```
SOURCE ACCESS NAME=      PPC2.P359.SRC.TRINSIC
OBJECT ACCESS NAME=      PPC2.P359.OBJ.TRINSICS
LISTING ACCESS NAME=     PPC2.P359.LST.TRINSICS
ERROR ACCESS NAME=
OPTIONS=                 XREF
MACRO LIBRARY PATHNAME=


LINE     KEY   NAME

 0002      A   VERSION
                    =>PPC2.P359.SRC.P359
```

```
0032                    IDT  'TRNSIC'
0033          ********************************************************************
0034          *
0035          *  TTTTT  RRRRR   IIIII  N      N   SSSS   IIIII   CCCC      *
0036          *    T    R    R    I    NN     N  S    S    I    C    C     *
0037          *    T    R    R    I    N N    N  S         I    C          *
0038          *    T    RRRRR     I    N  N   N   SSSS     I    C          *
0039          *    T    R  R      I    N   N  N       S    I    C          *
0040          *    T    R   R     I    N    NN  S    S    I    C    C     *
0041          *    T    R    R  IIIII  N      N   SSSS   IIIII   CCCC      *
0042          *                                                            *
0043          *                                                            *
0044          *          PPPP      3333     555555     9999               *
0045          *          P   P    3    3    5         9    9              *
0046          *          P   P         3    5         9    9              *
0047          *          PPPP       3333    55555     99999              *
0048          *          P             3         5        9               *
0049          *          P         3   3    5    5    9    9              *
0050          *          P          3333     5555     9999               *
0051          *                                                            *
0052          ********************************************************************
```

```
0057                        DEF   PWR$$,LOG$$,EXP$$,SQR$$
0058                        DEF   TAN$$,ATN$$,SIN$$,COS$$
0059                        DEF   ROLOUT,ROLIN
0060                        DEF   GRINT
0061                  *
0062                        REF   VDPWD,WRVDP,GROM,GRMWA
0063                        REF   GETV1,PUTV1,GET,GETV,C16,C100
0064                        REF   CFI,SMULT,FDIV,FLTONE
0065                        REF   SCOMPB,OVEXP,SSUB,FMULT,FCOMPB,FADD
0066                        REF   SDIV,FSUB,SADD
0067                        REF   R1LB,R2LB,R3LB,R5LB,R9LB,R12LB
0068                        REF   VROA$,FPSIGN,PROA$,P$,Q$,C$,INTRIN
0069                        REF   ARG,FAC,VSPTR,VSPTR1,PAD,POPSTK,STKADD
0070                        REF   ARG1,ARG2,FAC1,FAC8,FAC10,CBHA
0071                        REF   VRAM,VDPRD,SIGN,EXP,ROUNU
0072 0000
0073 0000
0074 0000 4101    CBH411    DATA  >4101
0075 0002
0076 0002   3F    CBH3F     BYTE  >3F
0077 0003   44    CBH44     BYTE  >44
0078 0004                   EVEN
0079                  *
0080                  *VROA$   EQU   >3C0          VDP ROLL OUT AREA
0081                  *FPSIGN  EQU   >3DC
0082                  *PROA$   EQU   PAD+>10       PROCESSOR ROLL OUT AREA
0083                  *P$      EQU   PAD+>12
0084                  *Q$      EQU   PAD+>16
0085                  *C$      EQU   PAD+>1A
0086                  *SGN$    EQU   PAD+>75
0087                  *EXP$    EQU   PAD+>76
0088                  *OE$     EQU   PAD+>14
```

```
0090    0000  EXC127  EQU  >00
0091    0008  FHALF   EQU  >08
0092    0010  SQRTEN  EQU  >10
0093    0018  LOG10E  EQU  >18
0094    0020  LN10    EQU  >20
0095    0028  PI2     EQU  >28
0096    0030  RPI2    EQU  >30
0097    0038  PI4     EQU  >38
0098    0040  TANPI8  EQU  >40
0099    0048  TAN3P8  EQU  >48
0100    0050  SQRP    EQU  >50
0101    006A  SQRQ    EQU  >6A
0102    006A  FPOS1   EQU  >6A
0103    007C  EXPP    EQU  >7C
0104    0096  EXPQ    EQU  >96
0105    00B8  LOGP    EQU  >B8
0106    00E2  LOGQ    EQU  >E2
0107    010C  SINP    EQU  >10C
0108    014E  ATNP    EQU  >14E
```

```
0111                ***************************************************************
0112                *       INVOLUTION                                              *
0113                *                                                               *
0114                *       FAC - exponent                                          *
0115                *       Top of stack - base                                     *
0116                *                                                               *
0117                *       If integer base and integer exponent do multiplies      *
0118                *       to keep result exact; otherwise, use logarithms to      *
0119                *       calculate value                                         *
0120                ***************************************************************
0121 0004          PWR$$
0122 0004 C28B             MOV   R11,R10
0123 0006 06A0             BL    @SAVRTN             SAVE RETURN
     0008 0624'
0124 000A 06A0             BL    @POPSTK             GET Base INTO ARG
     000C 0000
0125 000E C020             MOV   @FAC,R0             IF Exponent=0
     0010 0000
0126 0012 1359             JEQ   PWRG01              THEN RESULT = 1
0127 0014 C020             MOV   @ARG,R0             IF Base=0
     0016 0000
0128 0018 1352             JEQ   PWRG02              THEN RETURN 0 OR WARNING
0129 001A A820             A     @C8,@VSPTR          USE Base ON STACK
     001C 0666'
     001E 0000
0130 0020 06A0             BL    @PUSH               CHECK TO SEE IF E IS FLOATING
     0022 0664'
0131                *                                    INTEGER
0132 0024 06A0             BL    @GRINT              Convert 1 copy of exp to INT
     0026 055E'
0133 0028 D820             MOVB  @C8,@SIGN           ASSUME SIGN IS POSITIVE
     002A 0666'
     002C 0000
0134 002E 06A0             BL    @XTFAC$             FAC=ARG  STACK = INT(ARG)
     0030 06A6'
0135 0032 06A0             BL    @SCOMPB             Integer exponent?
     0034 0000
0136 0036 164D             JNE   PWR$$3              NO, TRY FLOATING CODE
0137                *
0138                *       COMPUTE INTEGER POWER B^E
0139                *
0140 0038 06A0             BL    @PUSH               PUT Exp above Base ON STACK
     003A 0664'
0141 003C D820             MOVB  @C8,@FAC10          ASSUME NO ERROR
     003E 0666'
     0040 0000
0142 0042 06A0             BL    @CFI                TRY TO CONVERT E TO INTEGER
     0044 0000
0143 0046 0760   CBH7      ABS   @FAC                ABSOLUTE VALUE OF EXPONENT
     0048 0010'
0144 004A C320             MOV   @FAC,R12            SAVE INTEGER EXPONENT
     004C 0048'
0145 004E 06A0             BL    @POP                RETURN E TO FAC; B ON STACK
     0050 0688'
0146 0052 D020             MOVB  @FAC10,R0           IF E > 32767
     0054 0040'
0147 0056 1648             JNE   PWR$$1                 RETURN TO FLOATING PT. CODE
0148 0058 06A0             BL    @XTFAC$             GET Base in accumulator
     005A 06A6'
0149 005C 06A0             BL    @PUSH               PUT E ON STACK FOR LATER SIGN
```

```
     005E 0664'
0150                 *                              CHECK
0151 0060 060C            DEC  R12                  REDUCE EXPONENT BY ONE SINCE
0152                 *                                 ACCUMULATOR STARTS WITH B
0153 0062 1312            JEQ  PWRJ40               IF 0 THEN DONE ALREADY
0154 0064 091C  PWRJ30    SRL  R12,1                CHECK L.S. BIT
0155 0066 1705            JNC  PWRJ10               IF 0, SKIP THE WORK
0156 0068 06A0            BL   @SMULT               MULTIPLY IN THIS POWER
     006A 0000
0157 006C A820            A    @C8,@VSPTR           RESTORE STACK
     006E 0666'
     0070 001E'
0158 0072 C30C  PWRJ10    MOV  R12,R12              FINISHED?
0159 0074 1309            JEQ  PWRJ40               YES
0160 0076 06A0            BL   @XTFAC$              NO, EXCHANGE: B IN FAC; ACCUM
     0078 06A6'
0161                 *                                 ON STACK
0162 007A 06A0            BL   @PUSH                COPY B ONTO STACK
     007C 0664'
0163 007E 06A0            BL   @SMULT               SQUARE IT FOR NEW B
     0080 006A'
0164 0082 06A0            BL   @XTFAC$              RESTORE ORDER; B ON STACK
     0084 06A6'
0165                 *                                 ACCUMULATOR IN FAC
0166 0086 10EE            JMP  PWRJ30               LOOP FOR NEXT BIT
0167                 *
0168 0088 6820  PWRJ40    S    @C16,@VSPTR          DONE, CLEAN UP
     008A 0000
     008C 0070'
0169 008E C0E0            MOV  @VSPTR,R3            GET STACK POINTER
     0090 008C'
0170 0092 0223            AI   R3,8                 TEST EXPONENT SIGN NOW
     0094 0008
0171 0096 06A0            BL   @GETV1               GET IT
     0098 0000
0172 009A 1102            JLT  PWRJ41               IF NEGATIVE, COMPUTE NEGATIVE
0173                 *
0174 009C 0460  PWRRTN    B    @ROLIN2              Use common code to return
     009E 0652'
0175                 *
0176 00A0 D020  PWRJ41    MOVB @FAC10,R0            IF OVERFLOW HAS OCCURRED
     00A2 0054'
0177 00A4 1606            JNE  PWRJ45                  GO MAKE IT ZERO
0178 00A6 06A0            BL   @MOVROM              GET A FLOATING POINT ONE
     00A8 05E2'
0179 00AA 006A            DATA FPOS1                   INTO ARG
0180 00AC 06A0            BL   @FDIV                COMPUTE THE INVERSE
     00AE 0000
0181 00B0 10F5            JMP  PWRRTN                  AND RETURN
0182                 *
0183 00B2 04E0  PWRJ45    CLR  @FAC                 IF OVERFLOW, THE RESULT = 0
     00B4 004C'
0184 00B6 D820            MOVB @FAC,@FAC10          INDICATE NO ERROR
     00B8 00B4'
     00BA 00A2'
0185 00BC 10EF            JMP  PWRRTN               AND RETURN
0186                 *
0187 00BE D020  PWRG02    MOVB @FAC,R0              IS Exp NEGATIVE?
     00C0 00B8'
0188 00C2 1139            JLT  PWRG05               YES-divide by 0 =>put in ovflw
```

```
0189 00C4 10F6          JMP   PWRJ45            NO, RESULT IS ZERO AND RETURN
0190                    *
0191 00C6 0200  PWRGO1  LI    R0,FAC            Need to put floating 1 in FAC
     00C8 00C0'
0192 00CA 06A0          BL    @MOVRM1           GET THE FLOATING 1
     00CC 05E6'
0193 00CE 006A          DATA  FPOS1                INTO FAC
0194 00D0 10E5          JMP   PWRRTN            AND RETURN
0195                    *
0196 00D2 06A0  PWR$$3  BL    @GETV             CHECK FOR NEGATIVE
     00D4 0000
0197 00D6 0090'         DATA  VSPTR             ON THE STACK
0198 00D8 1517          JGT   PWR$$2            IF OK
0199 00DA D820          MOVB  @ERRNIP,@FAC10    ELSE ERROR CODE
     00DC 0130'
     00DE 00BA'
0200 00E0 6820          S     @C8,@VSPTR        THROW AWAY ENTRY ON STACK
     00E2 0666'
     00E4 00D6'
0201 00E6 10DA          JMP   PWRRTN            AND RETURN
0202                    *
0203                    *
0204                    *       INTEGER EXPONENT OUT OF INTEGER RANGE
0205                    *
0206 00E8 06A0  PWR$$1  BL    @GETV             POSITIVE OR NEGATIVE BASE?
     00EA 00D4'
0207 00EC 00E4'         DATA  VSPTR
0208 00EE 150C          JGT   PWR$$2            POSITIVE BASE
0209                    *       Negative base-so see if exponent is even or odd
0210                    *       to set the sign of the result
0211 00F0 04C1  PWR$$4  CLR   R1                FOR DOUBLE
0212 00F2 D060          MOVB  @FAC,R1           GET EXPONENT
     00F4 00C8'
0213 00F6 0741          ABS   R1                WORK WITH POSITIVE
0214 00F8 0281          CI    R1,>4600          TOO BIG TO HAVE ONE'S BYTE?
     00FA 4600
0215 00FC 1505          JGT   PWR$$2            YES, ASSUME NUMBER IS EVEN
0216 00FE 06C1          SWPB  R1                GET IN LOW ORDER BYTE
0217                    *-----------------CONDITIONAL ASSEMBLY----------------------*
0218                    ASMIF VERS=DX10
0219
0220                    AI    R1,>000B          POINT INTO THE FAC
0221                    AI    R1,PAD            ADD CPU-RAM OFFSET
0222
0223                    ASMELS
0224 0100
0225 0100 0221          AI    R1,>830B          NO, GET ONE'S RADIX DIGIT
     0102 830B
0226                    *                          LOCATION IN FAC
0227                    ASMEND
0228                    *-----------------END OF CONDITIONAL ASSEMBLY--------------*
0229 0104 D051          MOVB  *R1,R1            GET THE DIGIT
0230 0106 0A71          SLA   R1,7              IF LAST BIT SET, SET TOP BIT
0231 0108 0204  PWR$$2  LI    R4,FPSIGN         SAVE SIGN OF RESULT
     010A 0000
0232 010C 06A0          BL    @PUTV1               IN A PERMANENT PLACE
     010E 0000
0233 0110 06A0          BL    @XTFAC$           Base IN FAC; Exponent ON STACK
     0112 06A6'
0234 0114 0760          ABS   @FAC              MUST WORK WITH POSITIVE
```

```
                0116 00F4'
0235 0118 06A0          BL    @LOG$$            COMPUTE LOG(B) IN FAC
     011A 0234'
0236 011C 06A0          BL    @SMULT            COMPUTE E*LOG(B) IN FAC
     011E 0080'
0237 0120 06A0          BL    @EXP$$            LET EXP GIVE ERROR OR WARNING
     0122 013C'
0238 0124 0203          LI    R3,FPSIGN         CHECK SIGN OF RESULT
     0126 010A'
0239 0128 06A0          BL    @GETV1
     012A 0098'
0240 012C 1101          JLT   PWR$$5            IF E IS NEGATIVE
0241 012E 10B6          JMP   PWRRTN            IF E IS POSITIVE
0242      0130' ERRNIP  EQU   $
0243 0130 0520  PWR$$5  NEG   @FAC              MAKE IT NEGATIVE
     0132 0116'
0244 0134 10B3          JMP   PWRRTN
0245              *
0246 0136         PWRG05
0247 0136 06A0          BL    @OVEXP            RETURN OVERFLOW
     0138 0000
0248 013A 10B0          JMP   PWRRTN            AND RETURN
0249              *
```

```
0252                ****************************************************************
0253                *       EXPONENTIAL FUNCTION                                   *
0254                *                                                              *
0255                *       FAC := EXP(FAC)                                        *
0256                *                                                              *
0257                *       CALL:  BL   @EXP$$                                     *
0258                *                                                              *
0259                *       WARNINGS:   WRNOV  OVERFLOW                            *
0260                *                                                              *
0261                *       STACK LEVELS USED:                                     *
0262                *            X := FAC * LOG10(E)                               *
0263                *            SO EXP(FAC) = 10^X                                *
0264                *            MAKE SURE X IS IN RANGE  LOG100(X) = LOG10(X)/2*
0265                *            N := INT(X)                                       *
0266                *            R := X-N, 0 <= R < 1                              *
0267                *            IF R < .5 THEN S := R                             *
0268                *                      ELSE S := R - .5                        *
0269                *       A RATIONAL FUNCTION APPROXIMATION IS USED FOR 10^S      *
0270                *       (HART EXPD 1444)                                       *
0271                *       EXP := IF R .LT. .5 THEN 10^N * 10^S                   *
0272                *                           ELSE 10^N * 10^.5 * 10^S           *
0273                ****************************************************************
0274 013C          EXP$$
0275 013C C28B             MOV  R11,R10
0276 013E 06A0             BL   @ROLOUT          GET WORKSPACE AND SAVE RETURN
     0140 0602'
0277 0142 06A0             BL   @MOVROM          GET LOG10(E)
     0144 05E2'
0278 0146 0018             DATA LOG10E              INTO ARG
0279 0148 06A0             BL   @FMULT           X := FAC * LOG10(E)
     014A 0000
0280 014C 06A0             BL   @PUSH            SAVE X
     014E 0664'
0281 0150 06A0             BL   @GRINT           COMPUTE N := INT(X)
     0152 055E'
0282 0154 06A0             BL   @MOVROM          GET FLOATING 127
     0156 05E2'
0283 0158 0000             DATA EXC127              INTO ARG
0284 015A 06A0             BL   @FCOMPB          IS N > 127?
     015C 0000
0285 015E 1313             JEQ  EXP03            IF = 127
0286 0160 1106             JLT  EXP01            IF > 127
0287 0162 0520             NEG  @ARG             CHECK NEGATIVE RANGE
     0164 0016'
0288 0166 06A0             BL   @FCOMPB          IS N < -127?
     0168 015C'
0289 016A 110D             JLT  EXP03            N > -127
0290 016C 130C             JEQ  EXP03            N = -127
0291                *
0292                *      N IS OUT OF RANGE
0293                *
0294 016E 6820     EXP01   S    @C8,@VSPTR       POP X OFF STACK
     0170 0666'
     0172 00EC'
0295 0174 C820             MOV  @FAC,@EXP        RECALL EXPONENT SIGN
     0176 0132'
     0178 0000
0296 017A D820             MOVB @C8,@SIGN        RESULT IS POSITIVE
     017C 0666'
     017E 002C'
```

```
0297 0180 06A0          BL   @OVEXP          TAKE OVER OR UNDERFLOW ACTION
     0182 0138'
0298 0184 1055          JMP  BROLIN          RESTORE CPU RAM AND RETURN
0299                *
0300 0186 06A0  EXP03   BL   @PUSH           SAVE VALUE ON STACK
     0188 0664'
0301 018A 06A0          BL   @CFI            CONVERT TO INTEGER EXPONENT
     018C 0044'
0302 018E C320          MOV  @FAC,R12        GET IT IN REG TO MPY BY 2
     0190 0176'
0303 0192 0A1C          SLA  R12,1           COMPUTE 2*N
0304 0194 06A0          BL   @POP            RESTORE VALUE
     0196 0688'
0305 0198 06A0          BL   @SSUB           COMPUTE R = X - N
     019A 0000
0306 019C 06A0          BL   @MOVROM         GET A FLOATING .5
     019E 05E2'
0307 01A0 0008          DATA FHALF              INTO ARG
0308 01A2 06A0          BL   @FCOMPB         IS .5 > R?
     01A4 0168'
0309 01A6 1505          JGT  EXP04           YES, S=R
0310 01A8 0520          NEG  @ARG            -.5
     01AA 0164'
0311 01AC 06A0          BL   @FADD           COMPUTE S := R - .5
     01AE 0000
0312 01B0 058C          INC  R12             REMEMBER R >= .5, (2*N+1)
0313 01B2 06A0  EXP04   BL   @PUSH           SAVE A COPY OF S
     01B4 0664'
0314 01B6 06A0          BL   @POLYW          COMPUTE S * P(S^2)
     01B8 02F4'
0315 01BA 007C          DATA EXPP            POLY TO EVALUATE
0316 01BC 06A0          BL   @XTFAC$         FAC = S, STACK = S * P(S^2)
     01BE 06A6'
0317 01C0 06A0          BL   @POLYX          COMPUTE Q(S^2)
     01C2 0318'
0318 01C4 0096          DATA EXPQ            POLY TO EVALUATE
0319 01C6 06A0          BL   @POPSTK         S * P(S^2) -> ARG
     01C8 000C'
0320 01CA A820          A    @C8,@VSPTR
     01CC 0666'
     01CE 0172'
0321 01D0 06A0          BL   @PUSH           SAVE COMP OF Q(S^2)
     01D2 0664'
0322 01D4 06A0          BL   @FADD           Q(S^2) + S * P(S^2)
     01D6 01AE'
0323 01D8 0203          LI   R3,FAC          SAVE FAC IN A TEMP
     01DA 0190'
0324 01DC 0204          LI   R4,C$
     01DE 0000
0325 01E0 CD33          MOV  *R3+,*R4+       1ST TWO BYTES
0326 01E2 CD33          MOV  *R3+,*R4+       2ND TWO BYTES
0327 01E4 CD33          MOV  *R3+,*R4+       3RD TWO BYTES
0328 01E6 C513          MOV  *R3,*R4         LAST TWO BYTES
0329 01E8 06A0          BL   @POP            FAC = Q(S^S),STACK = S*P(S^2
     01EA 0688'
0330 01EC 06A0          BL   @XTFAC$         REVERSE SAME
     01EE 06A6'
0331 01F0 06A0          BL   @SSUB           COMPUTE Q(S^2)-S*P(S^2)
     01F2 019A'
0332 01F4 0203          LI   R3,C$           GET FAC BACK FROM TEMP
```

```
      01F6 01DE'
 0333 01F8 0204          LI    R4,ARG
      01FA 01AA'
 0334 01FC CD33          MOV   *R3+,*R4+        1ST TWO BYTES
 0335 01FE CD33          MOV   *R3+,*R4+        2ND TWO BYTES
 0336 0200 CD33          MOV   *R3+,*R4+        3RD TWO BYTES
 0337 0202 C513          MOV   *R3,*R4          LAST TWO BYTES
 0338 0204 06A0          BL    @FDIV            COMPUTE Q + P/Q - P
      0206 00AE'
 0339 0208         EXPSQT
 0340 0208 081C          SRA   R12,1            CHECK FLAG THAT WAS SET ABOVE
 0341 020A 1705          JNC   EXPSQ5           IF NOT SET
 0342 020C 06A0          BL    @MOVROM          GET SQR(10)
      020E 05E2'
 0343 0210 0010          DATA  SQRTEN             INTO ARG
 0344 0212 06A0          BL    @FMULT           MULT BY SQR(10) IF N ODD
      0214 014A'
 0345 0216 06A0  EXPSQ5  BL    @MOVROM          NEED A FLOATING 1
      0218 05E2'
 0346 021A 006A          DATA  FPOS1              INTO ARG
 0347 021C 081C          SRA   R12,1            CHECK ODD POWER OF TEN
 0348 021E 1703          JNC   EXPSQ8           IF NOT ODD POWER
 0349 0220 D820          MOVB  @CBHA,@ARG1      ODD POWER OF TEN (>0A)
      0222 0000
      0224 0000
 0350 0226 B820  EXPSQ8  AB    @R12LB,@ARG      ADD IN POWER OF 100 TO EXP
      0228 0000
      022A 01FA'
 0351 022C 06A0          BL    @FMULT
      022E 0214'
 0352 0230 0460  BROLIN  B     @ROLIN
      0232 0636'
 0353              *
```

```
0356                ********************************************************************
0357                *       LOGARITHM FUNCTION                                         *
0358                *                                                                  *
0359                *       FAC := LOG(FAC)                                            *
0360                *                                                                  *
0361                *       ERRORS: ERRLOG   LOG OF NEGATIVE NUMBER OR ZERO            *
0362                *                        ATTEMPTED.                                *
0363                *                                                                  *
0364                *       STACK LEVELS USED:                                         *
0365                *            IF FAC <= 0 THEN ERRLOG                               *
0366                *            LOG(FAC)=LN(FAC)=LOG10(FAC) * LN(10)                  *
0367                *            FAC := A * 10^N,   .1 <= A < 1                        *
0368                *            S := A * SQR(10), 1/SQR(10) <= S < SQR(10)            *
0369                *            LOG10(A) := LOG10(S/SQR(10))                          *
0370                *                     :=LOG10(S) - LOG10(SQR(10))                  *
0371                *                     :=LOG10(S) - .5                              *
0372                *            LOG := (N - .5 + LOG10(S)) * LN(10)                   *
0373                *                := (N - .5 * LN(10) + LN(S)                       *
0374                *       A RATIONAL FUNCTION APPROXIMATION IS USED FOR LN(S)        *
0375                *       (HART LOGE 2687)                                           *
0376                ********************************************************************
0377 0234          LOG$$
0378 0234 C28B             MOV   R11,R10
0379 0236 06A0             BL    @ROLOUT              GET WORKSPACE AND SAVE RETURN
     0238 0602'
0380 023A C020             MOV   @FAC,R0              CHECK FOR NEGATIVE OR ZERO
     023C 01DA'
0381 023E 1504             JGT   LOG$$3               IF POSITIVE
0382 0240 D820             MOVB  @ERRLOG,@FAC10       LOAD ERROR CODE
     0242 0248'
     0244 00DE'
0383 0246 10F4             JMP   BROLIN               RESTORE CPU AND RETURN
0384                *
0385       0248' ERRLOG    EQU   $
0386 0248 06A0   LOG$$3    BL    @CNSTEN              GET BASE 10 EXPONENT
     024A 06D6'
0387 024C 160B             JNE   LOG$$5
0388 024E 06A0             BL    @MOVROM              GET A FLOATING 1
     0250 05E2'
0389 0252 006A             DATA  FPOS1                   INTO ARG
0390                *                                MAKE IT A FLOATING 10
0391 0254 D820             MOVB  @CBHA,@ARG1             BY PUTTING IN >0A
     0256 0222'
     0258 0224'
0392 025A 06A0             BL    @FMULT               MULTIPLY FAC BY 10
     025C 022E'
0393 025E 06A0             BL    @CNSTEN              GET NEW EXPONENT OF 10
     0260 06D6'
0394 0262 1002             JMP   LOG$5A               COMPENSATE FOR MULT
0395 0264 05A0   LOG$$5    INC   @EXP                 COMPENSATE FOR WHERE RADIX
     0266 0178'
0396                *                                   POINT IS
0397                *                                PUT  A  IN PROPER RANGE
0398 0268 D820   LOG$5A    MOVB  @CBH3F,@FAC             BY PUTTING IN >3F
     026A 0002'
     026C 023C'
0399 026E C320             MOV   @EXP,R12
     0270 0266'
0400 0272 06A0             BL    @MOVROM              GET SQR(10)
     0274 05E2'
```

```
0401 0276 0010          DATA SQRTEN                INTO ARG
0402 0278 06A0          BL   @FMULT                S := A * SQR(10)
     027A 025C'
0403 027C 06A0          BL   @FORMA                Z := (S-1) / (S+1)
     027E 036E'
0404 0280 06A0          BL   @PUSH                 PUSH Z
     0282 0664'
0405 0284 06A0          BL   @POLYW                COMPUTE Z * P(Z^2)
     0286 02F4'
0406 0288 00B8          DATA LOGP                  POLY TO EVALUATE
0407 028A 06A0          BL   @XTFAC$
     028C 06A6'
0408 028E 06A0          BL   @POLYX                COMPUTE Q(Z^2)
     0290 0318'
0409 0292 00E2          DATA LOGQ                  POLY TO EVALUATE
0410 0294 06A0          BL   @SDIV                 COMPUTE Z*P(Z^2)/Q(Z^2)
     0296 0000
0411 0298 06A0          BL   @PUSH                 PUSH IT
     029A 0664'
0412 029C 0200          LI   R0,ARG                BUILD ENTRY IN ARG
     029E 022A'
0413 02A0 CC0C          MOV  R12,*R0+              PUT IN EXPONENT
0414 02A2 04F0          CLR  *R0+                     AND
0415 02A4 04F0          CLR  *R0+                        CLEAR THE
0416 02A6 04D0          CLR  *R0                            REST
0417              *     STATUS WAS SET BY THE MOVE ABOVE
0418 02A8 130E          JEQ  LOG$$7                IF ZERO EXPONENT
0419 02AA 0760          ABS  @ARG                  WORK WITH ABS VALUE
     02AC 029E'
0420 02AE C020          MOV  @ARG,R0                  IN REGISTER
     02B0 02AC'
0421 02B2 0280          CI   R0,99                 TOO LARGE?
     02B4 0063
0422 02B6 1514          JGT  LOG$$9                YES
0423 02B8 D820          MOVB @FLTONE,@ARG          EXPONENT = >40
     02BA 0000
     02BC 02B0'
0424 02BE D30C  LOG$$6  MOVB R12,R12               EXPONENT POSITIVE?
0425 02C0 1302          JEQ  LOG$$7                YES
0426 02C2 0520          NEG  @ARG                  NO, MAKE IT NEGATIVE
     02C4 02BC'
0427 02C6 06A0  LOG$$7  BL   @MOVRM5               NEED A FLOATING .5
     02C8 05DC'
0428 02CA 0008          DATA FHALF                    IN FAC
0429 02CC 06A0          BL   @FSUB                 COMPUTE N - .5
     02CE 0000
0430 02D0 06A0          BL   @MOVROM               NEED LN(10)
     02D2 05E2'
0431 02D4 0020          DATA LN10                     INTO ARG
0432 02D6 06A0          BL   @FMULT                COMPUTE (N-.5) * LN(10)
     02D8 027A'
0433 02DA 06A0          BL   @SADD                 ADD TO LN(S)
     02DC 0000
0434 02DE 10A8          JMP  BROLIN                RESTORE CPU AND RETURN
0435              *
0436 02E0 6820  LOG$$9  S    @C100,@ARG            SUBTRACT FIRST 100
     02E2 0000
     02E4 02C4'
0437 02E6 D820          MOVB @ARG1,@ARG2
     02E8 0258'
```

```
      02EA 0000
0438                *                              LOAD EXPONENT AND
0439 02EC C820              MOV   @CBH411,@ARG         LEADING DIGIT OF >4101
     02EE 0000'
     02F0 02E4'
0440 02F2 10E5              JMP   LOG$$6
0441                *
```

```
0444                ***************************************************************
0445                *                                                             *
0446                *       EVALUATE X * P(X^^2)                                  *
0447                *                                                             *
0448                *       ON CALL:   P$    POINTER TO POLYNOMIAL COEFFICIENTS   *
0449                *                  FAC   CONTAINS X                           *
0450                *                  BL    @POLYW                               *
0451                *                  FAC   RETURNS X * P(X^^2)                  *
0452                *                                                             *
0453                ***************************************************************
0454 02F4 C83B  POLYW   MOV   *R11+,@P$          GET THE POLY TO EVALUATE
     02F6 0000
0455 02F8 C28B          MOV   R11,R10
0456 02FA 06A0          BL    @SAVRTN           SAVE RETURN ADDRESS
     02FC 0624'
0457 02FE 06A0          BL    @PUSH             PUSH THE ARGUMENT
     0300 0664'
0458 0302 06A0          BL    @POLYX1           COMPUTE P(X^^2)
     0304 031C'
0459 0306 06A0          BL    @SMULT            COMPUTE X * P(X^^2)
     0308 011E'
0460 030A 104E          JMP   PWRTN2            AND RETURN
0461                *
0462 030C C83B  POLY    MOV   *R11+,@P$
     030E 02F6'
0463 0310 C28B          MOV   R11,R10
0464 0312 06A0          BL    @SAVRTN           SAVE RETURN ADDRESS
     0314 0624'
0465 0316 1009          JMP   POLY01            AND MERGE IN BELOW
0466                *
0467 0318 C83B  POLYX   MOV   *R11+,@P$
     031A 030E'
0468 031C C28B  POLYX1  MOV   R11,R10
0469 031E 06A0          BL    @SAVRTN           SAVE RETURN ADDRESS
     0320 0624'
0470 0322 06A0          BL    @PUSH             NEED TO COPY FAC
     0324 0664'
0471                *                               INTO ARG TO SQUARE IT
0472 0326 06A0          BL    @SMULT            SQUARE X (SMULT POPS INTO ARG)
     0328 0308'
0473 032A       POLY01
0474 032A 06A0          BL    @PUSH             PUSH THE ARGUMENT
     032C 0664'
0475 032E C0E0          MOV   @P$,R3            GET THE POLY TO EVALUATE
     0330 031A'
0476 0332 0200          LI    R0,FAC                INTO FAC
     0334 026C'
0477 0336 06A0          BL    @MOVRM2
     0338 05E8'
0478 033A 100F          JMP   POLY03
0479                *
0480 033C 06A0  POLY02  BL    @POPSTK           GET X BACK
     033E 01C8'
0481 0340 A820          A     @C8,@VSPTR        KEEP IT ON STACK
     0342 0666'
     0344 01CE'
0482 0346 06A0          BL    @FMULT            MULTIPLY PREVIOUS RESULT BY X
     0348 02D8'
0483 034A C0E0          MOV   @P$,R3
     034C 0330'
```

```
0484 034E 0200          LI    R0,ARG              GET POLYNOMIAL TO EVALUATE
     0350 02F0'
0485 0352 06A0          BL    @MOVRM2                INTO ARG
     0354 05E8'
0486 0356 06A0          BL    @FADD               ADD IN THIS COEFFICIENT
     0358 01D6'
0487 035A A820  POLY03  A     @C8,@P$             POINT TO NEXT COEFFICIENT
     035C 0666'
     035E 034C'
0488                    *                         AND GET FIRST TWO BYTES
0489                    *                            INTO ARG
0490                    *--------------------CONDITIONAL ASSEMBLY--------------------
0491                           ASMIF VERS=DX10
0492
0493                           CB    *R15+,@CBH80  READ SECOND BYTE
0494
0495                           ASMELS
0496 0360
0497 0360 981D                 CB    *R13,@CBH80   READ FIRST BYTE
     0362 046F'
0498 0364
0499                           ASMEND
0500                    *--------------------END OF CONDITIONAL ASSEMBLY------------
0501                    *                         AND TEST IT TO SEE IF DONE
0502 0364 16EB                 JNE   POLY02        NO, CONTINUE COMPUTING POLY
0503 0366 6820                 S     @C8,@VSPTR    POP X OFF STACK
     0368 0666'
     036A 0344'
0504 036C 101D                 JMP   PWRTN2        RETURN WITH POLY IN FAC
0505                    *
```

```
0508 036E C28B  FORMA   MOV   R11,R10
0509 0370 06A0          BL    @SAVRTN           SAVE RETURN ADDRESS
     0372 0624'
0510 0374 06A0          BL    @PUSH             SAVE X ON STACK
     0376 0664'
0511 0378 06A0          BL    @FORMA2
     037A 0394'
0512 037C 06A0          BL    @FORMA2
     037E 0394'
0513 0380 06A0          BL    @XTFAC$           SWAP (X-1) AND X
     0382 06A6'
0514 0384 06A0          BL    @MOVROM           GET A FLOATING 1
     0386 05E2'
0515 0388 006A          DATA  FPOS1                INTO ARG
0516 038A 06A0          BL    @FADD             X+1
     038C 0358'
0517 038E 06A0          BL    @SDIV             (X - 1) / (X + 1)
     0390 0296'
0518 0392 100A          JMP   PWRTN2            AND RETURN
0519              *
0520 0394 C28B  FORMA2  MOV   R11,R10
0521 0396 06A0          BL    @SAVRTN           SAVE RETURN ADDRESS
     0398 0624'
0522 039A 06A0          BL    @MOVROM           GET A FLOATING .5
     039C 05E2'
0523 039E 0008          DATA  FHALF                INTO ARG
0524 03A0 0520          NEG   @ARG
     03A2 0350'
0525 03A4 06A0          BL    @FADD             X - .5
     03A6 038C'
0526 03A8       PWRTN2
0527 03A8 0460          B     @ROLIN2
     03AA 0652'
0528              *
```

```
0531                    ***************************************************************
0532                    *
0533                    *       SQUARE ROOT FUNCTION
0534                    *
0535                    *       REFERENCE FOR SCIENTIFIC FUNCTION APPROXIMATIONS,
0536                    *       JOHN F. HART, ET AL,  COMPUTER APPROXIMATIONS,
0537                    *         JOHN WILEY & SONS, 1968.
0538                    *
0539                    *       FAC := SQR(FAC)
0540                    *
0541                    *       ERRORS:  ERRSQR   SQUARE ROOT OF NEGATIVE NUMBER
0542                    *                         ATTEMPTED.
0543                    *
0544                    *       STACK LEVELS USED:
0545                    *            IF FAC = 0 THEN SQR := 0
0546                    *            IF FAC < 0 THEN ERRSQR
0547                    *            FAC := A * 100^N,  .01 <= A < 1
0548                    *            SQR := 10^N * SQR(A)
0549                    *
0550                    *       NEWTON'S METHOD WITH A FIXED NUMBER OF ITERATIONS IS
0551                    *       USED TO APPROXIMATE SQR(A):
0552                    *       A RATIONAL FUNCTION APPROXIMATION IS USED FOR Y(0)
0553                    *           (HART SQRT 0231)
0554                    *
0555                    *       Y(N+1) = (Y(N))/2
0556                    *
0557                    ***************************************************************
0558 03AC              SQR$$
0559 03AC C28B                  MOV   R11,R10
0560 03AE 06A0                  BL    @ROLOUT           GET WORKSPACE AND SAVE RETURN
     03B0 0602'
0561 03B2 C320                  MOV   @FAC,R12          CHECK EXPONENT
     03B4 0334'
0562 03B6 1334                  JEQ   SQR03             FAC IS ZERO, RETURN ZERO
0563 03B8 1130                  JLT   SQR02             FAC IS < 0, ERROR
0564 03BA D820                  MOVB  @CBH3F,@FAC       CREATE A IN RANGE .01 <=  A <
     03BC 0002'
     03BE 03B4'
0565                    *                                  BY LOADING >3F
0566 03C0 022C                  AI    R12,>C100         REMOVE BIAS (-63)
     03C2 C100
0567 03C4 088C                  SRA   R12,8             SIGN EXTEND
0568 03C6 0A1C                  SLA   R12,1             SAVE 2 * N
0569 03C8 06A0                  BL    @PUSH             SAVE A
     03CA 0664'
0570 03CC 06A0                  BL    @PUSH             SAVE A AGAIN
     03CE 0664'
0571 03D0 06A0                  BL    @POLY             COMPUTE P(A)
     03D2 030C'
0572 03D4 0050                  DATA  SQRP              POLY TO EVALUATE
0573 03D6 06A0                  BL    @XTFAC$           STACK := P(A),FAC := A
     03D8 06A6'
0574 03DA 06A0                  BL    @POLY             COMPUTE Q(A)
     03DC 030C'
0575 03DE 006A                  DATA  SQRQ              POLY TO EVALUATE
0576 03E0 06A0                  BL    @SDIV             COMPUTE P(A)/Q(A)
     03E2 0390'
0577 03E4 C820                  MOV   @C3,@P$           SAVE IN PERMANENT
     03E6 047A'
     03E8 035E'
```

```
0578 03EA 06A0  SQR01  BL    @POPSTK           POP INTO ARG
     03EC 033E'
0579 03EE A820         A     @C8,@VSPTR        BUT KEEP IT ON STACK
     03F0 0666'
     03F2 036A'
0580 03F4 06A0         BL    @PUSH             PUSH Y(N)
     03F6 0664'
0581 03F8 06A0         BL    @FDIV             COMPUTE A/Y(N)
     03FA 0206'
0582 03FC 06A0         BL    @SADD             COMPUTE A/Y(N) + Y(N)
     03FE 02DC'
0583 0400 06A0         BL    @MOVROM           NEED A FLOATING .5
     0402 05E2'
0584 0404 0008         DATA  FHALF                INTO ARG
0585 0406 06A0         BL    @FMULT            COMPUTE .5 * (A/Y(N) + Y(N))
     0408 0348'
0586 040A 0620         DEC   @P$               DECREMENT LOOP COUNTER
     040C 03E8'
0587 040E 16ED         JNE   SQR01             LOOP THREE TIMES
0588 0410 6820         S     @C8,@VSPTR        POP OFF STACK
     0412 0666'
     0414 03F2'
0589 0416 0460         B     @EXPSQT           TO FINISH UP
     0418 0208'
0590              *
0591 041A D820  SQR02  MOVB  @ERRSQR,@FAC10    LOAD ERROR CODE FOR RETURN
     041C 0420'
     041E 0244'
0592      0420' ERRSQR EQU   $
0593 0420 0460  SQR03  B     @ROLIN            RESTORE CPU RAM AND RETURN
     0422 0636'
0594              *
```

```
0597                    ********************************************************************
0598                    *
0599                    *       COSINE FUNCTION
0600                    *
0601                    *       FAC := COS(FAC)
0602                    *
0603                    *       COS(FAC) := SIN(FAC + PI/2)
0604                    *
0605                    ********************************************************************
0606 0424              COS$$
0607 0424 C30B                 MOV   R11,R12
0608 0426 06A0                 BL    @MOVROM           NEED TO GET PI / 2
     0428 05E2'
0609 042A 0028                 DATA  PI2                  INTO ARG
0610 042C 06A0                 BL    @FADD             COMPUTE FAC + PI/2
     042E 03A6'
0611 0430 C2CC                 MOV   R12,R11           AND FALL INTO SIN CODE
0612                    *
```

```
0615                ****************************************************************
0616                *                                                              *
0617                *       SINE FUNCTION                                          *
0618                *                                                              *
0619                *       FAC := SIN(FAC)                                        *
0620                *                                                              *
0621                *       STACK LEVELS USED:                                     *
0622                *            IF FAC < 0 THEN SIN(FAC) := -SIN(-FAC)            *
0623                *            X := 2/PI * FAC                                   *
0624                *            K := INT(X)                                       *
0625                *            R := X - K,   0 <= R < 1                          *
0626                *            Q := K MOD 4                                      *
0627                *            SO K := 4 * N + Q                                 *
0628                *            FAC := PI/2 * K + PI/2 * R                        *
0629                *                := 2*PI*N + PI/2*Q + PI/2*R                   *
0630                *            SIN(FAC) := SIN(PI/2 * Q + PI/2 * R)              *
0631                *                                                              *
0632                *       QUADRANT  Q  IDENTITY                                  *
0633                *       I         0  SIN(FAC) := SIN(PI/2 * R)                 *
0634                *       II        1  SIN(FAC) := SIN(PI/2 + PI/2*R)            *
0635                *                            := SIN(PI - (PI/2 + PI/2*R))      *
0636                *                            := SIN(PI/2 * (1 - R))            *
0637                *       III       2  SIN(FAC) := SIN(PI+PI/2*R)                *
0638                *                            := SIN(PI - (PI+PI/2*R))          *
0639                *                            := SIN(PI/2 * (R-1))              *
0640                *       IV        3  SIN(FAC) := SIN(3*PI/2 + PI/2*R)          *
0641                *                            := SIN(3*PI/2 + PI/2*R - 2*PI)    *
0642                *                            := SIN(PI/2 * (R-1))              *
0643                *                                                              *
0644                *       QUADRANT  Q  ARGUMENT TO APPROXIMATION POLYNOMIAL      *
0645                *       I         0        R   = R        0  <=  R   <  1      *
0646                *       II        1      1-R   = 1-R      0  < 1-R   <= 1      *
0647                *       III       2       -R   = -R      -1 <   -R   <= 0      *
0648                *       IV        3      R-1   = -(1-R)  -1 <=  R-1 <  0      *
0649                *                                                              *
0650                *       A POLYNOMIAL APPROXIMATION IS USED FOR SIN(PI/2 * R)   *
0651                *           -1 <= R <1                                         *
0652                *       (HART SIN 3344)                                        *
0653                ****************************************************************
0654 0432           SIN$$
0655 0432 C28B              MOV   R11,R10
0656 0434 06A0              BL    @ROLOUT             GET WORKSPACE AND SAVE RETURN
     0436 0602'
0657 0438 06A0              BL    @MOVROM             GET 2/PI
     043A 05E2'
0658 043C 0030              DATA RPI2                    INTO ARG
0659 043E 06A0              BL    @FMULT              X := 2/PI * FAC
     0440 0408'
0660 0442 D320              MOVB @FAC,R12             SAVE SIGN
     0444 03BE'
0661 0446 0760              ABS   @FAC                CONSIDER POSITIVE NUMBERS
     0448 0444'
0662 044A 9820              CB    @FAC,@CBH44         CHECK EXPONENT RANGE
     044C 0448'
     044E 0003'
0663                *                                    BY CHECKING WITH >44
0664 0450 152C              JGT   TRIERR              ERR IN RANGE OF EXPONENT
0665 0452 06A0              BL    @PUSH               SAVE X
     0454 0664'
0666 0456 06A0              BL    @GRINT              K := INT(K)
```

```
     0458 055E'
0667 045A 04C1          CLR  R1                 ASSUME Q IS ZERO
0668 045C 04C0          CLR  R0
0669 045E D020          MOVB @FAC,R0            IS FAC ZERO?
     0460 044C'
0670 0462 130C          JEQ  SIN02              YES, Q IS ZERO
0671 0464 0220          AI   R0,>BA00           BIAS EXPONENT  (->46 BYTE)
     0466 BA00
0672                *                           IS K TOO BIG FOR (K MOD 4) TO
0673                *                              TO HAVE A SIGNIFICANCE?
0674 0468 1507          JGT  SIN01              YES, DEFAULT Q TO ZERO
0675 046A 0220          AI   R0,>51*256         (FAC+7-PAD)*256
     046C 5100
0676      046F' CBH80   EQU  $+1                CONSTANT >80
0677 046E 0980          SRL  R0,8
0678 0470 0220          AI   R0,PAD
     0472 0000
0679 0474 D810          MOVB *R0,@R1LB          NO, GET 10'S AND 1'S PLACE OF K
     0476 0000
0680      047A' C3      EQU  $+2
0681 0478 0241  SIN01   ANDI R1,3               Q := (K MOD 4)
     047A 0003
0682 047C C801  SIN02   MOV  R1,@Q$
     047E 0000
0683 0480 06A0          BL   @SSUB              R := X - K
     0482 01F2'
0684 0484 C060          MOV  @Q$,R1
     0486 047E'
0685 0488 0911          SRL  R1,1               IS Q EVEN?
0686 048A C801          MOV  R1,@Q$
     048C 0486'
0687 048E 1705          JNC  SIN03              YES
0688 0490 06A0          BL   @MOVROM            GET A FLOATING 1
     0492 05E2'
0689 0494 006A          DATA FPOS1                 INTO ARG
0690 0496 06A0          BL   @FSUB              COMPUTE 1 - R
     0498 02CE'
0691 049A C060  SIN03   MOV  @Q$,R1             QUADRANT III OR IV?
     049C 048C'
0692 049E 1301          JEQ  SIN04              NO
0693 04A0 054C          INV  R12                YES, CHANGE SIGN OF RESULT
0694 04A2 06A0  SIN04   BL   @POLYW             EVALUATE IT
     04A4 02F4'
0695 04A6 010C          DATA SINP               GET POLY P'S COEFFICIENTS
0696 04A8 1054          JMP  ATNSGN             AND SET SIGN
0697 04AA       TRIERR
0698 04AA D820          MOVB @CBH7,@FAC10       TRIG ERROR ( >07 IN FAC+10)
     04AC 0046'
     04AE 041E'
0699 04B0 1054          JMP   ATNSG3
0700                *
0701                *
```

```
0704                    ***************************************************************
0705                    *                                                             *
0706                    *       TANGENT FUNCTION                                      *
0707                    *                                                             *
0708                    *       FAC := TAN(FAC)                                       *
0709                    *                                                             *
0710                    *       TAN(FAC) := SIN(FAC) / COS(FAC)                       *
0711                    *                                                             *
0712                    ***************************************************************
0713 04B2              TAN$$
0714 04B2 C28B                  MOV   R11,R10
0715 04B4 06A0                  BL    @SAVRTN           SAVE RETURN ADDRESS
     04B6 0624'
0716 04B8 06A0                  BL    @PUSH             SAVE FAC ON STACK
     04BA 0664'
0717 04BC 06A0                  BL    @SIN$$            COMPUTE SIN
     04BE 0432'
0718 04C0 06A0                  BL    @XTFAC$
     04C2 06A6'
0719 04C4 06A0                  BL    @COS$$            COMPUTE COS
     04C6 0424'
0720 04C8 06A0                  BL    @POPSTK           POP STACK INTO ARG
     04CA 03EC'
0721 04CC 9820                  CB    @FAC10,@CBH7      CHECK FOR ERROR
     04CE 04AE'
     04D0 0046'
0722 04D2 1305                  JEQ   PWRTN3            IF ERROR
0723 04D4 C020                  MOV   @FAC,R0           IS COS = ZERO?
     04D6 0460'
0724 04D8 1304                  JEQ   TAN01             YES
0725 04DA 06A0                  BL    @FDIV             NO, TAN := SIN(ARG)/COS(ARG)
     04DC 03FA'
0726 04DE 0460  PWRTN3 B        @ROLIN2
     04E0 0652'
0727 04E2 D820  TAN01  MOVB     @ARG,@SIGN
     04E4 03A2'
     04E6 017E'
0728 04E8 06A0                  BL    @OVEXP            ISSUE OVERFLOW MESSAGE
     04EA 0182'
0729 04EC 10F8                  JMP   PWRTN3            CLEAN UP AND EXIT
0730                    *
```

```
0733                ***************************************************************
0734                *
0735                *       INVERSE TANGENT FUNCTION
0736                *                                                               *
0737                *       FAC := ATN(FAC)                                         *
0738                *                                                               *
0739                *       STACK LEVELS USED:                                      *
0740                *            IF FAC < 0 THEN ARCTAN(FAC) = -ARCTAN(-FAC)        *
0741                *            IF 0 <= FAC <= TAN(PI/8)                           *
0742                *                THEN T = FAC, ARCTAN(FAC) := ARCTAN(T)         *
0743                *            IF TAN(PI/8) < FAC < TAN(3*PI/8)                   *
0744                *                THEN T = (FAC-1) / (FAC+1),                    *
0745                *                     ARCTAN(FAC) := PI/4 + ARCTAN(T)           *
0746                *            IF TAN(3*PI/8) <= FAC                              *
0747                *                THEN T = -1/FAC,                               *
0748                *                     ARCTAN(FAC) := PI/2 + ARCTAN(T)           *
0749                *                                                               *
0750                *       A POLYNOMIAL APPROXIMATION IS USED FOR ARCTAN(T),       *
0751                *       -TAN(PI/8) <= T <= TAN(PI/8)                            *
0752                *
0753                *       (HART  ARCTN 4967)
0754                ***************************************************************
0755 04EE          ATN$$
0756 04EE C28B             MOV  R11,R10
0757 04F0 06A0             BL   @ROLOUT            GET WORKSPACE AND SAVE RETURN
     04F2 0602'
0758 04F4 D320             MOVB @FAC,R12           SAVE SIGN
     04F6 04D6'
0759 04F8 0760             ABS  @FAC               USE ABS(FAC)
     04FA 04F6'
0760 04FC 04E0             CLR  @Q$                ASSUME ARG IS IN RANGE
     04FE 049C'
0761 0500 06A0             BL   @MOVROM            NEED TAN(PI/8)
     0502 05E2'
0762 0504 0040             DATA TANPI8                INTO ARG
0763 0506 06A0             BL   @FCOMPB            IS TAN(PI/8) >= ARG?
     0508 01A4'
0764 050A 1317             JEQ  ATN02              IF =
0765 050C 1516             JGT  ATN02              IF >
0766 050E 06A0             BL   @MOVROM            NEED TAN(3*PI/8)
     0510 05E2'
0767 0512 0048             DATA TAN3P8                INTO ARG
0768 0514 06A0             BL   @FCOMPB            IS TAN(3*PI/8) > ARG?
     0516 0508'
0769 0518 150A             JGT  ATN01              YES, USE CASE 2
0770 051A 06A0             BL   @MOVROM            GET A FLOATING 1
     051C 05E2'
0771 051E 006A             DATA FPOS1                 INTO ARG
0772 0520 0520             NEG  @ARG               USE CASE 3 TO COMPUTE
     0522 04E4'
0773 0524 06A0             BL   @FDIV              T = -1/ARG
     0526 04DC'
0774 0528 0203             LI   R3,PI2             GET PI/2
     052A 0028
0775 052C 1004             JMP  ATN02A             ADD IT IN AT THE END
0776                *
0777 052E 06A0     ATN01   BL   @FORMA             CASE 2 : T := (ARG-1)/ARG+1)
     0530 036E'
0778 0532 0203             LI   R3,PI4             GET PI/4
     0534 0038
```

```
0779 0536 C803  ATN02A MOV  R3,@Q$          SET UP TO EVALUATE
     0538 04FE'
0780 053A 06A0  ATN02  BL   @POLYW          ATN(T) := T * P(T^^2)
     053C 02F4'
0781 053E 014E         DATA ATNP            POLY TO EVALUATE
0782 0540 C0E0         MOV  @Q$,R3          CASE 1?
     0542 0538'
0783 0544 1306         JEQ  ATNSGN          YES, DON'T ADD ANYTHING IN
0784 0546 0200         LI   R0,ARG
     0548 0522'
0785 054A 06A0         BL   @MOVRM2
     054C 05E8'
0786 054E 06A0         BL   @FADD           ADD IN THE CONSTANT
     0550 042E'
0787             *
0788 0552 054C  ATNSGN INV  R12             CHECK SIGN OF RESULT
0789 0554 1102         JLT  ATNSG3          IF SIGN IS ALREADY OK
0790 0556 0520         NEG  @FAC               ELSE NEGATE IT
     0558 04FA'
0791 055A 0460  ATNSG3 B    @ROLIN          AND RETURN
     055C 0636'
0792             *
```

```
0795                    ***************************************************************
0796                    *       GREATEST INTEGER FUNCTION
0797                    ***************************************************************
0798 055E               GRINT
0799 055E C1CB                  MOV   R11,R7            SAVE RETURN ADDRESS
0800 0560 D820                  MOVB  @FAC,@SIGN        SAVE RESULT SIGN
     0562 0558'
     0564 04E6'
0801 0566 0760                  ABS   @FAC              ABSOLUTE VALUE
     0568 0562'
0802 056A D160                  MOVB  @FAC,R5           GET EXPONENT
     056C 0568'
0803 056E 0985                  SRL   R5,8              Make into a word
0804 0570 C805                  MOV   R5,@EXP           FOR ROUNDING
     0572 0270'
0805 0574 0285                  CI    R5,>40            EXPONENT < 0?
     0576 0040
0806 0578 1124                  JLT   BITINT            YES - HANDLE IT
0807 057A 0285                  CI    R5,>45            EXPONENT > 10^5 ?
     057C 0045
0808 057E 1519                  JGT   INT02             YES - HANDLE IT
0809 0580 0225                  AI    R5,->46           LOCATE POSITION
     0582 FFBA
0810 0584 D820                  MOVB  @R5LB,@FAC10      SAVE FOR ROUNDING
     0586 0000
     0588 04CE'
0811 058A 04C2                  CLR   R2
0812 058C 0203                  LI    R3,FAC8
     058E 0000
0813 0590 A0C5                  A     R5,R3             POINT TO 1ST FRACTIONAL DIGIT
0814 0592 F093  INT01           SOCB  *R3,R2            REMEMBER IF NON-ZERO
0815 0594 DCE0                  MOVB  @R2LB,*R3+        CLEAR THE DIGIT
     0596 0000
0816 0598 0585                  INC   R5
0817 059A 16FB                  JNE   INT01
0818 059C D020                  MOVB  @SIGN,R0          GET THE SIGN
     059E 0564'
0819 05A0 150D                  JGT   INT03             IF NON-NEGATIVE(ie POSITIVE)
0820 05A2 D082                  MOVB  R2,R2
0821 05A4 1306                  JEQ   INT02
0822 05A6 B820                  AB    @CBH7,@FAC10      WHERE TO ROUND UP
     05A8 0046'
     05AA 0588'
0823 05AC 06A0                  BL    @ROUNU            DO THE ROUNDING
     05AE 0000
0824 05B0 1005                  JMP   INT03
0825                    *
0826 05B2 D020  INT02           MOVB  @SIGN,R0          CHECK THE SIGN
     05B4 059E'
0827 05B6 1502                  JGT   INT03             IF POSITIVE DON'T NEGATE
0828 05B8 0520                  NEG   @FAC              MAKE RESULT NEGATIVE
     05BA 056C'
0829 05BC 04E0  INT03           CLR   @FAC10            INDICATE NO ERROR
     05BE 05AA'
0830 05C0 0457                  B     *R7               <<<<  RETURN FROM HERE
0831                    *
0832 05C2 0200  BITINT          LI    R0,FAC            ZERO OR -1
     05C4 05BA'
0833 05C6 0201                  LI    R1,>BFFF          DEFAULT TO -1
     05C8 BFFF
```

```
0834 05CA D0A0          MOVB  @SIGN,R2          NEGATIVE OR POSITIVE?
     05CC 05B4'
0835 05CE 1101          JLT   INT04             IF REALLY NEGATIVE PUT IN -1
0836 05D0 04C1          CLR   R1                IF POSITIVE PUT IN A 0
0837 05D2 CC01  INT04   MOV   R1,*R0+           COPY IN 0 OR -1
0838 05D4 04F0          CLR   *R0+                AND
0839 05D6 04F0          CLR   *R0+                  CLEAR
0840 05D8 04D0          CLR   *R0                     THE
0841 05DA 10F0          JMP   INT03                     REST
```

```
0844                    *
0845                    *       MOVE 8 BYTES FROM ROM(R3) TO CPU AT R0
0846                    *
0847 05DC 0200  MOVRM5  LI    R0,FAC            MOVE TO FAC
     05DE 05C4'
0848 05E0 1002          JMP   MOVRM1            MERGE INTO COMMON CODE
0849 05E2 0200  MOVROM  LI    R0,ARG            MOVE TO ARG
     05E4 0548'
0850 05E6 C0FB  MOVRM1  MOV   *R11+,R3          CONSTANT TO LOAD
0851 05E8 0202  MOVRM2  LI    R2,8              CONSTANTS ARE 8 BYTES LONG
     05EA 0008
0852 05EC A0E0          A     @INTRIN,R3        ADD IN GROM OFFSET
     05EE 0000
0853                    *--------------------CONDITIONAL ASSEMBLY--------------------*
0854                            ASMIF VERS=DX10
0855
0856                            MOV   R3,R15            PUT INTO GROM READ REGISTER
0857                            AI    R15,GROM          ADD IN GROM OFFSET
0858                    MOVRM4  MOVB  *R15+,*R0+        READ A BYTE
0859
0860                            ASMELS
0861 05F0
0862 05F0 DB43                  MOVB  R3,@GRMWA(R13)    WRITE MSBYTE OF ADDRESS
     05F2 0000
0863 05F4 06C3                  SWPB  R3                BARE THE LSBYTE
0864 05F6 DB43                  MOVB  R3,@GRMWA(R13)    WRITE THE LSBYTE
     05F8 05F2'
0865 05FA DC1D  MOVRM4  MOVB  *R13,*R0+         READ A BYTE
0866 05FC
0867                            ASMEND
0868                    *--------------------END OF CONDITIONAL ASSEMBLY------------*
0869 05FC
0870 05FC 0602                  DEC   R2                MOVED THEM ALL YET?
0871 05FE 16FD                  JNE   MOVRM4            NO-COPY THE NEXT ONE
0872 0600 045B                  RT                      YES-RETURN
```

```
0874                    *
0875                    *       ROLL OUT CPU AREA FOR WORKSPACE
0876                    *
0877 0602 0201  ROLOUT  LI    R1,PROA$          PROCESSOR ROLL OUT AREA
     0604 0000
0878       0608' CVROA$ EQU   $+2
0879 0606 0203          LI    R3,VROA$          VDP ROLL OUT AREA
     0608 0000
0880                    *---------------------CONDITIONAL ASSEMBLY------------------*
0881                            ASMIF VERS=DX10
0882                            MOV   R3,R14
0883                            AI    R14,VRAM
0884                            LI    R0,26
0885                    ROLOT1  MOVB  *R1+,*R14+
0886
0887                            ASMELS
0888 060A
0889 060A D7E0                  MOVB  @R3LB,*R15
     060C 0000
0890 060E 0263                  ORI   R3,WRVDP
     0610 0000
0891 0612 D7C3                  MOVB  R3,*R15
0892 0614 0200                  LI    R0,26
     0616 001A
0893 0618 D831  ROLOT1  MOVB  *R1+,@VDPWD
     061A 0000
0894                            ASMEND
0895                    *-------------------END OF CONDITIONAL ASSEMBLY-------------*
0896 061C 0600                  DEC   R0
0897 061E 16FC                  JNE   ROLOT1
0898 0620 04E0                  CLR   @FAC8             AND SAVE RETURN ADDRESS
     0622 058E'
0899                    *
0900                    *       SAVE RETURN ADDRESS
0901                    *
0902 0624 05E0  SAVRTN  INCT  @STKADD
     0626 0000
0903 0628 D260                  MOVB  @STKADD,R9
     062A 0626'
0904 062C 0989                  SRL   R9,8
0905 062E 0229                  AI    R9,PAD
     0630 0472'
0906 0632 C64A                  MOV   R10,*R9
0907 0634 045B                  RT
```

```
0909                    *
0910                    *       ROLL IN CPU AREA AFTER WORK IS DONE
0911                    *
0912 0636 0201  ROLIN   LI    R1,PROA$          PROCESSOR ROLL OUT AREA
     0638 0604'
0913                    *---------------------CONDITIONAL ASSEMBLY------------------*
0914                            ASMIF VERS=DX10
0915                            LI    R14,VROA$         VDP roll-out area
0916                            AI    R14,VRAM
0917                            LI    R0,26
0918                    ROLIN1  MOVB  *R14+,*R1+
0919
0920                            ASMELS
0921 063A
0922 063A D7E0                  MOVB  @CVROA$+1,*R15    LSByte of address
     063C 0609'
0923 063E D7E0                  MOVB  @CVROA$,*R15      MSByte of address
     0640 0608'
0924 0642 0200                  LI    R0,26             Number of bytes rolled out
     0644 001A
0925 0646 DC60  ROLIN1  MOVB  @VDPRD,*R1+
     0648 0000
0926                            ASMEND
0927                    *------------------END OF CONDITIONAL ASSEMBLY-------------*
0928 064A 0600                  DEC   R0
0929 064C 16FC                  JNE   ROLIN1
0930 064E 04E0                  CLR   @FAC8
     0650 0622'
0931 0652 D260  ROLIN2  MOVB  @STKADD,R9
     0654 062A'
0932 0656 0989                  SRL   R9,8
0933 0658 0229                  AI    R9,PAD
     065A 0630'
0934 065C C2D9                  MOV   *R9,R11
0935 065E 0660                  DECT  @STKADD
     0660 0654'
0936 0662 045B                  RT
```

```
0938                    *
0939                    *       PUSH FAC ONTO STACK
0940                    *
0941        0666'       C8      EQU   $+2
0942 0664 0200          PUSH    LI    R0,8                NUMBER TO PUSH
     0666 0008
0943 0668 A800                  A     R0,@VSPTR           BUMP STACK POINTER
     066A 0414'
0944 066C C060                  MOV   @VSPTR,R1           GET STACK POINTER
     066E 066A'
0945                    *---------------------CONDITIONAL ASSEMBLY-------------------*
0946                            ASMIF VERS=DX10
0947                            MOV   R1,R14
0948                            AI    R14,VRAM            BASE ADDRESS FOR MOVE
0949                            LI    R1,FAC                 SOURCE
0950                    PUSH1   MOVB  *R1+,*R14+          MOVE A BYTE
0951
0952                            ASMELS
0953 0670
0954 0670 D7E0                  MOVB  @R1LB,*R15
     0672 0476'
0955 0674 0261                  ORI   R1,WRVDP
     0676 0610'
0956 0678 D7C1                  MOVB  R1,*R15
0957 067A 0201                  LI    R1,FAC
     067C 05DE'
0958 067E D831          PUSH1   MOVB  *R1+,@VDPWD
     0680 061A'
0959                            ASMEND
0960                    *--------------------END OF CONDITIONAL ASSEMBLY------------*
0961 0682 0600                  DEC   R0
0962 0684 15FC                  JGT   PUSH1
0963 0686 045B                  RT
0964 0688
0965 0688
0966                    *
0967                    *       POP VALUE OFF STACK INTO FAC
0968                    *
0969 0688               POP
0970 0688 0202                  LI    R2,FAC
     068A 067C'
0971                    *--------------------CONDITIONAL ASSEMBLY--------------------*
0972                            ASMIF VERS=DX10
0973                            LI    R0,8
0974                            MOV   @VSPTR,R14
0975                            S     R0,@VSPTR
0976                            AI    R14,VRAM            ADD IN BASE
0977                    POP1    MOVB  *R14+,*R2+          MOVE A BYTE
0978
0979                            ASMELS
0980 068C
0981 068C D7E0                  MOVB  @VSPTR1,*R15        LSByte of address
     068E 0000
0982 0690 0200                  LI    R0,8
     0692 0008
0983 0694 D7E0                  MOVB  @VSPTR,*R15         MSByte of address
     0696 066E'
0984 0698 6800                  S     R0,@VSPTR
     069A 0696'
0985 069C DCA0          POP1    MOVB  @VDPRD,*R2+
```

```
      069E 0648'
0986                        ASMEND
0987                  *-----------------END OF CONDITIONAL ASSEMBLY-------------
0988 06A0 0600              DEC   R0
0989 06A2 15FC              JGT   POP1
0990 06A4 045B              RT
```

```
0992                    *
0993                    *       EXCHANGE TOP OF STACK AND FAC
0994                    *
0995 06A6               XTFAC$
0996 06A6 C28B                  MOV   R11,R10           SAVE RETURN ADDRESS
0997 06A8 06A0                  BL    @PUSH             PUT FAC ON TOP
     06AA 0664'
0998 06AC 0203                  LI    R3,8              WORKING WITH 8-BYTE ENTRIES
     06AE 0008
0999 06B0 C143                  MOV   R3,R5             NEED ANOTHER COPY FOR BELOW
1000 06B2 6803                  S     R3,@VSPTR         POINT BACK TO OLD TOP
     06B4 069A'
1001 06B6 06A0                  BL    @POP              PUT IT IN FAC
     06B8 0688'
1002 06BA A803                  A     R3,@VSPTR         RESTORE PTR TO OLD TOP
     06BC 06B4'
1003 06BE C120                  MOV   @VSPTR,R4         PLACE TO MOVE TO
     06C0 06BC'
1004 06C2 A0C4                  A     R4,R3             PLACE TO MOVE FROM
1005 06C4 06A0  XTFAC1          BL    @GETV1            GET A BYTE
     06C6 012A'
1006 06C8 06A0                  BL    @PUTV1            PUT A BYTE
     06CA 010E'
1007 06CC 0583                  INC   R3
1008 06CE 0584                  INC   R4
1009 06D0 0605                  DEC   R5                DONE?
1010 06D2 16F8                  JNE   XTFAC1            NO
1011 06D4 045A                  B     *R10              YES, RETURN
1012 06D6
1013 06D6
1014                    *
1015                    *       GET BASE 10 EXPONENT OF THE NUMBER IN FAC
1016                    *
1017                    *       EXP:    GETS THE BASE 10 EXPONENT
1018                    *       OE$:    0 IF EXP IS EVEN AND 1 IF EXP IS ODD
1019                    *
1020 06D6 04C0  CNSTEN  CLR   R0                GET BASE 100 EXPONENT
1021 06D8 D020          MOVB  @FAC,R0           PUT IN MSB
     06DA 068A'
1022 06DC 0220          AI    R0,>C000          REMOVE BIAS (SUBT >64
     06DE C000
1023                    *                          FROM MSB)
1024 06E0 0A10          SLA   R0,1              MULTIPLY IT BY 2
1025 06E2 0880          SRA   R0,8              SIGN FILL HIGH ORDER BYTE
1026 06E4 04C3          CLR   R3
1027                    *                          AND PUT IN LSB
1028 06E6 9820          CB    @FAC1,@CBHA       1ST DIGIT OF FAC ONE DECIMAL
     06E8 0000
     06EA 0256'
1029                    *                          DIGIT?
1030 06EC 1102          JLT   CNST10            YES, BASE 10 EXP IS EVEN
1031 06EE 0580          INC   R0                NO, TAKE THIS INTO ACCOUNT IN
1032                    *                          EXPONENT
1033 06F0 0583          INC   R3                THIS MAKES BASE 10 EXP ODD
1034 06F2 C800  CNST10  MOV   R0,@EXP
     06F4 0572'
1035 06F6 C0C3          MOV   R3,R3             SET CONDITION FOR RETURN
1036 06F8 045B          RT
```

```
1039              *       MISCELLANEOUS CONSTANTS:
1040              *CBH411
1041              *EXC127 BYTE >41,1,27,0,0,0,0,0          127
1042              *FHALF  BYTE >3F,50                      .5
1043              *ZER3   BYTE 0,0,0,0,0,0
1044              *SQRTEN BYTE >40,3,16,22,77,66,01,69     SQR(10)
1045              *LOG10E BYTE >3F,43,42,94,48,19,03,25    LOG10(E)
1046              *LN10   BYTE >40,2,30,25,85,09,29,94     LN(10)
1047              *CBH7   EQU  $+3
1048              *PI2    BYTE >40,1,57,7,96,32,67,95      PI/2
1049              *RPI2   BYTE >3F,63,66,19,77,23,67,58    2/PI
1050              *PI4    BYTE >3F,78,53,98,16,33,97,45    PI/4
1051              *CBHA   EQU  $+7
1052              *CBH3F
1053              *TANPI8 BYTE >3F,41,42,13,56,23,73,10    TAN(PI/8)=SQR(2)-1
1054              *TAN3P8 BYTE >40,2,41,42,13,56,23,73     TAN(3*PI/8)=SQR(2)+1
1055              **      SQR POLYNOMIALS  (HART SQRT 0231)
1056              *SQRP   BYTE >3F,58,81,22,90,00,00,00  P02=.58812 29E+00
1057              *       BYTE >3F,52,67,87,50,00,00,00  P01=.52678 75E+00
1058              *       BYTE >3E,58,81,20,00,00,00,00  P00=.58812 E-02
1059              *       DATA SGNBIT
1060              *FLTONE
1061              *FPOS1
1062              *SQRQ   BYTE >40,01,00,00,00,00,00,00  Q01=.1 E+01
1063              *       BYTE >3F,09,99,99,80,00,00,00  Q00=.99999 8 E-01
1064              *       DATA SGNBIT
1065              **      EXP POLYNOMIALS   (HART EXPD 1444)
1066              **      P02 = .18312 36015 92753 84761 54 E+02
1067              *EXPP   BYTE >40,18,31,23,60,15,92,75
1068              **      P01 = .83140 67212 93711 03487 3446 E+03
1069              *       BYTE >41,08,31,40,67,21,29,37
1070              **      P00 = .51780 91991 51615 35743 91297 E+04
1071              *       BYTE >41,51,78,09,19,91,51,62
1072              *       DATA SGNBIT
1073              **      Q03 = .1 E+01
1074              *EXPQ   BYTE >40,1,0,0,0,0,0,0
1075              **      Q02 = .15937 41523 60306 52437 552 E+03
1076              *       BYTE >41,01,59,37,41,52,36,03
1077              **      Q01 = .27093 16940 85158 99126 11636 E+04
1078              *       BYTE >41,27,09,31,69,40,85,16
1079              **      Q00 = .44976 33557 40578 41762 54723 E+04
1080              *       BYTE >41,44,97,63,35,57,40,58
1081              *       DATA SGNBIT
1082              **      LOG POLYNOMIALS  (HART LOGE 2687)
1083              **      P04 = .35670 51030 88437 69 E+00
1084              *LOGP   BYTE >3F,35,67,05,10,30,88,44
1085              **      P03 = -.11983 03331 36876 1464 E+02
1086              *       BYTE >BF,>F5,98,30,33,31,36,88
1087              **      P02 = .63775 48228 86166 05782 E+02
1088              *       BYTE >40,63,77,54,82,28,86,17
1089              **      P01 = -.10883 71223 55838 3228 E+03
1090              *       BYTE >BE,>FF,08,83,71,22,35,58
1091              **      P00 = .57947 38138 44442 78265 7 E+02
1092              *       BYTE >40,57,94,73,81,38,44,44
1093              *       DATA SGNBIT
1094              *LOGQ
1095              **      Q04 = .1 E+01
1096              *       BYTE >40,01,0,0,0,0,0,0
1097              **      Q03 = -.13132 59772 88464 0339 E+02
1098              *       BYTE >BF,>F3,13,25,97,72,88,46
```

```
1099                **      Q02 = .47451 82236 02606 00365 E+02
1100                *       BYTE >40,47,45,18,22,36,02,61
1101                **      Q01 = -.64076 45807 52556 00596 E+02
1102                *       BYTE >BF,>C0,07,64,58,07,52,56
1103                **      Q00 = .28973 69069 22217 71601 9 E+02
1104                *       BYTE >40,28,97,36,90,69,22,22
1105                *       DATA SGNBIT
1106                **      SIN POLYNOMIAL  (HART SIN 3344)
1107                *SINP
1108                **      REFLECTS CHANGE IN 99/4 CONSTANT TO CORRECT VALUES
1109                **      OF SIN AND COS >1
1110                **      P07 = -.64462 13674 9 E-09
1111                **      BYTE >C4,>FA,44,62,13,67,49,00
1112                **      P07 = -.64473 16000 0 E-09
1113                *       BYTE >C4,>FA,44,73,16,00,00,00
1114                **      P06 = .56882 03332 688 E-07
1115                *CBH44  EQU  $+2
1116                *       BYTE >3C,05,68,82,03,33,26,88
1117                **      P05 = -.35988 09117 03133 E-05
1118                *       BYTE >C2,>FD,59,88,09,11,70,31
1119                **      P04 = .16044 11684 69828 31 E-03
1120                *       BYTE >3E,01,60,44,11,68,46,98
1121                **      P03 = -.46817 54131 06023 168 E-02
1122                *       BYTE >C1,>D2,81,75,41,31,06,02
1123                **      P02 = .79692 62624 56180 0806 E-01
1124                *       BYTE >3F,07,96,92,62,62,45,62
1125                **      P01 = -.64596 40975 06219 07082 E+00
1126                *       BYTE >C0,>C0,59,64,09,75,06,22
1127                **      P00 = .15707 96323 79489 63959 E+01
1128                *       BYTE >40,01,57,07,96,32,67,95
1129                *       DATA SGNBIT
1130                **      ATN POLYNOMIAL  (HART ARCTN 4967)
1131                *ATNP
1132                **      P09 = -.25357 18798 82 E-01
1133                *       BYTE >C0,>FE,53,57,18,79,88,20
1134                **      P08 = .50279 13843 885 E-01
1135                *       BYTE >3F,05,02,79,13,84,38,85
1136                **      P07 = -.65069 99940 1396 E-01
1137                *       BYTE >C0,>FA,50,69,99,94,01,40
1138                **      P06 = .76737 12439 1641 E-01
1139                *       BYTE >3F,07,67,37,12,43,91,64
1140                **      P05 = -.90895 47919 67196 E-01
1141                *       BYTE >C0,>F7,08,95,47,91,96,72
1142                **      P04 = .11111 04992 50526 62 E+00
1143                *       BYTE >3F,11,11,10,49,92,50,53
1144                **      P03 = -.14285 71269 75961 157 E+00
1145                *       BYTE >C0,>F2,28,57,12,69,75,96
1146                **      P02 = .19999 99997 89961 5228 E+00
1147                *       BYTE >3F,19,99,99,99,97,89,96
1148                **      P01 = -.33333 33333 32253 4275 E+00
1149                *       BYTE >C0,>DF,33,33,33,33,32,25
1150                **      P00 = .99999 99999 99999 08253 E+00
1151                *       BYTE >40,01,0,0,0,0,0,0
1152                *       DATA SGNBIT
1153                        END
NO ERRORS,      NO WARNINGS
```

```
$          06FA'          0242 0385 0592 0676 0680 0878 0941
ARG     R  05E4'  0069    0127 0287 0310 0333 0350 0412 0419 0420 042
                          0426 0436 0439 0484 0524 0727 0772 0784 084
ARG1    R  02E8'  0070    0349 0391 0437
ARG2    R  02EA'  0070    0437
ATN$$   D  04EE'  0755    0058
ATNO1      052E'  0777    0769
ATNO2      053A'  0780    0764 0765
ATNO2A     0536'  0779    0775
ATNP       014E   0108    0781
ATNSG3     055A'  0791    0699 0789
ATNSGN     0552'  0788    0696 0783
BITINT     05C2'  0832    0806
BROLIN     0230'  0352    0298 0383 0434
C$      R  01F6'  0068    0324 0332
C100    R  02E2'  0063    0436
C16     R  008A'  0063    0168
C3         047A'  0680    0577
C8         0666'  0941    0129 0133 0141 0157 0200 0294 0296 0320 0481
                          0487 0503 0579 0588
CBH3F      0002'  0076    0398 0564
CBH411     0000'  0074    0439
CBH44      0003'  0077    0662
CBH7       0046'  0143    0698 0721 0822
CBH80      046F'  0676    0497
CBHA    R  06EA'  0070    0349 0391 1028
CFI     R  018C'  0064    0142 0301
CNST10     06F2'  1034    1030
CNSTEN     06D6'  1020    0386 0393
COS$$   D  0424'  0606    0058 0719
CVR0A$     0608'  0878    0922 0923
DX10       0001   0003    0004 0218 0491 0854 0881 0914 0946 0972
ERRLOG     0248'  0385    0382
ERRNIP     0130'  0242    0199
ERRSQR     0420'  0592    0591
EXC127     0000   0090    0283
EXP     R  06F4'  0071    0295 0395 0399 0804 1034
EXP$$   D  013C'  0274    0057 0237
EXP01      016E'  0294    0286
EXP03      0186'  0300    0285 0289 0290
EXP04      01B2'  0313    0309
EXPP       007C   0103    0315
EXPQ       0096   0104    0318
EXPSQ5     0216'  0345    0341
EXPSQ8     0226'  0350    0348
EXPSQT     0208'  0339    0589
FAC     R  06DA'  0069    0125 0143 0144 0183 0184 0187 0191 0212 0234
                          0243 0295 0302 0323 0380 0398 0476 0561 0564
                          0660 0661 0662 0669 0723 0758 0759 0790 0800
                          0801 0802 0828 0832 0847 0957 0970 1021
FAC1    R  06E8'  0070    1028
FAC10   R  05BE'  0070    0141 0146 0176 0184 0199 0382 0591 0698 0721
                          0810 0822 0829
FAC8    R  0650'  0070    0812 0898 0930
FADD    R  0550'  0065    0311 0322 0486 0516 0525 0610 0786
FCOMPB  R  0516'  0065    0284 0288 0308 0763 0768
FDIV    R  0526'  0064    0180 0338 0581 0725 0773
FHALF      0008   0091    0307 0428 0523 0584
FLTONE  R  02BA'  0064    0423
FMULT   R  0440'  0065    0279 0344 0351 0392 0402 0432 0482 0585 0659
```

```
LABEL     VALUE   DEFN  REFERENCES

FORMA     036E'   0508  0403  0777
FORMA2    0394'   0520  0511  0512
FPOS1     006A    0102  0179  0193  0346  0389  0515  0689  0771
FPSIGN R  0126'   0068  0231  0238
FSUB   R  0498'   0066  0429  0690
GET    R          0063
GETV   R  00EA'   0063  0196  0206
GETV1  R  06C6'   0063  0171  0239  1005
GRINT  D  055E'   0798  0060  0132  0281  0666
GRMWA  R  05F8'   0062  0862  0864
GROM   R          0062
INTO1     0592'   0814  0817
INTO2     05B2'   0826  0808  0821
INTO3     05BC'   0829  0819  0824  0827  0841
INTO4     05D2'   0837  0835
INTRIN R  05EE'   0068  0852
LN10      0020    0094  0431
LOG$$  D  0234'   0377  0057  0235
LOG$$3    0248'   0386  0381
LOG$$5    0264'   0395  0387
LOG$$6    02BE'   0424  0440
LOG$$7    02C6'   0427  0418  0425
LOG$$9    02E0'   0436  0422
LOG$5A    0268'   0398  0394
LOG10E    0018    0093  0278
LOGP      00B8    0105  0406
LOGQ      00E2    0106  0409
MOVRM1    05E6'   0850  0192  0848
MOVRM2    05E8'   0851  0477  0485  0785
MOVRM4    05FA'   0865  0871
MOVRM5    05DC'   0847  0427
MOVROM    05E2'   0849  0178  0277  0282  0306  0342  0345  0388  0400  0430
                        0514  0522  0583  0608  0657  0688  0761  0766  0770
OVEXP  R  04EA'   0065  0247  0297  0728
P$     R  040C'   0068  0454  0462  0467  0475  0483  0487  0577  0586
P359      0000    0003  0003
PAD    R  065A'   0069  0678  0905  0933
PI2       0028    0095  0609  0774
PI4       0038    0097  0778
POLY      030C'   0462  0571  0574
POLYO1    032A'   0473  0465
POLYO2    033C'   0480  0502
POLYO3    035A'   0487  0478
POLYW     02F4'   0454  0314  0405  0694  0780
POLYX     0318'   0467  0317  0408
POLYX1    031C'   0468  0458
POP       0688'   0969  0145  0304  0329  1001
POP1      069C'   0985  0989
POPSTK R  04CA'   0069  0124  0319  0480  0578  0720
PROA$  R  0638'   0068  0877  0912
PUSH      0664'   0942  0130  0140  0149  0162  0280  0300  0313  0321  0404
                        0411  0457  0470  0474  0510  0569  0570  0580  0665
                        0716  0997
PUSH1     067E'   0958  0962
PUTV1  R  06CA'   0063  0232  1006
PWR$$  D  0004'   0121  0057
PWR$$1    00E8'   0206  0147
PWR$$2    0108'   0231  0198  0208  0215
PWR$$3    00D2'   0196  0136
PWR$$4    00F0'   0211
```

```
PWR$$5    0130'   0243  0240
PWRG01    00C6'   0191  0126
PWRG02    00BE'   0187  0128
PWRG05    0136'   0246  0188
PWRJ10    0072'   0158  0155
PWRJ30    0064'   0154  0166
PWRJ40    0088'   0168  0153  0159
PWRJ41    00A0'   0176  0172
PWRJ45    00B2'   0183  0177  0189
PWRRTN    009C'   0174  0181  0185  0194  0201  0241  0244  0248
PWRTN2    03A8'   0526  0460  0504  0518
PWRTN3    04DE'   0726  0722  0729
Q$      R 0542'   0068  0682  0684  0686  0691  0760  0779  0782
R0        0000          0125  0127  0146  0176  0187  0191  0380  0412  0413
                        0414  0415  0416  0420  0421  0476  0484  0668  0669
                        0671  0675  0677  0678  0679  0723  0784  0818  0826
                        0832  0837  0838  0839  0840  0847  0849  0865  0892
                        0896  0924  0928  0942  0943  0961  0982  0984  0988
                        1020  1021  1022  1024  1025  1031  1034
R1        0001          0211  0212  0213  0214  0216  0225  0229  0229  0230
                        0667  0681  0682  0684  0685  0686  0691  0833  0836
                        0837  0877  0893  0912  0925  0944  0955  0956  0957
                        0958
R10       000A          0122  0275  0378  0455  0463  0468  0508  0520  0559
                        0655  0714  0756  0906  0996  1011
R11       000B          0122  0275  0378  0454  0455  0462  0463  0467  0468
                        0508  0520  0559  0607  0611  0655  0714  0756  0799
                        0850  0934  0996
R12       000C          0144  0151  0154  0158  0158  0302  0303  0312  034J
                        0347  0399  0413  0424  0424  0561  0566  0567  0568
                        0607  0611  0660  0693  0758  0788
R12LB   R 0228'   0067  0350
R13       000D          0497  0862  0864  0865
R15       000F          0889  0891  0922  0923  0954  0956  0981  0983
R1LB    R 0672'   0067  0679  0954
R2        0002          0811  0814  0820  0820  0834  0851  0870  0970  0985
R2LB    R 0596'   0067  0815
R3        0003          0169  0170  0238  0323  0325  0326  0327  0328  0332
                        0334  0335  0336  0337  0475  0483  0774  0778  0779
                        0782  0812  0813  0814  0815  0850  0852  0862  0863
                        0864  0879  0890  0891  0998  0999  1000  1002  1004
                        1007  1026  1033  1035  1035
R3LB    R 060C'   0067  0889
R4        0004          0231  0324  0325  0326  0327  0328  0333  0334  0335
                        0336  0337  1003  1004  1008
R5        0005          0802  0803  0804  0805  0807  0809  0813  0816  0999
                        1009
R5LB    R 0586'   0067  0810
R7        0007          0799  0830
R9        0009          0903  0904  0905  0906  0931  0932  0933  0934
R9LB    R       0067
ROLIN   D 0636'   0912  0059  0352  0593  0791
ROLIN1    0646'   0925  0929
ROLIN2    0652'   0931  0174  0527  0726
ROLOT1    0618'   0893  0897
ROLOUT  D 0602'   0877  0059  0276  0379  0560  0656  0757
ROUNU   R 05AE'   0071  0823
RPI2      0030    0096  0658
SADD    R 03FE'   0066  0433  0582
SAVRTN    0624'   0902  0123  0456  0464  0469  0509  0521  0715
```

```
SCOMPB  R  0034'   0065  0135
SDIV    R  03E2'   0066  0410  0517  0576
SIGN    R  05CC'   0071  0133  0296  0727  0800  0818  0826  0834
SIN$$   D  0432'   0654  0058  0717
SIN01      0478'   0681  0674
SIN02      047C'   0682  0670
SIN03      049A'   0691  0687
SIN04      04A2'   0694  0692
SINP       010C    0107  0695
SMULT   R  0328'   0064  0156  0163  0236  0459  0472
SQR$$   D  03AC'   0558  0057
SQR01      03EA'   0578  0587
SQR02      041A'   0591  0563
SQR03      0420'   0593  0562
SQRP       0050    0100  0572
SQRQ       006A    0101  0575
SQRTEN     0010    0092  0343  0401
SSUB    R  0482'   0065  0305  0331  0683
STKADD  R  0660'   0069  0902  0903  0931  0935
TAN$$   D  04B2'   0713  0058
TAN01      04E2'   0727  0724
TAN3P8     0048    0099  0767
TANPI8     0040    0098  0762
TRIERR     04AA'   0697  0664
VDPRD   R  069E'   0071  0925  0985
VDPWD   R  0680'   0062  0893  0958
VERMAC  M          A0001 0003
VERS       0000    0003  0004  0218  0491  0854  0881  0914  0946  0972
VRAM    R          0071
VROA$   R  0608'   0068  0879
VSPTR   R  06C0'   0069  0129  0157  0168  0169  0197  0200  0207  0294  0320
                         0481  0503  0579  0588  0943  0944  0983  0984  1000
                         1002  1003
VSPTR1  R  068E'   0069  0981
WRVDP   R  0676'   0062  0890  0955
XTFAC$     06A6'   0995  0134  0148  0160  0164  0233  0316  0330  0407  0513
                         0573  0718
XTFAC1     06C4'   1005  1010
```