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

M 10

FILEID**UVXGSINCO

```
UU      UU  VV      VV  XX      XX    GGGGGGGG    SSSSSSSS    IIIIII    NN      NN    CCCCCCCC    OOOOOO
UU      UU  VV      VV  XX      XX    GGGGGGGG    SSSSSSSS    IIIIII    NN      NN    CCCCCCCC    OOOOOO
UU      UU  VV      VV  XX      XX  GG          SS              II      NN      NN  CC          OO      OO
UU      UU  VV      VV  XX      XX  GG          SS              II      NN      NN  CC          OO      OO
UU      UU  VV      VV    XX  XX    GG          SS              II      NNNN    NN  CC          OO      OO
UU      UU  VV      VV    XX  XX    GG          SS              II      NNNN    NN  CC          OO      OO
UU      UU  VV      VV      XX      GG            SSSSSS        II      NN  NN  NN  CC          OO      OO
UU      UU  VV      VV      XX      GG            SSSSSS        II      NN  NN  NN  CC          OO      OO
UU      UU  VV      VV    XX  XX    GG  GGGGGG          SS      II      NN    NNNN  CC          OO      OO
UU      UU  VV      VV    XX  XX    GG  GGGGGG          SS      II      NN    NNNN  CC          OO      OO
UU      UU    VV  VV    XX      XX  GG      GG          SS      II      NN      NN  CC          OO      OO      ....
UU      UU    VV  VV    XX      XX  GG      GG          SS      II      NN      NN  CC          OO      OO      ....
UUUUUUUUUU      VV      XX      XX    GGGGGG    SSSSSSSS      IIIIII    NN      NN    CCCCCCCC    OOOOOO        ....
UUUUUUUUUU      VV      XX      XX    GGGGGG    SSSSSSSS      IIIIII    NN      NN    CCCCCCCC    OOOOOO        ....

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

Table of contents

```
0000          .TITLE  UVX$GSINCOS      ; Floating Point Sine, Cosine and Sincos
0000                                   ;     Functions
0000          .IDENT /2-004/           ; File: MTHGSINCOS.MAR  EDIT: JCW2004
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;
0000    ; FACILITY:     MATH LIBRARY
0000    ;++
0000    ; ABSTRACT:
0000    ;
0000    ; MTH$GSIN and MTH$GCOS are functions which return the floating point
0000    ; sine or cosine value of their single precision floating point argu-
0000    ; ment (radians). The call is standard call-by-reference.
0000    ; MTH$GSIN_R7 and MTH$GCOS_R7 are special routines which are the same
0000    ; as MTH$GSIN and MTH$GCOS  except  a faster non-standard JSB call is
0000    ; used with the argument in R0 and no registers are saved.
0000    ;
0000    ; MTH$GSINCOS  is  a routine which returns the floating point sine and
0000    ; cosine value of its single precision floating point radian argument.
0000    ; The call is standard call-by-reference.  MTH$GSINCOS_R7 is a special
0000    ; routine  which  is  the  same  as  MTH$GSINCOS, except a faster non-
0000    ; standard JSB call is used with the argument in R0 and no  registers
0000    ; are saved.
0000    ;
0000    ; MTH$GSIND and MTH$GCOSD are functions which return the floating point
0000    ; sine  or  cosine value of their single precision floating point argu-
0000    ; ment (degrees). The call is standard call-by-reference.
0000    ; MTH$GSIND_R7 and MTH$GCOSD_R7 are special routines which are the same
0000    ; as MTH$GSIND and MTH$GCOSD  except  a faster non-standard JSB call is
0000    ; used with the argument in R0 and no registers are saved.
0000    ;
0000    ; MTH$GSINCOSD is  a routine which returns the floating point sine and
0000    ; cosine value of its single precision floating point degree argument.
0000    ; The call is standard call-by-reference. MTH$GSINCOSD_R7 is a special
0000    ; routine  which  is  the  same  as MTH$GSINCOSD, except a faster non-
```

```
0000    ; standard JSB call is used with the argument in R0 and no  registers
0000    ; are saved.
0000    ;
0000    ;--
0000    ;
0000    ; VERSION:       1
0000    ;
0000    ; HISTORY:
0000    ; AUTHOR:
0000    ;       MARY PAYNE & JUD LEONARD, 25-MAY-78:    Version 0
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ; 1-1   Tryggve Fossum, 28-May-78
0000    ;
0000    ;
0000    ; VERSION:       2
0000    ;
0000    ; HISTORY:
0000    ; AUTHOR:
0000    ;       BOB HANEK, 25-MAY-78:   Version 2
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ; 2-004 Jeffrey C. Wiener 1-APR-83
0000    ;
0000    ; Edit history for Version 2:
0000    ;
0000    ; 2-001 - Original
0000    ; 2-002 - Change MTH$AL_4_OV_PI to MTH$AL_4_OV_PI_V.  RNH 29-Sep-81
0000    ; 2-003 - Modified REDUCE_LARGE to correct bug reported in QAR 896.
0000    ;         RNH 14-Jan-82
0000    ; 2-004 - This code is a modified version of MTH$DSINCOS. The need to
0000    ;         modify the previous version of MTH$GSINCOS was brought about by
0000    ;         the needs of microVAX. G_floating routines are not permitted to
0000    ;         use H_floating point instructions. JCW 1-APR-83.
```

```
                            0000        .SBTTL  DECLARATIONS    -        Declarative Part of Module
                            0000
                            0000  ;
                            0000  ; INCLUDE FILES:         MTH$JACKET.MAR
                            0000
                            0000  ; EXTERNAL SYMBOLS:
                            0000  ;
                            0000        .DSABL  GBL
                            0000        .EXTRN  MTH$AL_4_OV_PI_V
                            0000        .EXTRN  MTH$$SIGNAL
                            0000        .EXTRN  MTH$K_FLOUNDMAT
                            0000        .EXTRN  MTH$$JACKET_TST
                            0000  ;
                            0000  ; EQUATED SYMBOLS:
                            0000
                   0000C16C 0000        X_1_OV_45          = ^XC16C
                            0000
                            0000  ;
                            0000  ; MACROS:
                            0000
                            0000        $SFDEF                              ; Define SF$ (stack frame) symbols
                            0000        $PSLDEF                             ; Define PSL$ symbols
                            0000
                            0000  ; PSECT DECLARATIONS:
                            0000
                        00000000        .PSECT  _MTH$CODE        PIC,SHR,LONG,EXE,NOWRT
                            0000                                 ; program section for math routines
                            0000  ;
                            0000  ; OWN STORAGE:  none
                            0000  ;
                            0000  ; CONSTANTS:
                            0000
                            0000  G_M1:
          00000000 0000C010 0000        .QUAD   ^X000000000000C010       ; -1
                            0008  G_PI_OV_4:
          2D185444 21FB4009 0008        .QUAD   ^X2D18544421FB4009       ; 0.7853981633974483E+00
                            0010  G_9_PI_OV_4:
          B2BBBECC 463A403C 0010        .QUAD   ^XB2BBBECC463A403C       ; 0.7068583470577035E+01
                            0018  G_3_PI_OV_4:
          21D27F33 D97C4022 0018        .QUAD   ^X21D27F33D97C4022       ; 0.2356194490192345E+01
                            0020  G_5_PI_OV_4:
          385E2955 6A7A402F 0020        .QUAD   ^X385E29556A7A402F       ; 0.3926990816987241E+01
                            0028  G_7_PI_OV_4:
          A775E9BB FDBB4035 0028        .QUAD   ^XA775E9BBFDBB4035       ; 0.5497787143782138E+01
                            0030  G_2_OV_PI:
          C8826DC9 5F304004 0030        .QUAD   ^XC8826DC95F304004       ; 0.6366197723675813E+00
                            0038
                            0038  G_45:
          00000000 80004066 0038        .QUAD   ^X0000000080004066       ; 0.4500000000000000E+02
                            0040  G_M45:
          00000000 8000C066 0040        .QUAD   ^X000000008000C066       ; -.4500000000000000E+02
                            0048  G_SMALLD:
          C1F81A63 A5DC3EBC 0048        .QUAD   ^XC1F81A63A5DC3EBC       ; 0.4268868231257969E-06
                            0050  G_1_OV_45:
          6C1716C1 C16C3FB6 0050        .QUAD   ^X6C1716C1C16C3FB6       ; 0.2222222222222222E-01
                            0058  G_CONVERT:
          D3912529 F46A3F7D 0058        .QUAD   ^XD3912529F46A3F7D       ; 0.1828292519943295E-02
```

```
                     0060  G_90_OV_PI:
C1F81A63 A5DC405C    0060          .QUAD   ^XC1F81A63A5DC405C      ; 0.2864788975654116E+02
                     0068  G_SMALLEST_DEG:
C1F81A63 A5DC006C    0068          .QUAD   ^XC1F81A63A5DC006C      ; 0.3187183529933799-306
                     0070
                     0070
                     0070  PI_OV_2:
                     0070  ; pi/2
00005000 21FB4019    0070          .QUAD   ^X0000500021FB4019      ; 0.1570796326794897E+01
00006000 10B43E71    0078          .QUAD   ^X0000600010B43E71
00003000 A6263CB1    0080          .QUAD   ^X00003000A6263CB1      ; 0.5721188726109832E-17
00003000 A2E03B08    0088          .QUAD   ^X00003000A2E03B08
0000A000 7344396C    0090          .QUAD   ^X0000A0007344396C      ; 0.4335905065061890E-34
00008800 24E037C0    0098          .QUAD   ^X0000880024E037C0
                     00A0  ; pi
00005000 21FB4029    00A0          .QUAD   ^X0000500021FB4029      ; 0.3141592653589793E+01
00006100 10B43E81    00A8          .QUAD   ^X0000610010B43E81
00003000 62633C8A    00B0          .QUAD   ^X0000300062633C8A      ; 0.1144237745221966E-16
0000E000 5C063AC4    00B8          .QUAD   ^X0000E0005C063AC4
00000000 D12938FC    00C0          .QUAD   ^X00000000D12938FC      ; 0.8671810130123781E-34
00005300 27043744    00C8          .QUAD   ^X0000530027043744
                     00D0
                     00D0  ; 3*pi/2
00007800 D97C4032    00D0          .QUAD   ^X00007800D97C4032      ; 0.4712388980384690E+01
00004800 CC873E9C    00D8          .QUAD   ^X00004800CC873E9C
00004800 79393CCA    00E0          .QUAD   ^X0000480079393CCA      ; 0.3834758505292833E-16
00002800 7A283B22    00E8          .QUAD   ^X000028007A283B22
0000E800 59CD3965    00F0          .QUAD   ^X0000E80059CD3965      ; 0.1300771519518567E-33
00005000 6EA137A0    00F8          .QUAD   ^X000050006EA137A0
                     0100  ;2*pi
00005000 21FB4039    0100          .QUAD   ^X0000500021FB4039      ; 0.6283185307179586E+01
00006100 10B43E91    0108          .QUAD   ^X0000610010B43E91
00003000 62633C9A    0110          .QUAD   ^X0000300062633C9A      ; 0.2288475490443933E-16
0000E000 5C063AD4    0118          .QUAD   ^X0000E0005C063AD4
00000000 D129390C    0120          .QUAD   ^X00000000D129390C      ; 0.1734362026024756E-33
00005300 70443752    0128          .QUAD   ^X0000530070443752
                     0130
```

```
                   0130      .SBTTL  COEFFICIENT TABLES      -        Series Coefficients
                   0130
                   0130
                   0130
                   0130
                   0130 ;
                   0130 ; Polynomial Coefficient tables for arguments in radians
                   0130 ;
                   0130
                   0130 COSTBR1:              ; GCOS coefficients for arguments less than 1/2
CD345EF0 F6ADBDC8  0130         .QUAD   ^XCD345EF0F6ADBDC8      ; C7 = -.1135212320578394E-10
C93D4799 EE953E41  0138         .QUAD   ^XC93D4799EE953E41      ; C6 = 0.2087555514567788E-08
19916FE2 7E4FBEB2  0140         .QUAD   ^X19916FE27E4FBEB2      ; C5 = -.2755731286569608E-06
767E19AD 01A03F1A  0148         .QUAD   ^X767E19AD01A03F1A      ; C4 = 0.2480158728289946E-04
362D16C1 C16CBF76  0150         .QUAD   ^X362D16C1C16CBF76      ; C3 = -.1388888888885896E-02
55335555 55553FC5  0158         .QUAD   ^X5533555555553FC5      ; C2 = 0.4166666666666643E-01
00000000 0000C000  0160         .QUAD   ^X000000000000C000      ; C1 = -.5000000000000000E+00
00000000 00004010  0168         .QUAD   ^X0000000000004010      ; C0 = 0.1000000000000000E+01
         00000008  0170 COSLENR1 = .-COSTBR1/8
                   0170
                   0170 COSTBR2:              ; GCOS coefficients for arguments greater than 1/2
CD345EF0 F6ADBDC8  0170         .QUAD   ^XCD345EF0F6ADBDC8      ; C7 = -.1135212320578394E-10
C93D4799 EE953E41  0178         .QUAD   ^XC93D4799EE953E41      ; C6 = 0.2087555514567788E-08
19916FE2 7E4FBEB2  0180         .QUAD   ^X19916FE27E4FBEB2      ; C5 = -.2755731286569608E-06
767E19AD 01A03F1A  0188         .QUAD   ^X767E19AD01A03F1A      ; C4 = 0.2480158728289946E-04
362D16C1 C16CBF76  0190         .QUAD   ^X362D16C1C16CBF76      ; C3 = -.1388888888885896E-02
55335555 55553FC5  0198         .QUAD   ^X5533555555553FC5      ; C2 = 0.4166666666666643E-01
5EE62C40 80673C80  01A0         .QUAD   ^X5EE62C4080673C80      ; C1 = 0.7156417079102195E-17
5D30B344 297FBC05  01A8         .QUAD   ^X5D30B344297FBC05      ; C0 = -.3584999999999999E-19
         00000008  01B0 COSLENR2 = .-COSTBR2/8
                   01B0
                   01B0 SINTBR:               ; GSIN coefficients
B5E36E13 D8403E05  01B0         .QUAD   ^XB5E36E13D8403E05      ; C6 = 0.1589413523004633E-09
3319B5B3 E5E2BE7A  01B8         .QUAD   ^X3319B5B3E5E2BE7A      ; C5 = -.2505070582636817E-07
5AD851F3 1DE33EE7  01C0         .QUAD   ^X5AD851F31DE33EE7      ; C4 = 0.2755731329888568E-05
2F3C19B9 01A0BF4A  01C8         .QUAD   ^X2F3C19B901A0BF4A      ; C3 = -.1984126982840175E-03
F30C1110 11113FA1  01D0         .QUAD   ^XF30C111011113FA1      ; C2 = 0.8333333333320002E-02
55435555 5555BFE5  01D8         .QUAD   ^X554355555555BFE5      ; C1 = -.1666666666666662E+00
033809DB 8D82BC6E  01E0         .QUAD   ^X 33809DB8D82BC6E      ; C0 = -.3312537470886997E-17
         00000007  01E8 SINLENR = .-SINTBR/8
                   01E8
                   01E8
                   01E8
                   01E8
                   01E8
                   01E8 ;
                   01E8 ; Polynomial coefficients for arguments in cycles
                   01E8 ;
                   01E8
                   01E8 COSTBC1:              ; GCOS coefficients for arguments less than 2/pi
815BB995 2586BD7B  01E8         .QUAD   ^X815BB9952586BD7B      ; C7 = -.3857762037200000E-12
4DE15D51 9CC13DFF  01F0         .QUAD   ^X4DE15D519CC13DFF      ; C6 = 0.1150049702426300E-09
8016C3D4 6D1EBE7A  01F8         .QUAD   ^X8016C3D46D1EBE7A      ; C5 = -.2461136382637005E-07
3F246830 1F503EEE  0200         .QUAD   ^X3F2468301F503EEE      ; C4 = 0.3590860445885820E-05
8D5C7E3C 5D3CBF55  0208         .QUAD   ^X8D5C7E3C5D3CBF55      ; C3 = -.3259918869266876E-03
5AAA081B 3C1F3FB0  0210         .QUAD   ^X5AAA081B3C1F3FB0      ; C2 = 0.1585434424381541E-01
```

```
45DEC9BE BD3CBFF3  0218                .QUAD   ^X45DEC9BEBD3CBFF3        ; C1 = -.3084251375340424E+00
5D30B344 297FBC05  0220                .QUAD   ^X5D30B344297FBC05        ; C0 = -.3584999999999999E-19
         00000008  0228     COSLENC1 = .-COSTBC1/8
                   0228
                   0228     COSTBC2:           ; GCOS coefficients for arguments greater than 2/pi
815BB995 2586BD7B  0228                .QUAD   ^X815BB9952586BD7B        ; C7 = -.3857762037200000E-12
4DE15D51 9CC13DFF  0230                .QUAD   ^X4DE15D519CC13DFF        ; C6 = 0.1150049702426300E-09
8016C3D4 6D1EBE7A  0238                .QUAD   ^X8016C3D46D1EBE7A        ; C5 = -.2461136382637005E-07
3F246830 1F503EEE  0240                .QUAD   ^X3F2468301F503EEE        ; C4 = 0.3590860445885820E-05
8D5C7E3C 5D3CBF55  0248                .QUAD   ^X8D5C7E3C5D3CBF55        ; C3 = -.3259918869266876E-03
5AAA081B 3C1F3FB0  0250                .QUAD   ^X5AAA081B3C1F3FB0        ; C2 = 0.1585434424381541E-01
2EF24DF2 E9E6BFCD  0258                .QUAD   ^X2EF24DF2E9E6BFCD        ; C1 = -.5842513753404245E-01
5D30B344 297FBC05  0260                .QUAD   ^X5D30B344297FBC05        ; C0 = -.3584999999999999E-19
         00000008  0268     COSLENC2 = .-COSTBC2/8
                   0268
                   0268     SINTBC:            ; GSIN coef for arg in cycles
CB82386D 3EED3DBE  0268                .QUAD   ^XCB82386D3EED3DBE        ; C6 = 0.6877100349000000E-11
0141399B 3006BE3E  0270                .QUAD   ^X 141399B3006BE3E        ; C5 = -.1757149292755000E-08
1EF8FCA3 07823EB5  0278                .QUAD   ^X1EF8FCA307823EB5        ; C4 = 0.3133616216619040E-06
52FACE2D 2D2CBF23  0280                .QUAD   ^X52FACE2D2D2CBF23        ; C3 = -.3657620415845570E-04
86FF6775 66BC3F84  0288                .QUAD   ^X86FF677566BC3F84        ; C2 = 0.2490394570188736E-02
BE41E625 ABBCBFD4  0290                .QUAD   ^XBE41E625ABBCBFD4        ; C1 = -.8074551218828054E-01
D1844442 1FB53FC2  0298                .QUAD   ^XD18444421FB53FC2        ; C0 = 0.3539816339744831E-01
         00000007  02A0     SINLENC = .-SINTBC/8
                   02A0
                   02A0
                   02A0
                   02A0
                   02A0
                   02A0     ;
                   02A0     ; Polynomial coefficients for arguments in degrees
                   02A0     ;
                   02A0
                   02A0     COSDTB2:           ; GCOS coefficients for arguments less than 90/pi
20197263 613EB8AD  02A0                .QUAD   ^X20197263613EB8AD        ; C7 = -.2762868673216389E-35
ADD69711 EA1439E0  02A8                .QUAD   ^XADD69711EA1439E0        ; C6 = 0.1667886312398853E-29
B183A296 F623BB0B  02B0                .QUAD   ^XB183A296F623BB0B        ; C5 = -.7227873495985314E-24
D1DD5C04 83AB3C2F  02B8                .QUAD   ^XD1DD5C0483AB3C2F        ; C4 = 0.2135494301985904E-18
78AF5BBC 19B8BD46  02C0                .QUAD   ^X78AF5BBC19B8BD46        ; C3 = -.3925831985734635E-13
DC736A83 9B113E50  02C8                .QUAD   ^XDC736A839B113E50        ; C2 = 0.3866323851562971E-08
1FB9DB14 F6A1BF43  02D0                .QUAD   ^X1FB9DB14F6A1BF43        ; C1 = -.1523087098933543E-03
00000000 00004010  02D8                .QUAD   ^X0000000000004010        ; C0 = 0.1000000000000000E+01
         00000007  02E0     COSDLN2 = .-COSDTB2/8 - 1
                   02E0
                   02E0     COSDTB1:           ; GCOS coefficients for arguments greater than 90/pi
20197263 613EB8AD  02E0                .QUAD   ^X20197263613EB8AD        ; C7 = -.2762868673216389E-35
ADD69711 EA1439E0  02E8                .QUAD   ^XADD69711EA1439E0        ; C6 = 0.1667886312398853E-29
B183A296 F623BB0B  02F0                .QUAD   ^XB183A296F623BB0B        ; C5 = -.7227873495985314E-24
D1DD5C04 83AB3C2F  02F8                .QUAD   ^XD1DD5C0483AB3C2F        ; C4 = 0.2135494301985904E-18
78AF5BBC 19B8BD46  0300                .QUAD   ^X78AF5BBC19B8BD46        ; C3 = -.3925831985734635E-13
DC736A83 9B113E50  0308                .QUAD   ^XDC736A839B113E50        ; C2 = 0.3866323851562971E-08
FDCCD8A0 B50EBF1F  0310                .QUAD   ^XFDCCD8A0B50EBF1F        ; C1 = -.3023839739335430E-04
5D30B344 297FBC05  0318                .QUAD   ^X5D30B344297FBC05        ; C0 = -.3584999999999999E-19
         00000007  0320     COSDLN1 = .-COSDTB1/8 - 1
                   0320
                   0320     SINDTB:            ; GSIN coefficients
17E0B735 04203947  0320                .QUAD   ^X17E0B73504203947        ; C6 = 0.2216372140147286E-32
```

```
C8145B97 B6BEBA76  0328   304           .QUAD   ^XC8145B97B6BEBA76      ; C5 = -.1146755972041862E-26
85EFDBD8 4A5F3B9F  0330   305           .QUAD   ^X85EFDBD84A5F3B9F      ; C4 = 0.4141266526843263E-21
BA169F5A 368DBCBC  0338   306           .QUAD   ^XBA169F5A368DBCBC      ; C3 = -.9788384855269454E-16
D93DEAE0 AD943DCD  0340   307           .QUAD   ^XD93DEAE0AD943DCD      ; C2 = 0.1349601623161096E-10
2F5E0D94 BB82BECD  0348   308           .QUAD   ^X2F5E0D94BB82BECD      ; C1 = -.8860961557012952E-06
D3912529 F46A3F7D  0350   309           .QUAD   ^XD3912529F46A3F7D      ; C0 = 0.1828292519943296E-02
         00000006  0358   310 SINDLN = .-SINDTB/8 - 1
                   0358   311
```

```
          0358        .SBTTL  MTH$GSINCOS     -       Radian arguments
          0358
          0358
          0358  ;
          0358  ; FUNCTIONAL DESCRIPTION:
          0358  ;
          0358  ; The GSIN, GCOS and GSINCOS routines are based on octant reduction.  Given an
          0358  ; argument, x, it is written in the form
          0358  ;
          0358  ;               x = I1*(2*pi) + I*(pi/4) + Y1,
          0358  ;
          0358  ; where I1 and I are integers, 0 =< I < 8 and 0 =< Y1 < pi/4.  Since GSIN and
          0358  ; GCOS have a period of 2*pi it follows that
          0358  ;
          0358  ;               GSIN(x) = GSIN(I*(pi/4) + Y1)) and
          0358  ;               GCOS(x) = GCOS(I*(pi/4) + Y1)).
          0358  ;
          0358  ; Using the trigonometric identities for the sum and difference of two angles,
          0358  ; the following table can be generated:
          0358  ;
          0358  ;       If I =          then GSIN(x) =          and GCOS(x) =
          0358  ;       ------          -------------           -------------
          0358  ;         0               GSIN(Y1)                GCOS(Y1)
          0358  ;         1               GCOS(pi/4-Y1)           GSIN(pi/4-Y1)
          0358  ;         2               GCOS(Y1)               -GSIN(Y1)
          0358  ;         3               GSIN(pi/4-Y1)          -GCOS(pi/4-Y1)
          0358  ;         4              -GSIN(Y1)               -GCOS(Y1)
          0358  ;         5              -GCOS(pi/4-Y1)          -GSIN(pi/4-Y1)
          0358  ;         6              -GCOS(Y1)                GSIN(Y1)
          0358  ;         7              -GSIN(pi/4-Y1)           GCOS(pi/4-Y1)
          0358  ;
          0358  ; Let Y be defined as Y = Y1 if I is even and Y = pi/4 - Y1, if I is odd, then
          0358  ; each entry of the above table is of the for +/-GSIN(Y) or +/-GCOS(Y).  Based
          0358  ; on the above remarks, the GSIN, GCOS and GSINCOS routines process the input
          0358  ; argument x, to obtain I and Y, and based on I selects a suitable polynomial
          0358  ; approximation, p(Y), to evaluate the desired fuction.
          0358  ;
          0358  ;
          0358  ; INPUT PARAMETERS:
          0358  ;
00000004  0358        LONG   = 4
00000004  0358        x      = 1*LONG          ; x is input angle in radians
00000008  0358        sine   = 2*LONG          ; sine is GSIN(x)
0000000C  0358        cosine = 3*LONG          ; cosine is GCOS(x)
          0358
```

```
                               0358
                               0358
                               0358    ;
                               0358    ; Return sine and cosine of argument
                               0358    ;
                               0358
                               0358
                         00FC  0358            .ENTRY  MTH$GSINCOS, ^M<R2, R3, R4, R5, R6,R7>
                               035A
                               035A            MTH$FLAG_JACKET
                               035A
     6D     00000000'GF    9E  035A            MOVAB   G^MTH$$JACKET_HND, (FP)
                               0361                                            ; set handler address to jacket
                               0361                                            ; handler
                               0361
                               0361
      50    04 BC  50FD        0361            MOVG    @x(AP), R0
        000003E6'EF    16      0366            JSB     MTH$GSINCOS_R7
        08 BC    50    7D      036C            MOVQ    R0, @sine(AP)
        0C BC    52    7D      0370            MOVQ    R2, @cosine(AP)
                       04      0374            RET
                               0375
                               0375
                               0375
                               0375            .SBTTL  MTH$GSIN
                               0375
                               0375    ;
                               0375    ; Return sine of argument
                               0375    ;
                               0375
                               0375
                         00FC  0375            .ENTRY  MTH$GSIN, ^M<R2, R3, R4, R5, R6, R7>
                               0377
                               0377            MTH$FLAG_JACKET
                               0377
     6D     00000000'GF    9E  0377            MOVAB   G^MTH$$JACKET_HND, (FP)
                               037E                                            ; set handler address to jacket
                               037E                                            ; handler
                               037E
                               037E
      50    04 BC  50FD        037E            MOVG    @x(AP), R0
        000004FB'EF    16      0383            JSB     MTH$GSIN_R7
                       04      0389            RET
                               038A
                               038A
                               038A
                               038A            .SBTTL  MTH$GCOS
                               038A
                               038A    ;
                               038A    ; Return cosine of argument
                               038A    ;
                               038A
                               038A
                         00FC  038A            .ENTRY  MTH$GCOS, ^M<R2, R3, R4, R5, R6, R7>
                               038C
                               038C            MTH$FLAG_JACKET
                               038C
```

```
      6D  00000000'GF    9E  038C          MOVAB   G^MTH$$JACKET_HND, (FP)
                             0393                                         ; set handler address to jacket
                             0393                                         ; handler
                             0393
                             0393
         50   04 BC  50FD  0393          MOVG    @x(AP), R0
          0000058A'EF   16  0398          JSB     MTH$GCOS_R7
                         04  039E          RET
                             039F
```

```
          039F          .SBTTL  MTH$GSINCOSD    -       Degrees
          039F
          039F
          039F  ;
          039F  ; FUNCTIONAL DESCRIPTION:
          039F  ;
          039F  ; The GSIND, GCOSD and GSINCOSD routines are based on octant reduction.  Given
          039F  ; an argument, x, it is written in the form
          039F  ;
          039F  ;               x = I1*360 + I*45 + Y1,
          039F  ;
          039F  ; where I1 and I are integers, 0 =< I < 8 and 0 =< Y1 < 45.  Since GSIND and
          039F  ; GCOSD have a period of 360 it follows that
          039F  ;
          039F  ;               GSIND(x) = GSIND(I*45 + Y1) and
          039F  ;               GCOSD(x) = GCOSD(I*45 + Y1).
          039F  ;
          039F  ; Using the trigonometric identities for the sum and difference of two angles,
          039F  ; the following table can be generated:
          039F  ;
          039F  ;       If I =          then GSIND(x) =                 and GCOSD(x) =
          039F  ;       ------          -------------                   -------------
          039F  ;         0               GSIND(Y1)                       GCOSD(Y1)
          039F  ;         1               GCOSD(45-Y1)                    GSIND(45-Y1)
          039F  ;         2               GCOSD(Y1)                      -GSIND(Y1)
          039F  ;         3               GSIND(45-Y1)                   -GCOSD(45-Y1)
          039F  ;         4              -GSIND(Y1)                      -GCOSD(Y1)
          039F  ;         5              -GCOSD(45-Y1)                   -GSIND(45-Y1)
          039F  ;         6              -GCOSD(Y1)                       GSIND(Y1)
          039F  ;         7              -GSIND(45-Y1)                    GCOS(45-Y1)
          039F  ;
          039F  ; Let Y be defined as Y = Y1 if I is even and Y = 45 - Y1, if I is odd, then
          039F  ; each entry of the above table is of the for +/-GSIN(Y) or +/-GCOS(Y).  Based
          039F  ; on the above remarks, the GSIND, GCOSD and GSINCOSD routines process the input
          039F  ; argument x, to obtain I and Y, and based on I selects a suitable polynomial
          039F  ; approximation, p(Y), to evaluate the desired fuction.
          039F
          039F
00000004  039F          LONG = 4
00000008  039F          sind = 2*LONG
0000000C  039F          cosd = 3*LONG
          039F
```

```
                       00FC  039F  454        .ENTRY  MTH$GSINCOSD    ^M<R2, R3, R4, R5, R6, R7>
                             03A1  455
                             03A1  456        MTH$FLAG_JACKET
                             03A1
      6D   00000000'GF   9E  03A1             MOVAB   G^MTH$$JACKET_HND, (FP)
                             03A8                                                     ; set handler address to jacket
                             03A8                                                     ; handler
                             03A8
                             03A8  457
        50    04 BC 50FD     03A8  458        MOVG    @X(AP), R0
      0000061F'EF     16     03AD  459        JSB     MTH$GSINCOSD_R7
        08 BC     50  7D     03B3  460        MOVQ    R0, @sind(AP)
        0C BC     52  7D     03B7  461        MOVQ    R2, @cosd(AP)
                             03BB  462
                         04  03BB  463        RET
                             03BC  464
                             03BC  465
                             03BC  466
                       00FC  03BC  467        .ENTRY  MTH$GSIND       ^M<R2, R3, R4, R5, R6, R7>
                             03BE  468
                             03BE  469        MTH$FLAG_JACKET
                             03BE
      6D   00000000'GF   9E  03BE             MOVAB   G^MTH$$JACKET_HND, (FP)
                             03C5                                                     ; set handler address to jacket
                             03C5                                                     ; handler
                             03C5
                             03C5  470
        50    04 BC 50FD     03C5  471        MOVG    @X(AP), R0
      0000067D'EF     16     03CA  472        JSB     MTH$GSIND_R7
                             03D0  473
                         04  03D0  474        RET
                             03D1  475
                             03D1  476
                             03D1  477
                       00FC  03D1  478        .ENTRY  MTH$GCOSD       ^M<R2, R3, R4, R5, R6, R7>
                             03D3  479
                             03D3  480        MTH$FLAG_JACKET
                             03D3
      6D   00000000'GF   9E  03D3             MOVAB   G^MTH$$JACKET_HND, (FP)
                             03DA                                                     ; set handler address to jacket
                             03DA                                                     ; handler
                             03DA
                             03DA  481
        50    04 BC 50FD     03DA  482        MOVG    @X(AP), R0
      000006E3'EF     16     03DF  483        JSB     MTH$GCOSD_R7
                             03E5  484
                         04  03E5  485        RET
                             03E6  486
```

```
                        03E6        .SBTTL MTH$GSINCOS_R7
                        03E6
                        03E6  ; This routine computes the GSIN and GCOS of the G-format value of R0/R1.  The
                        03E6  ; computation is performed one of three ways depending on the size of the
                        03E6  ; input argument, X:
                        03E6  ;
                        03E6  ;     1) If |X| < pi/4, then X is used directly in polynomial approximation
                        03E6  ;        of GSIN and GCOS.
                        03E6  ;     2) If pi/4 =< |x| < 9*pi/4, then the subroutine REDUCE_MEDIUM is called
                        03E6  ;        to reduce the argument to an equivalent argument in radians, Y, and
                        03E6  ;        the octant, I, containing the argument.  Y is then evaluated in two
                        03E6  ;        polynomials chosen as a function of I, to compute GSIN(X) and GCOS(X).
                        03E6  ;     3) If 9*pi/4 =< |X|, then the subroutine REDUCE_LARGE is called to
                        03E6  ;        reduce the argument to an equivalent argument in cycles, Y, and the
                        03E6  ;        octant, I, containing the argument.  Y is then evaluated in two
                        03E6  ;        polynomials chosen as a function of I, to compute GSIN(X) and GCOS(X).
                        03E6
                        03E6  MTH$GSINCOS_R7::
       56    50 50FD    03E6          MOVG    R0, R6                  ; R6 = X
             10   18    03EA          BGEQ    POS_SINCOS              ;
  000003F7'EF     16    03EC          JSB     SINCOS                  ; R0/R1 = GSIN(|X|), R2/R3 = GCOS(X)
       50    50 52FD    03F2          MNEGG   R0, R0                  ; R0/R1 = GSIN(X)
                  05    03F6          RSB
                        03F7
                        03F7  SINCOS:
  56    8000 8F   AA    03F7          BICW    #^X8000, R6             ; R6/R7 = |X|
                        03FC  POS_SINCOS:
  56    FC07 CF 51FD    03FC          CMPG    G_PI_OV_4, R6           ; Compare pi/4 with |X|
             35   14    0402          BGTR    SMALL_SINCOS            ; No argument reduction is necessary
  56    FC07 CF 51FD    0404          CMPG    G_9_PI_OV_4, R6         ; Compare 9*pi/4 with |X|
             03   18    040A          BGEQ    1$
           00C2   31    040C          BRW     LARGE_SINCOS            ; Use special logic for |X| > 9*pi/4
                        040F
                        040F  ;
                        040F  ;  pi/4 =< |X| < 9*pi/4
                        040F  ;
  00000720'EF     16    040F  1$:     JSB     REDUCE_MEDIUM           ; Medium argument reduction routine
                        0415                                          ; R4/R7 = Y = reduced argument
                        0415                                          ; R2 = octant
        7E   54   7D    0415          MOVQ    R4, -(SP)               ; Save reduced argument on stack
        7E   56   7D    0418          MOVQ    R6, -(SP)
             52   DD    041B          PUSHL   R2                      ; Save octant bits on stack
  000005A8'EF     16    041D          JSB     M_COS                   ; R0/R1 = GCOS(X)
        52   8E   D0    0423          MOVL    (SP)+, R2               ; R2 = Octant bits
        56   8E   7D    0426          MOVQ    (SP)+, R6               ;
        54   8E   7D    0429          MOVQ    (SP)+, R4               ; R4/R7 = reduced argument
        7E   50   7D    042C          MOVQ    R0, -(SP)               ; Save GCOS(X) on stack
  00000526'EF     16    042F          JSB     M_SIN                   ; R0/R1 = GSIN(X)
        52   8E   7D    0435          MOVQ    (SP)+, R2               ; R2/R3 = GCOS(X)
                  05    0438          RSB
                        0439  ;
                        0439  ; Logic for small arguments.  |X| < pi/4.
                        0439  ;
                        0439
                        0439  SMALL_SINCOS:
  56    4000 8F   B1    0439          CMPW    #^X4000, R6             ; Compare 1/2 with |X|
             39   15    043E          BLEQ    2$                      ; Sufficent overghang not available
```

```
      56   3E60 8F    B1  0440        CMPW    #^X3E60, R6                ; Compare with 2^-27
             28  18  0445        BGEQ    1$                         ; No polynomial evaluation is needed
      54   56  56 45FD  0447        MULG3   R6, R6, R4                 ; R4/R5 = X*X
           7E  54   7D  044C        MOVQ    R4, -(SP)                  ; Put X*X on stack
 FCDA CF   07  54 55FD  044F        POLYG   R4, #COSLENR1-1, COSTBR1; RO/R1 = GCOS(X)
           54  6E   7D  0456        MOVQ    (SP), R4                   ; R4/R5 = X*X
           6E  50   7D  0459        MOVQ    R0, (SP)                   ; Save GCOS(X) on stack
 FD4D CF   06  54 55FD  045C        POLYG   R4, #SINLENR-1, SINTBR     ; RO/R1 = g(X^2)
           50  56 44FD  0463        MULG2   R6, R0                     ; RO/R1 = X*g(X^2)
           50  56 40FD  0467        ADDG2   R6, R0                     ; RO/R1 = GSIN(X)
           52  8E   7D  046B        MOVQ    (SP)+, R2                  ; R2/R3 = GCOS(X)
                    05  046E        RSB
                        046F
           52  08 50FD  046F  1$:   MOVG    #1.0, R2                   ; R2/R3 = 1.0 = GCOS(X)
      50   8000 8F  AA  0473        BICW2   #^X8000, R0                ; RO/R1 = !X!
                    05  0478        RSB
                        0479
                        0479
      54   56  56 45FD  0479  2$:   MULG3   R6, R6, R4                 ; R4/R5 = X^2
           7E  54   7D  047E        MOVQ    R4, -(SP)                  ; Save X^2
 FCE8 CF   07  54 55FD  0481        POLYG   R4, #COSLENR2-1, COSTBR2; RO/R1 = Q(Y^2)
           54  56   7D  0488        MOVQ    R6, R4                     ; R4/R5 = X
  55   FFFF07FF 8F  CA  048B        BICL2   #^XFFFF07FF, R5            ; R4/R5 = XHI, first 26 signif bits
   7E      56  54 43FD  0492        SUBG3   R4, R6, -(SP)              ; (SP) = XLO, Last 27 bits
   52      54  56 41FD  0497        ADDG3   R6, R4, R2                 ; R2/R3 = X + XHI
           52  8E 44FD  049C        MULG2   (SP)+, R2                  ; R2/R3 = XLO*(X + XHI) = A2
               07   13  04A0        BEQL    3$                         ; Check for A2 = 0
           52  10   A2  04A2        SUBW2   #^X0010, R2                ; R2/R3 = A2/2
           50  52 42FD  04A5        SUBG2   R2, R0                     ; RO/R1 = Q(Y^2) - A2/2
           54  54 44FD  04A9  3$:   MULG2   R4, R4                     ; R4/R5 = XHI^2
           54  10   A2  04AD        SUBW2   #^X0010, R4                ; R4/R5 = XHI^2/2
           54  08 42FD  04B0        SUBG2   #1, R4                     ; R4/R5 = XHI^2/2 - 1
           50  54 42FD  04B4        SUBG2   R4, R0                     ; RO/R1 = GCOS(X)
           52  6E   7D  04B8        MOVQ    (SP), R2                   ; R2/R3 = X^2
           6E  50   7D  04BB        MOVQ    R0, (SP)                   ; Save GCOS(X)
 FCEB CF   06  52 55FD  04BE        POLYG   R2, #SINLENR-1, SINTBR     ; RO/R1 = Q(X^2)
           50  56 44FD  04C5        MULG2   R6, R0                     ; RO/R1 = X*Q(X^2)
           50  56 40FD  04C9        ADDG2   R6, R0                     ; RO/R1 = GSIN(X)
           52  8E   7D  04CD        MOVQ    (SP)+, R2                  ; R2/R3 = GCOS(X)
                    05  04D0        RSB
                        04D1
                        04D1
                        04D1  LARGE_SINCOS:
     000007C2'EF    16  04D1        JSB     REDUCE_LARGE               ; R4/R7 = reduced argument (in cycles)
                        04D7                                           ; R2 = octant bits
               52   DD  04D7        PUSHL   R2                         ; Save octant bits on stack
           7E  56   7D  04D9        MOVQ    R6, -(SP)                  ; Save reduced
           7E  54   7D  04DC        MOVQ    R4, -(SP)                  ;    argument on stack
     000005F3'EF    16  04DF        JSB     L_COS                      ; RO/R1 = GCOS(X)
           54  8E   7D  04E5        MOVQ    (SP)+, R4                  ; Reduced argument
           56  8E   7D  04E8        MOVQ    (SP)+, R6                  ;      in R4/R7
           52  8E   D0  04EB        MOVL    (SP)+, R2                  ; R2 = octant bits
           7E  50   7D  04EE        MOVQ    R0, -(SP)                  ; R2/R3 = GCOS(X)
     0000055E'EF    16  04F1        JSB     L_SIN                      ; RO/R1 = GSIN(X)
           52  8E   7D  04F7        MOVQ    (SP)+, R2                  ; R2/R3 = GCOS(X)
                    05  04FA        RSB
```

```
                         04FB              .SBTTL MTH$GSIN_R7
                         04FB
                         04FB      ; This routine computes the GSIN of the G-format value of R0/R1.  The
                         04FB      ; computation is performed one of three ways depending on the size of the
                         04FB      ; input argument, X:
                         04FB      ;
                         04FB      ;       1) If |X| < pi/4, then X is used directly in a polynomial approximation
                         04FB      ;          of GSIN.
                         04FB      ;       2) If pi/4 =< |x| < 9*pi/4, then the subroutine REDUCE_MEDIUM is called
                         04FB      ;          to reduce the argument to an equivalent argument in radians, Y, and
                         04FB      ;          the octant, I, containing the argument.  Y is then evaluated in a
                         04FB      ;          polynomial chosen as a function of I to compute GSIN(X).
                         04FB      ;       3) If 9*pi/4 =< |X|, then the subroutine REDUCE_LARGE is called to
                         04FB      ;          reduce the argument to an equivalent argument in cycles, Y, and the
                         04FB      ;          octant, I, contanining the argument.  Y is then evaluated in a
                         04FB      ;          polynomial chosen as a function of I to compute GSIN(X).
                         04FB
                         04FB      MTH$GSIN_R7::
               50 53FD   04FB              TSTG    R0                      ; Check the sign of R0
               10   18   04FE              BGEQ    POS_SIN                 ;
      0000050B'EF   16   0500              JSB     SIN                     ; R0/R1 = GSIN(|X|)
         50    50 52FD   0506              MNEGG   R0, R0                  ; R0/R1 = GSIN(X)
                    05   050A              RSB
                         050B
                         050B      SIN:
 50    8000 8F      AA   050B              BICW2   #^X8000, R0             ; R0/R1 = |X|
                         0510      POS_SIN:
 50    FAF3 CF 51FD      0510              CMPG    G_PI_OV_4, R0           ; Compare pi/4 with |X|
               22   14   0516              BGTR    SMALL_SIN               ; No argument reduction is necessary
 50    FAF3 CF 51FD      0518              CMPG    G_9_PI_OV_4, R0         ; Compare 9*pi/4 with |X|
               38   19   051E              BLSS    LARGE_SIN               ; Use special logic for |X| > 9*pi/4
                         0520
                         0520      ;
                         0520      ;  pi/4 =< |X| < 9*pi/4
                         0520      ;
      00000720'EF   16   0520              JSB     REDUCE_MEDIUM           ; Medium argument reduction routine
                         0526                                              ; R4/R7 = Y = reduced argument
                         0526                                              ; R2 = octant
 07    01    52     8F   0526      M_SIN:  CASEB   R2, #1, #7              ; Branch to one of four polynomial
                         052A                                              ;    evaluations depending on the
                 061E'   052A      1$:     .WORD   P_COS_R-1$
                 061E'   052C              .WORD   P_COS_R-1$
                 06CA'   052E              .WORD   N_SIN_R-1$
                 06CA'   0530              .WORD   N_SIN_R-1$
                 0671'   0532              .WORD   N_COS_R-1$
                 0671'   0534              .WORD   N_COS_R-1$
                 06D4'   0536              .WORD   P_SIN_R-1$
                 06D4'   0538              .WORD   P_SIN_R-1$              ;    octant bits.
                         053A
                         053A      ;
                         053A      ; Logic for small arguments.  |X| < pi/4.
                         053A      ;
                         053A
                         053A      SMALL_SIN:
 50    3E60 8F      B1   053A              CMPW    #^X3E60, R0             ; Compare with 2^-27
               16   18   053F              BGEQ    1$                      ; No polynomial evaluation is needed
       56    50     7D   0541              MOVQ    R0, R6                  ; R6/R7 = X
```

```
          50   50 44FD  0544   659          MULG2  R0,R0                 ; R0/R1 = X*X
FC61 CF   06   50 55FD  0548   660          POLYG  R0, #SINLENR-1, SINTBR ; R0/R1 = g(x^2)
          50   56 44FD  054F   661          MULG2  R6, R0                ; R0/R1 = X*g(x^2)
          50   56 40FD  0553   662          ADDG2  R6, R0                ; R0/R1 = GSIN(X)
                    05  0557   663 1$:      RSB
                        0558   664
                        0558   665
                        0558   666 LARGE_SIN:
    000007C2'EF     16  0558   667          JSB    REDUCE_LARGE          ; R4/R7 = reduced argument (in cycles)
                        055E   668                                       ; R2 = octant bits
               54   D5  055E   669 L_SIN:   TSTL   R4                    ; Check for degenerate case
               14   13  0560   670          BEQL   DEGENERATE_CASE_SIN
                        0562   671
       07   00   52 8F  0562   672          CASEB  R2, #0, #7
                        0566   673
                 0771'  0566   674 1$:      .WORD  P_SIN_C-1$
                 06B9'  0568   675          .WORD  P_COS_C-1$
                 06B9'  056A   676          .WORD  P_COS_C-1$
                 0771'  056C   677          .WORD  P_SIN_C-1$
                 0767'  056E   678          .WORD  N_SIN_C-1$
                 070B'  0570   679          .WORD  N_COS_C-1$
                 070B'  0572   680          .WORD  N_COS_C-1$
                 0767'  0574   681          .WORD  N_SIN_C-1$
                        0576   682
                        0576   683
                        0576   684 DEGENERATE_CASE_SIN:
                        0576   685
          52   01   8A  0576   686          BICB2  #1, R2                ; Compute index as (R2 - 1)/2
   52   52   FF 8F  9C  0579   687          ROTL   #-1, R2, R2
       03   00   52 8F  057E   688          CASEB  R2, #0, #3
                        0582   689
                 086B'  0582   690 1$:      .WORD  P_ONE-1$
                 087D'  0584   691          .WORD  UNFL -1$
                 0870'  0586   692          .WORD  N_ONE-1$
                 087D'  0588   693          .WORD  UNFL -1$
                        058A   694
```

```
                           058A
                           058A
                           058A          .SBTTL MTH$GCOS_R7
                           058A
                           058A   ; This routine computes the GCOS of the G-format value of R0/R1.  The
                           058A   ; computation is performed one of three ways depending on the size of the
                           058A   ; input argument, X.  The processing is the same as described for MTH$GSIN_R4.
                           058A   ;
                           058A
                           058A   MTH$GCOS_R7::
                 50 53FD   058A          TSTG    R0                           ; Check for reserved operand
      50   8000 8F    AA   058D          BICW2   #^X8000, R0                  ; R0/R1 = |X|
      50   FA71 CF  51FD   0592          CMPG    G_PI_OV_4, R0                ; Compare pi/4 with |X|
                 22   14   0598          BGTR    SMALL_COS                    ; No argument reduction is necessary
      50   FA71 CF  51FD   059A          CMPG    G_9_PI_OV_4, R0              ; Compare 9*pi/4 with |X|
                 4B   19   05A0          BLSS    LARGE_COS                    ; Use special logic for |X| > 9*pi/4
                           05A2
                           05A2   ;
                           05A2   ;  pi/4 =< |X| < 9*pi/4
                           05A2   ;
     00000720'EF      16   05A2          JSB     REDUCE_MEDIUM                ; Medium argument reduction routine
                           05A8                                               ; R4/R7 = Y = reduced argument
                           05A8                                               ; R2 = octant
      07   01   52    8F   05A8   M_COS: CASEB   R2, #1, #7                   ; Branch to one of four polynomial
                           05AC                                               ;     evaluations depending on the
                    0648'  05AC   1$:    .WORD   N_SIN_R-1$
                    0648'  05AE          .WORD   N_SIN_R-1$
                    05EF'  05B0          .WORD   N_COS_R-1$
                    05EF'  05B2          .WORD   N_COS_R-1$
                    0652'  05B4          .WORD   P_SIN_R-1$
                    0652'  05B6          .WORD   P_SIN_R-1$
                    059C'  05B8          .WORD   P_COS_R-1$
                    059C'  05BA          .WORD   P_COS_R-1$                   ;     octant bits.
                           05BC
                           05BC   ;
                           05BC   ; Logic for small arguments.  |X| < pi/4.
                           05BC   ;
                           05BC
                           05BC   SMALL_COS:
      50   4000 8F    B1   05BC          CMPW    #^X4000, R0                  ; Compare 1/2 with |X|
                 12   14   05C1          BGTR    1$                           ; Sufficent overghang is available
            56   50   7D   05C3          MOVQ    R0, R6                       ; R6/R7 = X
   57  FFFF07FF 8F    CA   05C6          BICL2   #^XFFFF07FF, R7              ; R6/R7 = XHI, 26 most signif bits
      54   50   56  43FD   05CD          SUBG3   R6, R0, R4                   ; R4/R5 = XLO, 27 least signif bits
               0580   31   05D2          BRW     NEEDS_DOUBLE                 ; Use special logic to obtain overhang
      50   3E60 8F    B1   05D5   1$:    CMPW    #^X3E60, R0                  ; Compare with 2**-27
                 0C   18   05DA          BGEQ    2$                           ; No polynomial evaluation is needed
            50   50 44FD   05DC          MULG2   R0,R0                        ; R0/R1 = X*X
 FB49 CF   07   50  55FD   05E0          POLYG   R0, #COSLENR1-1, COSTBR1;    R0/R1 = GCOS(X)
                      05   05E7          RSB
                           05E8
           50   08  50FD   05E8   2$:    MOVG    #1.0, R0                     ; R0/R1 = 1.0 = GCOS(X)
                      05   05EC          RSB
                           05ED
                           05ED
                           05ED   LARGE_COS:
     000007C2'EF      16   05ED          JSB     REDUCE_LARGE                 ; R4/R7 = reduced argument (in cycles)
```

```
                            05F3   753                                   ; R2 = octant bits
                    54  D5  05F3   754 L_COS:  TSTL    R4                ; Check for degenerate case
                    14  13  05F5   755         BEQL    DEGENERATE_CASE_COS
                            05F7   756
       07   00   52     8F  05F7   757         CASEB   R2, #0, #7
                     0624'  05FB   758 1$:     .WORD   P_COS_C-1$
                     06DC'  05FD   759         .WORD   P_SIN_C-1$
                     06D2'  05FF   760         .WORD   N_SIN_C-1$
                     0676'  0601   761         .WORD   N_COS_C-1$
                     0676'  0603   762         .WORD   N_COS_C-1$
                     06D2'  0605   763         .WORD   N_SIN_C-1$
                     06DC'  0607   764         .WORD   P_SIN_C-1$
                     0624'  0609   765         .WORD   P_COS_C-1$
                            060B   766
                            060B   767
                            060B   768 DEGENERATE_CASE_COS:
                            060B   769
            52      01  8A  060B   770         BICB2   #1, R2            ; Compute index as (R2 - 1)/2
  52    52    FF 8F     9C  060E   771         ROTL    #-1, R2, R2
       03   C0   52     8F  0613   772         CASEB   R2, #0, #3
                            0617   773
                     07E8'  0617   774 1$:     .WORD   UNFL -1$
                     07DB'  0619   775         .WORD   N_ONE-1$
                     07E8'  061B   776         .WORD   UNFL -1$
                     07D6'  061D   777         .WORD   P_ONE-1$
                            061F   778
```

```
                        061F          .SBTTL  MTH$GSINCOSD_R7
                        061F
                        061F  ; This routine computes the GSIND and GCOSD of the G-format value of R0/R1.
                        061F  ; The computation is performed one of two ways depending on the size of the
                        061F  ; input argument, X:
                        061F  ;
                        061F  ;       1) If |X| < 45, then X is used directly in polynomial approximation
                        061F  ;          of GSIND and GCOSD.
                        061F  ;       2) If 45 =< |x|, then the subroutine REDUCE_DEGREES is called to reduce
                        061F  ;          the argument to an equivalent argument in degrees, Y, and the
                        061F  ;          octant, I, containing the argument.  Y is then evaluated in two
                        061F  ;          polynomials chosen as a function of I, to compute GSIND(X) and
                        061F  ;          GCOSD(X).
                        061F
                        061F  MTH$GSINCOSD_R7::
              50 53FD   061F          TSTG    R0
              10   18   0622          BGEQ    SINCOSD
    50   8000 8F   AA   0624          BICW2   #^X8000, R0             ; R0/R1 = |X|
    00000634'EF    16   0629          JSB     SINCOSD                 ; R0/R1 = GSIND(|X|)
                        062F                                          ; R2/R3 = GCOSD(|X|)
         50    50 52FD  062F          MNEGG   R0, R0                  ; R0/R1= -GSIND(|X|)
                   05   0633          RSB
                        0634
                        0634  SINCOSD:
    50   F9FF CF 51FD   0634          CMPG    G_45, R0                ; Compare 45 to |X|
              24   14   063A          BGTR    SMALL_SINCOSD           ; special processing for small arg
    00000A8C'EF    16   063C          JSB     REDUCE_DEGREES          ; R6/R7 = reduced argument
                        0642                                          ; R3 = octant
         7E   56   7D   0642          MOVQ    R6, -(SP)               ; Save reduced arg
              53   DD   0645          PUSHL   R3                      ; Save octant bits
    000006F9'EF    16   0647          JSB     EVAL_COSD               ; R0/R1 = GCOSD(Y)
         53   8E   D0   064D          MOVL    (SP)+, R3               ; R3 = octant bits
         56   6E   7D   0650          MOVQ    (SP), R6                ; R6/R7 = reduced argument
         6E   50   7D   0653          MOVQ    R0, (SP)                ; Save GCOSD(Y)
    000006A0'EF    16   0656          JSB     EVAL_SIND               ; R0/R1 = GSIND(Y)
         52   8E   7D   065C          MOVQ    (SP)+, R2               ; R2/R3 = GCOSD(Y)
                   05   065F          RSB
                        0660
                        0660
                        0660  SMALL_SINCOSD:
         5E   10   C2   0660          SUBL2   #16, SP                 ; Allocate 4 longwords on stack
         6E   50   7D   0663          MOVQ    R0, (SP)                ; Save argument
    0000070D'EF    16   0666          JSB     SMALL_COSD              ; R0/R1 = GCOSD(|X|)
      08 AE   50   7D   066C          MOVQ    R0, 8(SP)               ; Save GCOSD(|X|)
         50   8E   7D   0670          MOVQ    (SP)+, R0               ; R0/R1 = argument
    000006B4'EF    16   0673          JSB     SMALL_SIND              ; R0/R1 = GSIND(X)
         52   8E   7D   0679          MOVQ    (SP)+, R2               ; R2/R3 = GCOSD(|X|)
                   05   067C          RSB
```

```
                              067D          .SBTTL  MTH$GSIND_R7
                              067D
                              067D  ; This routine computes the GSIND of the G-format value of R0/R1.  The
                              067D  ; computation is performed one of two ways depending on the size of the input
                              067D  ; argument, X:
                              067D  ;
                              067D  ;       1) If |X| < 45, then X is used directly in polynomial approximation
                              067D  ;          of GSIND.
                              067D  ;       2) If 45 =< |x|, then the subroutine REDUCE_DEGREES is called to reduce
                              067D  ;          the argument to an equivalent argument in degrees, Y, and the
                              067D  ;          octant, I, containing the argument.  Y is then evaluated in two
                              067D  ;          polynomials chosen as a function of I, to compute GSIND(X).
                              067D
                              067D  MTH$GSIND_R7::
                   50 53FD    067D          TSTG    R0                      ; R0/R1 = X
                   10   18    0680          BGEQ    POS_SIND                ;
            0000068D'EF   16    0682          JSB     NEG_SIND                ; R0/R1 = GSIND(|X|)
             50    50 52FD    0688          MNEGG   R0, R0                  ; R0/R1 = -GSIND(|X|)
                        05    068C          RSB
                              068D
                              068D  NEG_SIND:
      50    8000 8F     AA    068D          BICW2   #^X8000, R0             ; R0/R1 = |X|
                              0692  POS_SIND:
      50    F9A1 CF 51FD     0692          CMPG    G_45, R0                ; Compare 45 to |X|
                   1A   14    0698          BGTR    SMALL_SIND              ; special processing for small arg
            00000A8C'EF   16    069A          JSB     REDUCE_DEGREES          ; R6/R7 = reduced argument
                              06A0                                          ; R3 = octant
                              06A0
                              06A0  EVAL_SIND:
      07    00    53    8F    06A0          CASEB   R3, #0, #7
                     072D'    06A4  1$:     .WORD   P_SIN_D-1$
                     0665'    06A6          .WORD   P_COS_D-1$
                     0665'    06A8          .WORD   P_COS_D-1$
                     072D'    06AA          .WORD   P_SIN_D-1$
                     0729'    06AC          .WORD   N_SIN_D-1$
                     06C0'    06AE          .WORD   N_COS_D-1$
                     06C0'    06B0          .WORD   N_COS_D-1$
                     0729'    06B2          .WORD   N_SIN_D-1$
                              06B4
                              06B4
                              06B4  SMALL_SIND:
      50    F98F CF 51FD     06B4          CMPG    G_SMALLD, R0            ; Compare 180/pi*2^-27 with |x|
                   06   14    06BA          BGTR    1$                      ; No polynomial evaluation is
             56    50   7D    06BC          MOVQ    R0, R6                  ;   necessary
                 070F   31    06BF          BRW     P_SIN_D                 ;
                   50 53FD    06C2  1$:     TSTG    R0                      ; Check for zero
                   1B   13    06C5          BEQL    3$                      ; Return if R0 = 0
      50    F99C CF 51FD     06C7          CMPG    G_SMALLEST_DEG, R0      ; Check for possible underflow on
                   03   15    06CD          BLEQ    2$                      ;   conversion to radians
                 072D   31    06CF          BRW     UNFL                    ; Underflow will occur on conversion
 52    50   F981 CF 45FD     06D2  2$:     MULG3   G_CONVERT, R0, R2       ; R2/R3 = (pi/180 - 2^-6)*|x|
      50    0060 8F     A2    06D9          SUBW2   #^X0060, R0             ; R0/R1 = |X|*2^-6
             50    52 40FD    06DE          ADDG2   R2, R0                  ; R0/R1 = GSIND(|X|) = (pi/180)|X|
                        05    06E2  3$:     RSB
```

```
                         06E3         .SBTTL  MTH$GCOSD_R7
                         06E3
                         06E3   ; This routine computes the GCOSD of the G-format value of R0.  The computation
                         06E3   ; is performed one of two ways depending on the size of the input argument, X:
                         06E3   ; Details are given in the discussion on MTH$GCOSD_R4.
                         06E3
                         06E3   MTH$GCOSD_R7::
            50 53FD      06E3         TSTG    R0                      ; Check for reserved operand
50   8000 8F    AA       06E6         BICW2   #^X8000, R0             ; R0/R1 = |X|
50   F948 CF 51FD        06EB         CMPG    G_45, R0                ; Compare 45 to |X|
            1A  14       06F1         BGTR    SMALL_COSD              ;
   00000A8C'EF  16       06F3         JSB     REDUCE_DEGREES          ; R6/R7 = reduced argument
                         06F9                                         ; R3 = octant
                         06F9
                         06F9   EVAL_COSD:
07    00   53   8F       06F9         CASEB   R3, #0, #7
              060C'      06FD   1$:   .WORD   P_COS_D-1$
              06D4'      06FF         .WORD   P_SIN_D-1$
              06D0'      0701         .WORD   N_SIN_D-1$
              0667'      0703         .WORD   N_COS_D-1$
              0667'      0705         .WORD   N_COS_D-1$
              06D0'      0707         .WORD   N_SIN_D-1$
              06D4'      0709         .WORD   P_SIN_D-1$
              060C'      070B         .WORD   P_COS_D-1$
                         070D
                         070D
                         070D   SMALL_COSD:
50   F936 CF 51FD        070D         CMPG    G_SMALLD, R0            ; Compare 180/pi*2^-27  with |X|
            06  14       0713         BGTR    1$                      ; Check if polyinomial evaluation is
      56    50  7D       0715         MOVQ    R0, R6                  ;    necessary.
          05EE  31       0718         BRW     P_COS_D                 ; POLY needed
50     08 50FD           071B   1$:   MOVG    #1, R0                  ; R0 = 1. = GCOSD(|X|)
                05       071F         RSB
                         0720
```

```
                         0720
                         0720
                         0720           .SBTTL  REDUCE_MEDIUM
                         0720
                         0720   ;
                         0720   ; This routine assumes that the absolute value of the argument, X,  is in R0/R1
                         0720   ; and that pi/4 =< |X| < 9*pi/4.  It returns a pair of G-format values for the
                         0720   ; reduced argument:  YHI in R6/R7, and YLO in R4/R5.  The octant bits are
                         0720   ;  returned in R2.
                         0720   ;
                         0720   ; The reduced argument is obtained by locating the octant that X is in through
                         0720   ; a binary search and then subtracting off a suitable multiple of pi/2
                         0720   ;
                         0720
                         0720   REDUCE_MEDIUM:
      50   8000 8F    AA 0720           BICW2   #^X8000, R0                     ; R0/R1 = |X|
      50   F8F6 CF  51FD 0725           CMPG    G_5_PI_OV_4, R0                 ;
              12    15   072B           BLEQ    5$                              ; |X| >= 5*pi/4
      50   F8E6 CF  51FD 072D           CMPG    G_3_PI_OV_4, R0                 ;
              05    15   0733           BLEQ    3$                              ; |X| >= 3*pi/4
      52      01    D0   0735           MOVL    #1, R2                          ; First quadrant
              17    11   0738           BRB     SUBTRACT
                         073A
      52      03    D0   073A   3$:     MOVL    #3, R2                          ; Second quadrant
              12    11   073D           BRB     SUBTRACT
                         073F
      50   F8E4 CF  51FD 073F   5$:     CMPG    G_7_PI_OV_4, R0 ;
              05    15   0745           BLEQ    7$                              ; |X| >= 7*pi/4
      52      05    D0   0747           MOVL    #5, R2                          ; Third quadrant
              05    11   074A           BRB     SUBTRACT
                         074C
      52      07    D0   074C   7$:     MOVL    #7, R2                          ; Fourth quadrant
              00    11   074F           BRB     SUBTRACT
                         0751
                         0751
                         0751   SUBTRACT:
                         0751   ; New
   57   52    02    C5   0751           MULL3   #2, R2, R7
   53   FA  A747    3E   0755           MOVAW   -6(R7)[R7], R3                  ; R3 = index into PI_OV_2 table
 53   F911 CF43     DE   075A           MOVAL   PI_OV_2[R3], R3                 ; R3 = pointer into PI_OV_2 table
                         0760   ; NEW
      56      50    D0   0760           MOVL    R0, R6
57   51   FFFF07FF 8F CB 0763           BICL3   #^XFFFF07FF, R1, R7             ; R6/R7 = XH = 26 HIGH BITS OF X
      7E   50   56 43FD  076B           SUBG3   R6, R0, -(SP)                   ; SP = XL = 27 LOW ORDER BITS
           56   83 42FD  0770           SUBG2   (R3)+, R6
           6E   83 42FD  0774           SUBG2   (R3)+, (SP)
           56   8E 40FD  0778           ADDG2   (SP)+, R6
                         077C   ; END NEW
                         077C
                         077C   ;       SUBG3   (R3)+, R0, R6                   ; R6/R7 = 1st approximation to YHI
              02    15   077C           BLEQ    1$                              ;       = YHI'
              52    D6   077E           INCL    R2                              ; Adjust octant bits
   54   56   8000 8F  AB 0780   1$:     BICW3   #^X8000, R6, R4                 ; R4 = high 16 bits of |YHI'|
        54   3D10 8F  B1 0786           CMPW    #^X3D10, R4                     ; Check for at least 6 significant bits,
                         078B                                                   ;  ie, CMPW exp of 53-6=47 with (R4)
                         078B                                                   ;  so there is at least 6 bits of overhang
              0E    14   078B           BGTR    NOT_ENOUGH_BITS                 ;
```

```
                               078D
          54   56   7D         078D                 MOVQ    R6, R4                  ; R4/R5 = YHI'
               57   D4         0790                 CLRL    R7                      ; R6/R7 = high 21 bits of YHI' = YHI
          54   56 42FD         0792                 SUBG2   R6, R4                  ; R4/R5 = low bits of YHI'
          54   63 42FD         0796                 SUBG2   (R3), R4                ; R4/R5 = YLO
                    05         079A                 RSB
                               079B
                               079B  NOT_ENOUGH_BITS:
          50   63   7D         079B                 MOVQ    (R3), R0                ;
                               079E  ;              BICL2   #^X003F0000, R0         ;
 50    00070000 8F  CA         079E                 BICL2   #^X00070000, R0         ;
               51   D4         07A5                 CLRL    R1                      ;
      54   83  50 43FD         07A7                 SUBG3   R0, (R3)+, R4           ;
           56  50 42FD         07AC                 SUBG2   R0, R6                  ; R6/R7 = YHI
           54  63 40FD         07B0                 ADDG2   (R3), R4                ; R4/R5 = -YLO
      54   8000 8F  AC         07B4                 XORW    #^X8000, R4             ; R4/R5 = YLO
                    05         07B9                 RSB
                               07BA
                               07BA
```

```
                          07BA          .SBTTL  REDUCE_LARGE
                          07BA
                          07BA  ;
                          07BA  ; This routine is used to reduce large arguments (|X| >= 9*pi/4) modulo pi/4.
                          07BA  ; It returns the reduced argument, Y, in R4/R7 in units of cycles, and returns
                          07BA  ; the octant bits, I, in R2.
                          07BA  ;
                          07BA  ; The method of reduction is as follows:
                          07BA  ;
                          07BA  ;       x*(4/pi) = 2^n*f*(4/pi) where n is an integer and 1/2 =< f < 1
                          07BA  ;                = 2^(n-53)*(2^53*f)*(4/pi)
                          07BA  ;                = (2^53*f)*(2^(n-53)*4/pi)
                          07BA  ;                = K*C, where K = 2^53*f is an integer and C = 2*(n-53)*4/pi
                          07BA  ;       Let L = K*C modulo 8, where 0 =< L < 8, and let I = the integer(L) and
                          07BA  ;       h = fract(L), then if I is even Y = h, otherwise Y = 1-h
                          07BA  ;
                          07BA  ; CONSTANTS:
                          07BA
                000003D0  07BA          L_INT_WEIGHT  = ^X03D0              ; weights exponent by 61
                00000200  07BA          W_TERM_WEIGHT = ^X0200              ; weights exponent by 32
                00000400  07BA          W_MAX_WEIGHT  = ^X0400              ; maximum unbiased exponent
                000003B6  07BA          W_ADJUST      = ^X3B6               ; Used to locate binary point in
                          07BA                                              ;   MTH$AL_4_OV_PI_V table
                          07BA  G_2_TO_32:
       00000000 00004210  07BA          .QUAD   ^X4210                      ; 2^32
                          07C2
                          07C2
                          07C2
                          07C2  REDUCE_LARGE:
                          07C2  ;
                          07C2  ; The first step is to obtain the location of the binary point in the represen-
                          07C2  ; tation of C = 2^(n-53)*(4/pi) in two parts - the number of longwords from
                          07C2  ; the start and the number of bits from the most significant bit of the next
                          07C2  ; longword. Also K = 2^53*f must be obtained.
                          07C2  ;
        50   8000 8F   AA 07C2          BICW2   #^X8000, R0                 ; R0/R1 = |X|
     53  50    FC 8F   9C 07C7          ROTL    #-4, R0, R3                 ; Shift exponext field 4 bits right
        53   03B6 8F   A2 07CC          SUBW2   #W_ADJUST, R3               ; Unbias exp and adjust for leading
                          07D1                                              ;   zeroes.  R3 = location of binary
                          07D1                                              ;   point
     54  53    FD 8F   9C 07D1          ROTL    #-3, R3, R4                 ; Divide R3 by 32 and mull by 4 to get
     54   FFFFFF03 8F  CA 07D6          BICL2   #^XFFFFFF03, R4             ;   R4 = # of longwords (in bytes) to
                          07DD                                              ;   binary point.
     52   00000000'EF  DE 07DD          MOVAL   MTH$AL_4_OV_PI_V, R2        ; Get base address of MTH$AL_4_OV_PI_V
                          07E4                                              ;   table
     52   00000000'EF  C0 07E4          ADDL2   MTH$AL_4_OV_PI_V, R2        ; R2 = address of MTH$AL_4_OV_PI table
              52   54  C2 07EB          SUBL2   R4, R2                      ; R2 points to 1st quadword of interest
        53     E0 8F   8A 07EE          BICB2   #^XE0, R3                   ; R3(7:0) = # of bits within longword
                          07F2
        50   7FF0 8F   AA 07F2          BICW2   #^X7FF0, R0                 ; Clear exponent field
        50   4150 8F   A8 07F7          BISW    #^X4150, R0                 ; R0 = 2^21*f
             50    50 4AFD 07FC         CVTGL   R0, R0                      ; R0 = High 21 bits of K
        51   51    10  9C 0800          ROTL    #16, R1, R1                 ; R1 = Low 32 bits of K
                   02  18 0804          BGEQ    1$                          ; Check for high bit of R1 set
                   50  D6 0806          INCL    R0                          ; Adjust R0 if R1 is negative
                          0808
                          0808  ;
```

```
                                0808  ; The next step is to generate an approximation to C, call it C'' to be used
                                0808  ; in computing x*4/pi.  C'' will consist of the first three integer bits of
                                0808  ; C and the first 61 fraction bits of C.  These bits will be obtained from a
                                0808  ; constant stored in the interger array MTH$AL_4_OV_PI_V.
                                0808  ;
                                0808  ; NOTE:  The ASHQ, ADDL, and MULL instructions in the follow sections may
                                0808  ; result in an integer overflow trap.  The overflow incurred is intentional,
                                0808  ; so that the IV bit must be turned off.  The IV bit is not restored until
                                0808  ; after all of the necessary fraction bits hav been generated.
                                0808  ;
                         7E  DC 0808  1$:     MOVPSL  -(SP)                   ; Put current PSL on stack
         6E    FFFFFFDF  8F  CA 080A          BICL    #^C<PSL$M_IV>, (SP)     ; (SP) = current IV bit
                         20  B9 0811          BICPSW  #PSL$M_IV               ; Clear integer overflow bit
                                0813
                                0813  ; The necessary calculation to produce the reduced argument can require up to
                                0813  ; nine longwords of temporary work space.  This work space will be allocated
                                0813  ; on the stack.  The work space will be accessed though the use of three
                                0813  ; registers: R4, R5, and SP.  For the purposes of comments the temporary work
                                0813  ; space will be referred to as locations T0 though T8.  The stack and its
                                0813  ; pointers will look something like this:
                                0813  ;
                                0813  ;                       |                    |
                                0813  ;                       |--------------------|
                                0813  ;                   T0: |                    |   <---- SP
                                0813  ;                       |--------------------|
                                0813  ;                   T1: |                    |
                                0813  ;                       |--------------------|
                                0813  ;                   T2: |                    |
                                0813  ;                       |--------------------|
                                0813  ;                   T3: |                    |
                                0813  ;                       |--------------------|
                                0813  ;                   T4: |                    |   <---- R4
                                0813  ;                       |--------------------|
                                0813  ;                   T5: |                    |
                                0813  ;                       |--------------------|
                                0813  ;                   T6: |                    |
                                0813  ;                       |--------------------|
                                0813  ;                   T7: |                    |
                                0813  ;                       |--------------------|
                                0813  ;                   T8: |                    |   <---- R5
                                0813  ;                       |--------------------|
                                0813  ;                       |                    |
                                0813  ;
                                0813  ; The following code allocates the storage and sets up the pointers.
                                0813
                     5E  24  C2 0813          SUBL2   #36, SP                 ; Allocate 9 longwords on the stack
                 54  5E  10  C1 0816          ADDL3   #16, SP, R4             ; R4 points to T4
                 55  5E  20  C1 081A          ADDL3   #32, SP, R5             ; R5 points to T8
                                081E
                                081E  ;
                                081E  ; Get C'' = C(0):C(1):C(2):C(3) in T5/T8.  C(0) though  C(3) are unsigned
                                081E  ; integers generated from the binary representation of C.  The high three bits
                                081E  ; of C(0) are the first three bits to the left of the binary point of C.
                                081E  ; The remaining bits C(0) and C(1) though C(3) are the first 125 bits to the
                                081E  ; right of the binary point of C. Note that the C(i)'s are adjusted to
                                081E  ; compensate for their signed (rather than unsigned) interpretation in the EMUL
                                081E  ; instruction.  Note also that the representation of C has no more than 15
```

```
                                   081E  ; consequtive ones, so that no carry is possible from the adjustment.
                                   081E  ;
                                   081E
            57   54    0C     C1   081E           ADDL3   #12, R4, R7                  ; Initailize loop counter.  R7 points
                                   0822                                                ;   to T7
            67   62    53     79   0822           ASHQ    R3, (R2), (R7)               ; Shift the proper quadword so that
                                   0826                                                ;   T8 has C(0)in it
                 57    04     C2   0826           SUBL    #4, R7                       ; R7 points to T6
                 52    04     C2   0829  2$:      SUBL    #4, R2                       ; R2 points to next quadword in
                                   082C                                                ;   MTH$AL_4_OV_PI_V table
            67   62    53     79   082C           ASHQ    R3, (R2), (R7)               ; Shift quadword so that C(n) is in
                                   0830                                                ;   T(8-n) , n = 0,1,2,3
                       03     18   0830           BGEQ    3$                           ; Check for high bit of C(n) set
                    08 A7     D6   0832           INCL    8(R7)                        ; Bit set.  Adjust C(n-1)
FFEA 57   FFFFFFFC 8F    54   F1   0835  3$:      ACBL    R4, #-4, R7, 2$              ; Loop until C(0) though C(3) are in
                                   083F                                                ;   T5 though T8
                                   083F
                                   083F  ;
                                   083F  ; Generate the low 128 bits of the product K*C'' = L.  This product is
                                   083F  ; equivalent to multiplying K times C'' modulo 8.  The result of the
                                   083F  ; product is in T4/T7 with bits 31:29 of T4 the octant bits, and the remaining
                                   083F  ; 125 bits the faction bits of the product.  The last 53 fraction bits (bits
                                   083F  ; 20:0 of T6 and 31:0 of T5) are non-valid fraction bits that will be used
                                   083F  ; later if more fraction bits need to be generated.
                                   083F  ;
                                   083F  ;
                                   083F  ; Multiply the high order bits of K (R0) times C'' and store the result in
                                   083F  ; T0/T2.
                                   083F
       6E    00    04 A4    50    7A   083F       EMUL    R0, 4(R4), #0, (SP)          ; T0/T1 = KHI*C(3)
04 AE    04 AE    08 A4    50    7A   0845       EMUL    R0, 8(R4), 4(SP), 4(SP)      ; T0/T2 = KHI*[C(2):C(3)]
             64    0C A4    50    C5   084D       MULL3   R0, 12(R4), (R4)             ; T4 = Low 32 bits of KHI*C(1)
                   08 AE    64    C0   0852       ADDL2   (R4), 8(SP)                  ; T0/T2 = KHI*C'' modulo 8
                                   0856
                                   0856  ; Multiply the low order bits of K (R1) times C'' and store the result in
                                   0856  ; T4/T8.
                                   0856
       64    00    04 A4    51    7A   0856       EMUL    R1, 4(R4), #0, (R4)          ; T4/T5 = KLO*C(3)
04 A4    04 A4    08 A4    51    7A   085C       EMUL    R1, 8(R4), 4(R4), 4(R4)      ; T4/T6 = KLO*[C(2):C(3)]
08 A4    08 A4    0C A4    51    7A   0864       EMUL    R1, 12(R4), 8(R4), 8(R4)     ; T4/T7 = KLO*[C(1):C(2):C(3)]
                      65    51    C4   086C       MULL2   R1, (R5)                     ; T8 = KLO*C(0)
                   0C A4    65    C0   086F       ADDL2   (R5), 12(R4)                 ; T4/T7 = KLO*C'' modulo 8
                                   0873
                                   0873  ; Add KHI*C'' to KLO*C'' to get K*C''.  Store the result in T4/T7.
                                   0873
                   04 A4    6E    C0   0873       ADDL2   (SP), 4(R4)                  ;
          08 A4    04 AE    D8   0877       ADWC    4(SP), 8(R4)                 ;
          0C A4    08 AE    D8   087C       ADWC    8(SP), 12(R4)                ; T4/T7 = K*C'' modulo 8
                                   0881
                                   0881  ;
                                   0881  ; At this point there may or may not be enough valid bits in R3/R4 to generate
                                   0881  ; Y.  If the first 12 fratction bits are all 1's or 0's, there a possibility of
                                   0881  ; loss of significance when computing Y.  Consequently, we must check for loss
                                   0881  ; of significance before converting T4/T7 to Y and I.
                                   0881  ;
                                   0881
   65    FC A5    00008000 8F    C1   0881       ADDL3   #^X8000, -4(R5), (R5)        ; If the first 14 fraction bits are 1's
```

```
 65   3FFF0000 8F   D3  088A          BITL    #^X3FFF0000, (R5)        ;   and the reduced arg = 1-f or the
                37   12  0891          BNEQ    CONVERT                  ;   first 14 bit are 0 and the reduced
                         0893                                           ;   arg = f, then (and only then) bits
                         0893                                           ;   29:16 are 0 and significance will
                         0893                                           ;   be lost.
                         0893
                         0893  ;
                         0893  ; More bits need to be generated to cover the loss of significance.  There are
                         0893  ; not enough registers to hold all the potential entra bits, so that the bits
                         0893  ; already generated must be put on the stack.
                         0893  ;
                         0893
      000009C5'EF   16   0893          JSB     GEN_MORE_BITS            ; Generate 85 additional bits and add
                         0899                                           ;   them to existing bits.  Results are
                         0899                                           ;   stored in T3/T7
          54   04   C2   0899          SUBL2   #4, R4                   ; Adjust R4 to reflect the addition of
                         089C                                           ;   another longword of K*C''
     15 FC A5  1D   E0   089C          BBS     #29, -4(R5), 4$          ; Check if loss of significance is due
                         08A1                                           ;   to leading ones or zeros
                         08A1
                         08A1  ; Lost significance due to leading zeros
                         08A1
 65   10 A4   0F   00   EA   08A1          FFS     #0, #15, 16(R4), (R5)    ; If at least one bit is set.  This
                    21   12   08A7          BNEQ    CONVERT                  ;   means lost significance was minor.
 0C A4   0EFFFFFF 8F   D1   08A9          CMPL    #^X0EFFFFFF, 12(R4)      ; If one of the three high bits is set,
                    17   15   08B1          BLEQ    CONVERT                  ;   lost significance was minor.
                  00B6   31   08B3          BRW     LEADING_ZEROS
                              08B6
                              08B6  ; Lost significance due to leading ones
                              08B6
 65   10 A4   0F   00   EB   08B6  4$:     FFC     #0, #15, 16(R4), (R5)    ; If at least one bit is clear. This
                    0C   12   08BC          BNEQ    CONVERT                  ;   meanslost significance was minor.
 0C A4   F8000000 8F   D1   08BE          CMPL    #^XF8000000, 12(R4)      ; If one of the three high bits is
                    02   1E   08C6          BGEQU   CONVERT                  ;   clear, lost significance was minor.
                    3D   11   08C8          BRB     LEADING_ONES
                              08CA
                              08CA
                              08CA  CONVERT:
                              08CA
                              08CA  ;     Isolate octant bits and convert fraction bits to a pair of D-format
                              08CA  ;     quantities YHI and YLO
                              08CA
 65   FC A5   03   1D   EF   08CA          EXTZV   #29, #3, -4(R5), (R5)    ; T8 = octant bits
  FC A5   E0000000 8F   CA   08D0          BICL2   #^XE0000000, -4(R5)      ; Clear octant bits
        54   55   0C   C3   08D8          SUBL3   #12, R5, R4              ; R4 points to low order bits of h
       00000A0B'EF   16   08DC          JSB     CVT_TO_DOUBLE            ; R0/R1 = 2^29*h_lo
                              08E2                                           ; R6/R7 = 2^29*h_hi
        56   01D0 8F   A2   08E2          SUBW2   #^X01D0, R6              ; R6/R7 = h_hi
                    02   14   08E7          BGTR    3$                       ; Check for h_hi = 0
                    56   D4   08E9          CLRL    R6                       ; Restore h_hi to 0
                    50   B5   08EB  3$:     TSTW    R0                       ; Check for h_lo = 0
                    05   13   08ED          BEQL    1$
        50   01D0 8F   A2   08EF          SUBW2   #^X01D0, R0              ; R0/R1 = h_lo
                 09 65   E9   08F4  1$:     BLBC    (R5), 2$                 ; Check for odd or even octant bits
                              08F7
                              08F7  ;     Octant bits are odd.  Reduced argument equals 1 - h.
                              08F7
```

```
         56  08   56 43FD  08F7          SUBG3   R6, #1, R6                 ; R6/R7 = Y = 1 - h_hi
             50   50 52FD  08FC          MNEGG   R0, R0                     ; R0/R1 = -h_lo
                           0900
                           0900  ;    Get octant bits
                           0900
             52   20 AE  D0  0900  2$:   MOVL    32(SP), R2                 ; R2 = octant bits
                   00AA  31  0904        BRW     GET_YHI_YLO
                           0907
                           0907
                           0907  ;
                           0907  ; At this point it has been determined that there is a major loss of
                           0907  ; significance and the processing begins a looping phase.  Each iteration of
                           0907  ; the loop will generate additional extra bits of K*C' until enough significant
                           0907  ; bits to compute Y are available.  During this time the nine longwords
                           0907  ; allocated on the stack will be used as follows:
                           0907  ;
                           0907  ;       T0/T2   Temporary storage used when generating extra bits.
                           0907  ;
                           0907  ;       T3/T7   Contains all significant bits generated so far.
                           0907  ;
                           0907  ;       T8      Contains a counter, W, indicating the appropriate exponent
                           0907  ;                 of the last longword of fraction bits used in converting
                           0907  ;                 to Y.
                           0907
                           0907  LEADING_ONES:
                           0907
                           0907  ; If processing continues here it is known that the loss of significance is due
                           0907  ; to a string of leading ones.
                           0907
      65   000003D0 8F  D0  0907         MOVL    #L_INT_WEIGHT, (R5)        ; T8 = exp bias for last longword
                           090E                                             ;     of the product K*C'
                           090E
   0C A4   FFFF0000 8F  D1  090E  LOOP_1: CMPL   #^XFFFF0000, 12(R4)        ; Check for enough significant bits
                     2E  1A  0916        BGTRU   CONVERT_1                  ; Enough bits.  Convert to floating.
           000009C5'EF  16  0918         JSB     GEN_MORE_BITS              ; T2/T7 contains K*C''
   0C A4   FFFFFFFF 8F  D1  091E         CMPL    #-1, 12(R4)                ; Check for all 1's
                     1E  1A  0926        BGTRU   CONVERT_1                  ; Not all 1's.  Enough precision bits
                           0928                                             ;     to compute Y
        08 A4   04 A4  7D  0928          MOVQ    4(R4), 8(R4)               ; Compress representation
           64   FC A4  7D  092D          MOVQ    -4(R4), (R4)               ;     of K*C''
FFD3 65   0200 8F  0400 8F  3D  0931     ACBW    #W_MAX_WEIGHT, #W_TERM_WEIGHT, (R5), LOOP_1
                           093B                                             ; Increment weighting factor.  If
                           093B                                             ;   weighting factor is greater than
                           093B                                             ;   1024 then no more bits need to be
                           093B                                             ;   generated.
                           093B
                           093B  ;
                           093B  ; The weighting factor is greater than 1024.  This means that the reduced
                           093B  ; argument is either not distinguishable from 1 or too small to be represented
                           093B  ; in F-format (i.e. underflow.)  Zero is returned in R4 for the reduced
                           093B  ; argument to signal this occurance.  Note that under these conditions the
                           093B  ; correct function value is one of the values 0, +/-1.  The
                           093B  ; correct choice is determined by the calling program based on the octant bits
                           093B  ; returned in R1.
                           093B  ;
                     53  D4  093B        CLRL    R3                         ; Reduced argument is zero
   52   08 AE   03   1D  EF  093D        EXTZV   #29, #3, 8(SP), R2         ; R2 = octant bits
```

```
                 0078   31  0943              BRW     RESTORE
                            0946
                            0946
                            0946      CONVERT_1:
              54   04   C0  0946              ADDL2   #4, R4                  ; R4 points to low bits of h
         00000A0B'EF    16  0949              JSB     CVT_TO_DOUBLE           ; R0/R1 = 2^W*h_lo
                            094F                                              ; R6/R7 = 2^W*h_hi
   56    FE65 CF   56 43FD  094F              SUBG3   R6, G_2_TO_32, R6       ; R6/R7 = 2^W*(1 - h_hi)
              50   50 52FD  0956              MNEGG   R0, R0                  ; R0/R1 = -2^W*h_lo
 52    FC A5  03   1D   EF  095A              EXTZV   #29, #3, -4(R5), R2     ; R2 = octant bits
              56   65   C2  0960              SUBL    (R5), R6                ; R6/R7 = 1 - h_hi
                   50   B5  0963              TSTW    R0                      ; Check for h_lo = 0
                   4A   13  0965              BEQL    GET_YHI_YLO             ;
              50   65   C2  0967              SUBL    (R5), R0                ; R0/R1 = - h_lo
                   45   11  096A              BRB     GET_YHI_YLO
                            096C
                            096C
                            096C      LEADING_ZEROS:
                            096C
                            096C      ; If processing continues here it is known that the loss of significance is due
                            096C      ; to a string of leading zeros.  Note that it is known that the loop for
                            096C      ; leading zeros will terminate before an underflow condition occurs so that the
                            096C      ; loop does not include a test for underflow.
                            096C
       65     000003D0 8F   D0  096C              MOVL    #L_INT_WEIGHT, (R5)     ; T8 = exp bias for last longword
                            0973                                              ;     of the product K*C'
                            0973
   0C A4      0000FFFF 8F   D1  0973      LOOP_0: CMPL    #^X0000FFFF, 12(R4)     ; Check enough fraction bits
                   1B   19  097B              BLSS    CONVERT_0               ; Enough bits.  Convert to floating
         000009C5'EF    16  097D              JSB     GEN_MORE_BITS           ; T2/R7 contain K*C''
                 0C A4  D5  0983      1$:     TSTL    12(R4)                  ; Check for all 0's
                   10   12  0986              BNEQ    CONVERT_0               ; Not all 0's.  Enough precision bits.
        08 A4   04 A4   7D  0988              MOVQ    4(R4), 8(R4)            ; Compress representation
           64   FC A4   7D  098D              MOVQ    -4(R4), (R4)            ;     of K*C''
          65    0200 8F A0  0991              ADDW    #W_TERM_WEIGHT, (R5)    ; Increment weighting factor.
                   DB   11  0996              BRB     LOOP_0
                            0998
                            0998      CONVERT_0:
              54   04   C0  0998              ADDL    #4, R4                  ; R4 points to low bits of h
         00000A0B'EF    16  099B              JSB     CVT_TO_DOUBLE           ; R0/R1 = 2^W*h_lo
                            09A1                                              ; R6/R7 = 2^W*h_hi
 52    FC A5  03   1D   EF  09A1              EXTZV   #29, #3, -4(R5), R2     ; R2 = octant bits
              56   65   C2  09A7              SUBL2   (R5), R6                ; R6/R7 = h_hi
                   50   B5  09AA              TSTW    R0                      ; Check for h_lo = 0
                   03   13  09AC              BEQL    GET_YHI_YLO             ;
              50   65   C2  09AE              SUBL2   (R5), R0                ; R0/R1 = h_lo
                            09B1
                            09B1      GET_YHI_YLO:
              54   56   7D  09B1              MOVQ    R6, R4                  ; R4/R5 = high bits of Y
                   57   D4  09B4              CLRL    R7                      ; R6/R7 = high 24 bits Y = YHI
              54   56 42FD  09B6              SUBG2   R6, R4                  ;
              54   50 40FD  09BA              ADDG2   R0, R4                  ; R4/R5 = YLO
                            09BE      RESTORE:
                 24 AE  B8  09BE              BISPSW  36(SP)                  ; Restore IV bit and exit
              5E   28   C0  09C1              ADDL    #40, SP                 ; Remove mask and temporary storage
                        05  09C4              RSB
                            09C5
```

```
                              09C5
                              09C5   GEN_MORE_BITS:
                              09C5
                              09C5   ;
                              09C5   ; This subroutine generates 85 extra fraction bits and puts them to the
                              09C5   ; existing bits.  NOTE:  This routine is always entered via a JSB instruction.
                              09C5   ; Consequently, SP points to the first longword BEFORE T0, rather than T0
                              09C5   ; itself.
                              09C5   ;
                              09C5
              52    04    C2  09C5           SUBL2   #4, R2                 ; Adjust pointer to get next quadword
                              09C8                                          ;   from MTH$AL_4_OV_PI_V
        56    62    53    79  09C8           ASHQ    R3, (R2), R6           ; R7 = C(n)
                    17    18  09CC           BGEQ    1$                     ; Branch if high bit is clear
                              09CE
                              09CE   ; Logic to process unsigned values greater than 2^31 - 1
                              09CE
 04 AE    00    57    51    7A  09CE           EMUL    R1, R7, #0, 4(SP)      ;
          08 AE    51    C0  09D4           ADDL2   R1, 8(SP)              ; T0/T1 = KLO*C(n)
 08 AE  08 AE    57    50    7A  09D8           EMUL    R0, R7, 8(SP), 8(SP)   ;
          0C AE    50    C0  09DF           ADDL2   R0, 12(SP)             ; T0/T2 = K*C(n)
                    0D    11  09E3           BRB     2$
                              09E5
                              09E5   ; Logic to process unsigned values less than 2^31
                              09E5
 04 AE    00    57    51    7A  09E5   1$:     EMUL    R1, R7, #0, 4(SP)      ; T0/T1 = KLO*C(n)
 08 AE  08 AE    57    50    7A  09EB           EMUL    R0, R7, 8(SP), 8(SP)   ; T0/T2 = K*C(n)
                              09F2
                              09F2   ; Add new bits to old
                              09F2
        64    08 AE    C0  09F2   2$:     ADDL2   8(SP), (R4)            ;
     04 A4    0C AE    D8  09F6           ADWC    12(SP), 4(R4)          ;
                    08    1E  09FB           BCC     3$                     ; Check for carry from previous add
              08 A4    D6  09FD           INCL    8(R4)                  ; Propagate carry
                    03    1E  0A00           BCC     3$                     ; Check for carry from previous add
              0C A4    D6  0A02           INCL    12(R4)                 ; Propagate carry
     FC A4    04 AE    D0  0A05   3$:     MOVL    4(SP), -4(R4)          ; Move new low order bits to end of
                              0A0A                                          ;   of old low order bits
                    05        0A0A           RSB                            ;
                              0A0B
                              0A0B
                              0A0B
                              0A0B
                              0A0B
                              0A0B   CVT_TO_DOUBLE:
                              0A0B
                              0A0B   ;
                              0A0B   ; This routine converts an array of three longword pointed to by R4 to a pair
                              0A0B   ; of G-format values.  The results are returned in R0/R1 (low 48 bits) and
                              0A0B   ; R6/R7 (high 48 bits).  ;
                              0A0B
        50    84  4EFD  0A0B           CVTLG   (R4)+, R0              ; R0/R1 = Low 32 bits of h
                    0E    13  0A0F           BEQL    2$                     ;
                    07    14  0A11           BGTR    1$                     ; Adjust for signed
                    64    D6  0A13           INCL    (R4)                   ;   conversion error
                    03    1E  0A15           BCC     1$                     ; If necessary,
              04 A4    D6  0A17           INCL    4(R4)                  ;   propagate carry
```

```
              50    0200 8F     A2  0A1A  1$:     SUBW2   #W_TERM_WEIGHT, R0        ; R0/R1 = (low 32 bits of h)/2^32
FC A4    64   FFFF0000 8F     CB  0A1F  2$:     BICL3   #^XFFFF0000, (R4), -4(R4)
                 64   FC A4     C2  0A28          SUBL2   -4(R4), (R4)              ;
                 52   FC A4   4EFD  0A2C          CVTLG   -4(R4), R2                ;
                    50   52   40FD  0A31          ADDG2   R2, R0                    ; R0/R1 = (low 48 bits of h)/2^32
                         05     13  0A35          BEQL    3$                        ;
              50    0200 8F     A2  0A37          SUBW2   #W_TERM_WEIGHT, R0        ; R0/R1 = (low 48 bits of h)/2^64
                    52   84   4EFD  0A3C  3$:     CVTLG   (R4)+, R2                 ; R2/R3 = next 16 bits of h
                         09     13  0A40          BEQL    5$                        ;
                         02     14  0A42          BGTR    4$                        ; Adjust for signed conversion error.
                         64     D6  0A44          INCL    (R4)                      ;   Note that no carry is possible
              52    0200 8F     A2  0A46  4$:     SUBW2   #W_TERM_WEIGHT, R2        ; R2/R3 = (next 16 bits of h)/2^32
                    56   64   4EFD  0A4B  5$:     CVTLG   (R4), R6                  ; R6/R7 = high 32 bits of h
                         06     18  0A4F          BGEQ    6$                        ; Adjust for signed conversion
              56    FD64 CF   40FD  0A51          ADDG2   G_2_TO_32, R6             ;   error
                    56   52   40FD  0A57  6$:     ADDG2   R2, R6                    ; R6/R7 = (high 48 bits of h)/2^32
                              05    0A5B          RSB
                                    0A5C
```

```
                           0A5C           .SBTTL  REDUCE_DEGREES
                           0A5C
                           0A5C   ; This routine assumes that the absolute value of the argument is in R0/R1.
                           0A5C   ; The reduction process is performed in two stages.  The first stage of
                           0A5C   ; the reduction reduces the argument modulo 360 to a value less that 2^55,
                           0A5C   ; and the second stage reduces the argument modulo 45 to a value less than 45.
                           0A5C
                           0A5C   ; Constants used in this reduction:
                           0A5C   ;
                           0A5C
                           0A5C  POWER_MOD_360_0:          ; Powers of 2 modulo 360 for t1 = 0
     0008 0004 0002 0001   0A5C           .WORD      1,       2,       4,      8
     0080 0040 0020 0010   0A64           .WORD     16,      32,      64,    128
     00F8 0130 0098 0100   0A6C           .WORD    256,     152,     304,    248
                           0A74
                           0A74  POWER_MOD_360_1:          ; Powers of 2 modulo 360 for t1 <> 0
     0008 00B8 0110 0088   0A74           .WORD    136,     272,     184,      8
     0080 0040 0020 0010   0A7C           .WORD     16,      32,      64,    128
     00F8 0130 0098 0100   0A84           .WORD    256,     152,     304,    248
                           0A8C
                           0A8C
                           0A8C
                           0A8C  REDUCE_DEGREES:
       50    4360 8F   B1  0A8C           CMPW    #^X4360, R0                  ; Compare |x| with 2^53
                  4E   14  0A91           BGTR    LAST_STEP                    ; Branch to special logic for med arg
                           0A93
                           0A93   ;
                           0A93   ; It is assumed here that the argument is greater than 2^53.
                           0A93   ;
                           0A93   ; The argument is reduced as follows:
                           0A93   ;       Let x = 2^t*f, where t > 53 and 1/2 =< f < 1.  And let J = 2^53*f =
                           0A93   ;       2^30*J1 + J2 and K = 2^(t-53).  Since 2^30 = 64 modulo 360, we have that
                           0A93   ;       J = 64*J1 + J2 modulo 360.  Now let t' = t - 53 = 12*t1 + t2.  Note that
                           0A93   ;       (2^12)^2 = (2^9)*(2^15) = (2^9)*(2^3) = 2^12 modulo 360.  Hence, if t1 is
                           0A93   ;       not zero, K = 2^t' = 2^(12*t1+t2) = (2^12)*(2^t2) = 136*2^t2 modulo 360.
                           0A93   ;       For t1 = 0 K = 2^t2.  Consequently, define K' congruent to 2^t2 if t1 = 0
                           0A93   ;       and congruent to 136*2^t2 otherwise, where 0 =< K' < 360.  Then x' =
                           0A93   ;       K'*(64*J1 + J2) is congruent to s modulo 360 and x' < 2^53.
                           0A93
             52    50  D0  0A93           MOVL    R0, R2                       ; R2 = high longword of X
   50   00007FF0 8F    CA  0A96           BICL2   #^X7FF0, R0                  ; Clear exp bits of X
       50    4350 8F   A8  0A9D           BISW    #^X4350, R0                  ; R0/R1 = J
             52    50  C2  0AA2           SUBL    R0, R2                       ; R2 = t'*2^7
                           0AA5
             53    50  7D  0AA5           MOVQ    R0, R3                       ; R3/R4 = J
   51   FFFF3FFF 8F    CA  0AA8           BICL    #^XFFFF3FFF, R1              ; R0/R1 = J1*2^30
             53    50 42FD 0AAF           SUBG2   R0, R3                       ; R3/R4 = J2
       50    0180 8F   A2  0AB3           SUBW2   #^X0180, R0                  ; R0/R1 = 64*J1
             50    53 40FD 0AB8           ADDG2   R3, R0                       ; R0/R1 = 64*J1 + J2 = J modulo 45
                           0ABC
   52    52    FC 8F   9C  0ABC           ROTL    #-4, R2, R2                  ; R2 = t'
     53    52    0C    A7  0AC1           DIVW3   #12, R2, R3                  ; R3 = t1
             53    0C  A4  0AC5           MULW2   #12, R3                      ; R3 = 12*t1
             52    53  A2  0AC8           SUBW2   R3, R2                       ; R2 = t2
                           0ACB
                   53  B5  0ACB           TSTW    R3                           ; Check for t1 = 0 and choose K'
                   08  12  0ACD           BNEQ    1$                           ;   accordingly
```

**

```
            52    88 AF42 4DFD  0ACF              CVTWG   POWER_MOD_360_0[R2], R2 ; R2/R3 = K'
                       06   11  0AD5              BRB     2$                      ;
            52    98 AF42 4DFD  0AD7  1$:         CVTWG   POWER_MOD_360_1[R2], R2 ; R2/R3 = K'
            50       52 44FD    0ADD  2$:         MULG2   R2, R0                  ; R0/R1 = X' (mod 45) 0 =< R0 < 2^55
                                0AE1
                                0AE1
                                0AE1  LAST_STEP:
                                0AE1  ;
                                0AE1  ; Argument reduction scheme for arguments with absolute value less than 2^55
                                0AE1  ;
                                0AE1  ; The reduced argument Y is computed as follows:
                                0AE1  ;    Let I = int(X/45)
                                0AE1  ;        if I is even
                                0AE1  ;           then Y = X - 45*I
                                0AE1  ;           else Y = (I+1)*45 - x
                                0AE1
                                0AE1
            50    4240 8F   B1  0AE1              CMPW    #^X4240, R0             ; Compare 2^36 with |X|
                       28   18  0AE6              BGEQ    NO_OVERFLOW             ;
       56   50    F563 CF 45FD  0AE8              MULG3   G_T_OV_45, R0, R6       ; R6/R7 = |X|/45
                                0AEF
                                0AEF  ;
                                0AEF  ; Turn off IV to avoid an exception in EMODD
                                0AEF  ;
                       52   DC  0AEF              MOVPSL  R2                      ; Move PSL to R2
       52     FFFFFFDF 8F   CA  0AF1              BICL2   #^C<PSL$M_IV>, R2       ; Save current IV bit
                       20   B9  0AF8              BICPSW  #PSL$M_IV               ; Turn off integer overflow trap
                                0AFA
  54   53   56    00    08 54FD 0AFA              EMODG   #1, #0, R6, R3, R4      ; R3 = low 32 integer bits of |X|/45
                                0B01                                              ; R4/R5 = fractional part of |X|/45
                                0B01
                       52   B8  0B01              BISPSW  R2                      ; Restore IV bit
                                0B03
            56       54 42FD    0B03              SUBG2   R4, R6                  ; R6/R7 = Integer part of |X|/45 = I
                    2F 53   E9  0B07              BLBC    R3, EVEN                ;
            56       08 40FD    0B0A              ADDG2   #1, R6                  ; R6/R7 = I + 1
                       16   11  0B0E              BRB     ODD
                                0B10
                                0B10
                                0B10  NO_OVERFLOW:
  53   50   C16C 8F  F53B CF 54FD 0B10            EMODG   G_1_OV_45, #X_1_OV_45, R0, R3, R4
                          54    0B1A
                                0B1B                                              ; R3 = I = integer part of |X|/45
                    17 53   E9  0B1B              BLBC    R3, CVT                 ; Branch if octant bits are even
            56   53   01    C1  0B1E              ADDL3   #1, R3, R6              ; R6 = I + 1
                 56   56 4EFD   0B22              CVTLG   R6, R6                  ; R6/R7 = I + 1
            56   F50D CF 44FD   0B26  ODD:        MULG2   G_45, R6                ; R6/R7 = 45*(I+1)
                 56   50 42FD   0B2C              SUBG2   R0, R6                  ; R6/R7 = Y
           53    F8 8F      8A  0B30              BICB    #^XF8, R3               ; Save only last three octant bits
                            05  0B34              RSB
                                0B35
            56       53 4EFD    0B35  CVT:        CVTLG   R3, R6                  ; R6/R7 = I
            56   F502 CF 44FD   0B39  EVEN:       MULG2   G_M45, R6               ; R6/R7 = -45*I
                 56   50 40FD   0B3F              ADDG2   R0, R6                  ; R6/R7 = Y
           53    F8 8F      8A  0B43              BICB    #^XF8, R3               ; Save only last three octant bits
                            05  0B47              RSB
                                0B48
```

```
0B48  1527
```

```
                                0B48
                                0B48            .SBTTL  RADIAN_POLYNOMIALS      ; Polynomials for arguments in radians
                                0B48
                                0B48
                                0B48
                                0B48    ;
                                0B48    ; Polynomial evaluation for DCOS(Y) for Y in radians
                                0B48    ;
                                0B48
                                0B48    P_COS_R:
 53    56   8000 8F      AB     0B48            BICW3   #^X8000, R6, R3         ;
       53   4000 8F      B1     0B4E            CMPW    #^X4000, R3             ; Compare 1/2 with |YHI|
                  36     14     0B53            BGTR    LEQL_HALF               ; Sufficent overhang is available
                                0B55    NEEDS_DOUBLE:
        7E        54     7D     0B55            MOVQ    R4, -(SP)               ; Save YLO
        7E   54   56   41FD     0B58            ADDG3   R6, R4, -(SP)           ; Save Y
        50   6E   6E   45FD     0B5D            MULG3   (SP), (SP), R0          ; R0/R1 = Y^2
 F607   CF   07   50   55FD     0B62            POLYG   R0, #COSLENR2-1, COSTBR2; R0/R1 = Q(Y^2)
        54   56   8E   41FD     0B69            ADDG3   (SP)+, R6, R4           ; R4/R5 = Y + YHI
             54   8E   44FD     0B6E            MULG2   (SP)+, R4               ; R4/R5 = YLO*(Y + YHI) = A2
                  03     13     0B72            BEQL    1$                      ; Check for A2 = 0
             54   10     A2     0B74            SUBW2   #^X0010, R4             ; R4/R5 = A2/2
             50   54   42FD     0B77    1$:     SUBG2   R4, R0                  ; R0/R1 = Q(Y^2) - A2/2
             56   56   44FD     0B7B            MULG2   R6, R6                  ; R6/R7 = YHI^2
             56   10     A2     0B7F            SUBW2   #^X0010, R6             ; R6/R7 = YHI^2/2
             56   08   42FD     0B82            SUBG2   #1, R6                  ; R6/R7 = -(1 - YHI^2/2)
             50   56   42FD     0B86            SUBG2   R6, R0                  ; R0/R1 = GCOS(Y)
                         05     0B8A            RSB
                                0B8B
                                0B8B    LEQL_HALF:
             56   54   40FD     0B8B            ADDG2   R4, R6                  ; R6/R7 = Y
             56   56   44FD     0B8F            MULG2   R6, R6                  ; R6/R7 = Y^2
 F596   CF   07   56   55FD     0B93            POLYG   R6, #COSLENR1-1, COSTBR1; R0/R1 = GCOS(Y)
                         05     0B9A            RSB
                                0B9B
                                0B9B
                                0B9B    ;
                                0B9B    ; Polynomial evaluation for -GCOS(Y)
                                0B9B    ;
                                0B9B
                                0B9B    N_COS_R:
 53    56   8000 8F      AB     0B9B            BICW3   #^X8000, R6, R3         ;
       53   4000 8F      B1     0BA1            CMPW    #^X4000, R3             ; Compare 1/2 with |YHI|
                  37     14     0BA6            BGTR    2$                      ; Sufficent overhang is available
        7E        54     7D     0BA8            MOVQ    R4, -(SP)               ; Save YLO
        7E   54   56   41FD     0BAB            ADDG3   R6, R4, -(SP)           ; Save Y
        50   6E   6E   45FD     0BB0            MULG3   (SP), (SP), R0          ; R0/R1 = Y^2
 F5B4   CF   07   50   55FD     0BB5            POLYG   R0, #COSLENR2-1, COSTBR2; R0/R1 = Q(Y^2)
        54   56   8E   41FD     0BBC            ADDG3   (SP)+, R6, R4           ; R4/R5 = Y + YHI
             54   8E   44FD     0BC1            MULG2   (SP)+, R4               ; R4/R5 = YLO*(Y + YHI) = A2
                  03     13     0BC5            BEQL    1$                      ; Check for A2 = 0
             54   10     A2     0BC7            SUBW2   #^X0010, R4             ; R4/R5 = A2/2
             50   54   42FD     0BCA    1$:     SUBG2   R4, R0                  ; R0/R1 = Q(Y^2) - A2/2
             56   56   44FD     0BCE            MULG2   R6, R6                  ; R6/R7 = YHI^2
             56   10     A2     0BD2            SUBW2   #^X0010, R6             ; R6/R7 = YHI^2/2
             56   08   42FD     0BD5            SUBG2   #1, R6                  ; R6/R7 = -(1 - YHI^2/2)
        50   56   50   43FD     0BD9            SUBG3   R0, R6, R0              ; R0/R1 = -GCOS(Y)
```

```
                          05  0BDE  1586          RSB
                              0BDF  1587
          56     54  40FD     0BDF  1588 2$:      ADDG2   R4, R6                  ; R6/R7 = Y
          56     56  44FD     0BE3  1589          MULG2   R6, R6                  ; R6/R7 = Y^2
F542 CF   07     56  55FD     0BE7  1590          POLYG   R6, #COSLENR1-1, COSTBR1; R0/R1 = GCOS(Y)
          50   8000  8F   AC  0BEE  1591          XORW    #^X8000, R0             ; R0/R1 = -GCOS(Y)
                          05  0BF3  1592          RSB
                              0BF4  1593
                              0BF4  1594 ;
                              0BF4  1595 ; Polynomial evaluation for -GSIN(Y)
                              0BF4  1596 ;
                              0BF4  1597
                              0BF4  1598 N_SIN_R:
          54   8000  8F   AC  0BF4  1599          XORW    #^X8000, R4             ;
          56   8000  8F   AC  0BF9  1600          XORW    #^X8000, R6             ; R4/R7 = -Y
                              0BFE  1601
                              0BFE  1602 ;
                              0BFE  1603 ; Polynomial evaluation for GSIN(Y)
                              0BFE  1604 ;
                              0BFE  1605
                              0BFE  1606 P_SIN_R:
          7E     54    7D     0BFE  1607          MOVQ    R4, -(SP)               ; Save YLO
     7E   56     54  41FD     0C01  1608          ADDG3   R4, R6, -(SP)           ; Save Y
     54   6E     6E  45FD     0C06  1609          MULG3   (SP), (SP), R4          ; R4 = Y^2
F59E CF   06     54  55FD     0C0B  1610          POLYG   R4, #SINLENR-1, SINTBR  ; R0/R1 = P(Y^2)
          50     8E  44FD     0C12  1611          MULG2   (SP)+, R0               ; R0/R1 = Y*P(Y^2)
          50     8E  40FD     0C16  1612          ADDG2   (SP)+, R0               ; R0/R1 = YLO + Y*P(Y^2)
          50     56  40FD     0C1A  1613          ADDG2   R6, R0                  ; R0/R1 = Y + Y*P(Y^2) = GSIN(Y)
                          05  0C1E  1614          RSB
                              0C1F  1615
                              0C1F  1616
                              0C1F  1617
```

```
                       0C1F          .SBTTL  CYCLE_POLYNOMIALS          ; Polynomials for arguments in cycles
                       0C1F
                       0C1F
                       0C1F
                       0C1F     ;
                       0C1F     ; Polynomial evaluation for GCOS(Y) for Y in cycles
                       0C1F     ;
                       0C1F
                       0C1F     P_COS_C:
   56   F40C CF 51FD   0C1F          CMPG    G_2_OV_PI, R6             ; Compare 2/pi with |YHI|
             36   18   0C25          BGEQ    2$                        ; Sufficent overhang is available
        7E   54   7D   0C27          MOVQ    R4, -(SP)                 ; Save YLO
   7E   54   56 41FD   0C2A          ADDG3   R6, R4, -(SP)             ; Save Y
   50   6E   6E 45FD   0C2F          MULG3   (SP), (SP), R0            ; R0/R1 = Y^2
F5ED CF 07   50 55FD   0C34          POLYG   R0, #COSLENC2-1, COSTBC2; R0/R1 = Q(Y^2)
   54   56   8E 41FD   0C3B          ADDG3   (SP)+, R6, R4             ; R4/R5 = Y + YHI
        54   8E 44FD   0C40          MULG2   (SP)+, R4                 ; R4/R5 = YLO*(Y + YHI) = A2
             03   13   0C44          BEQL    1$                        ; Check for A2 = 0
        54   20   A2   0C46          SUBW2   #^X0020, R4               ; R4/R5 = A2/4
        50   54 42FD   0C49     1$:  SUBG2   R4, R0                    ; R0/R1 = Q(Y^2) - A2/4
        56   56 44FD   0C4D          MULG2   R6, R6                    ; R6/R7 = YHI^2
        56   20   A2   0C51          SUBW2   #^X0020, R6               ; R6/R7 = YHI^2/4
        56   08 42FD   0C54          SUBG2   #1, R6                    ; R6/R7 = -(1 - YHI^2/4)
        50   56 42FD   0C58          SUBG2   R6, R0                    ; R0/R1 = GCOS(Y)
                  05   0C5C          RSB
                       0C5D
        56   54 40FD   0C5D     2$:  ADDG2   R4, R6                    ; R6/R7 = Y
        56   56 44FD   0C61          MULG2   R6, R6                    ; R6/R7 = Y^2
F57C CF 07   56 55FD   0C65          POLYG   R6, #COSLENC1-1, COSTBC1; R0/R1 = GCOS(Y) - 1
        50   08 40FD   0C6C          ADDG2   #1, R0                    ; R0/R1 = GCOS(Y)
                  05   0C70          RSB
                       0C71
                       0C71
                       0C71     ;
                       0C71     ; Polynomial evaluation for -GCOS(Y)
                       0C71     ;
                       0C71
                       0C71     N_COS_C:
   56   F3BA CF 51FD   0C71          CMPG    G_2_OV_PI, R6             ; Compare 2/pi with |YHI|
             37   18   0C77          BGEQ    2$                        ; Sufficent overhang is available
        7E   54   7D   0C79          MOVQ    R4, -(SP)                 ; Save YLO
   7E   54   56 41FD   0C7C          ADDG3   R6, R4, -(SP)             ; Save Y
   50   6E   6E 45FD   0C81          MULG3   (SP), (SP), R0            ; R0/R1 = Y^2
F59B CF 07   50 55FD   0C86          POLYG   R0, #COSLENC2-1, COSTBC2; R0/R1 = Q(Y^2)
   54   56   8E 41FD   0C8D          ADDG3   (SP)+, R6, R4             ; R4/R5 = Y + YHI
        54   8E 44FD   0C92          MULG2   (SP)+, R4                 ; R4/R5 = YLO*(Y + YHI) = A2
             03   13   0C96          BEQL    1$                        ; Check for A2 = 0
        54   20   A2   0C98          SUBW2   #^X0020, R4               ; R4/R5 = A2/4
        50   54 42FD   0C9B     1$:  SUBG2   R4, R0                    ; R0/R1 = Q(Y^2) - A2/4
        56   56 44FD   0C9F          MULG2   R6, R6                    ; R6/R7 = YHI^2
        56   20   A2   0CA3          SUBW2   #^X0020, R6               ; R6/R7 = YHI^2/4
        56   08 42FD   0CA6          SUBG2   #1, R6                    ; R6/R7 = -(1 - YHI^2/4)
   50   56   50 43FD   0CAA          SUBG3   R0, R6, R0                ; R0/R1 = -GCOS(Y)
                  05   0CAF          RSB
                       0CB0
        56   54 40FD   0CB0     2$:  ADDG2   R4, R6                    ; R6/R7 = Y
        56   56 44FD   0CB4          MULG2   R6, R6                    ; R6/R7 = Y^2
```

```
        F529 CF   07    56 55FD  0CB8          POLYG   R6, #COSLENC1-1, COSTBC1; R0/R1 = GCOS(Y) - 1
50 00000000 0000C010 8F 50 43FD  0CBF          SUBG3   R0, #-1, R0             ; R0/R1 = -GCOS(Y)
                          05     0CCC          RSB
                                 0CCD
                                 0CCD    ;
                                 0CCD    ; Polynomial evaluation for -SIN(Y)
                                 0CCD    ;
                                 0CCD
                                 0CCD    N_SIN_C:
          54   8000 8F      AC   0CCD          XORW    #^X8000, R4             ;
          56   8000 8F      AC   0CD2          XORW    #^X8000, R6             ; R4/R7 = - Y
                                 0CD7
                                 0CD7    ;
                                 0CD7    ; Polynomial evaluation for GSIN(Y)
                                 0CD7    ;
                                 0CD7
                                 0CD7    P_SIN_C:
               7E    54     7D   0CD7          MOVQ    R4, -(SP)               ; Save YLO
          7E   56    54   41FD   0CDA          ADDG3   R4, R6, -(SP)           ; Save Y
          54   6E    6E   45FD   0CDF          MULG3   (SP), (SP), R4          ; R4 = Y^2
     F57D CF   06    54   55FD   0CE4          POLYG   R4, #SINLENC-1, SINTBC  ; R0/R1 = P(Y^2)
               50    8E   44FD   0CEB          MULG2   (SP)+, R0               ; R0/R1 = Y*P(Y^2)
               50    6E   40FD   0CEF          ADDG2   (SP), R0                ; R0/R1 = Y*P(Y^2) + YLO
               6E    20     A2   0CF3          SUBW2   #^X0020, (SP)           ; (SP) = YLO/4
               50    8E   42FD   0CF6          SUBG2   (SP)+, R0               ; R0/R1 = Y*P(Y^2) + 3/4*YLO
               54    56     7D   0CFA          MOVQ    R6, R4                  ; R4/R5 = YHI
               54    20     A2   0CFD          SUBW2   #^X0020, R4             ; R4/R5 = YHI/4
               56    54   42FD   0D00          SUBG2   R4, R6                  ; R6/R7 = 3/4*YHI
               50    56   40FD   0D04          ADDG2   R6, R0                  ; R0/R1 = GSIN(Y)
                          05     0D08          RSB
                                 0D09
```

```
                                0D09      .SBTTL  DEGREE_POLYNOMIALS
                                0D09
                                0D09
                                0D09 P_COS_D:
          56    F352 CF 51FD    0D09            CMPG    G_90_OV_PI, R6          ; Compare 90/pi with Y
                  40     18    0D0F            BGEQ    2$                      ; Double precision isn't needed
          54    56    56 45FD    0D11            MULG3   R6, R6, R4              ; R0/R1 = Y^2
     F5C3 CF    07    54 55FD    0D16            POLYG   R4, #COSDLN1, COSDTB1   ; R0/R1 = Q(Y^2)
   54    56   00070000 8F    CB    0D1D            BICL3   #^X70000, R6, R4        ;
                                0D25
                  55     D4    0D25            CLRL    R5                      ; R4/R5 = YHI
          52    56    54 43FD    0D27            SUBG3   R4, R6, R2              ; R2 = YLO
                56    54 40FD    0D2C            ADDG2   R4, R6                  ; R6/R7 = Y + YHI
                56    52 44FD    0D30            MULG2   R2, R6                  ; R6/R7 = YLO*(Y + YHI) = A2
                  05     13    0D34            BEQL    1$                      ; Check for A2 = 0
          56    00D0 8F    A2    0D36            SUBW2   #^X00D0, R6             ; R6/R7 = A2/2^13
                50    56 42FD    0D3B 1$:        SUBG2   R6, R0                  ; R0/R1 = Q(Y^2) - A2/2^13
                54    54 44FD    0D3F            MULG2   R4, R4                  ; R4/R5 = YHI^2
          54    00D0 8F    A2    0D43            SUBW2   #^X00D0, R4             ; R4/R5 = YHI^2/2^13
                54    08 42FD    0D48            SUBG2   #1, R4                  ; R4/R5 = -(1 - YHI^2/2^13)
                50    54 42FD    0D4C            SUBG2   R4, R0                  ; R0/R1 = GCOS(Y)
                         05    0D50            RSB
                                0D51
          56          56 44FD    0D51 2$:        MULG2   R6, R6                  ; R6/R7 = Y^2
                  08     13    0D55            BEQL    3$                      ; Check for Y = 0
     F542 CF    07    56 55FD    0D57            POLYG   R6, #COSDLN2, COSDTB2   ; R0/R1 = Q(Y^2)
                         05    0D5E            RSB
                                0D5F
          50          08 50FD    0D5F 3$:        MOVG    #1, R0                  ; R0/R1 = GCOS(Y)
                         05    0D63            RSB
                                0D64
                                0D64
                                0D64 N_COS_D:
          56    F2F7 CF 51FD    0D64            CMPG    G_90_OV_PI, R6          ; Compare 90/pi with Y
                  41     18    0D6A            BGEQ    2$                      ; Double precision isn't needed
          54    56    56 45FD    0D6C            MULG3   R6, R6, R4              ; R0/R1 = Y^2
     F568 CF    07    54 55FD    0D71            POLYG   R4, #COSDLN1, COSDTB1   ; R0/R1 = Q(Y^2)
   54    56   00070000 8F    CB    0D78            BICL3   #^X70000, R6, R4        ;
                  55     D4    0D80            CLRL    R5                      ; R4/R5 = YHI
          52    56    54 43FD    0D82            SUBG3   R4, R6, R2              ; R2 = YLO
                56    54 40FD    0D87            ADDG2   R4, R6                  ; R6/R7 = Y + YHI
                56    52 44FD    0D8B            MULG2   R2, R6                  ; R6/R7 = YLO*(Y + YHI) = A2
                  05     13    0D8F            BEQL    1$                      ; Check for A2 = 0
          56    00D0 8F    A2    0D91            SUBW2   #^X00D0, R6             ; R6/R7 = A2/2^13
                50    56 42FD    0D96 1$:        SUBG2   R6, R0                  ; R0/R1 = Q(Y^2) - A2/2^13
                54    54 44FD    0D9A            MULG2   R4, R4                  ; R4/R5 = YHI^2
          54    00D0 8F    A2    0D9E            SUBW2   #^X00D0, R4             ; R4/R5 = YHI^2/2^13
                54    08 42FD    0DA3            SUBG2   #1, R4                  ; R4/R5 = -(1 - YHI^2/2^13)
          50    54    50 43FD    0DA7            SUBG3   R0, R4, R0              ; R0/R1 = -GCOS(Y)
                         05    0DAC            RSB
                                0DAD
          56          56 44FD    0DAD 2$:        MULG2   R6, R6                  ; R6/R7 = Y^2
                  0D     13    0DB1            BEQL    3$                      ; Check for Y = 0
     F4E6 CF    07    56 55FD    0DB3            POLYG   R6, #COSDLN2, COSDTB2   ; R0/R1 = GCOSD(Y)
          50    8000 8F    AC    0DBA            XORW    #^X8000, R0             ; R0/R1 = -GCOSD(Y)
                         05    0DBF            RSB
                                0DC0
```

```
     50   00000000 0000C010 8F 50FD  0DC0  3$:       MOVG    #-1, R0                  ; R0/R1 = GCOS(Y)
                                05   0DCC            RSB
                                     0DCD
                                     0DCD  N_SIN_D:
                    56     56 52FD   0DCD            MNEGG   R6, R6                   ; R6/R7 = -Y
                                     0DD1  P_SIN_D:
              50    56     56 45FD   0DD1            MULG3   R6, R6, R0               ; R0/R1 = Y^2
                           14   13   0DD6            BEQL    RETURN
         F541 CF    06     50 55FD   0DD8            POLYG   R0, #SINDLN, SINDTB      ; R0/R1 = P(Y^2)
                    50     56 44FD   0DDF            MULG2   R6, R0                   ; R0/R1 = Y*P(Y^2)
              56    0060 8F   A2     0DE3            SUBW2   #^X0060, R6              ; R6/R7 = Y/2^6
                    50     56 40FD   0DE8            ADDG2   R6, R0                   ; R0/R1 = GSIN(Y)
                                05   0DEC  RETURN:   RSB
                                     0DED
```

```
                                  0DED
                                  0DED
                                  0DED              .SBTTL  DEGENERATE_SOLUTIONS
                                  0DED
                                  0DED  P_ONE:
               50     08 50FD     0DED              MOVG    #1, R0                      ; Answer is 1
                           05     0DF1              RSB
                                  0DF2
                                  0DF2
                                  0DF2  N_ONE:
     50   00000000 0000C010 8F 50FD  0DF2           MOVG    #-1, R0                     ; Answer is -1
                           05     0DFE              RSB
                                  0DFF
                                  0DFF
                                  0DFF  UNFL:
                                  0DFF  ;
                                  0DFF  ; Underflow; if user has FU set, signal error.  Always return 0.0
                                  0DFF  ;
                     52    DC     0DFF              MOVPSL  R2                          ; R2 = user's or jacket routine's PSL
         00000000'GF  00   FB     0E01              CALLS   #0, G^MTH$$JACKET_TST       ; R0 = TRUE if JSB from jacket routine
                  04  50   E9     0E08              BLBC    R0, 10$                     ; branch if user did JSB
            52    04  AD   3C     0E0B              MOVZWL  SF$W_SAVE_PSW(FP), R2       ; get user PSL saved by CALL
                      50   D4     0E0F  10$:        CLRL    R0                          ; R0 = result. LIB$SIGNAL will save in
                                  0E11                                                  ; CHF$L_MCH_R0/R1 so any handler can
                                  0E11                                                  ;   fixup
               0D 52  06   E1     0E11              BBC     #6, R2, 20$                 ; has user enabled floating underflow?
                      6E   DD     0E15              PUSHL   (SP)                        ; yes, return PC from special routine
            7E    00'8F    9A     0E17              MOVZBL  #MTH$K_FLOUNDMAT, -(SP)     ; trap code for hardware floating
                                  0E1B                                                  ;   underflow convert to MTH$_FLOUNDMAT
                                  0E1B                                                  ;   (32-bit VAX-11 exception code)
         00000000'GF  02   FB     0E1B              CALLS   #2, G^MTH$$SIGNAL           ; signal (condition, PC)
                           05     0E22  20$:        RSB                                 ; return
                                  0E23
                                  0E23              .END
```

```
CONVERT                          000008CA R     02
CONVERT_0                        00000998 R     02
CONVERT_1                        00000946 R     02
COSD                           = 0000000C
COSDLN1                        = 00000007
COSDLN2                        = 00000007
COSDTB1                          000002E0 R     02
COSDTB2                          000002A0 R     02
COSINE                         = 0000000C
COSLENC1                       = 00000008
COSLENC2                       = 00000008
COSLENR1                       = 00000008
COSLENR2                       = 00000008
COSTBC1                          000001E8 R     02
COSTBC2                          00000228 R     02
COSTBR1                          00000130 R     02
COSTBR2                          00000170 R     02
CVT                              00000B35 R     02
CVT_TO_DOUBLE                    00000A0B R     02
DEGENERATE_CASE_COS              0000060B R     02
DEGENERATE_CASE_SIN              00000576 R     02
EVAL_COSD                        000006F9 R     02
EVAL_SIND                        000006A0 R     02
EVEN                             00000B39 R     02
GEN_MORE_BITS                    000009C5 R     02
GET_YHI_YLO                      000009B1 R     02
G_1_OV_45                        00000050 R     02
G_2_OV_PI                        00000030 R     02
G_2_TO_32                        000007BA R     02
G_3_PI_OV_4                      00000018 R     02
G_45                             00000038 R     02
G_5_PI_OV_4                      00000020 R     02
G_7_PI_OV_4                      00000028 R     02
G_90_OV_PI                       00000060 R     02
G_9_PI_OV_4                      00000010 R     02
G_CONVERT                        00000058 R     02
G_M1                             00000000 R     02
G_M45                            00000040 R     02
G_PI_OV_4                        00000008 R     02
G_SMALLD                         00000048 R     02
G_SMALLEST_DEG                   00000068 R     02
LARGE_COS                        000005ED R     02
LARGE_SIN                        00000558 R     02
LARGE_SINCOS                     000004D1 R     02
LAST_STEP                        00000AE1 R     02
LEADING_ONES                     00000907 R     02
LEADING_ZEROS                    0000096C R     02
LEQL_HALF                        00000B8B R     02
LONG                           = 00000004
LOOP_0                           00000973 R     02
LOOP_1                           0000090E R     02
L_COS                            000005F3 R     02
L_INT_WEIGHT                   = 000003D0
L_SIN                            0000055E 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                         0000038A RG    02
MTH$GCOSD                        000003D1 RG    02
MTH$GCOSD_R7                     000006E3 RG    02
MTH$GCOS_R7                      0000058A RG    02
MTH$GSIN                         00000375 RG    02
MTH$GSINCOS                      00000358 RG    02
MTH$GSINCOSD                     0000039F RG    02
MTH$GSINCOSD_R7                  0000061F RG    02
MTH$GSINCOS_R7                   000003E6 RG    02
MTH$GSIND                        000003BC RG    02
MTH$GSIND_R7                     0000067D RG    02
MTH$GSIN_R7                      000004FB RG    02
MTH$K_FLOUNDMAT                  ********   X   00
M_COS                            000005A8 R     02
M_SIN                            00000526 R     02
NEEDS_DOUBLE                     00000B55 R     02
NEG_SIND                         0000068D R     02
NOT_ENOUGH_BITS                  0000079B R     02
NO_OVERFLOW                      00000B10 R     02
N_COS_C                          00000C71 R     02
N_COS_D                          00000D64 R     02
N_COS_R                          00000B9B R     02
N_ONE                            00000DF2 R     02
N_SIN_C                          00000CCD R     02
N_SIN_D                          00000DCD R     02
N_SIN_R                          00000BF4 R     02
ODD                              00000B26 R     02
PI_OV_2                          00000070 R     02
POS_SIN                          00000510 R     02
POS_SINCOS                       000003FC R     02
POS_SIND                         00000692 R     02
POWER_MOD_360_0                  00000A5C R     02
POWER_MOD_360_1                  00000A74 R     02
PSL$M_IV                       = 00000020
P_COS_C                          00000C1F R     02
P_COS_D                          00000D09 R     02
P_COS_R                          00000B48 R     02
P_ONE                            00000DED R     02
P_SIN_C                          00000CD7 R     02
P_SIN_D                          00000DD1 R     02
P_SIN_R                          00000BFE R     02
REDUCE_DEGREES                   00000A8C R     02
REDUCE_LARGE                     000007C2 R     02
REDUCE_MEDIUM                    00000720 R     02
RESTORE                          000009BE R     02
RETURN                           00000DEC R     02
SF$W_SAVE_PSW                  = 00000004
SIN                              0000050B R     02
SINCOS                           000003F7 R     02
SINCOSD                          00000634 R     02
SIND                           = 00000008
SINDLN                         = 00000006
SINDTB                           00000320 R     02
SINE                           = 00000008
SINLENC                        = 00000007
SINLENR                        = 00000007
```

```
SINTBC                            00000268 R     02
SINTBR                            000001B0 R     02
SMALL_COS                         000005BC R     02
SMALL_COSD                        0000070D R     02
SMALL_SIN                         0000053A R     02
SMALL_SINCOS                      00000439 R     02
SMALL_SINCOSD                     00000660 R     02
SMALL_SIND                        000006B4 R     02
SUBTRACT                          00000751 R     02
UNFL                              00000DFF R     02
W_ADJUST                        = 000003B6
W_MAX_WEIGHT                    = 00000400
W_TERM_WEIGHT                   = 00000200
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                         00000E23  (  3619.)  02 (  2.)    PIC    USR    CON    REL    LCL   SHR   EXE   RD   NOWRT NOVEC LONG

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

Phase                        Page faults    CPU Time       Elapsed Time
-----                        -----------    --------       ------------
Initialization                       33    00:00:00.08    00:00:01.14
Command processing                  119    00:00:00.64    00:00:03.50
Pass 1                              208    00:00:05.88    00:00:18.83
Symbol table sort                     0    00:00:00.26    00:00:00.52
Pass 2                              318    00:00:03.91    00:00:10.83
Symbol table output                  16    00:00:00.14    00:00:00.64
Psect synopsis output                 3    00:00:00.02    00:00:00.02
Cross-reference output                0    00:00:00.00    00:00:00.00
Assembler run totals                699    00:00:10.95    00:00:35.56

The working set limit was 1650 pages.
39338 bytes (77 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 195 non-local and 57 local symbols.
1873 source lines were read in Pass 1, producing 36 object records in Pass 2.
10 pages of virtual memory were used to define 9 macros.

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

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

131 GETS were required to define 5 macros.
```

There were no errors, warnings or information messages.

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

0265 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

OTSPOWHH LIS

OTSPOWII LIS

OTSPOWRJ LIS

UVXPOWGG LIS

OTSPOWGLU LIS

OTSPOWHLU LIS

OTSPOWHJ LIS

UVXPOWCJ LIS

UVXPOWRR LIS

OTSPOWLUL LIS

OTSPOWRR LIS

OTSPOWJJ LIS

UVXEXP LIS

UVXGSINCO LIS

OTSPOWRLU LIS