; Polynomials for arguments in radians
                                0A30  1455
                                0A30  1456
                                0A30  1457
                                0A30  1458 ;
                                0A30  1459 ; Polynomial evaluation for GCOS(Y) for Y in radians
                                0A30  1460 ;
                                0A30  1461
                                0A30  1462 P_COS_R:
      54    8000 8F    AA      0A30  1463          BICW    #^X8000, R4             ; R4/R7 = |Y|
      54    4000 8F    B1      0A35  1464          CMPW    #^X4000, R4             ; Compare 1/2 with |Y|
                  28    14      0A3A  1465          BGTR    LESS_THAN_HALF                  ; Sufficent overhang is available
                                0A3C  1466 NEEDS_DOUBLE:
            54    54 64FD       0A3C  1467          MULH2   R4, R4                  ; R4/R7 = Y^2
            7E    54    7D      0A40  1468          MOVQ    R4, -(SP)               ; Save high half of Y^2
                                0A43  1469 NEEDS_DOUBLE_SINCOS:
            54    54 76FD       0A43  1470          CVTHG   R4, R4                  ; R4/R5 = Y^2
 F6AA CF    07    54 55FD       0A47  1471          POLYG   R4, #COSLENR2-1, COSTBR2; R0/R1 = Q(Y^2)
            54    8E    7D      0A4E  1472          MOVQ    (SP)+, R4               ; R4/R7 = Y^2
                  54    B7      0A51  1473          DECW    R4                      ; R4/R7 = Y^2/2
            54    08 62FD       0A53  1474          SUBH2   #1, R4                  ; R4/R7 = Y^2/2 - 1
            50    50 56FD       0A57  1475          CVTGH   R0, R0                  ; R0/R3 = Q(Y^2)
            50    54 62FD       0A5B  1476          SUBH2   R4, R0                  ; R0/R3 = GCOS(Y)
            50    50 76FD       0A5F  1477          CVTHG   R0, R0                  ; R0/R1 = GCOS(Y)
                        05      0A63  1478          RSB
                                0A64  1479 LESS_THAN_HALF:
            56    54 76FD       0A64  1480          CVTHG   R4, R6                  ; R6/R7 = |Y|
            56    56 44FD       0A68  1481          MULG2   R6, R6                  ; R6/R7 = Y^2
 F645 CF    07    56 55FD       0A6C  1482          POLYG   R6, #COSLENR1-1, COSTBR1; R0/R1 = GCOS(Y)
                        05      0A73  1483          RSB
                                0A74  1484
                                0A74  1485
                                0A74  1486 ;
                                0A74  1487 ; Polynomial evaluation for -GCOS(Y)
                                0A74  1488 ;
                                0A74  1489
                                0A74  1490 N_COS_R:
      54    8000 8F    AA      0A74  1491          BICW    #^X8000, R4             ; R4/R7 = |Y|
      54    4000 8F    B1      0A79  1492          CMPW    #^X4000, R4             ; Compare 1/2 with |Y|
                  28    14      0A7E  1493          BGTR    1$                      ; Sufficent overhang is available
            54    54 64FD       0A80  1494          MULH2   R4, R4                  ; R4/R7 = Y^2
            7E    54    7D      0A84  1495          MOVQ    R4, -(SP)               ; Save high half of Y^2
            54    54 76FD       0A87  1496          CVTHG   R4, R4                  ; R4/R5 = Y^2 in G-format
 F666 CF    07    54 55FD       0A8B  1497          POLYG   R4, #COSLENR2-1, COSTBR2; R0/R1 = Q(Y^2)
            54    8E    7D      0A92  1498          MOVQ    (SP)+, R4               ; R4/R7 = Y^2
                  54    B7      0A95  1499          DECW    R4                      ; R4/R7 = Y^2/2
            54    08 62FD       0A97  1500          SUBH2   #1, R4                  ; R4/R7 = -(1 - Y^2/2)
            50    50 56FD       0A9B  1501          CVTGH   R0, R0                  ; R0/R3 = Q(Y^2)
            54    50 62FD       0A9F  1502          SUBH2   R0, R4                  ; R4/R7 = -GCOS(Y)
            50    54 76FD       0AA3  1503          CVTHG   R4, R0                  ; R0/R1 = -GCOS(Y)
                        05      0AA7  1504          RSB
                                0AA8  1505
            56    54 76FD       0AA8  1506 1$:      CVTHG   R4, R6                  ; R6/R7 = |Y|
            56    56 44FD       0AAC  1507          MULG2   R6, R6                  ; R6/R7 = Y^2
 F601 CF    07    56 55FD       0AB0  1508          POLYG   R6, #COSLENR1-1, COSTBR1; R0/R1 = GCOS(Y)
      50    8000 8F    AC      0AB7  1509          XORW    #^X8000, R0             ; R0/R1 = -GCOS(Y)
```

```
                     05  0ABC  1510          RSB
                         0ABD  1511
                         0ABD  1512 ;
                         0ABD  1513 ; Polynomial evaluation for -GSIN(Y)
                         0ABD  1514 ;
                         0ABD  1515
                         0ABD  1516 N_SIN_R:
        54   8000 8F     AC  0ABD  1517          XORW    #^X8000, R4
                         0AC2  1518
                         0AC2  1519 ;
                         0AC2  1520 ; Polynomial evaluation for GSIN(Y)
                         0AC2  1521 ;
                         0AC2  1522
                         0AC2  1523 P_SIN_R:
            7E   54    7D  0AC2  1524          MOVQ    R4, -(SP)                ; Save high half of Y
            7E   54 76FD  0AC5  1525          CVTHG   R4, -(SP)                ; Save Y in G-format
        54  6E   6E 45FD  0AC9  1526          MULG2   (SP), (SP), R4           ; R4 = Y^2
  F663 CF   06   54 55FD  0ACE  1527          POLYG   R4, #SINLENR-1, SINTBR   ; R0/R1 = P(Y^2)
            50   8E 44FD  0AD5  1528          MULG2   (SP)+, R0                ; R0/R1 = Y*P(Y^2)
            54   8E    7D  0AD9  1529          MOVQ    (SP)+, R4                ; R4/R7 = Y
            50   50 56FD  0ADC  1530          CVTGH   R0, R0                   ; R0/R3 = Y*P(Y^2)
            50   54 60FD  0AE0  1531          ADDH2   R4, R0                   ; R0/R3 = GSIN(Y)
            50   50 76FD  0AE4  1532          CVTHG   R0, R0                   ; R0/R1 = GSIN(Y)
                     05  0AE8  1533          RSB
                         0AE9  1534
                         0AE9  1535
                         0AE9  1536
```

```
                        0AE9  1538          .SBTTL  CYCLE_POLYNOMIALS       ; Polynomials for arguments in cycles
                        0AE9  1539
                        0AE9  1540
                        0AE9  1541
                        0AE9  1542 ;
                        0AE9  1543 ; Polynomial evaluation for GCOS(Y) for Y in cycles
                        0AE9  1544 ;
                        0AE9  1545
                        0AE9  1546 P_COS_C:
      54    F5BA CF 71FD  0AE9  1547          CMPH    H_2_OV_PI, R4           ; Compare 2/pi with .Y!
                  29   18  0AEF  1548          BGEQ    1$                      ; Sufficent overhang is available
            54    54 64FD  0AF1  1549          MULH2   R4, R4                  ; R4/R7 = Y^2
            7E    54   7D  0AF5  1550          MOVQ    R4, -(SP)               ; Save high half of Y^2
            54    54 76FD  0AF8  1551          CVTHG   R4, R4                  ; R4/R5 = Y^2
 F6AD CF    07    54 55FD  0AFC  1552          POLYG   R4, #COSLENC2-1, COSTBC2; R0/R1 = Q(Y^2)
            54    8E   7D  0B03  1553          MOVQ    (SP)+, R4               ; R4/R7 = Y^2
            54    02   A2  0B06  1554          SUBW    #2, R4                  ; R4/R7 = Y^2/4
            54    08 62FD  0B09  1555          SUBH2   #1, R4                  ; R4/R7 = Y^2/4 - 1
            50    50 56FD  0B0D  1556          CVTGH   R0, R0                  ; R0/R3 = Q(Y^2)
            50    54 62FD  0B11  1557          SUBH2   R4, R0                  ; R0/R3 = GCOS(Y)
            50    50 76FD  0B15  1558          CVTHG   R0, R0                  ; R0/R1 = GCOS(Y)
                     05    0B19  1559          RSB
                        0B1A  1560
            56    54 76FD  0B1A  1561 1$:      CVTHG   R4, R6                  ; R6/R7 = !Y!
            56    56 44FD  0B1E  1562          MULG2   R6, R6                  ; R6/R7 = Y^2
 F647 CF    07    56 55FD  0B22  1563          POLYG   R6, #COSLENC1-1, COSTBC1; R0/R1 = GCOS(Y) - 1
            50    08 40FD  0B29  1564          ADDG2   #1, R0                  ; R0/R1 = GCOS(Y)
                     05    0B2D  1565          RSB
                        0B2E  1566
                        0B2E  1567
                        0B2E  1568 ;
                        0B2E  1569 ; Polynomial evaluation for -GCOS(Y)
                        0B2E  1570 ;
                        0B2E  1571
                        0B2E  1572 N_COS_C:
      54    8000 8F   AA  0B2E  1573          BICW    #^X8000, R4             ; R4/R7 = !Y!
      54    F570 CF 71FD  0B33  1574          CMPH    H_2_OV_PI, R4           ; Compare 2/pi with !Y!
                  29   18  0B39  1575          BGEQ    1$                      ; Sufficent overhang is available
            54    54 64FD  0B3B  1576          MULH2   R4, R4                  ; R4/R7 = Y^2
            7E    54   7D  0B3F  1577          MOVQ    R4, -(SP)               ; Save high half of Y^2
            54    54 76FD  0B42  1578          CVTHG   R4, R4                  ; R4/R5 = Y^2
 F663 CF    07    54 55FD  0B46  1579          POLYG   R4, #COSLENC2-1, COSTBC2; R0/R1 = Q(Y^2)
            54    8E   7D  0B4D  1580          MOVQ    (SP)+, R4               ; R4/R7 = Y^2
            54    02   A2  0B50  1581          SUBW    #2, R4                  ; R4/R7 = Y^2/4
            54    08 62FD  0B53  1582          SUBH2   #1, R4                  ; R4/R7 = -(1 - Y^2/4)
            50    50 56FD  0B57  1583          CVTGH   R0, R0                  ; R0/R3 = Q(Y^2)
            54    50 62FD  0B5B  1584          SUBH2   R0, R4                  ; R4/R7 = -GCOS(Y)
            50    54 76FD  0B5F  1585          CVTHG   R4, R0                  ; R0/R1 = -GCOS(Y)
                     05    0B63  1586          RSB
                        0B64  1587
            56    54 76FD  0B64  1588 1$:      CVTHG   R4, R6                  ; R6/R7 = !Y!
            56    56 44FD  0B68  1589          MULG2   R6, R6                  ; R6/R7 = Y^2
 F5FD CF    07    56 55FD  0B6C  1590          POLYG   R6, #COSLENC1-1, COSTBC1; R0/R1 = GCOS(Y) - 1
 50   F487 CF    50 43FD  0B73  1591          SUBG3   R0, G_M1, R0            ; R0/R1 = -GCOS(Y)
                     05    0B7A  1592          RSB
                        0B7B  1593
                        0B7B  1594 ;
```

```
                     0B7B  1595 ; Polynomial evaluation for -SIN(Y)
                     0B7B  1596 ;
                     0B7B  1597
                     0B7B  1598 N_SIN_C:
        54   8000 8F  AC  0B7B  1599          XORW    #^X8000, R4             ; R4/R7 = - Y
                     0B80  1600
                     0B80  1601 ;
                     0B80  1602 ; Polynomial evaluation for GSIN(Y)
                     0B80  1603 ;
                     0B80  1604
                     0B80  1605 P_SIN_C:
             7E   54 7DFD  0B80  1606          MOVO    R4, -(SP)               ; Save argument in H format
             7E   54 76FD  0B84  1607          CVTHG   R4, -(SP)               ; Save argument in G format
        54   6E   6E 45FD  0B88  1608          MULG3   (SP), (SP), R4          ; R4/R5 = Y^2
  F65C CF    06   54 55FD  0B8D  1609          POLYG   R4, #SINLENC-1, SINTBC  ; R0/R1 = P(Y^2)
             50   8E 44FD  0B94  1610          MULG2   (SP)+, R0               ; R0/R1 = Y*P(Y^2)
             54   6E   7D  0B98  1611          MOVQ    (SP), R4                ; R4/R7 = Y
             6E   02   A2  0B9B  1612          SUBW    #2, (SP)                ; (SP) = Y/4
             54   8E 62FD  0B9E  1613          SUBH2   (SP)+, R4               ; R4/R7 = 3/4*Y
             50   50 56FD  0BA2  1614          CVTGH   R0, R0                  ; R0/R3 = Y*P(Y^2)
             50   54 60FD  0BA6  1615          ADDH2   R4, R0                  ; R0/R3 = GSIN(Y)
             50   50 76FD  0BAA  1616          CVTHG   R0, R0                  ; R0/R1 = GSIN(Y)
                       05  0BAE  1617          RSB
                     0BAF  1618
```

```
                                0BAF  1620          .SBTTL  DEGREE_POLYNOMIALS
                                0BAF  1621
                                0BAF  1622
                                0BAF  1623 P_COS_D:
      56   F4A4 CF   51FD       0BAF  1624          CMPG    G_90_OV_PI, R6          ; Compare 90/pi with Y
                     49    18   0BB5  1625          BGEQ    2$                      ; Double precision isn't needed
           54   56   56FD       0BB7  1626          CVTGH   R6, R4                  ; R4/R7 = Y
           54   54   64FD       0BBB  1627          MULH2   R4, R4                  ; R4/R7 = Y^2
           7E   54     7D       0BBF  1628          MOVQ    R4, -(SP)               ; Save high half of Y^2
           54   54   76FD       0BC2  1629          CVTHG   R4, R4                  ; R4/R5 = Y^2
 F69B CF   07   54   55FD       0BC6  1630          POLYG   R4, #COSDLN1, COSDTB1   ; R0/R1 = Q(Y^2)
           50   50   56FD       0BCD  1631          CVTGH   R0, R0                  ; R0/R3 = Q(Y^2)
           54   8E     7D       0BD1  1632          MOVQ    (SP)+, R4               ; R4/R7 = Y^2
           54   0D     A2       0BD4  1633          SUBW    #13, R4                 ; R4/R7 = Y^2/2^13
 57   00001000 8F     D1       0BD7  1634          CMPL    #^X1000, R7             ; Check for loss of significanc when
                     13    1A   0BDE  1635          BGTRU   1$                      ;    subtracting 1 and adjust
           7E   54   7DFD       0BE0  1636          MOVO    R4, -(SP)               ; Save Y^2/2^13 on stack
 57   FFFF0000 8F     CA       0BE4  1637          BICL    #^XFFFF0000, R7         ; R4/R7 = High order bits of Y^2/2^13
           6E   54   62FD       0BEB  1638          SUBH2   R4, (SP)                ; (SP) = Low order bits
           50   8E   62FD       0BEF  1639          SUBH2   (SP)+, R0               ; R0/R3 = Q(Y^2) - low bits of Y^2/2^13
           54   08   62FD       0BF3  1640 1$:      SUBH2   #1, R4                  ; R4/R7 = -(1 - Y^2/2^13)
           50   54   62FD       0BF7  1641          SUBH2   R4, R0                  ; R0/R3 = GCOS(Y)
           50   50   76FD       0BFB  1642          CVTHG   R0, R0                  ; R0/R1 = GCOS(Y)
                       05       0BFF  1643          RSB
                                0C00  1644
           56   56   44FD       0C00  1645 2$:      MULG2   R6, R6                  ; R6/R7 = Y^2
                     08    13   0C04  1646          BEQL    3$                      ; Check for Y = 0
 F61B CF   07   56   55FD       0C06  1647          POLYG   R6, #COSDLN2, COSDTB2   ; R0/R1 = Q(Y^2)
                       05       0C0D  1648          RSB
                                0C0E  1649
           50   08   50FD       0C0E  1650 3$:      MOVG    #1, R0                  ; R0/R1 = GCOS(Y)
                       05       0C12  1651          RSB
                                0C13  1652
                                0C13  1653
                                0C13  1654 N_COS_D:
      56   F440 CF   51FD       0C13  1655          CMPG    G_90_OV_PI, R6          ; Compare 90/pi with Y
                     49    18   0C19  1656          BGEQ    2$                      ; Double precision isn't needed
           54   56   56FD       0C1B  1657          CVTGH   R6, R4                  ;
           54   54   64FD       0C1F  1658          MULH2   R4, R4                  ; R4/R7 = Y^2
           7E   54     7D       0C23  1659          MOVQ    R4, -(SP)               ;
           54   54   76FD       0C26  1660          CVTHG   R4, R4                  ; R4/R5 = Y^2
 F637 CF   07   54   55FD       0C2A  1661          POLYG   R4, #COSDLN1, COSDTB1   ; R0/R1 = Q(Y^2)
           50   50   56FD       0C31  1662          CVTGH   R0, R0                  ; R0/R3 = Q(Y^2)
           54   8E     7D       0C35  1663          MOVQ    (SP)+, R4               ; R4
           54   0D     A2       0C38  1664          SUBW    #13, R4                 ; R4/R7 = Y^2/2^13
 57   00001000 8F     D1       0C3B  1665          CMPL    #^X1000, R7             ; Check for loss of significanc when
                     13    1A   0C42  1666          BGTRU   1$                      ;    subtracting 1 and adjust
           7E   54   7DFD       0C44  1667          MOVO    R4, -(SP)               ; Save Y^2/2^13 on stack
 57   FFFF0000 8F     CA       0C48  1668          BICL    #^XFFFF0000, R7         ; R4/R7 = High order bits of Y^2/2^13
           6E   54   62FD       0C4F  1669          SUBH2   R4, (SP)                ; (SP) = Low order bits
           50   8E   62FD       0C53  1670          SUBH2   (SP)+, R0               ; R0/R3 = Q(Y^2) - low bits of Y^2/2^13
           54   08   62FD       0C57  1671 1$:      SUBH2   #1, R4                  ; R4/R7 = -(1 - Y^2/2^13)
           54   50   62FD       0C5B  1672          SUBH2   R0, R4                  ; R4/R7 = -COS(Y)
           50   54   76FD       0C5F  1673          CVTHG   R4, R0                  ; R0/R1 = -GCOS(Y)
                       05       0C63  1674          RSB
                                0C64  1675
           56   56   44FD       0C64  1676 2$:      MULG2   R6, R6                  ; R6/R7 = Y^2
```

```
                     0D   13  0C68  1677          BEQL    3$                      ; Check for Y = 0
F5B7 CF    07   56 55FD  0C6A  1678          POLYG   R6, #COSDLN2, COSDTB2  ; R0/R1 = GCOSD(Y)
     50    8000 8F   AC  0C71  1679          XORW    #^X8000, R0             ; R0/R1 = -GCOSD(Y)
                       05  0C76  1680          RSB
                           0C77  1681
     50    F384 CF 50FD  0C77  1682 3$:       MOVG    G_M1, R0                ; R0/R1 = GCOS(Y)
                       05  0C7D  1683          RSB
                           0C7E  1684
                           0C7E  1685 N_SIN_D:
     56    56   52FD  0C7E  1686          MNEGG   R6, R6                  ; R6/R7 = -Y
                           0C82  1687 P_SIN_D:
     50    56   56 45FD  0C82  1688          MULG3   R6, R6, R0              ; R0/R1 = Y^2
                 14   13  0C87  1689          BEQL    RETURN
F618 CF    06   50 55FD  0C89  1690          POLYG   R0, #SINDLN, SINDTB     ; R0/R1 = P(Y^2)
     50    56   56 44FD  0C90  1691          MULG2   R6, R0                  ; R0/R1 = Y*P(Y^2)
     56    0060 8F   A2  0C94  1692          SUBW    #^X60, R6               ; R6/R7 = Y/2^6
     50    56   56 40FD  0C99  1693          ADDG2   R6, R0                  ; R0/R1 = GSIN(Y)
                       05  0C9D  1694 RETURN: RSB
                           0C9E  1695
```

```
                        0C9E  1697
                        0C9E  1698
                        0C9E  1699          .SBTTL  DEGENERATE_SOLUTIONS
                        0C9E  1700
                        0C9E  1701 P_ONE:
        50    08 50FD   0C9E  1702          MOVG    #1, R0                  ; Answer is 1
              05        0CA2  1703          RSB
                        0CA3  1704
                        0CA3  1705
                        0CA3  1706 N_ONE:
     50    F358 CF 50FD 0CA3  1707          MOVG    G_M1, R0                ; Answer is -1
              05        0CA9  1708          RSB
                        0CAA  1709
                        0CAA  1710
                        0CAA  1711 UNFL:
                        0CAA  1712 ;
                        0CAA  1713 ; Underflow; if user has FU set, signal error.  Always return 0.0
                        0CAA  1714 ;
           52   DC      0CAA  1715          MOVPSL  R2                      ; R2 = user's or jacket routine's PSL
00000000'GF     00   FB 0CAC  1716          CALLS   #0, G^MTH$$JACKET_TST   ; R0 = TRUE if JSB from jacket routine
          04 50   E9    0CB3  1717          BLBC    R0, 10$                 ; branch if user did JSB
      52  04 AD   3C    0CB6  1718          MOVZWL  SF$W_SAVE_PSW(FP), R2   ; get user PSL saved by CALL
           50   D4      0CBA  1719 10$:     CLRL    R0                      ; R0 = result. LIB$SIGNAL will save in
                        0CBC  1720                                          ; CHF$L_MCH_R0/R1 so any handler can
                        0CBC  1721                                          ;   fixup
      0D 52   06   E1   0CBC  1722          BBC     #6, R2, 20$             ; has user enabled floating underflow?
              6E   DD   0CC0  1723          PUSHL   (SP)                    ; yes. return PC from special routine
      7E   00'8F   9A   0CC2  1724          MOVZBL  #MTH$K_FLOUNDMAT, -(SP) ; trap code for hardware floating
                        0CC6  1725                                          ;   underflow convert to MTH$_FLOUNDMAT
                        0CC6  1726                                          ;   (32-bit VAX-11 exception code)
00000000'GF     02   FB 0CC6  1727          CALLS   #2, G^MTH$$SIGNAL       ; signal (condition, PC)
              05        0CCD  1728 20$:     RSB                             ; return
                        0CCE  1729
                        0CCE  1730          .END
```

```
CONVERT                         00000803 R      02
CONVERT_0                       000008AA R      02
CONVERT_1                       00000B6D R      02
COSD                          = 0000000C
COSDLN1                       = 00000007
COSDLN2                       = 00000007
COSDTB1                         00000268 R      02
COSDTB2                         00000228 R      02
COSINE                        = 0000000C
COSLENC1                      = 00000008
COSLENC2                      = 00000008
COSLENR1                      = 00000008
COSLENR2                      = 00000008
COSTBC1                         00000170 R      02
COSTBC2                         000001B0 R      02
COSTBR1                         000000B8 R      02
COSTBR2                         000000F8 R      02
CVT                             00000A1D R      02
CVT_LONG_TO_H                   00000910 R      02
DEGENERATE_CASE_COS             00000570 R      02
DEGENERATE_CASE_SIN             000004E6 R      02
EVAL_COSD                       0000065E R      02
EVAL_SIND                       00000605 R      02
EVEN                            00000A21 R      02
GEN_MORE_BITS                   000008CA R      02
G_1_OV_45                       00000048 R      02
G_3_PI_OV_4                     00000018 R      02
G_45                            00000030 R      02
G_5_PI_OV_4                     00000020 R      02
G_7_PI_OV_4                     00000028 R      02
G_90_OV_PI                      00000058 R      02
G_9_PI_OV_4                     00000010 R      02
G_CONVERT                       00000050 R      02
G_M1                            00000000 R      02
G_M45                           00000038 R      02
G_PI_OV_4                       00000008 R      02
G_SMALLD                        00000040 R      02
G_SMALLEST_DEG                  00000060 R      02
H_2_OV_PI                       000000A8 R      02
H_2_PI                          00000098 R      02
H_2_TO_32                       000006F0 R      02
H_3_PI_OV_2                     00000088 R      02
H_PI                            00000078 R      02
H_PI_OV_2                       00000068 R      02
LARGE_COS                       00000552 R      02
LARGE_SIN                       000004C8 R      02
LARGE_SINCOS                    00000445 R      02
LAST_STEP                       000009C9 R      02
LEADING_ONES                    00000838 R      02
LEADING_ZEROS                   00000889 R      02
LESS_THAN_HALF                  00000A64 R      02
LONG                          = 00000004
LOOP_0                          0000088C R      02
LOOP_1                          00000838 R      02
L_COS                           00000558 R      02
L_INT_WEIGHT                  = 0000003D
L_SIN                           000004CE R      02
MTH$$JACKET_HND                 ********   X    02
MTH$$JACKET_TST                 ********   X    00
MTH$$SIGNAL                     ********   X    00
MTH$AL_4_OV_PI_V                ********   X    00
MTH$GCOS                        00000312 RG     02
MTH$GCOSD                       00000359 RG     02
MTH$GCOSD_R7                    00000648 RG     02
MTH$GCOS_R7                     000004FA RG     02
MTH$GSIN                        000002FD RG     02
MTH$GSINCOS                     000002E0 RG     02
MTH$GSINCOSD                    00000327 RG     02
MTH$GSINCOSD_R7                 00000584 RG     02
MTH$GSINCOS_R7                  0000036E RG     02
MTH$GSIND                       00000344 RG     02
MTH$GSIND_R7                    000005E2 RG     02
MTH$GSIN_R7                     0000046B RG     02
MTH$K_FLOUNDMAT                 ********     X  00
M_COS                           00000518 R      02
M_SIN                           00000496 R      02
NEEDS_DOUBLE                    00000A3C R      02
NEEDS_DOUBLE_SINCOS             00000A43 R      02
NEG_SIND                        000005F2 R      02
NO_OVERFLOW                     000009F8 R      02
N_COS_C                         00000B2E R      02
N_COS_D                         00000C13 R      02
N_COS_R                         00000A74 R      02
N_ONE                           00000CA3 R      02
N_SIN_C                         00000B7B R      02
N_SIN_D                         00000C7E R      02
N_SIN_R                         00000ABD R      02
ODD                             00000A0E R      02
POS_SIN                         00000480 R      02
POS_SINCOS                      00000384 R      02
POS_SIND                        000005F7 R      02
POWER_MOD_360_0                 00000944 R      02
POWER_MOD_360_1                 0000095C R      02
PSL$M_IV                      = 00000020
P_COS_C                         00000AE9 R      02
P_COS_D                         00000BAF R      02
P_COS_R                         00000A30 R      02
P_ONE                           00000C9E R      02
P_SIN_C                         00000B80 R      02
P_SIN_D                         00000C82 R      02
P_SIN_R                         00000AC2 R      02
REDUCE_DEGREES                  00000974 R      02
REDUCE_LARGE                    00000700 R      02
REDUCE_MEDIUM                   00000685 R      02
RESTORE                         000008C3 R      02
RETURN                          00000C9D R      02
SF$W_SAVE_PSW                 = 00000004
SIN                             0000047B R      02
SINCOS                          0000037F R      02
SINCOSD                         00000599 R      02
SIND                          = 00000008
SINDLN                        = 00000006
SINDTB                          000002A8 R      02
SINE                          = 00000008
```

```
SINLENC                         = 00000007
SINLENR                         = 00000007
SINTBC                            000001F0 R      02
SINTBR                            00000138 R      02
SMALL_COS                         0000052C R      02
SMALL_COSD                        00000672 R      02
SMALL_SIN                         000004AA R      02
SMALL_SINCOS                      000003BD R      02
SMALL_SINCOSD                     C00005C5 R      02
SMALL_SIND                        00000619 R      02
UNFL                              00000CAA R      02
W_ADJUST                        = 00000386
W_MAX_WEIGHT                    = 00000400
W_TERM_WEIGHT                   = 00000020
X                               = 00000004
X_1_OV_45                       = 0000C16C
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_MTH$CODE	00000CCE (3278.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.06	00:00:01.12
Command processing	109	00:00:00.63	00:00:03.25
Pass 1	203	00:00:05.64	00:00:14.74
Symbol table sort	0	00:00:00.27	00:00:00.49
Pass 2	309	00:00:03.82	00:00:12.50
Symbol table output	16	00:00:00.13	00:00:00.67
Psect synopsis output	2	00:00:00.02	00:00:00.10
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	670	00:00:10.60	00:00:32.90

```
The working set limit was 1500 pages.
37048 bytes (73 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 196 non-local and 50 local symbols.
1790 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
_255DUA28:[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$:MTHGSINCO/OBJ-OBJ$:MTHGSINCO MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MS