```
                1          TITLE PSCAN359
                2  ********************************************************
                3  *                                                      *
                4  *       PPPPP      SSSS      CCCCC        AAA      N      N       *
                5  *       P    P    S    S    C      C      A   A     NN     N       *
                6  *       P    P    S         C            A     A    N N    N       *
                7  *       PPPPP      SSSS     C            AAAAAAA    N  N   N       *
                8  *       P              S    C            A     A    N   N N       *
                9  *       P         S    S    C      C     A     A    N    NN       *
               10  *       P          SSSS     CCCCC       A     A    N     N       *
               11  *                                                      *
               12  *                                                      *
               13  *             33333      555555      99999            *
               14  *            3     3     5          9     9           *
               15  *                  3     5          9     9           *
               16  *               3333      55555      999999           *
               17  *                  3           5          9           *
               18  *                  3           5         9            *
               19  *            3     3     5     5        9             *
               20  *             33333       5555        9              *
               21  *                                                      *
               22  ********************************************************
A040           23  CPUBAS EQU  >A040              Base of Expansion RAM
               24  ********************************************************
A000           25  M$EXEC EQU  >A000          Module EXEC branch table address
6010           26  M$EDIT EQU  >6010          Module EDIT branch table address
8000           27  M$FLMG EQU  >8000          Module FLMGR branch table address
6038           28  M$MSG  EQU  M$EDIT+>28     Start of message area
               29  ********************************************************
A004           30  EXEC   EQU  M$EXEC+>04
A00A           31  ASC    EQU  M$EXEC+>0A
A00E           32  EXEC6D EQU  M$EXEC+>0E
A010           33  DELINK EQU  M$EXEC+>10
A014           34  SQUISH EQU  M$EXEC+>14
               35
6012           36  TOPL15 EQU  M$EDIT+>02
601C           37  CHRTAB EQU  M$EDIT+>0C
802C           38  GRSUB2 EQU  M$FLMG+>2C
802E           39  GRSUB3 EQU  M$FLMG+>2E
               40  *
               41  *       LINKAGE TO ERROR MESSAGES IN FILE "EDIT"
               42  **
               43  *
6038           44  MSGERR EQU  M$EDIT+>28         * ERROR
6040           45  MSGFST EQU  M$EDIT+>30         * READY
6048           46  MSGBRK EQU  M$EDIT+>38         * BREAKPOINT
6053           47  MSGTA  EQU  M$EDIT+>43         * TRY AGAIN
605C           48  MSGWRN EQU  M$EDIT+>4C         * WARNING
               49  *
6065           50  MSG10  EQU  M$EDIT+>55         * NUMERIC OVERFLOW
6076           51  MSG14  EQU  M$EDIT+>66         * SYNTAX ERROR
6083           52  MSG16  EQU  M$EDIT+>73         * ILLEGAL AFTER SUBPROGRAM
609C           53  MSG17  EQU  M$EDIT+>8C         * UNMATCHED QUOTES
60AD           54  MSG19  EQU  M$EDIT+>9D         * NAME TOO LONG
60BB           55  MSG24  EQU  M$EDIT+>AB         * STRING-NUMBER MISMATCH
```

```
60D2      56  MSG25  EQU  M$EDIT+>C2        * OPTION BASE ERROR
60E4      57  MSG28  EQU  M$EDIT+>D4        * IMPROPERLY USED NAME
60F9      58  MSG34  EQU  M$EDIT+>E9        * UNRECOGNIZED CHARACTER
6110      59  MSG36  EQU  M$EDIT+>100       * IMAGE ERROR
611C      60  MSG39  EQU  M$EDIT+>10C       * MEMORY FULL
6128      61  MSG40  EQU  M$EDIT+>118       * STACK OVERFLOW
6137      62  MSG43  EQU  M$EDIT+>127       * NEXT WITHOUT FOR
6148      63  MSG44  EQU  M$EDIT+>138       * FOR-NEXT NESTING
6159      64  MSG47  EQU  M$EDIT+>149       * SUBEND NOT IN SUBPROGRAM
616F      65  MSG48  EQU  M$EDIT+>15F       * RECURSIVE SUBPROGRAM CALL
6189      66  MSG49  EQU  M$EDIT+>179       * MISSING SUBEND
6198      67  MSG51  EQU  M$EDIT+>188       * RETURN WITHOUT GOSUB
61AD      68  MSG54  EQU  M$EDIT+>19D       * STRING TRUNCATED
61BE      69  MSG57  EQU  M$EDIT+>1AE       * BAD SUBSCRIPT
61CC      70  MSG60  EQU  M$EDIT+>1BC       * LINE NOT FOUND
61DB      71  MSG61  EQU  M$EDIT+>1CB       * BAD LINE NUMBER
62C5      72  MSG62  EQU  M$EDIT+>2B5       * LINE TOO LONG
61EB      73  MSG67  EQU  M$EDIT+>1DB       * CAN'T CONTINUE
61FA      74  MSG69  EQU  M$EDIT+>1EA       * COMMAND ILLEGAL IN PROGRAM
6215      75  MSG70  EQU  M$EDIT+>205       * ONLY LEGAL IN A PROGRAM
622D      76  MSG74  EQU  M$EDIT+>21D       * BAD ARGUMENT
623A      77  MSG78  EQU  M$EDIT+>22A       * NO PROGRAM PRESENT
624D      78  MSG79  EQU  M$EDIT+>23D       * BAD VALUE
6257      79  MSG81  EQU  M$EDIT+>247       * INCORRECT ARGUMENT LIST
626F      80  MSG83  EQU  M$EDIT+>25F       * INPUT ERROR
627B      81  MSG84  EQU  M$EDIT+>26B       * DATA ERROR
6286      82  MSG97  EQU  M$EDIT+>276       * PROTECTION VIOLATION
629B      83  MSG109 EQU  M$EDIT+>28B       * FILE ERROR
62A6      84  MSG130 EQU  M$EDIT+>296       * I/O ERROR
62B0      85  MSG135 EQU  M$EDIT+>2A0       * SUBPROGRAM NOT FOUND
          86  *
630A      87  MSGCIS EQU  M$EDIT+>2FA       * UDF REFS ITSELF
6319      88  MSGCF  EQU  M$EDIT+>309       * CALLED FROM
6324      89  MSG56  EQU  M$EDIT+>314       * SPEECH STRING TOO LONG
          90  *
8012      91  CLSALL EQU  M$FLMG+>12
801A      92  OUTREC EQU  M$FLMG+>1A
          93
          94  ********************************************************
0036      95  TONE2  EQU  >36               BAD BEEP
          96  ********************************************************
0072      97  STACK  EQU  >72               STACK FOR DATA          STATUS BLOCK
0073      98  SUBSTK EQU  >73               SUBROUTINE STACK
0074      99  KEYBD  EQU  >74               KEYBOARD SELECTION
0075     100  RKEY   EQU  >75               KEY CODE
0079     101  TIMER  EQU  >79               TIMING REGISTER
         102  ********************************************************
         103  *                  PSCAN temporary workspace
0000     104  VARO   EQU  >00                    TEMPORARY
0001     105  VARV   EQU  >01                    TEMPORARY
0002     106  ACCUM  EQU  >02                    #OF BYTES ACCUMULATOR (4 BYTES
0004     107  PABPTR EQU  >04
0006     108  DFLTLM EQU  >06                    Default array limit (10)
0006     109  CCPPTR EQU  >06                    OFFSET WITHIN RECORD (1)
0007     110  RECLEN EQU  >07                    LENGTH OF CURRENT RECORD
```

```
0008    111  VARC    EQU  >08
0008    112  CCPADD  EQU  >08                 RAM address of current column
000A    113  CALIST  EQU  >0A                 Call list for resolving refs
000C    114  BYTE    EQU  >0C                 BYTE COUNTER       3
000C    115  NMPTR   EQU  BYTE                 Pointer save for pscan
000E    116  CHSAV   EQU  >0E
000E    117  CURINC  EQU  >0E                 Increment for auto-num mode
0010    118  TOPSTK  EQU  >10                 Top of data stack pointer
0012    119  LINUM   EQU  >12                 Used to determine end of scan
0014    120  NMLEN   EQU  >14                 # bytes needed for s.t. entry
0014    121  CURLIN  EQU  >14                 Current line for auto-num
0016    122  XFLAG   EQU  >16                 SCAN FLAG-BITS USED AS BELOW
        123  *             BIT 7  ENTER(0)  ENTERX(1)
        124  *             BIT 6  IFFLAG(1) Scanning an if-statement
        125  *             BIT 5  SAFLG(1)  Scanning subprogram arguments
        126  *             BIT 4  STRFLG(1) Scanning a string variable.
        127  *             BIT 3  SUBFLG(1) Scanning a subprogram
        128  *             BIT 2  FUNCTION FLAG(1) Entering a UDF in s.t.
        129  *             BIT 1  OPTION BASE DECLARED  Opt.base or array
        130  *             BIT 0  REMODE(1) REM only mode
0017    131  DSRFLG  EQU  >17                 INTERNAL=60, EXTERNAL=0 (1)
0017    132  FORNET  EQU  >17                 Nesting level of for/next
        133  ****************************************************************
        134  *                  Permanent workspace
0018    135  STRSP   EQU  >18                 Start of string space
001A    136  STREND  EQU  >1A                 End of string space
001C    137  SREF    EQU  >1C                 Temporary string pointer
001E    138  SMTSRT  EQU  >1E                 Beginning of statement
0020    139  VARW    EQU  >20                 Starting screen address
0022    140  ERRCOD  EQU  >22                 Return vector from ALC
0024    141  STVSPT  EQU  >24                 Base of value stack
        142  *                                NOT USED IN PRESCAN
        143  *
002A    144  VARA    EQU  >2A                 Ending screen address
002C    145  PGMPTR  EQU  >2C                 Program text pointer
002E    146  EXTRAM  EQU  >2E                 Line # table pointer
0030    147  STLN    EQU  >30                 Start of line number table
0032    148  ENLN    EQU  >32                 End of line number table
0034    149  DATA    EQU  >34                 DATA pointer for READ
0036    150  LNBUF   EQU  >36                 Line table pointer for DATA
        151  *       EQU  >38
003A    152  SUBTAB  EQU  >3A                 Subprogram symbol table ptr
003E    153  SYMTAB  EQU  >3E                 Symbol table pointer
0040    154  FREPTR  EQU  >40                 Free space pointer
0042    155  CHAT    EQU  >42                 Current chracter or token
0043    156  BASE    EQU  >43                 Option base value
0044    157  PRGFLG  EQU  >44                 Program/imperative flag
0045    158  FLAG    EQU  >45                 General flag byte
0046    159  BUFLEV  EQU  >46                 Crunch-buffer destruction lev
0048    160  LSUBP   EQU  >48                 Last subprogram block on stack
004A    161  FAC     EQU  >4A                 Floating-point accumulator
004B    162  FAC1    EQU  FAC+1
004C    163  FAC2    EQU  FAC+2
004D    164  FAC3    EQU  FAC+3
004E    165  FAC4    EQU  FAC+4
```

```
004F    166  FAC5    EQU  FAC+5
0050    167  FAC6    EQU  FAC+6
0051    168  FAC7    EQU  FAC+7
0052    169  FAC8    EQU  FAC+8
0054    170  FAC10   EQU  FAC+10
0056    171  FAC12   EQU  FAC+12
0057    172  FAC13   EQU  FAC+13
0058    173  FAC14   EQU  FAC+14
0059    174  FAC15   EQU  FAC+15
005A    175  FAC16   EQU  FAC+16
005B    176  FAC17   EQU  FAC+17
0054    177  DDD1    EQU  FAC+10
0056    178  FFF1    EQU  FAC+12
0058    179  EEE1    EQU  FAC+14
005C    180  ARG     EQU  >5C              Floating-point argument
005D    181  ARG1    EQU  ARG+1
005E    182  ARG2    EQU  ARG+2
005F    183  ARG3    EQU  ARG+3
0060    184  ARG4    EQU  ARG+4
0061    185  ARG5    EQU  ARG+5
0062    186  ARG6    EQU  ARG+6
0063    187  ARG7    EQU  ARG+7
0064    188  ARG8    EQU  ARG+8
0067    189  ARG11   EQU  ARG+11
006B    190  ARG15   EQU  ARG+15
006C    191  ARG16   EQU  ARG+16
006E    192  VSPTR   EQU  >6E              Value stack pointer
0076    193  EXP$    EQU  >76
0084    194  RAMTOP  EQU  >84              Highest address in ERAM
0086    195  RAMFRE  EQU  >86              Free pointer in the ERAM
0089    196  RAMFLG  EQU  >89              ERAM program flag
0088    197  RSTK    EQU  >88              Base of subroutine stack
00AF    198  STKMIN  EQU  >AF              Base of data stack
00BD    199  STKMAX  EQU  >BD              Top of data stack
00CE    200  PRTNFN  EQU  >CE
        201  ********************************************************************
        202  *                    VDP equates
02E2    203  NLNADD  EQU  >2E2             New line screen address
02FE    204  ENDSCR  EQU  >2FE             End of screen address
0376    205  SYMBOL  EQU  >376             Saved symbol table pointer
0382    206  SPGMPT  EQU  >382             Saved text pointer for break
038C    207  BUFSRT  EQU  >38C             Edit buffer start addr(VARW)
038E    208  BUFEND  EQU  >38E             Edit buffer end addr(VARW)
0392    209  TABSAV  EQU  >392             Saved main symbol table ptr
039A    210  SSTEMP  EQU  >39A             To save sub program table.
039C    211  SSTMP2  EQU  >39C             Same as above. Used in SUBS.
0820    212  CRNBUF  EQU  >820             Crunch buffer
08BE    213  CRNEND  EQU  >8BE             End of crunch buffer
08C0    214  RECBUF  EQU  >8C0             Edit recall buffer
0958    215  VRAMVS  EQU  >958             Default base of value stack
        216  *       temporary
038A    217  ERRLN   EQU  >38A
0388    218  SAVEVP  EQU  >388
03B0    219  VALIDP  EQU  >3B0             Use as two values passing
03B2    220  VALIDL  EQU  >3B2             VALIDATE code to READL1
```

```
                221  ********************************************************************
                222  *                    Immediate values
000D            223  CHRTN   EQU  >0D
0000            224  NUMBR   EQU  >00                 NUMERIC validate
0001            225  DIGIT   EQU  >01                 DIGIT validate
0002            226  UALPH   EQU  >02                 UALPHA validate
                227  *       Bits in XFLAG
0000            228  REMODE  EQU  0                   REM only mode.
0001            229  OPTFLG  EQU  1                   Option base declared flag
0002            230  FNCFLG  EQU  2                   Scanning UDF
0003            231  SUBFLG  EQU  3                   Scanning a subprogram.
0004            232  STRFLG  EQU  4                   Scanning a string variable.
0005            233  SAFLG   EQU  5                   Scanning subprog arguments.
0006            234  IFFLAG  EQU  6                   Scanning an if-statement
0007            235  ENTXFL  EQU  7                   ENTERX flag
                236  *
                237  *       bits in FLAG
                238  *
0001            239  WRNPRT  EQU  1                 Warning print bit
0002            240  WRNSTP  EQU  2                 Warning stop bit
                241  ********************************************************************
                242  *                    Editting command equates
0002            243  BREAK   EQU  >02                 Break key
0003            244  DLETE   EQU  >03                 Delete key
0004            245  INSRT   EQU  >04                 Insert key
0006            246  RECALL  EQU  >06                 Edit-buffer recall
0007            247  CLRLN   EQU  >07                 Clear-line key
0008            248  BACK    EQU  >08                 Back-space key
0009            249  FORW    EQU  >09                 Forward-space key
000A            250  DOWN    EQU  >0A                 Down-arrow key
000B            251  MVUP    EQU  >0B                 Up-arrow key
                252  ********************************************************************
                253  *                    IMMEDIATE VALUES
0022            254  QUOTE   EQU  >22            "
0024            255  DOLLAR  EQU  >24            $
0060            256  OFFSET  EQU  >60                 OFFSET FOR SCREEN
007E            257  CURSOR  EQU  >1E+OFFSET
007F            258  EDGECH  EQU  >1F+OFFSET          EDGE CHARACTER
                259  ********************************************************************
                260  *                    XML Equates
007D            261  SCHSYM  EQU  >7D                 SEARCH SYMBOL TABLE
0077            262  VPUSH   EQU  >77                 PUSH ON VALUE STACK
0078            263  VPOP    EQU  >78                 POP FROM VALUE STACK
0079            264  PGMCHR  EQU  >79                 GET PROGRAM CHARACTER
0081            265  CONTIN  EQU  >81                 CONTINUE EXECUTION
0072            266  MEMCHK  EQU  >72                 MEMORY CHECK ROUTINE: VDP RAM
0083            267  SCROLL  EQU  >83                 SCROLL THE SCREEN
0084            268  IO      EQU  >84                 IO UITILITY (KW TABLE SEARCH)
008B            269  GVWITE  EQU  >8B                 WRITE DATA FROM GRAM TO VRAM
008F            270  SCNSMT  EQU  >8F                 SCAN STATEMENT FOR PRESCAN
```

```
                    272  *
                    273  *              B A S I C   T O K E N   T A B L E
                    274  *
                    275  *       EQU   >80          SPARE
0081                276  ELSE$    EQU   >81          "ELSE"
0082                277  SSEP$    EQU   >82          "::"
0083                278  TREM$    EQU   >83          "!"
0084                279  IF$      EQU   >84          "IF"
0085                280  GO$      EQU   >85          "GO"
0086                281  GOTO$    EQU   >86          "GOTO"
0087                282  GOSUB$   EQU   >87          "GOSUB"
0088                283  RETUR$   EQU   >88          "RETURN"
0089                284  DEF$     EQU   >89          "DEF"
008A                285  DIM$     EQU   >8A          "DIM"
008B                286  END$     EQU   >8B          "END"
008C                287  FOR$     EQU   >8C          "FOR"
008D                288  LET$     EQU   >8D          "LET"
008E                289  BREAK$   EQU   >8E          "BREAK"
008F                290  UNBRE$   EQU   >8F          "UNBREAK"
0090                291  TRACE$   EQU   >90          "TRACE"
0091                292  UNTRA$   EQU   >91          "UNTRACE"
0092                293  INPUT$   EQU   >92          "INPUT"
0093                294  DATA$    EQU   >93          "DATA"
0094                295  RESTO$   EQU   >94          "RESTORE"
0095                296  RANDO$   EQU   >95          "RANDOMIZE"
0096                297  NEXT$    EQU   >96          "NEXT"
0097                298  READ$    EQU   >97          "READ"
0098                299  STOP$    EQU   >98          "STOP"
0099                300  DELET$   EQU   >99          "DELETE"
009A                301  REM$     EQU   >9A          "REM"
009B                302  ON$      EQU   >9B          "ON"
009C                303  PRINT$   EQU   >9C          "PRINT"
009D                304  CALL$    EQU   >9D          "CALL"
009E                305  OPTIO$   EQU   >9E          "OPTION"
009F                306  OPEN$    EQU   >9F          "OPEN"
00A0                307  CLOSE$   EQU   >A0          "CLOSE"
00A1                308  SUB$     EQU   >A1          "SUB"
00A2                309  DISPL$   EQU   >A2          "DISPLAY"
00A3                310  IMAGE$   EQU   >A3          "IMAGE"
00A4                311  ACCEP$   EQU   >A4          "ACCEPT"
00A5                312  ERROR$   EQU   >A5          "ERROR"
00A6                313  WARN$    EQU   >A6          "WARNING"
00A7                314  SUBXT$   EQU   >A7          "SUBEXIT"
00A8                315  SUBND$   EQU   >A8          "SUBEND"
00A9                316  RUN$     EQU   >A9          "RUN"
00AA                317  LINPU$   EQU   >AA          "LINPUT"
                    318  *        EQU   >AB          "LIBRARY"      FUTURE
                    319  *        EQU   >AC          "REAL"         FUTURE
                    320  *        EQU   >AD          "INTEGER"      FUTURE
                    321  *        EQU   >AE          "SCRATCH"      FUTURE
                    322  *        EQU   >AF          SPARE
00B0                323  THEN$    EQU   >B0          "THEN"
00B1                324  TO$      EQU   >B1          "TO"
00B2                325  STEP$    EQU   >B2          "STEP"
00B3                326  COMMA$   EQU   >B3          ","
```

```
00B4    327   SEMIC$ EQU  >B4          ";"
00B5    328   COLON$ EQU  >B5          ":"
00B6    329   RPAR$  EQU  >B6          ")"
00B7    330   LPAR$  EQU  >B7          "("
00B8    331   CONC$  EQU  >B8          "&"  CONCATENATE
        332   *      EQU  >B9          SPARE
00BA    333   OR$    EQU  >BA          "OR"
00BB    334   AND$   EQU  >BB          "AND"
00BC    335   XOR$   EQU  >BC          "XOR"
00BD    336   NOT$   EQU  >BD          "NOT"
00BE    337   EQUAL$ EQU  >BE          "="
00BF    338   LESS$  EQU  >BF          "<"
00C0    339   GREAT$ EQU  >C0          ">"
00C1    340   PLUS$  EQU  >C1          "+"
00C2    341   MINUS$ EQU  >C2          "-"
00C3    342   MULT$  EQU  >C3          "*"
00C4    343   DIVI$  EQU  >C4          "/"
00C5    344   CIRCU$ EQU  >C5          "^"
        345   *      EQU  >C6          SPARE
00C7    346   STRIN$ EQU  >C7          QUOTED STRING
00C8    347   UNQST$ EQU  >C8          UNQUOTED STRING
00C8    348   NUM$   EQU  UNQST$       NUMERIC STRING
00C8    349   NUMCO$ EQU  UNQST$       NUMERIC CONSTANT
00C9    350   LN$    EQU  >C9          LINE NUMBER
00CA    351   EOF$   EQU  >CA          "EOF"
00CB    352   ABS$   EQU  >CB          "ABS"
00CC    353   ATN$   EQU  >CC          "ATN"
00CD    354   COS$   EQU  >CD          "COS"
00CE    355   EXP$$  EQU  >CE          "EXP"
00CF    356   INT$   EQU  >CF          "INT"
00D0    357   LOG$   EQU  >D0          "LOG"
00D1    358   SGN$$  EQU  >D1          "SGN"
00D2    359   SIN$   EQU  >D2          "SIN"
00D3    360   SQR$   EQU  >D3          "SQR"
00D4    361   TAN$   EQU  >D4          "TAN"
00D5    362   LEN$   EQU  >D5          "LEN"
00D6    363   CHR$$  EQU  >D6          "CHR$"
00D7    364   RND$   EQU  >D7          "RND"
00D8    365   SEG$$  EQU  >D8          "SEG$"
00D9    366   POS$   EQU  >D9          "POS"
00DA    367   VAL$   EQU  >DA          "VAL"
00DB    368   STR$$  EQU  >DB          "STR$"
00DC    369   ASC$   EQU  >DC          "ASC"
00DD    370   PI$    EQU  >DD          "PI"
00DE    371   REC$   EQU  >DE          "REC"
00DF    372   MAX$   EQU  >DF          "MAX"
00E0    373   MIN$   EQU  >E0          "MIN"
00E1    374   RPT$$  EQU  >E1          "RPT$"
        375   *
        376   *
00E8    377   NUMER$ EQU  >E8          "NUMERIC"
00E9    378   DIGIT$ EQU  >E9          "DIGIT"
00EA    379   UALPH$ EQU  >EA          "UALPHA"
00EB    380   SIZE$  EQU  >EB          "SIZE"
00EC    381   ALL$   EQU  >EC          "ALL"
```

```
00ED    382  USING$ EQU  >ED          "USING"
00EE    383  BEEP$  EQU  >EE          "BEEP"
00EF    384  ERASE$ EQU  >EF          "ERASE"
00F0    385  AT$    EQU  >F0          "AT"
00F1    386  BASE$  EQU  >F1          "BASE"
        387  *      EQU  >F2          "TEMPORARY"      FUTURE
00F3    388  VARIA$ EQU  >F3          "VARIABLE"       FUTURE
00F4    389  RELAT$ EQU  >F4          "RELATIVE"       FUTURE
00F5    390  INTER$ EQU  >F5          "INTERNAL"       FUTURE
00F6    391  SEQUE$ EQU  >F6          "SEQUENTIAL"
00F7    392  OUTPU$ EQU  >F7          "OUTPUT"
00F8    393  UPDAT$ EQU  >F8          "UPDATE"
00F9    394  APPEN$ EQU  >F9          "APPEND"
00FA    395  FIXED$ EQU  >FA          "FIXED"
00FB    396  PERMA$ EQU  >FB          "PERMANENT"
00FC    397  TAB$   EQU  >FC          "TAB"
00FD    398  NUMBE$ EQU  >FD          "#"
00FE    399  VALID$ EQU  >FE          "VALIDATE"
        400  *      EQU  >FF          ILLEGAL VALUE
```

```
                    402         GROM 3
                    403         ORG  >A70
6A70 50F5           404         BR   PRESCN
6A72 4A72           405         BR   $                    Spare
6A74 4F83           406         BR   LLIST
6A76 4A8E           407         BR   READLN
6A78 525D           408         BR   CHKEND               Check end of statements.
6A7A 4A7A           409         BR   $                    Was SEETWO - now spare
6A7C 50C1           410         BR   DISO
6A7E 5400           411         BR   ENTER
6A80 5446           412         BR   ENT09
6A82 4CB0           413         BR   WARN$$
6A84 4CFB           414         BR   ERR$$
6A86 4A98           415         BR   READL1
6A88 4AAF           416         BR   READ00
6A8A 4A8A           417         BR   $                    Spare
6A8C 4A8C           418         BR   $                    Spare
```

```
                    420  ********************************************************************
                    421  *       READLN - Read one logical line (up to four                 *
                    422  *       physical lines) from the keyboard.  Interpret              *
                    423  *       things like BACKSPACE, INSERT, DELETE and                  *
                    424  *       FORWARD.  The total number of characters can be            *
                    425  *       limited by changing the start value for ARG2               *
                    426  *       (upper limit), and entering ar READL1.                     *
                    427  *       VARW has to contain the start address of the field,        *
                    428  *       and VARA the current highest write address                 *
                    429  *            Entering at READL1 also enables us to                 *
                    430  *       prespecify the minimum number of characters to be          *
                    431  *       read.  Filling this field with a default answer            *
                    432  *       allows for default creation.                               *
                    433  *            Entering at READOO allows for specification           *
                    434  *       of the initial cursor-position.  In this case              *
                    435  *       ARG5 has to be set to the cursor-position.                 *
                    436  *            Please see to it that VARA, VARW, ARG2,               *
                    437  *       ARG4 have consistent values, i.e.                          *
                    438  *             VARW <= ARG5 <= VARA <= ARG2                         *
                    439  *       ARG4 indicates if the line has been changed.               *
                    440  *       If so, it contains a 0.  If you enter READLN through*
                    441  *       READOO, you have to initialize ARG4 to a nonzero           *
                    442  *       value, should you want to use this feature.                *
                    443  ********************************************************************
                    444  READLN
6A8E BE63FF         445    ST   >FF,@ARG7          Indicate non-check mode
6A91 BF5E03         446    DST  >37D,@ARG2         Set default upper limit
6A94 7D
6A95 BD2A20         447    DST  @VARW,@VARA        Default to "nothing entered ye
                    448    $
                    449    $    Please make sure that VARA points at an space
                    450    $    location, or at the end-of-field
                    451    $
                    452  READL1
6A98 BE6001         453    ST   1,@ARG4            This means "no change" in line
                    454  READL2
6A9B BD6120         455    DST  @VARW,@ARG5        Position cursor at start of field
                    456    $
                    457    $    This will cause the next test to fail initialy, since
                    458    $    VARW clearly equals ARG5 first time through
                    459    $
                    460  RBACK
6A9E C56120         461    $IF @ARG5 .DH. @VARW  This is the standard backup entry, 1
6AA1 4AAF
6AA3 9361           462      DDEC @ARG5              so we backup the current positio1
6AA5 D6B061         463      $IF RAM(@ARG5) .EQ. EDGECH THEN Skip border line       2
6AA8 7F4AAF
6AAB A76100         464        DSUB 4,@ARG5        Backup to previous line          2
6AAE 04
                    465      $END IF
                    466    $END IF
                    467    $
                    468    $    To get out of insert mode, we usually return here
                    469    $
                    470  READOO
```

```
6AAF 8664      471     CLR  @ARG8              Indicate normal operation mode
6AB1 BE017E    472     ST   CURSOR,@VARV       Use VARV for CURSOR/CHARACTER
               473   READ$1
               474     $
               475     $     Input one character and alternate current
               476     $     character position between normal and cursor
               477     $
6AB4 C0B061    478     EX   @VARV,RAM(@ARG5)  By alternating between the normal
6AB7 01
6AB8 8679      479     CLR  @TIMER              character and the cursor, we make
               480     $REPEAT                  the cursor "blink"
6ABA 03        481       SCAN                 Scan for a character
6ABB 0263      482       RAND 99              Force randomize to be random
6ABD 6AC6      483       BS   READ$2          Found one !!!!!
6ABF C6790E    484     $UNTIL @TIMER .H. 14   Time next character switch
6AC2 4ABA
6AC4 4AB4      485     BR   READ$1            Restart character blink cycle
               486     $
               487     $     Correct if we ended up with a displayed cursor
               488     $
               489   READ$2
6AC6 D6017E    490     $IF @VARV .NE. CURSOR THEN Will have to change once more
6AC9 6ACF
6ACB C0B061    491       EX   @VARV,RAM(@ARG5) Exchange for current cursor
6ACE 01
               492     $END IF
6ACF CA7520    493     $IF @RKEY .L. : : THEN Control character !!!!!
6AD2 6BBD
               494       $
               495       $  BREAK character handling comes first
               496       $
6AD4 D67502    497       $IF @RKEY .EQ. BREAK THEN Saw break character
6AD7 4AE7
6AD9 B245FE    498         RB   @FLAG,0         Reset AUTONUM mode
6ADC 8E446C    499         $IF  @PRGFLG .EQ. 0 GOTO BTOP15 If in run mode
6ADF F8
6AE0 BDA382    500         DST  @SMTSRT,RAM(SPGMPT) Save place for continue
6AE3 1E
6AE4 05A00E    501         B    EXEC6D       Interrupt execution
               502       $END IF
               503   *
               504   *       Edit buffer recall
               505   *
6AE7 D67506    506      $IF  @RKEY .EQ. RECALL THEN  If edit recall
6AEA 4B1D
6AEC 8E444A    507        $IF @PRGFLG .NE. 0 GOTO READ$1 Ignore if exec mode
6AEF B4
6AF0 B245FE    508        RB   @FLAG,0              Reset AUTONUM
6AF3 BF2003    509        DST  NLNADD+32,@VARW      Initialize to 32 below screen
6AF6 02
               510        $REPEAT
6AF7 0F83      511          XML  SCROLL             Scroll the screen
6AF9 A72000    512          DSUB 32,@VARW           Line start is 32 lower now
6AFC 20
6AFD D520A3    513        $UNTIL @VARW .DEQ. RAM(BUFSRT) Until reach recall star
```

```
6B00 8C4AF7
6B03 BD2AA3   514        DST  RAM(BUFEND),@VARA   Set old end of line
6B06 8E
6B07 BD4A2A   515        DST  @VARA,@FAC          Calculate length of old line
6B0A A54A20   516        DSUB @VARW,@FAC          Subtract start from end
6B0D 6B15     517        BS   READ$3              If no characters to recall
6B0F 344AB0   518        MOVE @FAC FROM RAM(RECBUF) TO RAM(@VARW) Recall line
6B12 20A8C0
              519   READ$3
6B15 BE63FF   520        ST   >FF,@ARG7           Non-check mode
6B18 BD6120   521        DST  @VARW,@ARG5         Cursor at beginning of line
6B1B 4AAF     522        BR   READOO              Allow edit of line
              523      $END IF
              524       $
              525       $  BACK-ARROW - Space back one position
              526       $
6B1D D67508   527       $IF @RKEY .EQ. BACK GOTO RBACK   Backup to previous pos
6B20 6A9E
              528       $
              529       $  RIGHT-ARROW - Forward space
              530       $
6B22 D67509   531       $IF @RKEY .EQ. FORW GOTO RFORW   Space one position
6B25 6CAC
              532       $
              533       $  INSERT - Start INSERT mode here
              534       $
6B27 D67504   535       $IF @RKEY .EQ. INSRT THEN        Set INSERT flag
6B2A 4B2F
6B2C BE6401   536         ST   1,@ARG8           Select INSERT mode
              537       $END IF
              538       $
              539       $  DELETE - Delete the current character
              540       $
6B2F D67503   541       $IF @RKEY .EQ. DLETE THEN        DELETE all right
6B32 4B84
6B34 8660     542         CLR  @ARG4             Indicate definite change in line
6B36 D52A61   543         $IF @VARA .DNE. @ARG5 THEN Not an empty line
6B39 6B7E
6B3B D6B02A   544           $IF  RAM(@VARA) .EQ. EDGECH THEN If pointing at end
6B3E 7F4B43
6B41 932A     545             DDEC @VARA           Backup up onto line
              546           $END IF
6B43 BD5C2A   547           DST  @VARA,@ARG  Move everything from the right
6B46 A55C61   548           DSUB @ARG5,@ARG    of the cursor to the left
6B49 345CB0   549           MOVE @ARG FROM RAM(1(ARG5)) TO RAM(@ARG5)
6B4C 61E001
6B4F 61
6B50 BD5C61   550           DST  @ARG5,@ARG Start at the beginning
6B53 B25DFC   551           AND  >FC,@ARG1
6B56 B65D1D   552           OR   >1D,@ARG1   Move over to the end of the line
6B59 C95C2A   553           $WHILE @ARG .DL. @VARA Update all errors
6B5C 6B6A
6B5E C0E004   554             EX   RAM(@ARG),RAM(4(ARG))  Restore edge chars
6B61 5CB05C
6B64 A35C00   555             DADD 32,@ARG   Next victim please
```

```
6B67 20
6B68 4B59      556          $END WHILE                                    3
6B6A 932A      557          DDEC @VARA        Pre-update end of string    3
6B6C D6B02A    558          $IF RAM(@VARA) .EQ. EDGECH THEN Hit edge character 4
6B6F 7F4B76
6B72 A72A00    559            DSUB 4,@VARA    Skip over edge characters   4
6B75 04
               560          $END IF
6B76 D6B02A    561          $IF RAM(@VARA) .EQ. : :+OFFSET GOTO READ00    3
6B79 806AAF
6B7C 912A      562          DINC @VARA        Locked at field position    3
               563        $END IF
6B7E BEB02A    564        ST   : :+OFFSET,RAM(@VARA) Clear last position  2
6B81 80
6B82 4AAF      565        BR   READ00                                       2
               566      $END IF
               567      $
               568      $  CLEAR - Clear the entire input line
               569      $
6B84 D67507    570      $IF @RKEY .EQ. CLRLN THEN       Found CLEAR command 2
6B87 4BA0
               571        $REPEAT             Current maximum to minimum
6B89 D6B02A    572          $IF RAM(@VARA) .NE. EDGECH THEN Don't clear edges 4
6B8C 7F6B93
6B8F BEB02A    573            ST   : :+OFFSET,RAM(@VARA) Blank line       4
6B92 80
               574          $END IF
6B93 932A      575          DDEC @VARA        Pre-update end-of-line      3
6B95 C92A20    576        $UNTIL @VARA .DL. @VARW Up to and including first pos2
6B98 6B89
6B9A 912A      577        DINC @VARA          Undo last subtraction       2
6B9C 8660      578        CLR  @ARG4          Indicate change             2
6B9E 4A9B      579        BR   READL2         And restart everything      2
               580      $END IF
               581      $
               582      $  General exit point.  Unidentified control codes
               583      $  don't have any effect !!!!!!!
               584      $
6BA0 D6750D    585      $IF @RKEY .NE. CHRTN THEN Only react on CR/UP/DOWN  2
6BA3 6BAF
6BA5 D6750B    586        $IF @RKEY .NE. MVUP THEN                          3
6BA8 6BAF
6BAA D6750A    587          $IF @RKEY .NE. DOWN GOTO READ$1                 3
6BAD 4AB4
               588        $END IF
               589      $END IF
6BAF D52A5E    590      $IF @VARA .DEQ. @ARG2 THEN Check for block on last pos. 2
6BB2 4BBC
6BB4 D6B02A    591        $IF RAM(@VARA) .NE. : :+OFFSET THEN    Blocked.....  3
6BB7 806BBC
6BBA 912A      592          DINC @VARA        Point beyond last character in lin3
               593        $END IF
               594      $END IF
6BBC 00        595      RTN                   ENTER the current line        1
               596    $END IF
```

```
6BBD 8E634C   597    $IF @ARG7 .EQ. 0 THEN check value of RKEY against given  1
6BC0 27
6BC1 BD5CA3   598      DST   RAM(VALIDP),@ARG pick up the standard stuff      1
6BC4 B0
6BC5 BC5CB0   599      ST  RAM(@ARG),@ARG  RAM(VALIDP):Ptr to the standard stu1
6BC8 5C
6BC9 DA5C04   600      $IF .BIT(UALPH) @ARG .EQ. 1 THEN specified UPPER CASE A2
6BCC 6BDD
6BCE C6755A   601        $IF @RKEY .H. :Z: GOTO VALI$2 too high for anything  2
6BD1 6C07
6BD3 CA7541   602        $IF @RKEY .HE. :A: GOTO VALI$9 o.k. already in range 2
6BD6 6C27
6BD8 D67520   603        $IF @RKEY .EQ. : : GOTO VALI$9 allow spaces in UALPHA2
6BDB 6C27
              604      $END IF
6BDD DA5C01   605      $IF .BIT(NUMBR) @ARG .EQ. 1 THEN specified NUMERIC     2
6BE0 6BF8
6BE2 D67545   606        $IF @RKEY .EQ. :E: GOTO VALI$9                       2
6BE5 6C27
6BE7 D6752E   607        $IF @RKEY .EQ. :.: GOTO VALI$9                       2
6BEA 6C27
6BEC D6752B   608        $IF @RKEY .EQ. :+: GOTO VALI$9                       2
6BEF 6C27
6BF1 D6752D   609        $IF @RKEY .EQ. :-: GOTO VALI$9                       2
6BF4 6C27
6BF6 4BFD     610        BR   VALI$1          now try DIGIT range             2
              611      $END IF
6BF8 DA5C02   612      $IF .BIT(DIGIT) @ARG .EQ. 1 THEN Digit range selectr   2
6BFB 6C07
              613  VALI$1
6BFD CA7530   614        $IF @RKEY .L. :0: GOTO VALI$2 no good                2
6C00 4C07
6C02 CA753A   615        $IF @RKEY .L. :9:+1 GOTO VALI$9 numeric allright     2
6C05 4C27
              616      $END IF
              617  VALI$2
6C07 BD5CA3   618      DST  RAM(VALIDP),@ARG copy start address of string     1
6C0A B0
6C0B BC50A3   619      ST   RAM(VALIDL+1),@FAC6 and string length             1
6C0E B3
6C0F 4C17     620      BR   VALI$4          now test given characters        1
              621  VALI$3
6C11 D475B0   622      $IF @RKEY .EQ. RAM(@ARG) GOTO VALI$9 valid !!!!!!      1
6C14 5C6C27
              623  VALI$4
6C17 915C     624      DINC @ARG            update actual address            1
6C19 9250     625      DEC  @FAC6           and count # of characters         1
6C1B 4C11     626      BR   VALI$3                                           1
              627      $REPEAT
6C1D 8E80CE   628      $UNTIL @PRTNFN .EQ. 0 wait for completion of previous  1
6C20 4C1D
6C22 060036   629      CALL TONE2             tone,and then ------BEEP------  1
6C25 4AB4     630      BR   READ$1          continue in whatever mode we're i1
              631    $END IF
              632  VALI$9
```

```
6C27 8E646C   633    $IF @ARG8 .NE. 0 THEN INSERT mode
6C2A 63
              634      $
              635      $  INSERT is COMPLICATED!!!!! because of those edge
              636      $  characters.......Shift up all things.....continue
              637      $  as a standard insert......VARA <= ARG2
              638      $
6C2B D52A5E   639      $IF @VARA .DEQ. @ARG2 GOTO READ$4 If end of screen
6C2E 6C3A
6C30 D6B02A   640      $IF RAM(@VARA) .EQ. EDGECH THEN  If at end of line
6C33 7F4C3A
6C36 A32A00   641         DADD 4,@VARA       Skip to next line
6C39 04
              642      $END IF
              643  READ$4
6C3A BD5C2A   644      DST  @VARA,@ARG        Use ARG as temp. for insert
6C3D C55C61   645      $WHILE @ARG .DH. @ARG5 Move everything up to cur. loc.
6C40 4C5C
6C42 935C     646        DDEC @ARG            Copy lower location to higher one
6C44 BCE001   647        ST   RAM(@ARG),RAM(1(ARG)) going from high to low
6C47 5CB05C
6C4A D6B05C   648        $IF RAM(@ARG) .EQ. EDGECH THEN Bumped into wall again
6C4D 7F4C5A
6C50 A75C00   649          DSUB 4,@ARG        Skip the wall
6C53 04
6C54 BCE005   650          ST   RAM(@ARG),RAM(5(ARG)) And move character over
6C57 5CB05C
              651        $END IF
6C5A 4C3D     652      $END WHILE
6C5C C92A5E   653      $IF @VARA .DL. @ARG2 THEN Only update VARA if upper
6C5F 6C63
6C61 912A     654        DINC @VARA           hasn't been reached yet
              655      $END IF
              656    $END IF                  Rest is standard copy-character
6C63 A27560   657    ADD  OFFSET,@RKEY        Create displayable character
6C66 BCB061   658    ST   @RKEY,RAM(@ARG5)    Display at current character pos.
6C69 75
6C6A 8660     659    CLR  @ARG4               Indicate change in line
              660  READ05
6C6C D5615E   661    $IF @ARG5 .DEQ. @ARG2 THEN Hit right margin
6C6F 4C76
6C71 060036   662      CALL TONE2             ---------- B E E P ----------
6C74 4AB4     663      BR   READ$1            Stay in current mode !!!!!!!
              664    $END IF
6C76 9161     665    DINC @ARG5               Update current address
6C78 D6B061   666    $IF RAM(@ARG5) .EQ. EDGECH THEN Correct for next line
6C7B 7F4C82
6C7E A36100   667      DADD 4,@ARG5           By skipping border
6C81 04
              668    $END IF
6C82 C5612A   669    $IF @ARG5 .DH. @VARA THEN Check for last new high limit
6C85 4C8A
6C87 BD2A61   670      DST  @ARG5,@VARA       Update new high limit
              671    $END IF
6C8A CB2A02   672    $IF @VARA .DL. >2FE GOTO READ$1 Still some space to go
```

```
6C8D FE4AB4
6C90 0F83      673     XML  SCROLL              Scroll the screen !!!!!
6C92 A72A00    674     DSUB 28,@VARA            Back to current start of line
6C95 1C
6C96 8E646C    675     $IF @ARG8 .NE. 0 THEN    If not insert mode then          1
6C99 9E
6C9A A72A00    676        DSUB 4,@VARA          Off by 4 more-correct it         1
6C9D 04
               677     $END IF
6C9E A72000    678     DSUB 32,@VARW            Backup line start address
6CA1 20
6CA2 A75E00    679     DSUB 32,@ARG2            Absolute high limit backs up too
6CA5 20
6CA6 A76100    680     DSUB 32,@ARG5            Current cursor position too
6CA9 20
6CAA 4AB4      681     BR   READ$1              Start with something else
               682     $
               683     $    Something special for forward cursor move
               684     $
               685  RFORW
6CAC 8664      686     CLR  @ARG8                 Leave INSERT mode - don't copy
6CAE 4C6C      687     BR   READ05                but use rest of input logic.
```

```
                    689  ********************************************************
                    690  *                                                      *
                    691  *       WARN$$  -  Checks the special warning handling *
                    692  *            conditions which can be set by an ON WARNING *
                    693  *            statement and  does the following based upon *
                    694  *            those conditions:                         *
                    695  *       ON WARNING PRINT - prints continues execution  *
                    696  *       ON WARNING STOP  - prints and stops            *
                    697  *       ON WARNING NEXT  - ignores the warning and goes on *
                    698  *                                                      *
                    699  *                                                      *
                    700  ********************************************************
                    701  *
                    702  *
6CB0 8722           703  WARN$$ DCLR @ERRCOD            Clear the error if from 9900
6CB2 8676           704         CLR  @EXP$
6CB4 8877           705         FETCH @EXP$+1          Get index into error table
6CB6 E37600         706         DSLL @EXP$,2           Multiply by 4
6CB9 02
6CBA A376B7         707         DADD  ERRTAB,@EXP$     Get addr of entry into table
6CBD 57
6CBE 330004         708         MOVE 4 FROM ROM(@EXP$) TO @FAC+10
6CC1 540000
6CC4 76
6CC5 8E446C         709         $IF @PRGFLG .EQ. 0 GOTO WRN$$3  If its imperative
6CC8 CE
                    710  *                              take default.
6CC9 DA4502         711         $IF .BIT(WRNPRT) @FLAG .EQ. 0 THEN If print turned o1
6CCC 4CE8
6CCE 0F83           712  WRN$$3   XML  SCROLL           Scroll the screen             1
6CD0 310009         713           MOVE 9 FROM ROM(MSGWRN) TO RAM(NLNADD)  * WARNING 1
6CD3 A2E260
6CD6 5C
6CD7 0F83           714           XML  SCROLL           Scroll the screen again.      1
6CD9 BF2002         715           DST  NLNADD+2,@VARW   Start address behind warning. 1
6CDC E4
6CDD 066DFF         716           CALL  TRACBK          Check for warning in UDF      1
6CE0 6CE5           717           BS    WRN$$5          Was UDF so message already ou1
6CE2 066D9E         718           CALL ERPNT5           Print the message             1
6CE5 BE7F03         719  WRN$$5   ST   3,@XPT                                        1
                    720         $END IF
                    721  *      If imperative then continue on normally
6CE8 8E446F         722         $IF @PRGFLG .EQ. 0 GOTO RETNOS  If its imperative
6CEB 48
                    723  *      If warning continue turned on then continue
6CEC DA4504         724         $IF .BIT(WRNSTP) @FLAG .EQ. 0 GOTO RETNOS If continue
6CEF 6F48
6CF1 067782         725  ERR$$4 CALL CLEAN             Clean up stack and s.t.
6CF4 BD6EA3         726  ERR$$5 DST  RAM(SAVEVP),@VSPTR   Restore value stack
6CF7 88
6CF8 056012         727  BTOP15 B    TOPL15            Finish up and go back
```

```
                    729  ********************************************************
                    730  *                                                      *
                    731  *      ERR$$ - Sets up an error stack entry based upo  *
                    732  *           the information passed to it by the caller *
                    733  *           and what it can gather from the error table.*
                    734  *           It then either prints the error message and*
                    735  *           aborts or goes to the line specified by a  *
                    736  *           previously executed ON ERROR statement.  The*
                    737  *           stack entry looks like:                    *
                    738  *-----------------------------------------------------*
                    739  *     !Error code!Severity! >69 !Luno #!EXTRAM!PGMPTR!  *
                    740  *     !^          ^         ^     ^      ^      ^     !  *
                    741  *     !FAC        FAC+1     FAC+2 FAC+3  FAC+4  FAC+6 !  *
                    742  *-----------------------------------------------------*
                    743  *                                                      *
                    744  *      ERROR CODE - the error number                   *
                    745  *      SEVERITY - Severity of the error                *
                    746  *           1 - Warning                                *
                    747  *           5 - Possibly recoverable                   *
                    748  *           9 - Fatal, unrecoverable                   *
                    749  *      >69  ERROR STACK ENTRY ID                       *
                    750  *      LUNO # - Luno # if file error or -1 if non-file *
                    751  *               error                                  *
                    752  *      EXTRAM,PGMPTR - Information to indicate the line #
                    753  *           of the error                               *
                    754  *                                                      *
                    755  ********************************************************
                    756  *
                    757  *
                    758  *
6CFB 8722           759  ERR$$  DCLR @ERRCOD              Clear error code if from 9900
6CFD A5140E         760         DSUB @CURINC,@CURLIN      Just in case in autonum mode
6D00 D73E08         761         $IF @SYMTAB .DEQ. CRNBUF THEN  If prescanning r.h.   1
6D03 204D0A
6D06 BD3EA8         762           DST RAM(CRNBUF+2),@SYMTAB   of UDF and parm in     1
6D09 22
                    763         $END IF                      crunch buffer, fix SYMTAB
6D0A 8676           764         CLR  @EXP$
6D0C 8877           765         FETCH @EXP$+1            Get index into error table
6D0E E37600         766         DSLL @EXP$,2             Multiply index by 4
6D11 02
6D12 A376B7         767         DADD  ERRTAB,@EXP$       Addr of table entry
6D15 57
6D16 330004         768         MOVE 4 FROM ROM(@EXP$) TO @FAC+10  Get table entry
6D19 540000
6D1C 76
6D1D BE738A         769         ST  RSTK+2,@SUBSTK       Init subr stack but allow for
                    770  *                                  GROM return address
6D20 8E574D         771         $IF  @FAC+13 .EQ. 0 THEN If message only             1
6D23 32
6D24 066D91         772  ERR$$R   CALL ERPRNT             Display the error message   1
6D27 D75460         773           $IF @FAC10 .DEQ. MSGFST THEN  If * READY *         2
6D2A 404D30
6D2D 068012         774              CALL CLSALL           Close all files            2
                    775           $END IF
```

```
6D30 4CF1      776           BR  ERR$$4              and clean up                1
               777         $END IF
               778  *
               779  *
6D32 8E446D    780         $IF  @PRGFLG .EQ. 0 GOTO ERR$1 If imperative-default
6D35 3B
6D36 8FA38A    781         $IF  RAM(ERRLN) .DEQ. 0 THEN If error turned off    1
6D39 4D42
6D3B 066DFF    782  ERR$1    CALL TRACBK           Check for UDF               1
6D3E 6CF1      783           BS   ERR$$4           Was UDF, message already out 1
6D40 4D24      784           BR   ERR$$R           Assume normal error         1
               785         $END IF
               786  *
               787  *      Error turned on.  Now build the error entry
               788  *
6D42 067782    789         CALL CLEAN              Clean up the stack
6D45 BD4A56    790         DST  @FAC12,@FAC       Put in error & severity
6D48 BE4C69    791         ST   >69,@FAC2         Error stack ID
6D4B D75462    792         $IF @FAC10 .DEQ. MSG130 THEN  If I/O error          1
6D4E A64D59
6D51 BC4DE0    793           ST  RAM(2(PABPTR)),@FAC3 Put in LUNO #            1
6D54 0204
6D56 B64B80    794           SB  @FAC1,7           And indicate an I/O error   1
               795         $END IF
6D59 BD502E    796         DST  @EXTRAM,@FAC6      Save line pointer
6D5C BD4E1E    797         DST  @SMTSRT,@FAC4      Save ptr to beginning of stmt
               798  *
6D5F BD5C6E    799         DST  @VSPTR,@ARG        Must check for room on stack
6D62 A35C00    800         DADD 24,@ARG            Need 24 to help out VPUSH
6D65 18
6D66 C51A5C    801         $IF  @STREND .NOT. .DH. @ARG THEN  If not room      1
6D69 6D80
6D6B 066D91    802           CALL ERPRNT           Put out the message anyway  1
6D6E BF5461    803           DST  MSG39,@FAC10     Memory full message         1
6D71 1C
6D72 8644      804           CLR  @PRGFLG          Don't print a line #        1
6D74 066D91    805           CALL ERPRNT           Print it too                1
6D77 310008    806           MOVE 8 FROM ROM(MSGERR) TO RAM(NLNADD-18)         1
6D7A A2D060
6D7D 38
6D7E 4CF4      807           BR  ERR$$5            And give up                 1
               808         $END IF
6D80 0F77      809         XML  VPUSH              Push the error entry
6D82 872E      810         DCLR @EXTRAM            Clear on-error entry
6D84 C12EA3    811         DEX  RAM(ERRLN),@EXTRAM  Set line ptr&clear on-error
6D87 8A
6D88 06802C    812         CALL GRSUB2             Read the ln. text ptr from
               813  *                             ERAM(use GREAD1)or VDP
6D8B 2E        814         DATA EXTRAM             @EXTRAM:Source addr in ERM/VDP
6D8C BD2C58    815         DST  @EEE1,@PGMPTR      Put the result in @PGMPTR
6D8F 0F81      816         XML  CONTIN             And go to the line
```

```
                    818  ****************************************************************
                    819  *
                    820  *       ERPRNT - Print an error or warning message
                    821  *
                    822  *       ERPRNT - Entry point for ERROR
                    823  *       ERPNT5 - Entry point for WARNING
                    824  ****************************************************************
6D91 06601C         825  ERPRNT CALL CHRTAB              Load the character table
6D94 0F83           826         XML  SCROLL              Scroll the screen
6D96 BEA2E2         827         ST   :*:+OFFSET,RAM(NLNADD)  Put the * in
6D99 8A
6D9A BF2002         828         DST  NLNADD+2,@VARW      Set up for the message
6D9D E4
                    829  ERPNT5
6D9E 8674           830         CLR  @KEYBD              Enable main console
6DA0 330001         831         MOVE 1 FROM ROM(@FAC10)  TO @ARG+1 Get message length
6DA3 5D0000
6DA6 54
6DA7 865C           832         CLR  @ARG
6DA9 325CB0         833         MOVE @ARG FROM ROM(1(FAC10))  TO RAM(@VARW) Display
6DAC 200001
6DAF 54
6DB0 A1205C         834         DADD @ARG,@VARW          Start location for " IN "
6DB3 D75462         835         $IF @FAC10 .DEQ. MSG130 THEN  "* I/O ERROR [xx]xy"  1
6DB6 A64DD0
6DB9 9120           836           DINC @VARW             Update for one space separati1
6DBB BC5FE0         837           ST   RAM(4(PABPTR)),@ARG2+1  Create high order res1
6DBE 0404
6DC0 865E           838           CLR  @ARG2             Only display high order d   .1
6DC2 0670C1         839           CALL DISO              Display this  number.        1
6DC5 BC5FE0         840           ST   RAM(5(PABPTR)),@ARG2+1  Get low order result 1
6DC8 0504
6DCA E65F05         841           SRL  @ARG2+1,5         Remove mode identification bi1
6DCD 0670C1         842           CALL DISO              Output the number in decimal 1
                    843         $END IF
6DD0 D75460         844         $IF @FAC10 .DNE. MSGFST THEN                          1
6DD3 406DFC
6DD6 060036         845           CALL TONE2               Wake up the idiot!!!       1
6DD9 8E446D         846           $IF @PRGFLG .NE. 0 THEN  If program, print line # 2
6DDC FC
6DDD C72002         847             $IF @VARW .DH. >2F6 THEN  If will pass EOL       3
6DE0 F64DE9
6DE3 0F83           848               XML  SCROLL          Display on next line       3
6DE5 BF2002         849               DST  NLNADD+1,@VARW  Indent for the 'IN'        3
6DE8 E3
                    850             $END IF
6DE9 BFE001         851             DST  :IN:+>6060,RAM(1(VARW)) Put in the 'IN'     2
6DEC 20A9AE
6DEF A32000         852             DADD 4,@VARW           Display location for line 2
6DF2 04
6DF3 BC7642         853             ST   @CHAT,@EXP$         ASC destroys CHAT       2
6DF6 06A00A         854             CALL ASC                 DISPLAY THE LINE #      2
6DF9 BC4276         855             ST   @EXP$,@CHAT         Restore CHAT            2
                    856           $END IF
                    857         $END IF
```

```
6DFC 0F83     858          XML   SCROLL
6DFE 00       859          RTN
```

```
                    861  ************************************************************
                    862  *      TRACBK - Is used to trace back the error levels     *
                    863  *           through nested function references and          *
                    864  *           subprogram calls.  It takes care of issuing    *
                    865  *           the trace back information messages in these   *
                    866  *           two cases.  It leaves the stack unchanged      *
                    867  *           except in the case of a prescan error          *
                    868  *           occurring in an external subprogram.  If any   *
                    869  *           messages are issued, it returns with the       *
                    870  *           status set, else reset.                        *
                    871  ************************************************************
                    872  TRACBK
6DFF BD526E         873     DST  @VSPTR,@FAC8       Get a temp stack pointer
6E02 C55224         874     $WHILE @FAC8 .DH. @STVSPT  While not end of stack      1
6E05 4E28
6E07 D6E002         875       $IF RAM(2(FAC8)) .EQ. >68 GOTO TRAC05  If UDF entry  1
6E0A 52686E
6E0D 29
6E0E D6E002         876       $IF RAM(2(FAC8)) .EQ. >70 THEN  If temp UDF entry    2
6E11 52704E
6E14 1B
6E15 A76E00         877         DSUB 8,@VSPTR           Trash it so DELINK won't   2
6E18 08
6E19 4E29           878         BR   TRAC05               mess up the symbol table 2
                    879       $END IF
6E1B D6E002         880       $IF RAM(2(FAC8)) .EQ. >6A GOTO TRAC50 If subprog     1
6E1E 526A6E
6E21 D8
6E22 A75200         881       DSUB 8,@FAC8              Goto next entry on stack   1
6E25 08
6E26 4E02           882     $END WHILE
6E28 00             883     RTN                       If no UDFs or subprogs active
                    884  *
                    885  *      Trace back UDF references
                    886  *
                    887  TRAC05
6E29 8656           888     CLR  @FAC12               To cheat on ERPRNT
6E2B C05644         889     EX   @PRGFLG,@FAC12       Force line # NOT to be printed
6E2E D65701         890     $IF  @FAC13 .EQ. 1 THEN  If warning message            1
6E31 4E38
                    891  *         Place for the message already set in WRN$3
6E33 066D9E         892       CALL ERPNT5             Don't restore char set       1
6E36 4E3B           893     $SELSE                                                 1
6E38 066D91         894       CALL ERPRNT             Print the real error message 1
                    895     $END IF
6E3B BC4456         896     ST   @FAC12,@PRGFLG       Restore prog/imper flag
6E3E BD5C2C         897     DST  @PGMPTR,@ARG         Get the place of error for FNDLNE
6E41 066F53         898     CALL FNDLNE               Find the line that the error is in
6E44 BFA2E4         899     DST  :IN:+>6060,RAM(NLNADD+2)  Say 'IN' xx
6E47 A9AE
6E49 BF2002         900     DST  NLNADD+5,@VARW       Start place of line number
6E4C E7
6E4D 0670C1         901     CALL DISO                 Put out the line number
6E50 0F83           902     XML  SCROLL
                    903  TRAC09
```

```
6E52 BD5CB0   904      DST  RAM(@FAC8),@ARG   Save PGMPTR from the entry
6E55 52
              905  TRAC10
6E56 A75200   906      DSUB 8,@FAC8           Go on to next entry
6E59 08
6E5A C55224   907      $IF @FAC8 .DH. @STVSPT THEN  If not end of stack      1
6E5D 4EC2
6E5F D6E002   908        $IF RAM(2(FAC8)) .EQ. >68 THEN  If function entry   2
6E62 52684E
6E65 A8
6E66 D5B052   909          $IF RAM(@FAC8) .DEQ. @ARG THEN  If recursive      3
6E69 5C4E93
6E6C 31000F   910            MOVE 15 FROM ROM(MSGCIS) TO RAM(NLNADD+2)       3
6E6F A2E463
6E72 0A
6E73 0F83     911            XML  SCROLL          *CALLS ITSELF              3
6E75 A75200   912  TRAC12    DSUB 8,@FAC8         Goto next entry on stack   3
6E78 08
6E79 D6E002   913            $WHILE RAM(2(FAC8)) .EQ. >68   While functions  4
6E7C 52684E
6E7F 8C
6E80 D5B052   914              $IF RAM(@FAC8) .DNE. @ARG GOTO TRAC09         4
6E83 5C4E52
6E86 A75200   915              DSUB 8,@FAC8   Goto next entry on stack       4
6E89 08
6E8A 4E79     916            $SEND WHILE                                     3
6E8C CEE002   917            $IF RAM(2(FAC8)) .LE. >65 GOTO TRAC12  If string o3
6E8F 52654E
6E92 75
              918  *                                                  numeric
              919          $END IF
6E93 31000B   920          MOVE 11 FROM ROM(MSGCF) TO RAM(NLNADD+2)          2
6E96 A2E463
6E99 19
6E9A 066F53   921          CALL FNDLNE          Find the line                2
6E9D BF2002   922          DST  NLNADD+14,@VARW  Place to display it         2
6EA0 F0
6EA1 0670C1   923          CALL DISO            Display the line number      2
6EA4 0F83     924          XML  SCROLL          *CALLED FROM                 2
6EA6 4E52     925          BR   TRAC09          Go on                        2
              926  *       Jump always
              927        $END IF
6EA8 CAE002   928        $IF RAM(2(FAC8)) .L. >66 GOTO TRAC10  If not perm   1
6EAB 52664E
6EAE 56
6EAF C75209   929        $WHILE @FAC8 .DH. VRAMVS While still not at bottom  2
6EB2 584EC2
6EB5 D6E002   930          $IF RAM(2(FAC8)) .EQ. >6A GOTO TRAC51  If subprog  2
6EB8 526A6E
6EBB E9
6EBC A75200   931          DSUB 8,@FAC8           Go down an entry            2
6EBF 08
6EC0 4EAF     932        $SEND WHILE                                          1
              933      $END IF
6EC2 8E446E   934      $IF @PRGFLG .NE. 0 THEN  If not imperative             1
```

```
6EC5 D6
6EC6 31000B    935        MOVE 11 FROM ROM(MSGCF) TO RAM(NLNADD+2)
6EC9 A2E463
6ECC 19
6ECD BF2002    936        DST  NLNADD+14,@VARW Place to display line #
6ED0 F0
6ED1 06A00A    937        CALL ASC             Display it
6ED4 0F83      938        XML  SCROLL
               939      $END IF
6ED6 4F45      940      BR   RTNSET            Return w/condition set
               941  *
               942  *       Trace back subprogram calls
               943  *
6ED8 D65701    944  TRAC50 $IF @FAC13 .EQ. 1 THEN  If warning message only
6EDB 4EE2
6EDD 066D9E    945           CALL ERPNT5          Don't restore char set
6EE0 4EE5      946         $SELSE
6EE2 066D91    947           CALL ERPRNT          Print the real message
               948         $END IF
6EE5 8E446F    949         $IF @PRGFLG .EQ. 0 GOTO RTNSET
6EE8 45
6EE9 8E446F    950  TRAC51 $IF @PRGFLG .EQ. 0 GOTO RETNOS
6EEC 48
6EED BFA2E4    951         DST  :IN:+>6060,RAM(NLNADD+2)  Display 'IN'
6EF0 A9AE
6EF2 BF5602    952         DST  NLNADD+6,@FAC12   Display loc of name
6EF5 E8
6EF6 BD5AB0    953  TRAC55 DST  RAM(@FAC8),@FAC16 Get s.t. pointer
6EF9 52
6EFA 8654      954         CLR  @FAC10            Need a double length
6EFC BC55E0    955         ST   RAM(1(FAC16)),@FAC10+1  Get the name length
6EFF 015A
6F01 BD5AE0    956         DST  RAM(4(FAC16)),@FAC16  Get the name pointer
6F04 045A
6F06 3454B0    957         MOVE @FAC10 FROM RAM(@FAC16) TO RAM(@FAC12)  Display
6F09 56B05A
               958         $REPEAT                Add offset to each char
6F0C A2B056    959           ADD  OFFSET,RAM(@FAC12)
6F0F 60
6F10 9156      960           DINC @FAC12
6F12 9354      961           DDEC @FAC10
6F14 8F544F    962         $UNTIL @FAC10 .DEQ. 0
6F17 0C
6F18 0F83      963         XML  SCROLL          Scroll the screen   'CALLED
6F1A 31000B    964         MOVE  11 FROM ROM(MSGCF) TO RAM(NLNADD+2)   FROM'
6F1D A2E463
6F20 19
6F21 BD5452    965         DST  @FAC8,@FAC10      In case at top level
6F24 BD52E0    966         DST  RAM(6(FAC8)),@FAC8  Get LSUBP off stack
6F27 0652
6F29 8F526F    967         $IF @FAC8 .DNE. 0 THEN  If not top level call
6F2C 33
6F2D BF5602    968           DST  NLNADD+15,@FAC12 Display loc of name
6F30 F1
6F31 4EF6      969           BR   TRAC55
```

```
                    970           $END IF
                    971   *       Now find original line number
6F33 BD5EEF         972           DST  RAM(-6(FAC10)),@ARG2 Get pointer to line number
6F36 FFFA54
6F39 066F49         973           CALL GETLN2               Get the actual line numger
6F3C BF2002         974           DST  NLNADD+15,@VARW      Place to put line number
6F3F F1
6F40 0670C1         975           CALL DISO                 Display the line number
6F43 0F83           976           XML  SCROLL               Scroll the mess up
                    977   *       RETURN WITH CONDITION BIT SET
6F45 D40000         978   RTNSET  CEQ  @0,@0                SET CONDITION BIT
6F48 01             979   RETNOS  RTNC
                    980   *
                    981   *
                    982   GETLN2
6F49 975E           983           DDECT @ARG2
6F4B 06802C         984           CALL GRSUB2               Read 2 bytes of data from ERAM
6F4E 5E             985           DATA ARG2                 (use GREAD1) or VDP
                    986   *
6F4F BD5E58         987           DST  @EEE1,@ARG2          Put the result into @ARG2
6F52 00             988           RTN
```

```
                    990  *       Given a specific PGMPTR(in ARG) find the line
                    991  *       number of the line it points into and put the
                    992  *       actual line number in ARG2
                    993  *
                    994  FNDLNE
6F53 BD6030         995          DST @STLN,@ARG4     Get pointer into line # buffer
6F56 9560           996          DINCT @ARG4         Point at the line pointer
6F58 BD5E60         997          DST  @ARG4,@ARG2    Get line pointer
6F5B 8762           998          DCLR @ARG6          Start with a zero value
                    999  *
6F5D C96032        1000          $WHILE @ARG4 .DL. @ENLN    While in line buffer     1
6F60 6F7C
6F62 06802C        1001            CALL GRSUB2        Get the ln # from ERAM(GREAD1)/V1
6F65 60            1002            DATA ARG4          @ARG4:Source addr. on ERAM/VDP   1
6F66 CD585C        1003            $IF @EEE1 .DLE. @ARG THEN                           2
6F69 6F76
                   1004  *         If pointer lower than the one we're looking for
6F6B C55862        1005              $IF @EEE1 .DH. @ARG6 THEN  If closer              3
6F6E 4F76
6F70 BD5E60        1006                DST @ARG4,@ARG2  Make it the one                3
6F73 BD6258        1007                DST @EEE1,@ARG6  Get the ptr to the line        3
                   1008              $END IF
                   1009            $END IF
6F76 A36000        1010            DADD 4,@ARG4     Goto next line in buffer         1
6F79 04
6F7A 4F5D          1011          $SEND WHILE
                   1012  *
6F7C 066F49        1013          CALL GETLN2             Get the line number.
6F7F B25E7F        1014          AND >7F,@ARG2           Reset the breakpt if any
6F82 00            1015          RTN
```

```
                1017  ********************************************************
                1018  *                                                      *
                1019  *      LLIST - Lists one program line on the screen. The *
                1020  *              entrypoint to the line is given in STPT. *
                1021  *                                                      *
                1022  *      In this routine, FAC2 is used as a flag to indicate *
                1023  *      that the most recent character output was an    *
                1024  *      alphanumeric character.  If the next character is *
                1025  *      also an alphanumeric character, then the two are *
                1026  *      separated by a space.                           *
                1027  *                                                      *
                1028  ********************************************************
                1029  LLIST
6F83 DA4580     1030        $IF .BIT7 @FLAG .EQ. 1 THEN  If program protected  1
6F86 6F8C
6F88 066CFB     1031  ERRPV   CALL ERR$$              * PROTECTION VIOLATION   1
6F8B 27         1032          DATA 39                                          1
                1033        $END IF
6F8C 06801A     1034        CALL OUTREC             Make room for a new line
6F8F BD5EB0     1035        DST  RAM(@EXTRAM),@ARG2 Prepare for line # printing
6F92 2E
6F93 B25E7F     1036        AND >7F,@ARG2           Reset possible B.P.
6F96 0670B0     1037        CALL OUTLN              Display line in free format
6F99 BD2008     1038        DST  @CCPADD,@VARW      Copy position for editing
6F9C 9120       1039        DINC @VARW              Leave room for space
6F9E BD2CE0     1040        DST  RAM(2(EXTRAM)),@PGMPTR Get pointer to line
6FA1 022E
6FA3 BF4C00     1041  LLIS$0 DST  : :,@FAC2         Clear blank fill and set space
6FA6 20
6FA7 0F79       1042  LLI$12 XML  PGMCHR            Get next token on line
6FA9 8E4270     1043        $IF @CHAT .EQ. 0 GOTO LLIS$9  Exit on end of line
6FAC AF
6FAD 8E4D6F     1044        $IF  @FAC3 .NE. 0 THEN If separator needed          1
6FB0 BA
6FB1 C04D42     1045          EX   @CHAT,@FAC3    Save CHAT and bare the sep  1
6FB4 0670D9     1046          CALL DSPCHR         Put the separator out       1
6FB7 C04D42     1047          EX   @CHAT,@FAC3    Restore CHAT                1
                1048        $END IF
                1049
                1050  *     Next thing to determine is whether or not we need
                1051  *     a space for separation with the next stuff
                1052
6FBA 864D       1053  LLI$15 CLR  @FAC3           Assume we'll get alphanumeric
6FBC D64282     1054        $IF @CHAT .EQ. SSEP$ GOTO LLI$16  If double-colon
6FBF 6FC6
6FC1 D642B5     1055        $IF @CHAT .EQ. COLON$ THEN If colon now and colon   1
6FC4 4FCB
6FC6 D64AB5     1056  LLI$16   $IF @FAC .EQ. COLON$ GOTO LLI$17 before-separate 1
6FC9 6FEB
                1057        $END IF
6FCB CA42B3     1058        $IF @CHAT .HE. COMMA$ THEN Figure out separator rang1
6FCE 4FD5
6FD0 CA42BA     1059          $IF @CHAT .L. OR$ GOTO LLIS$2                     1
6FD3 4FF7
                1060        $END IF
```

```
6FD5 C642BD  1061          $IF @CHAT .H. NOT$ THEN Figure out separator range  1
6FD8 4FDF
6FDA CA42C8  1062            $IF @CHAT .L. NUMCO$ GOTO LLIS$2                  1
6FDD 4FF7
             1063          $END IF
6FDF BE4D20  1064          ST   : :,@FAC3          Prepare for alfa indication
6FE2 8E4C6F  1065          $IF @FAC2 .NE. 0 THEN alfanum-alfanum combination  1
6FE5 F7
6FE6 D65420  1066            $IF @FAC10 .NE. : : THEN Don't output 2 spaces   2
6FE9 6FF7
6FEB BC4C42  1067  LLI$17      ST   @CHAT,@FAC2      Save CHAT somewhere       2
6FEE BE4220  1068              ST   : :,@CHAT        And display a space       2
6FF1 0670D9  1069              CALL DSPCHR                                     2
6FF4 BC424C  1070              ST   @FAC2,@CHAT      Retrieve CHAT             2
             1071            $END IF
             1072          $END IF
             1073
6FF7 C04C4D  1074  LLIS$2 EX   @FAC3,@FAC2          Could be for the next time toc
             1075
             1076  *       That takes care of all the extra spaces we might need
             1077
6FFA DA4280  1078          $IF .BIT7 @CHAT .EQ. 0 THEN just copy variable names1
6FFD 5011
             1079            $REPEAT
6FFF 0670D9  1080              CALL DSPCHR         Copy the character to output 2
7002 0F79    1081              XML  PGMCHR         Get the next character       2
7004 8E4270  1082              $IF @CHAT .EQ. 0 GOTO LLIS$9 But exit on EOL     2
7007 AF
7008 DA4280  1083            $UNTIL .BIT7 @CHAT .EQ. 1                         1
700B 6FFF
700D 864A    1084            CLR  @FAC             No spaces if ': ' or '::'   1
700F 4FBA    1085            BR   LLI$15                                       1
             1086          $END IF
7011 D642C8  1087          $IF @CHAT .NE. NUM$ THEN                            1
7014 701E
7016 D642C7  1088            $IF @CHAT .NE. STRIN$ GOTO LLIS$3                 1
7019 5048
701B 0670D6  1089            CALL DSPQUO           Display first quote of string1
             1090          $END IF
             1091
             1092  *       This place is the general location for strings
             1093  *       both quoted and unquoted.
             1094
701E 0F79    1095          XML  PGMCHR             Get string length in CHAT
7020 BC4A42  1096          ST   @CHAT,@FAC         Copy in temporary space
7023 8E4A70  1097          $WHILE @FAC .NE. 0 Also take care of empty strings  1
7026 3C
7027 0F79    1098            XML  PGMCHR           Get next character          1
7029 8E4C50  1099            $IF @FAC2 .EQ. 0 THEN Alpha means unquoted string 2
702C 35
702D D622  1100               $IF @CHAT .EQ. QUOTE THEN                       3
7030 5035
7032 0670D9  1101                CALL DSPCHR      Display two quotes for one   3
             1102              $END IF
             1103            $END IF
```

```
7035 0670D9  1104            CALL DSPCHR            Display 2nd quote or char    1
7038 924A    1105            DEC  @FAC              Update string length; get next1
703A 5023    1106          $SEND WHILE              We also fall through on 0 here
703C 8E4C50  1107          $IF @FAC2 .NE. 0 GOTO LLIS$1 non-alfa end means extra
703F 8E
7040 0670D6  1108          CALL DSPQUO              Display closing quote
7043 BE4C20  1109          ST   : :,@FAC2           Cause space before following
7046 508E    1110          BR   LLIS$1                  alpha
             1111
             1112  *       Try to decode line numbers and keywords
             1113
             1114  LLIS$3
7048 D642C9  1115          $IF @CHAT .EQ. LN$ THEN Decode line #                 1
704B 505C
704D 0F79    1116            XML  PGMCHR            Get the high order byte first1
704F BC5E42  1117            ST   @CHAT,@ARG2                                    1
7052 0F79    1118            XML  PGMCHR                                         1
7054 BC5F42  1119            ST   @CHAT,@ARG3         information as collected he1
7057 0670B0  1120            CALL OUTLN             Display the actual informatio1
705A 508E    1121            BR   LLIS$1            And continue                 1
             1122          $END IF
             1123
             1124  *       Now it has to be a normal keyword
             1125
705C BF4AB4  1126          DST  #KEYTAB,@FAC        Addr of keytab for search
705F E0
7060 0F84    1127          XML  IO                  Search keyword table
7062 00      1128          DATA 0                   Select table search
             1129  *                                FAC8 returns w/ptr to keyword
             1130  *                                FAC4 has length
7063 330001  1131  LLIS$6 MOVE 1 FROM ROM(@FAC8) TO @CHAT
7066 420000
7069 52
             1132
             1133  *       And output the thus found character
             1134
706A 0670D9  1135          CALL DSPCHR              Display character on screen
706D 9152    1136          DINC @FAC8               Update FAC8 for next reference
706F 924F    1137          DEC  @FAC5               Count number of characters
7071 5063    1138          BR   LLIS$6              Always less than 255
7073 D64A83  1139          $IF @FAC .EQ. TREM$ GOTO LLIS$7 No spaces after !
7076 708C
7078 D64A9A  1140          $IF @FAC .EQ. REM$ GOTO LLIS$7 No spaces after REM
707B 708C
707D CA4AB3  1141          $IF @FAC .L. COMMA$ GOTO LLIS$0 Master stuff =>space
7080 4FA3
7082 D64AED  1142          $IF @FAC .EQ. USING$ GOTO LLIS$0 Master stuff =>space
7085 6FA3
7087 D64AFD  1143          $IF @FAC .EQ. NUMBE$ THEN '#' never followed by spac1
708A 508E
708C 864C    1144  LLIS$7    CLR  @FAC2             Avoid spaces behind here     1
             1145          $END IF
708E 864D    1146  LLIS$1 CLR  @FAC3               Indicate separator not needed
7090 4FA7    1147          BR   LLI$12              Continue for next keyword
```

```
                    1149  ********************************************************************
                    1150  *       Convert a number from binary to ASCII                      *
                    1151  *       Input:  binary number in ARG2-3                            *
                    1152  *       Output: pointer to ASCII number in FAC11 with the          *
                    1153  *               actual number lying just before and ending         *
                    1154  *               with FAC10, i.e. the last digit of the             *
                    1155  *               ASCII representation is in FAC10; number           *
                    1156  *               of digits in the number in ARG5                    *
                    1157  ********************************************************************
                    1158  CVRTLN
7092 8661           1159          CLR   @ARG5                 Start with 0 characters
7094 BE6767         1160          ST    ARG11,@ARG11          Select first address + 1
                    1161          $REPEAT
7097 875C           1162            DCLR @ARG                 Clear upper 2 bytes of 4-byte1
7099 9267           1163            DEC  @ARG11               Go to next position          1
709B AF5C00         1164            DDIV 10,@ARG              Compute least significant rem1
709E 0A
709F A25F30         1165            ADD  >30,@ARG3            Always < 10 off course       1
70A2 BC9067         1166            ST   @ARG3,*ARG11         Store it in ARG              1
70A5 5F
70A6 BD5E5C         1167            DST  @ARG,@ARG2           Replace remainder by result  1
70A9 9061           1168            INC  @ARG5                Update total # of characters 1
70AB 8F5E50         1169          $UNTIL @ARG2 .DEQ. 0        Until whole number converted
70AE 97
70AF 00             1170  LLIS$9  RTN
                    1171
                    1172
                    1173  *       Output a line number to a device (or screen)
                    1174  OUTLN
70B0 067092         1175          CALL CVRTLN               Convert from binary to ASCII
70B3 BC4290         1176  OUTL$1  ST   *ARG11,@CHAT         Get the next character
70B6 67
70B7 0670D9         1177          CALL DSPCHR               Display the character
70BA 9067           1178          INC  @ARG11               Increment the char pointer
70BC 9261           1179          DEC  @ARG5                Decrement number of digits
70BE 50B3           1180          BR   OUTL$1               Output digit if not all out
70C0 00             1181          RTN
                    1182
                    1183
                    1184  *       Display number on the screen
70C1 067092         1185  DISO    CALL CVRTLN               Convert from binary to ASCII
70C4 BCB020         1186  DISP$1  ST   *ARG11,RAM(@VARW)    Get most significant character
70C7 9067
70C9 A2B020         1187          ADD  OFFSET,RAM(@VARW)    Display character on screen
70CC 60
70CD 9120           1188          DINC @VARW                Update screen pointer
70CF 9067           1189          INC  @ARG11               Get next position
70D1 9261           1190          DEC  @ARG5                Update count
70D3 50C4           1191          BR   DISP$1               And loop until finished
70D5 00             1192          RTN
                    1193
                    1194
                    1195  *       Put out a quote
                    1196  DSPQUO
70D6 BE4222         1197          ST   QUOTE,@CHAT          DISPLAY A QUOTE
```

```
                1198  *       Put out next character
70D9 C40607     1199  DSPCHR $IF @CCPPTR .H. @RECLEN THEN Action on end of screen1
70DC 50E5
70DE 06801A     1200           CALL OUTREC           Output current record        1
70E1 A72000     1201           DSUB 32,@VARW         Keep track of begin of line  1
70E4 20
                1202         $END IF
70E5 BCB008     1203         ST   @DSRFLG,RAM(@CCPADD) Put offset on screen
70E8 17
70E9 A0B008     1204         ADD  @CHAT,RAM(@CCPADD)  Add in the character
70EC 42
70ED 9108       1205         DINC @CCPADD          Bump output pointer
70EF 9006       1206         INC  @CCPPTR          Update current line position
70F1 BC5442     1207         ST   @CHAT,@FAC10     FAC10 may be used by OUTREC !!
70F4 00         1208         RTN
```

```
                1210  ************************************************************
                1211  *       Static scanner to build the main symbol table and  *
                1212  *       to build symbol tables for each subprogram and to  *
                1213  *       build the subprogram table.  Checks some errors and*
                1214  *       aborts if any detected.                            *
                1215  ************************************************************
                1216  PRESCN
70F5 870A       1217          DCLR @CALIST              Initialize call list
70F7 BF0600     1218          DST  10,@DFLTLM           Set default array size
70FA 0A
70FB 8716       1219          DCLR  @XFLAG              Initialize prescan flag bits
                1220  *                                     and FOR/NEXT counter
70FD 8E4451     1221          $IF  @PRGFLG .EQ. 0 THEN  If imperative                1
7100 0E
7101 BF2C08     1222            DST  CRNBUF,@PGMPTR     Pointer to 1st token         1
7104 20
7105 0F79       1223            XML  PGMCHR             Get the 1st token            1
7107 0F83       1224            XML  SCROLL             Scroll the screen            1
7109 06714E     1225            CALL SCAN10             Do the static scan of the lin1
710C 5116       1226          $SELSE                    If program                   1
710E 06711D     1227            CALL SCAN               Scan the program             1
7111 B24590     1228            AND  >90,@FLAG          Reset all flags but the TRACE1
                1229  *                                 & LIST/EDIT protection
7114 8748       1230            DCLR @LSUBP                                          1
                1231          $END IF
7116 BD6EA3     1232          DST  RAM(SAVEVP),@VSPTR   Initialize VSPTR.
7119 88
711A 05A004     1233          B    EXEC                 Execute the program or str
```

```
                1235  ****************************************************************
                1236  *                        Static scanner                        *
                1237  ****************************************************************
711D BD2E32     1238  SCAN    DST  @ENLN,@EXTRAM      1st addr of line # table
7120 A32E00     1239          DADD 3,@EXTRAM
7123 03
7124 873E       1240          DCLR @SYMTAB            Clear the symbol table
7126 873A       1241          DCLR @SUBTAB            Clear the subprogram table.
7128 8E8084     1242          $IF @RAMTOP .EQ. 0 THEN                                  1
712B 5134
712D BD4030     1243            DST  @STLN,@FREPTR    Initialize free-space pointer1
7130 9340       1244            DDEC @FREPTR          Back up from line # table    1
7132 513E       1245          $SELSE                                                   1
7134 BD8086     1246            DST  @STLN,@RAMFRE    Initialize ERAM free-space   1
7137 30
7138 938086     1247            DDEC @RAMFRE          pointer.                       1
713B BD4070     1248            DST  @>70,@FREPTR     Initialize with no prog in VD1
                1249          $END IF
713E 8643       1250          CLR  @BASE              OPTION BASE = 0
7140 BD1840     1251          DST  @FREPTR,@STRSP     Initialize string space
7143 BD1A18     1252          DST  @STRSP,@STREND
7146 BD1230     1253          DST  @STLN,@LINUM
7149 9512       1254          DINCT @LINUM            Point to last line in program
                1255  *THE FOLLOWING 20 STATEMENTS CANNOT BE SEPARATED OR THE
                1256  *ASSEMBLY LANGUAGE CODE WILL NOT WORK - SRH
714B 0F8F       1257          XML  SCNSMT             Scan the program
714D 00         1258          DATA 0                  Entire program flag
714E 0F8F       1259  SCAN10  XML  SCNSMT             Scan the statement
7150 02         1260          DATA 2                  Single statememtflag
7151 526D       1261          BR   SCANRT             Normal end of scan          0
7153 51B0       1262          BR   SCNDEF             Scan a def                  2
7155 5171       1263          BR   SCNDIM             Scan a dim                  4
7157 526E       1264          BR   CALLS              Scan a call                 6
7159 5180       1265          BR   SCNOPT             Scan an option base         8
715B 52C5       1266          BR   SUBS               Scan a sub                  A
715D 53D8       1267          BR   SUBNDS             Scan a subexit              C
715F 53D8       1268          BR   SUBNDS             Scan a subend               E
7161 523C       1269          BR   CALENT             Call ENTER                 10
7163 524E       1270          BR   ERROLP             * ONLY LEGAL IN A PROGRAM  12
7165 576A       1271          BR   ERRNWF             * NEXT WITHOUT FOR         14
7167 576E       1272          BR   ERRFNN             * FOR/NEXT NESTING         16
7169 5776       1273          BR   ERRMS              * MISSING SUBEND           18
716B 5259       1274          BR   ERRSYN             * SYNTAX ERROR             1A
716D 5766       1275          BR   ERRMEM             * MEMORY FULL              1C
716F 5756       1276          BR   ERRIBS             * ILLEGAL AFTER SUBPROGRAM1E
```

```
                    1278  *           SPECIALLY SCANNED STATEMENTS
                    1279
                    1280  *       DIM STATEMENT
7171 DA1640         1281  SCNDIM $IF  .BIT(IFFLAG) @XFLAG .EQ. 1 GOTO ERRSYN
7174 5259
                    1282          $REPEAT
7176 067400         1283            CALL ENTER              Declare this symbol
7179 D642B3         1284          $UNTIL @CHAT .NE. COMMA$ Loop if more
717C 7176
717E 51A9           1285          BR   SCAN25              Must have EOL now
                    1286
                    1287  *       OPTION BASE STATEMENT
7180 067245         1288  SCNOPT  CALL IMPIF               Can't be imperative or in 'IF'
7183 067743         1289          CALL PGMERR              OPTION-therefore must be BASE
7186 DA1602         1290          $IF .BIT(OPTFLG) @XFLAG .EQ. 1 GOTO ERROBE
7189 575E
                    1291  *                                Error if OPTFLG already set
718B 067252         1292          CALL SYNCHK              Must have a 'BASE'
718E F1             1293          DATA BASE$
718F 067252         1294          CALL SYNCHK              Must have a numeric constant
7192 C8             1295          DATA NUMCO$
7193 067252         1296          CALL SYNCHK              Must have 1-char numeric con
7196 01             1297          DATA 1
7197 8643           1298          CLR  @BASE               Assume BASE=0
7199 A64230         1299          SUB  >30,@CHAT           Must be 0 or 1
719C 71A4           1300          BS   SCAN20              OK if 0
719E 9242           1301          DEC  @CHAT               Check for a 1
71A0 575E           1302          BR   ERROBE              If it was not a 1 then ERR
71A2 9043           1303          INC  @BASE               Set OPTION BASE=1
71A4 B61602         1304  SCAN20  SB   @XFLAG,OPTFLG        Set the option base flag
71A7 0F79           1305  SCAN22  XML  PGMCHR              Now - check for end-of-line
                    1306
71A9 06725D         1307  SCAN25  CALL CHKEND              If not EOL or :: or ! - error
71AC 7242           1308          BS   CONSCN              If EOS-continue scan
71AE 5259           1309          BR   ERRSYN              * SYNTAX ERROR
                    1310
                    1311  *       DEF STATEMENT
71B0 067245         1312  SCNDEF  CALL IMPIF               Can't be imperative or in 'IF'
71B3 B61684         1313          OR   >84,@XFLAG          Set function bit
                    1314  *                                Set ENTERX bit
71B6 067400         1315          CALL ENTER               Enter the function name
                    1316  *                                ENTER resets function bit
71B9 DAB03E         1317          CLOG >07,RAM(@SYMTAB)    Did function have parm?
71BC 07
71BD 7236           1318          BS   SCAN55              No...
71BF B61680         1319          SB   @XFLAG,ENTXFL        ENTERX call for parm enter
71C2 B64508         1320          SB   @FLAG,3             Fake it so symbol table
                    1321  *                                  searches won't be made
71C5 067403         1322          CALL ENTERW              Enter the parameter
71C8 B245F7         1323          RB   @FLAG,3             Reset function bit
71CB 067252         1324          CALL SYNCHK              Complex symbol must be
71CE B6             1325          DATA RPAR$                  followed by ')='
71CF 067252         1326          CALL SYNCHK
71D2 BE             1327          DATA EQUAL$
71D3 35001D         1328          MOVE 29 FROM RAM(@SYMTAB) TO RAM(CRNBUF)
```

```
71D6 A820B0
71D9 3E
71DA BD00A8  1329          DST  RAM(CRNBUF+4),@VAR0  Get pointer to name
71DD 24
71DE 8E8084  1330          $IF  @RAMTOP .NE. 0 THEN  If ERAM program          1
71E1 71EE
             1331  *          If ERAM must fix up the name pointer
             1332  *          because the name was moved too
71E3 A5003E  1333            DSUB @SYMTAB,@VAR0   Offset into entry           1
71E6 A30008  1334            DADD CRNBUF,@VAR0    New location of name       1
71E9 20
71EA BDA824  1335            DST  @VAR0,RAM(CRNBUF+4) Put it in              1
71ED 00
             1336          $END IF
71EE BD40E0  1337          DST  RAM(2(SYMTAB)),@FREPTR Reset free space ptr
71F1 023E
71F3 BF3E08  1338          DST  CRNBUF,@SYMTAB    Point into crunch buffer
71F6 20
71F7 9340    1339          DDEC @FREPTR
71F9 06725D  1340  SCAN35  CALL CHKEND            If EOL or ! or ::
71FC 7230    1341          BS   SCAN50            Yes
71FE CE4200  1342          $IF  @CHAT .GT. 0 GOTO SCAN40
7201 7214
7203 D642C8  1343          $IF  @CHAT .EQ. NUM$ GOTO SCAN45 If numeric-skip it
7206 720D
7208 D642C7  1344          $IF  @CHAT .EQ. STRIN$ THEN If string-skip          1
720B 5210
720D 06774B  1345  SCAN45    CALL SKPSTR          Skip the string or numeric   1
             1346          $END IF
7210 0F79    1347          XML  PGMCHR            Get next character
7212 51F9    1348          BR   SCAN35
             1349  *       Jump always
7214 B61680  1350  SCAN40  SB   @XFLAG,ENTXFL     Make an ENTERX call
7217 067408  1351          CALL ENTERX            Enter the symbol
             1352
             1353  *****Relink to keep parameter at the beginning of the table
             1354
721A D73E08  1355          $IF  @SYMTAB .DEQ. CRNBUF GOTO SCAN35 If no entry
721D 2071F9
7220 BDE002  1356          DST  RAM(CRNBUF+2),RAM(2(SYMTAB))  Put link in
7223 3EA822
7226 BDA822  1357          DST  @SYMTAB,RAM(CRNBUF+2)  Put link in
7229 3E
722A BF3E08  1358          DST  CRNBUF,@SYMTAB    Put new pointer in
722D 20
722E 51F9    1359          BR   SCAN35            Go on
             1360  *       Jump always
7230 BD3EA8  1361  SCAN50  DST  RAM(CRNBUF+2),@SYMTAB  Delink the parameter
7233 22
7234 5242    1362          BR   CONSCN            Continue the scan
             1363
             1364
7236 067252  1365  SCAN55  CALL SYNCHK
7239 BE      1366          DATA EQUAL$
723A 5242    1367          BR   CONSCN
```

```
                1368
723C B61680     1369  CALENT SB    @XFLAG,ENTXFL      Set enterx flag
723F 067408     1370         CALL  ENTERX             Enter in symbol table
7242 0F8F       1371  CONSCN XML   SCNSMT              Return to 9900 code to resume
7244 01         1372         DATA  1                   Return call to 9900 code
                1373
7245 DA1640     1374  IMPIF  $IF   .BIT(IFFLAG) @XFLAG .EQ. 1 GOTO ERRSYN Not in if
7248 5259
724A 8E4452     1375  IMPILL $IF   @PRGFLG .NE. 0 GOTO SCANRT Program mode-OK-rtn
724D 6D
724E 066CFB     1376  ERROLP CALL  ERR$$               If imperative - error
7251 1B         1377         DATA  27                  Only legal in a program
                1378
                1379  *      Syntax required token routine
                1380
7252 884A       1381  SYNCHK FETCH @FAC
7254 D4424A     1382         $IF   @CHAT .EQ. @FAC GOTO PGMERR
7257 7743
7259 066CFB     1383  ERRSYN CALL  ERR$$
725C 03         1384         DATA  3                   Syntax error
725D DA4280     1385  CHKEND $IF   .BIT7 @CHAT .EQ. 1 THEN                          1
7260 726B
7262 CA4284     1386           $IF @CHAT .L. TREM$+1 THEN                           2
7265 726B
7267 D40000     1387             CEQ @0,@0           Force COND to "SET"             2
726A 01         1388             RTNC                                                2
                1389           $END IF
                1390         $END IF
726B 8E42       1391         CZ @CHAT                Set COND according to CHA
726D 01         1392  SCANRT RTNC
```

```
                    1394  ************************************************************
                    1395  *                 CALLS routine                            *
                    1396  *       This routine scans the CALL statement.             *
                    1397  *       Get the subprogram name, search the table and      *
                    1398  *       update the call list(value stack area) if         *
                    1399  *       necessary.  Share the same XML search routine      *
                    1400  *       as the symbol table code uses.                     *
                    1401  ************************************************************
                    1402  CALLS
726E 0F79           1403          XML   PGMCHR        Get token after call.
7270 067252         1404          CALL  SYNCHK        Check subprogram name.
7273 C8             1405          DATA  UNQST$        Must start w/ unquoted string.
7274 C6420F         1406          $IF @CHAT .H. 15 GOTO ERRNTL  * NAME TOO LONG!!
7277 775A
7279 BD002C         1407          DST   @PGMPTR,@VAR0     Save prog pointer to name.
727C BE5B4A         1408          ST    FAC,@FAC17        Set up a pointer.
727F BC5942         1409          ST    @CHAT,@FAC15      Save name length.
7282 BC5A42         1410          ST    @CHAT,@FAC16      Save name length as a counter.
                    1411  CALL20
7285 0F79           1412          XML   PGMCHR            Get one byte of name.
7287 BC905B         1413          ST    @CHAT,*FAC17      Store that character in FAC ar
728A 42
728B 905B           1414          INC   @FAC17            Increment pointer.
728D 925A           1415          DEC   @FAC16            Decrement counter.
728F 5285           1416          BR    CALL20            Get next character.
                    1417  *
                    1418  *                               Exchange call list address with
                    1419  *                               sbol table address to run the
                    1420  *                               same search routine used for
                    1421  *                               symbol table serach.
7291 C10A3E         1422          DEX   @SYMTAB,@CALIST
7294 0F7D           1423          XML   SCHSYM            Search to see if name there.
7296 C13E0A         1424          DEX   @CALIST,@SYMTAB   Exchange back both addresses.
7299 72C1           1425          BS    SCAN67            If name found do nothing.
729B 8E8089         1426          $IF @RAMFLG .NE. 0 THEN   If not imperative and ERAM1
729E 72B1
72A0 0F77           1427            XML   VPUSH           Put first 8 byte of name.     1
72A2 BD006E         1428            DST   @VSPTR,@VAR0    Pointing to new name loc.     1
72A5 CE5908         1429            $IF @FAC15 .GT. 8 THEN  If more charcters in name.2
72A8 52B1
72AA 350008         1430              MOVE 8 FROM @FAC8 TO @FAC  Move rest of the name2
72AD 4A52
72AF 0F77           1431              XML  VPUSH            Push one more time.       2
                    1432            $END IF
                    1433          $END IF
72B1 864A           1434          CLR   @FAC
72B3 BC4B59         1435          ST    @FAC+15,@FAC1     Put in name length.
72B6 BD4C0A         1436          DST   @CALIST,@FAC2     Put in call list link.
72B9 BD4E00         1437          DST   @VAR0,@FAC4       Put in pointer to name.
72BC 0F77           1438          XML   VPUSH             Put the entry in the VDP.
72BE BD0A6E         1439          DST   @VSPTR,@CALIST    Change pointer to call list.
                    1440
72C1 0F79           1441  SCAN67  XML   PGMCHR
72C3 5242           1442          BR    CONSCN
```

```
                1444  ********************************************************
                1445  *              SUBS routine                            *
                1446  *       This routine scans SUB statement in subprogram.*
                1447  *       First check the subprogram name and call list. *
                1448  *       Then builds subprogram table without argument list,*
                1449  *       scans symbols in the subprogram and create symbol *
                1450  *       table for the subprogram, make entry to the sub-  *
                1451  *       program table and add (if necessary) to call list.*
                1452  ********************************************************
                1453  SUBS
72C5 067245     1454        CALL IMPIF                  Can't be imperative or in 'IF'
72C8 8E1757     1455        $IF @FORNET .NE. 0 GOTO ERRFNN  Check FOR-NEXT nesting.
72CB 6E
72CC DA1601     1456        $IF .BIT(REMODE) @XFLAG .EQ. 0 THEN Called first time. 1
72CF 52DA
72D1 DA1608     1457          $IF .BIT(SUBFLG) @XFLAG .EQ. 1 GOTO ERRMS           1
72D4 5776
                1458  *       Cannot be in subprogram.    Can't start another one.
72D6 BDA392     1459          DST  @SYMTAB,RAM(TABSAV)        Finish off main tabl1
72D9 3E
                1460        $END IF
                1461  *                                   From the second SUB statement.
72DA 873E       1462          DCLR @SYMTAB              Start with empty symbol table.
72DC B61628     1463          OR   >28,@XFLAG           Set flag for SAFLG and SUBFLG.
72DF B216FE     1464          RB   @XFLAG,REMODE        Reset REMODE flag.
72E2 0F79       1465          XML  PGMCHR               Get name behind SUB statement.
72E4 067252     1466          CALL SYNCHK               Make sure it's unquoted string.
72E7 C8         1467          DATA UNQST$
72E8 C6420F     1468          $IF  @CHAT .H. 15 GOTO ERRNTL Length must be <= .
72EB 775A
72ED BC4B42     1469          ST   @CHAT,@FAC1          Save name length.
72F0 BD4E2C     1470          DST  @PGMPTR,@FAC4         Assume ptr to VDP name
72F3 8E8084     1471          $IF @RAMTOP .NE. 0 THEN But if ERAM save name in tab1
72F6 7311
72F8 864A       1472            CLR  @FAC                                         1
72FA 0F72       1473            XML  MEMCHK           FAC already has name length.1
72FC 7766       1474            BS   ERRMEM           * MEMORY FULL               1
72FE A5404A     1475            DSUB @FAC,@FREPTR     Get pointer to put name in. 1
7301 BD5840     1476            DST  @FREPTR,@EEE1    Re-do pointer to name.      1
7304 9158       1477            DINC @EEE1            Correct for one off.        1
7306 BD564A     1478            DST  @FAC,@FFF1       Set for XML GVWITE.         1
7309 BD542C     1479            DST  @PGMPTR,@DDD1    Set for XML GVWITE.         1
730C 0F8B       1480            XML  GVWITE           Move @FFF1 bytes from ERAM at 1
                1481  *                               DDD1 to VDP at EEE1.
                1482  *
                1483  *       Start building the subprogram table.
                1484  *
730E BD4E58     1485            DST  @EEE1,@FAC4      Put pointer in VRAM to name.  1
                1486          $END IF
7311 BF4A00     1487          DST  14,@FAC            Minimum table size for subprog.
7314 0E
7315 0F72       1488          XML  MEMCHK             Make sure enough room there.
7317 7766       1489          BS   ERRMEM             * MEMORY FULL
7319 864A       1490          CLR  @FAC               Prepare for name length.
731B BC4B42     1491          ST   @CHAT,@FAC1        Get the name length.
```

```
731E BD4C3A  1492          DST  @SUBTAB,@FAC2     Save subprogram table address.
7321 8750    1493          DCLR @FAC6             Mark end of argument list.
             1494  *
             1495  *       @FAC=name length       @FAC2= Subprogram table link.
             1496  *       @FAC4=pointer to name @FAC6=argument list=00
             1497  *       @FAC8=@PGMPTR          @FAC10=@EXTRAM
             1498  *       @FAC12=symbol table=00
             1499  *
7323 A12C4A  1500          DADD @FAC,@PGMPTR      Skip the name to look ahead.
7326 350004  1501          MOVE 4 FROM @PGMPTR TO @FAC8 Copy PGMPTR and EXTRAM.
7329 522C
732B 8756    1502          DCLR @FAC12            Assume subprog has no symbol ta
732D A74000  1503          DSUB 14,@FREPTR        Reset free pointer.
7330 0E
7331 BD3A40  1504          DST  @FREPTR,@SUBTAB   Copy.
7334 913A    1505          DINC @SUBTAB           Set new subtable pointer.
7336 35000E  1506          MOVE 14 FROM @FAC TO RAM(@SUBTAB) Put the table in!!
7339 B03A4A
             1507  *
             1508  *       Start fixing up subprogram's symbol table.
             1509  *
733C BDA39A  1510          DST  @SUBTAB,RAM(SSTEMP) Copy address of subtable.
733F 3A
7340 A3A39A  1511          DADD 6,RAM(SSTEMP)     Point to argument list.
7343 0006
7345 BDA39C  1512          DST  RAM(SSTEMP),RAM(SSTMP2) Duplicate for later use.
7348 A39A
             1513
734A 0F79    1514          XML  PGMCHR            Get next token
734C 06725D  1515          CALL CHKEND            Check if end of statement.
734F 73CE    1516          BS   SCAN90            Yes. Get out here quick.
             1517  *
             1518  *                              Start looking at arguments.
7351 067252  1519          CALL SYNCHK            Check for left par.
7354 B7      1520          DATA LPAR$
             1521  *
             1522  SCAN86
7355 B61680  1523          SB   @XFLAG,ENTXFL     Flag for ENTXFL.
7358 067408  1524          CALL ENTERX            Enter next parmeter.
735B BF4A00  1525          DST  2,@FAC            Get room for ptr. in sub block.
735E 02
735F 0F72    1526          XML  MEMCHK            See if we had space for 2 bytes
7361 7766    1527          BS   ERRMEM            * MEMORY FULL
7363 BD4AA3  1528          DST  RAM(SSTEMP),@FAC  Copy current arg list pointer.
7366 9A
7367 A54A3E  1529          DSUB @SYMTAB,@FAC      Find length from table address.
             1530  *                              Move symbol table down two byte
             1531  *                              to make space for next argument
736A 344AEF  1532          MOVE @FAC FROM RAM(@SYMTAB) TO RAM(-2(SYMTAB))
736D FFFE3E
7370 B03E
7372 973A    1533          DDECT @SUBTAB          Adjust the subtable pointer.
7374 97A39C  1534          DDECT RAM(SSTMP2)      Adjust to point to first arg.
7377 BD00A3  1535          DST  RAM(SSTEMP),@VAR0
737A 9A
```

```
737B BDEFFF  1536          DST  @SYMTAB,RAM(-2(VARO))  Put pointer in subtable.
737E FE003E
7381 BD4A3E  1537          DST  @SYMTAB,@FAC      Copy symbol table address.
7384 974A    1538          DDECT @FAC             Pointing to real s.t. addres.
             1539  SCAN88
7386 BD4CE0  1540          DST  RAM(4(FAC)),@FAC2  Copy pointer to symbol name.
7389 044A
738B 934C    1541          DDEC  @FAC2
738D C54C3A  1542          $IF @FAC2 .NOT. .DH. @SUBTAB THEN  If name moved als1
7390 7396
7392 97E004  1543            DDECT RAM(4(FAC))    correct for the movement.     1
7395 4A
             1544          $END IF
7396 8FE002  1545          $IF RAM(2(FAC)) .DNE. 0 THEN    If more symbol there1
7399 4A73A7
739C 97E002  1546            DDECT RAM(2(FAC))    Adjust the link address also. 1
739F 4A
73A0 BD4AE0  1547            DST   RAM(2(FAC)),@FAC Point to next s.t. address. 1
73A3 024A
73A5 5386    1548            BR    SCAN88         Check for more s.t. adjustment1
             1549          $END IF
73A7 BD4AA3  1550          DST  RAM(SSTMP2),@FAC Restore pointer to first arg.
73AA 9C
73AB D54AA3  1551          $WHILE @FAC .DNE. RAM(SSTEMP) Fix all pointers in ar1
73AE 9A73B8
73B1 97B04A  1552            DDECT RAM(@FAC)      Shift address by 2 bytes.     1
73B4 954A    1553            DINCT @FAC           Go to next argument pointer.  1
73B6 53AB    1554          $SEND WHILE
73B8 973E    1555          DDECT @SYMTAB          Restore s.t. pointer.
73BA 9740    1556          DDECT @FREPTR          Restore free pointer.
             1557  *
             1558  *       Done with building a subprogram table.
             1559  *
73BC D642B6  1560          $IF @CHAT .NE. RPAR$ THEN  Next char not ")"?        1
73BF 73C7
73C1 067252  1561            CALL SYNCHK          Must be ",".                  1
73C4 B3      1562            DATA COMMA$                                        1
73C5 5355    1563            BR   SCAN86          Go get more argument.         1
             1564          $END IF
73C7 0F79    1565          XML  PGMCHR            Finished...
73C9 06725D  1566          CALL CHKEND            Check if end of statement.
73CC 5259    1567          BR   ERRSYN            If not, error.
             1568  SCAN90
73CE B216DF  1569          RB   @XFLAG,SAFLG      Finished scanning sub arguments
73D1 A3A39A  1570          DADD 6,RAM(SSTEMP)     Pnt to loc of ptr in subtab
73D4 0006
73D6 5242    1571          BR   CONSCN            Start scanning subprogram.
```

```
                1573  ********************************************************
                1574  *              SUBNDS and SUBXTS                       *
                1575  *      This routine scans SUBEND and SUBEXIT statemt.  *
                1576  ********************************************************
                1577  *
                1578  SUBNDS
73D8 06724A     1579          CALL IMPILL             Can't be imperative.
73DB DA1608     1580          $IF .BIT(SUBFLG) @XFLAG .EQ. 0 GOTO ERRSNS
73DE 7772
                1581  *       * MUST BE IN SUBPROGRAM message above.
73E0 D642A8     1582          $IF @CHAT .EQ. SUBND$ THEN                        1
73E3 53FE
73E5 8E1757     1583            $IF @FORNET .NE. 0 GOTO ERRFNN Check FOR-NEXT nest1
73E8 6E
73E9 DA1601     1584            $IF .BIT(REMODE) @XFLAG .EQ. 1 GOTO ERRSNS      1
73EC 5772
73EE DA1640     1585            $IF .BIT(IFFLAG) @XFLAG .EQ. 1 GOTO ERRSYN      1
73F1 5259
73F3 BD00A3     1586            DST  RAM(SSTEMP),@VAR0                          1
73F6 9A
73F7 BDB000     1587            DST  @SYMTAB,RAM(@VAR0)                         1
73FA 3E
73FB B61601     1588            SB   @XFLAG,REMODE                              1
                1589          $END IF
73FE 51A7       1590          BR   SCAN22             Check for end of statement.
```

```
                    1592  *
                    1593  ***************************************************************
                    1594  *                  ENTER and ENTERX routines                  *
                    1595  *          These routines take care of entering a symbol      *
                    1596  *       into the symbol table.  If a symbol is encountered    *
                    1597  *       which is already in the table, the usage of the       *
                    1598  *       symbol is checked for consistency.                    *
                    1599  ***************************************************************
7400 067743         1600  ENTER  CALL PGMERR              Get next token - error if EOL
7403 D24200         1601  ENTERW $IF @CHAT .LT. 0 GOTO ERRSYN  If token - error
7406 5259
                    1602
7408 BE5949         1603  ENTERX ST   FAC-1,@FAC15        FOR INDIRECTION IN NAME SAVE
740B BD0C2C         1604         DST  @PGMPTR,@NMPTR      SAVE POINTER TO NAME
740E 930C           1605         DDEC @NMPTR              CORRECT FOR PGMCHR POST INCREM
                    1606  *
                    1607  *******Accumulate the name of the symbol
                    1608  *
7410 9059           1609  ENT01  INC  @FAC15              Count the character
7412 C65958         1610         $IF  @FAC15 .H. FAC14 GOTO ERRNTL
7415 775A
7417 BC9059         1611         ST   @CHAT,*FAC15        Save it
741A 42
741B 0F79           1612         XML  PGMCHR              Get the next one
741D CE4200         1613         $IF  @CHAT .GT. 0 GOTO ENT01  If not token or EOL
7420 7410
7422 BD6C2C         1614         DST  @PGMPTR,@ARG16      Save text pointer to put into
7425 936C           1615         DDEC @ARG16                 symbol table entry late
7427 D69059         1616         $IF  *FAC15 .EQ. DOLLAR THEN   String variable?     1
742A 245430
742D B61610         1617              SB  @XFLAG,STRFLG Set string flag               1
                    1618         $END IF
7430 A6594A         1619         SUB  FAC,@FAC15          Calc length of name
7433 9059           1620         INC  @FAC15              + Offset of 1
7435 D642B7         1621         $IF  @CHAT .EQ. LPAR$ GOTO ENT22   If complex
7438 7485
743A DA1680         1622         $IF  .BIT(ENTXFL) @XFLAG .EQ. 1 GOTO ENT08 If ENTERX
743D 5444
743F DA1604         1623         $IF  .BIT(FNCFLG) @XFLAG .EQ. 0 GOTO ERRSYN
7442 7259
                    1624  *           If not DEF then  DIM w/o  subscripted variable
```

```
                    1626  ********************************************************
                    1627  *              CODE FOR SIMPLE ENTRY INTO TABLE        *
                    1628  *       This includes all non-dimensioned varaibles as *
                    1629  *       well as phony entries for no-parameter functions.*
                    1630  *       ENT09 is the entry point for entering one of these *
                    1631  *       phony entries ENT10 is the code which checks for *
                    1632  *       consistent use of symbols within the user's program.*
                    1633  ********************************************************
7444 932C           1634  ENT08   DDEC @PGMPTR              Correct pointer overshoot
7446 BD0E2C         1635  ENT09   DST  @PGMPTR,@CHSAV       Save character pointer
7449 8680B0         1636          CLR  @STKMIN+1            Zero dimensions for simple
744C BE10B0         1637          ST   STKMIN+1,@TOPSTK     Save top of stack
744F DA4508         1638          $IF .BIT3 @FLAG .EQ. 1 GOTO ENT16 No search if funct
7452 5475
7454 0F7D           1639          XML  SCHSYM               Search symbol table
7456 5475           1640          BR   ENT16                Not found - must enter it
7458 912C           1641          DINC @PGMPTR              Correct pointer undershoot
                    1642  *
                    1643  *       Common code used by SIMPLE and COMPLEX
                    1644  *       When the symbol appears in the SYMBOL
                    1645  *       TABLE.  It verifies that the declarations
                    1646  *       are the same(# of parms/dims,string,function)
                    1647  *
                    1648  ENT10
745A DA1680         1649          $IF .BIT(ENTXFL) @XFLAG .EQ. 0 GOTO ERRMUV Redeclarin
745D 7762
                    1650  *      * IMPROPERLY USED NAME error message above.
745F DA1624         1651          CLOG >24,@XFLAG           If func or sub-arg
7462 5762           1652          BR   ERRMUV               Then redefining variable
                    1653  *                                    UDF
7464 BC00B0         1654          ST   RAM(@FAC),@VAR0      Fetch declaration
7467 4A
7468 B20007         1655          AND  >07,@VAR0            MASK FUNCTION AND STRING BITS
746B D40090         1656          $IF  @VAR0 .NE. *TOPSTK GOTO ERRMUV Not same # of dim
746E 105762
7471 B2166B         1657          AND  >6B,@XFLAG           Clear FNCFLG,STRFLG and ENTERX
7474 00             1658          RTN                       All OK-Type matches perfectly
                    1659
                    1660
7475 350010         1661  ENT16   MOVE 16 BYTES FROM @FAC TO @ARG  Save name
7478 5C4A
747A BF1400         1662          DST  14,@NMLEN            Need 14 bytes for a simple var
747D 0E
747E DA1614         1663          CLOG >14,@XFLAG           String or function?
7481 759F           1664          BS   ENT61                No-allocate & update table
7483 559B           1665          BR   ENT60                Yes-need 8 bytes for them
                    1666  *                                    Set count to 8 and update
```

```
                1668  ************************************************************
                1669  *                  CODE FOR A COMPLEX ENTER                *
                1670  ************************************************************
7485 BD0E2C     1671  ENT22  DST   @PGMPTR,@CHSAV     Save the line pointer
7488 BE72AF     1672          ST    STKMIN,@STACK     Initialize base of data stack
748B 350010     1673          MOVE  16 BYTES FROM @FAC TO @ARG   Save name
748E 5C4A
7490 DA1684     1674          CLOG  >84,@XFLAG        ENTERX or inside a DEF?
7493 54BE       1675          BR    ENT28             Yes-requires special scanning
                1676  *                             No - must be a dimension stmt
7495 0F79       1677  ENT24   XML   PGMCHR            Get next character
7497 067252     1678          CALL  SYNCHK            Must have numeric constant
749A C8         1679          DATA  NUMCO$
749B 0677C2     1680          CALL  CSINT             Convert dimension to integer
749E 777E       1681          BS    ERRBV             If got an error on conversion
74A0 8E4A54     1682          $IF   @FAC .EQ. 0 THEN If not BIG dim                  1
74A3 A9
74A4 C84B43     1683            $IF  @FAC1 .L. @BASE GOTO ERRBV   Dim<BASE         1
74A7 577E
                1684          $END IF
74A9 8C4B       1685          PUSH  @FAC1             Push this dimension
74AB 8C4A       1686          PUSH  @FAC              Both bytes
74AD C672BD     1687          $IF   @STACK .H. STKMAX GOTO ERRSYN If too many dims
74B0 7259
74B2 D642B3     1688          $IF   @CHAT .EQ. COMMA$ GOTO ENT24 If comma-more dims
74B5 7495
74B7 D642B6     1689          $IF   @CHAT .EQ. RPAR$ GOTO ENT40  Ok if end on rpar
74BA 7514
74BC 5259       1690          BR    ERRSYN            Didn't end on a rpar - err
                1691
                1692  *******Code for a non-DIM statement
                1693
74BE BE0001     1694  ENT28   ST    1,@VARO           Parenthesis level counter
                1695  *                               At first level
74C1 067743     1696  ENT29   CALL  PGMERR           Get next token-error if EOL
74C4 CE4200     1697          $IF @CHAT .GT. 0 THEN        If non-token - look mor
74C7 54D0
                1698  *                               During sub argument scanning
74C9 DA1620     1699            $IF .BIT(SAFLG) @XFLAG .EQ. 1 GOTO ERRBA not accep
74CC 577A
74CE 54C1       1700            BR ENT29              Get next token.              1
                1701          $END IF
74D0 D642B6     1702          $IF   @CHAT .EQ. RPAR$  GOTO ENT34
74D3 750C
74D5 DA1604     1703          $IF   .BIT(FNCFLG) @XFLAG .EQ. 1 GOTO ERRSYN
74D8 5259
74DA D642B3     1704          $IF   @CHAT .EQ. COMMA$ THEN                         1
74DD 54EF
74DF CE0001     1705            $IF   @VARO .GT. 1 GOTO ENT29 If not top-level comm
74E2 74C1
74E4 8C07       1706            PUSH @DFLTLM+1                                     1
74E6 8C06       1707            PUSH @DFLTLM            Push a default limit       1
74E8 CE72BD     1708            $IF   @STACK .LE. STKMAX GOTO ENT29 NOT too many di
74EB 54C1
74ED 5259       1709            BR    ERRSYN            Too many dims - so error   1
```

```
                1710  *       Jump always
                1711          $END IF
74EF DA1620     1712          $IF .BIT(SAFLG) @XFLAG .EQ. 1 GOTO ERRBA *BAD ARGUMEN
74F2 577A
74F4 D642C7     1713          $IF  @CHAT .EQ. STRIN$ THEN                          1
74F7 54FE
74F9 06774B     1714  ENT30     CALL SKPSTR                                        1
74FC 54C1       1715            BR   ENT29                                       1
                1716          $END IF
74FE D642C8     1717          $IF  @CHAT .EQ. NUMCO$ GOTO ENT30
7501 74F9
7503 D642B7     1718          $IF  @CHAT .EQ. LPAR$  THEN                          1
7506 550A
7508 9000       1719            INC  @VARO           Increase nesting level       1
                1720          $END IF
750A 54C1       1721          BR   ENT29             Not anything above. Get next c
                1722  *
750C 9200       1723  ENT34   DEC  @VARO             Decrease nesting level
750E 54C1       1724          BR   ENT29             Continue scan unless through
7510 8C07       1725          PUSH @DFLTLM+1         Push final default limit
7512 8C06       1726          PUSH @DFLTLM
```

```
                1728  ************************************************************
                1729  *    Calculate number of dims and search symbol table      *
                1730  ************************************************************
7514 BC0072     1731  ENT40  ST    @STACK,@VARO       Compute the # of dims
7517 A600AF     1732         SUB   STKMIN,@VARO
751A E60001     1733         SRL   @VARO,1            Divide by 2
751D 8C00       1734         PUSH  @VARO              Push the number of dims on top
751F BC1072     1735         ST    @STACK,@TOPSTK     Save stack top
7522 350010     1736         MOVE 16 BYTES FROM @ARG TO @FAC  Get name back
7525 4A5C
7527 0F7D       1737         XML   SCHSYM             Search symbol table for it
7529 5530       1738         BR    ENT44              Not found in table - ENTER it
752B BD2C0E     1739         DST   @CHSAV,@PGMPTR     Restore scan restart at '('
752E 545A       1740         BR    ENT10              And check for consistency
7530 DA1624     1741  ENT44  CLOG  >24,@XFLAG         If function or subprogram arg
7533 559B       1742         BR    ENT60                 then need 8 bytes
                1743
                1744  *      Calculate total number of array elements
                1745
7535 BC1072     1746         ST    @STACK,@TOPSTK     Save stack pointer
7538 9272       1747         DEC   @STACK             Skip # of dims
753A BC4A90     1748         POP   @FAC               Assume base=0
753D 7C
753E BC4B90     1749         POP   @FAC1
7541 7C
7542 914A       1750         DINC  @FAC
7544 8608       1751         CLR   @VARC              But correct if base=1
7546 BC0943     1752         ST    @BASE,@VARC+1      Handle 1st dim specially t
7549 A54A08     1753         DSUB  @VARC,@FAC         Avoid 1 multiply
754C BD144A     1754         DST   @FAC,@NMLEN        FAC gets # of elems in array
754F 05756C     1755         B     ENT53              Merge into loop
                1756  *
7552 BC4A90     1757  ENT50  POP   @FAC               Get next dimension
7555 7C
7556 BC4B90     1758         POP   @FAC1
7559 7C
755A 914A       1759         DINC  @FAC               Assume base=0
755C A54A08     1760         DSUB  @VARC,@FAC         But correct if base=1
755F BD0214     1761         DST   @NMLEN,@ACCUM
7562 A9024A     1762         DMUL  @FAC,@ACCUM        Accumulate size
7565 8F0257     1763         $IF   @ACCUM .DNE. 0 GOTO ERRMEM Out of memory
7568 66
7569 BD1404     1764         DST   @ACCUM+2,@NMLEN
                1765  *
756C D672AF     1766  ENT53  $IF @STACK .NE. STKMIN GOTO ENT50
756F 5552
7571 DA14E0     1767         CLOG  >E0,@NMLEN         If any of the top 3 bits set
7574 5766       1768         BR    ERRMEM                then * MEMORY FULL
7576 E31400     1769         DSLL  @NMLEN,1           Assume string - mem=elems*2
7579 01
757A DA1610     1770         $IF .BIT(STRFLG) @XFLAG .EQ. 0 THEN But if numeric  1
757D 5583
757F E31400     1771           DSLL @NMLEN,2          Memory = 4 * (2 * # of elems)1
7582 02
                1772         $END IF
```

```
7583 A31400  1773        DADD 6,@NMLEN           Need 6 more bytes for header
7586 06
7587 864A    1774        CLR  @FAC               For double
7589 BC4B90  1775        ST   *TOPSTK,@FAC1      Get # of dimensions
758C 10
758D E24B01  1776        SLL  @FAC1,1            Multiply by 2
7590 BD084A  1777        DST @FAC,@VARC          Save # of elements for later
7593 A1144A  1778        DADD @FAC,@NMLEN        Total # of bytes needed
7596 0C7766  1779        $IF .CARRY. GOTO ERRMEM
7599 559F    1780        BR   ENT61              Jump always
```

```
                    1782  *
759B BF1400         1783  ENT60  DST  8,@NMLEN          Funcs&simple strings need
759E 08
                    1784  *
                    1785  ********************************************************
                    1786  *       Check to see if enough memory in VDP RAM or ERAM  *
                    1787  *      Put symbol name in table if imperatively created   *
                    1788  *      or if executing an ERAM program.                   *
                    1789  ********************************************************
759F 8E8084         1790  ENT61  $IF  @RAMTOP .EQ. 0 THEN If not ERAM                1
75A2 55A8
75A4 8E4455         1791           $IF  @PRGFLG .NE. 0 GOTO ENT62 If program mode    1
75A7 C5
                    1792         $END IF
75A8 8E6B75         1793         $IF  @ARG15 .EQ. 0 GOTO ENT62 If 0-length(function)
75AB C5
                    1794  *      Move the name into the symbol table
75AC 8600           1795         CLR  @VARO                Re-do name and pointer
75AE BC016B         1796         ST   @ARG15,@VARO+1       Get length of name
75B1 BD4A00         1797         DST  @VARO,@FAC           Put length for MEMCHK
75B4 0F72           1798         XML  MEMCHK               Check enough memory for name
75B6 7766           1799         BS   ERRMEM               * MEMORY FULL
75B8 A54000         1800         DSUB @VARO,@FREPTR        Get space for the name
75BB BD0C40         1801         DST  @FREPTR,@NMPTR       Set new pointer to name
75BE 910C           1802         DINC @NMPTR               New pointer to name
75C0 3400B0         1803         MOVE @VARO FROM @ARG TO RAM(@NMPTR) Move the name
75C3 0C5C
                    1804  *
75C5 8651           1805  ENT62  CLR @FAC7                 Assume not simple numeric
75C7 8E8084         1806         $IF @RAMTOP .NE. 0 THEN  Set simple numeric variable1
75CA 763A
                    1807  *                                                  flag in FAC7
75CC BC7210         1808           ST @TOPSTK,@STACK      Get # of dimensions or parms 1
75CF BC5290         1809           POP  @FAC8                                          1
75D2 7C
75D3 DA1614         1810           CLOG >14,@XFLAG        If string or UDFfunction     1
75D6 55EE           1811           BR  ENT62A             Yes-don't set FAC7           1
                    1812  *                               No- if array ?
75D8 8E5255         1813           $IF @FAC8 .EQ. 0 THEN  Not array                    2
75DB EE
75DC 9051           1814             INC  @FAC7           Has to be a simple numeric va2
75DE BD0014         1815             DST @NMLEN,@VARO     Store NMLEN                  2
75E1 BF1400         1816             DST 8, @NMLEN        For later use - to locate 8  2
75E4 08
                    1817  *                               bytes in VDP for header & ptr
75E5 BD4A14         1818             DST @NMLEN,@FAC      Check enough memory in VDP   2
75E8 0F72           1819             XML MEMCHK                                        2
75EA 7766           1820             BS  ERRMEM           * MEMORY FULL                2
75EC 5625           1821             BR   ENT63           Check enough memory in ERAM  2
                    1822           $END IF
75EE 8650           1823  ENT62A   CLR @FAC6                                          1
75F0 DA1604         1824           $IF .BIT(FNCFLG) @XFLAG .EQ. 1 GOTO ENT63A        1
75F3 563A
                    1825  *                               UDFfunction
75F5 BC5052         1826           ST @FAC8,@FAC6                                    1
```

```
75F8 8E5076  1827            $IF @FAC6 .NE. 0 THEN  String or numeric array ?  2
75FB 3A
             1828  *         If numeric array goto ENT62B.
             1829  *         When checking subprogram arguments, numeric arra
             1830  *         is treated the same as string array case.
75FC DA1620  1831            $IF .BIT(SAFLG) @XFLAG .EQ. 1 GOTO ENT62C        2
75FF 5606
7601 DA1610  1832            $IF .BIT(STRFLG) @XFLAG .EQ. 0 GOTO ENT62B       2
7604 760A
7606 8650    1833  ENT62C     CLR  @FAC6     Clear FAC6 to indicate string arra2
7608 563A    1834             BR   ENT63A   So skip the next portion.          2
             1835  *
             1836  *       Numeric array case...
760A BD0014  1837  ENT62B     DST @NMLEN,@VARO   Store @NMLEN in temporary     2
760D BD1408  1838             DST @VARC,@NMLEN   # of bytes for dimension info2
7610 A31400  1839             DADD 8,@NMLEN      # of bytes need in the symbol2
7613 08
             1840  *                                   table entry in VDP RAM
7614 BD4A14  1841             DST @NMLEN,@FAC    Check enough memory in VDP RA2
7617 0F72    1842             XML MEMCHK                                       2
7619 7766    1843             BS  ERRMEM         * MEMORY FULL                 2
761B BD4A00  1844             DST @VARO,@FAC     Restore @NMLEN from VARO      2
761E A54A08  1845             DSUB @VARC,@FAC                                  2
7621 A74A00  1846             DSUB 6,@FAC        6 bytes header                2
7624 06
             1847  ENT63
7625 BD4C80  1848             DST  @RAMFRE,@FAC2 Get ERAM free pointer         2
7628 86
7629 A54C4A  1849             DSUB @FAC,@FAC2    Calculate lowest addr needed 2
762C 914C    1850             DINC @FAC2         One byte off here             2
762E CB4CA0  1851             $IF  @FAC2 .DL. CPUBAS GOTO ERRMEM  *MEMORY FULL2
7631 405766
7634 BD8086  1852             DST  @FAC2,@RAMFRE Set new ERAM freespace pointe2
7637 4C
7638 5641    1853             BR   ENT65                                       2
             1854           $END IF
             1855         $END IF
763A BD4A14  1856  ENT63A   DST  @NMLEN,@FAC      No -# of bytes needed
763D 0F72    1857           XML  MEMCHK          Check enough memory for entry
763F 7766    1858           BS   ERRMEM          * MEMORY FULL
             1859  *                             in VDP RAM
             1860  *       Now, construct the entry for the symbol table
             1861  *       in the FAC for ease and speed.  Then move it to
             1862  *       VDP RAM
             1863
7641 864A    1864  ENT65    CLR  @FAC            Clear the header byte
7643 DA1610  1865           $IF  .BIT(STRFLG) @XFLAG .EQ. 1 THEN  If string    1
7646 764B
7648 B64A80  1866             SB   @FAC,7        Set string bit in header      1
             1867           $END IF
764B DA1604  1868           $IF  .BIT(FNCFLG) @XFLAG .EQ. 1 THEN  If UDFunction 1
764E 7653
7650 B64A40  1869             SB   @FAC,6        Set function bit              1
             1870           $END IF
7653 BC7210  1871           ST @TOPSTK,@STACK    Get # of dimensions or parms
```

```
7656 BC5290  1872          POP @FAC8
7659 7C
765A 8E5276  1873          $IF  @FAC8 .NE. 0 THEN If array or parms                1
765D 69
765E B44A52  1874            OR   @FAC8,@FAC       Overlay # of dimensions       1
7661 DA1624  1875            CLOG >24,@XFLAG       If def or sub-arg             1
7664 5669    1876            BR   ENT67            Don't set opt flag            1
7666 B61602  1877            SB   @XFLAG,OPTFLG    Array so set OPT BASE flag    1
             1878          $END IF
7669 BC4B6B  1879  ENT67   ST   @ARG15,@FAC1       Save length of name
766C BD4C3E  1880          DST  @SYMTAB,@FAC2     Link to previous entry
766F BD4E0C  1881          DST  @NMPTR,@FAC4      Save pointer to the name
7672 A54014  1882          DSUB @NMLEN,@FREPTR    Set new table pointer
7675 9140    1883          DINC @FREPTR
             1884  *       Move the entry from the FAC to the symbol table
7677 350006  1885          MOVE 6 BYTES FROM @FAC TO RAM(@FREPTR)
767A B0404A
767D BD3E40  1886          DST  @FREPTR,@SYMTAB   Ptr to beginning of table
7680 DA4508  1887          $IF  .BIT3 @FLAG .EQ. 0 THEN If not run-function mod1
7683 568E
7685 DA1608  1888            $IF .BIT(SUBFLG) @XFLAG .NE. 1 THEN If not in subp2
7688 568E
768A BDA376  1889              DST  @SYMTAB,RAM(SYMBOL) Save ptr in VDP RAM     2
768D 3E
             1890            $END IF
             1891          $END IF
768E A34000  1892          DADD 6,@FREPTR
7691 06
7692 8E8084  1893          $IF @RAMTOP .NE. 0 THEN If ERAM exists then             1
7695 76C4
7697 D65101  1894            $IF @FAC7 .EQ. 1 THEN If simple numeric variable  2
769A 56AB
769C BD1400  1895              DST @VARO,@NMLEN    Restore NMLEN                 2
769F BDB040  1896              DST @RAMFRE,RAM(@FREPTR) Set the ptr into ERAM    2
76A2 8086
76A4 DA1620  1897              $IF .BIT(SAFLG) @XFLAG .EQ. 1 GOTO ENT69          2
76A7 56F6
76A9 56C2    1898            $SELSE                                              2
76AB DA1620  1899              $IF .BIT(SAFLG) @XFLAG .EQ. 1 GOTO ENT69          2
76AE 56F6
76B0 8E5076  1900              $IF @FAC6 .NE. 0 THEN If numeric array            3
76B3 C2
76B4 BD1400  1901                DST @VARO, @NMLEN   Restore NMLEN               3
76B7 BD0008  1902                DST @VARC,@VARO  Leave the space for dimension3
             1903  *                 info which is going to be filled in later
76BA A10040  1904                DADD @FREPTR,@VARO                              3
76BD BDB000  1905                DST  @RAMFRE,RAM(@VARO) Set ptr in ERAM         3
76C0 8086
             1906              $END IF
             1907            $END IF
76C2 56C9    1908          $SELSE                                                1
76C4 DA1620  1909            $IF .BIT(SAFLG) @XFLAG .EQ. 1 GOTO ENT69            1
76C7 56F6
             1910          $END IF
76C9 DA1604  1911          $IF  .BIT(FNCFLG) @XFLAG .EQ. 1 THEN If UDF-no d:     1
```

```
76CC 76D4
76CE BDB040  1912           DST  @ARG16,RAM(@FREPTR)  SAVE PTR TO '(' OR '='  1
76D1 6C
76D2 572B    1913           BR   ENT69B              Jump always              1
             1914         $END IF
             1915
             1916  *******Save the dimension information in the symbol table
76D4 CE72AF  1917         $IF  @STACK .LE. STKMIN GOTO ENT69 If non-array
76D7 56F6
76D9 BE72AF  1918         ST   STKMIN,@STACK       Get to bottom of stack
76DC 9072    1919  ENT68  INC  @STACK              Point at LSByte of next entry
76DE C87210  1920         $IF  @STACK .HE. @TOPSTK GOTO ENT69  If finished- out
76E1 76F6
76E3 BCE001  1921         ST   *STACK,RAM(1(FREPTR)) Put directly into table
76E6 409072
76E9 9072    1922         INC  @STACK              Point at MSByte of next entry
76EB BCB040  1923         ST   *STACK,RAM(@FREPTR)   Put directly into table
76EE 9072
76F0 9714    1924         DDECT @NMLEN             Used up 2 bytes in table
76F2 9540    1925         DINCT @FREPTR            Adjust pointer to unused bytes
76F4 56DC    1926         BR   ENT68               Get next dimension
             1927  *      Jump always
             1928
             1929  *******Now, zero the required amount of memory
76F6 8E8084  1930  ENT69  $IF @RAMTOP .NE. 0 THEN   If ERAM exists             1
76F9 771D
76FB DA1610  1931           $IF .BIT(STRFLG) @XFLAG .EQ. 1 GOTO ENT69D        1
76FE 571D
7700 D65101  1932           $IF @FAC7 .EQ. 1 THEN If simple numeric variable  2
7703 570B
7705 BF1400  1933             DST  8,@NMLEN        Zero 8 bytes of ERAM memory  2
7708 08
7709 5713    1934             BR   ENT69C                                       2
             1935           $END IF
770B 8E5077  1936           $IF @FAC6 .NE. 0 THEN   If numeric array          2
770E 1B
770F A71400  1937             DSUB 6, @NMLEN       Calc amount of ERAM to clear 2
7712 06
7713 0F84    1938  ENT69C     XML  IO              Special code to clear ERAM   2
7715 03      1939             DATA 3               Select the clear-ERAM code   2
7716 86      1940             DATA RAMFRE          Address of ERAM address      2
7717 14      1941             DATA NMLEN           Address of number of bytes   2
7718 938086  1942             DDEC @RAMFRE         Adjust ERAM free pointer     2
             1943           $END IF
771B 572B    1944         $SELSE                   VDP case                     1
771D A71400  1945  ENT69D   DSUB 7,@NMLEN          Now clear VDP RAM            1
7720 07
7721 86B040  1946           CLR  RAM(@FREPTR)      Clear 1st byte, then the rest1
7724 3414E0  1947           MOVE @NMLEN FROM RAM(@FREPTR) TO RAM(1(FREPTR))   1
7727 0140B0
772A 40
             1948         $END IF
772B BD403E  1949  ENT69B DST  @SYMTAB,@FREPTR   Set new free ptr @ the table
772E 9340    1950         DDEC @FREPTR           Now, set it at 1st free byte
7730 B216EB  1951         AND  >EB,@XFLAG        Clear STRFLG and FNCFLG
```

```
7733 DA1680  1952          $IF  .BIT(ENTXFL) @XFLAG .EQ. 1 THEN  If ENTERX call1
7736 7740
7738 DA1620  1953            $IF .BIT(SAFLG) @XFLAG .EQ. 0 THEN  If not scar   2
773B 5740
             1954  *                                      a subprogram argument then
773D BD2C0E  1955              DST  @CHSAV,@PGMPTR  Restore character pointer  2
             1956            $END IF
             1957          $END IF
7740 0F79    1958          XML  PGMCHR              Get next character
7742 00      1959          RTN
```

```
                1961  ********************************************************
                1962  *  THIS ROUTINE READS A CHARACTER AND WILL GIVE AN ERROR *
                1963  *        IF IT READS AN END OF LINE(PREMATURE END)       *
                1964  ********************************************************
7743 0F79       1965  PGMERR XML   PGMCHR
7745 06725D     1966         CALL  CHKEND
7748 7259       1967         BS    ERRSYN              Premature EOL
774A 00         1968         RTN
                1969  ********************************************************
                1970  *            THIS ROUTINE SKIPS QUOTED STRINGS         *
                1971  *          UNQUOTED STRINGS AND NUMERIC CONSTANTS      *
                1972  ********************************************************
774B 0F79       1973  SKPSTR XML   PGMCHR              Get the byte count
774D 8608       1974         CLR   @VARC               for double
774F BC0942     1975         ST    @CHAT,@VARC+1       Get count for add
7752 A12C08     1976         DADD  @VARC,@PGMPTR       Skip the string
7755 00         1977         RTN
                1978  *
                1979  ***
                1980  *       ERROR messages called in this file
                1981  ****
                1982  *
7756 066CFB     1983  ERRIBS CALL  ERR$$               * ILLEGAL AFTER SUBPROGRAM
7759 04         1984         DATA  4
                1985  *
775A 066CFB     1986  ERRNTL CALL  ERR$$               * NAME TOO LONG
775D 06         1987         DATA  6
                1988  *
775E 066CFB     1989  ERROBE CALL  ERR$$               * OPTION BASE ERROR
7761 08         1990         DATA  8
                1991  *
7762 066CFB     1992  ERRMUV CALL  ERR$$               * IMPROPERLY USED NAME
7765 09         1993         DATA  9
                1994  *
7766 066CFB     1995  ERRMEM CALL  ERR$$               * MEMORY FULL
7769 0B         1996         DATA  11
                1997  *
776A 066CFB     1998  ERRNWF CALL  ERR$$               * NEXT WITHOUT FOR
776D 0D         1999         DATA  13
                2000  *
776E 066CFB     2001  ERRFNN CALL  ERR$$               * FOR/NEXT NESTING
7771 0E         2002         DATA  14
                2003  *
7772 066CFB     2004  ERRSNS CALL  ERR$$               * MUST BE IN SUBPROGRAM
7775 0F         2005         DATA  15
                2006  *
7776 066CFB     2007  ERRMS  CALL  ERR$$               * MISSING SUBEND
7779 11         2008         DATA  17
                2009  *
777A 066CFB     2010  ERRBA  CALL  ERR$$               * BAD ARGUMENT
777D 1C         2011         DATA  28
                2012  *
777E 066CFB     2013  ERRBV  CALL  ERR$$               * BAD VALUE
7781 1E         2014         DATA  30
                2015  *
```

```
2016  *       Other error messages inside this porgram
2017  *
2018  * ERRSYN     * SYNTAX ERROR                 DATA 3
2019  * ERROLP     * ONLY LEGAL IN A PROGRAM      DATA 27
2020  * ERRPV      * PROTECTION VIOLATION         DATA 39
```

```
                    2022  ********************************************************
                    2023  *       Search and clean up stack and symbol table
                    2024  *       to not allow garbage to accumulate
                    2025  ********************************************************
7782 BD526E         2026  CLEAN  DST  @VSPTR,@FAC8       Get a temporary stack ptr
7785 C55224         2027  CLEAN1 $IF @FAC8 .DH. @STVSPT  While not end of stack      1
7788 57C1
778A BC58E0         2028           ST   RAM(2(FAC8)),@FAC14   Get stack ID byte      1
778D 0252
778F A65866         2029           SUB  >66,@FAC14       Check the range             1
7792 C65804         2030           $IF  @FAC14 .H. 4 THEN  If string, numeric,>70, >72
7795 579B
7797 0F78           2031              XML  VPOP          Throw it away(Must be on top!2
7799 5782           2032              BR   CLEAN                                     2
                    2033           $END IF
                    2034  *
779B 8A58           2035           CASE @FAC14                                      1
779D 57AA           2036              BR  CLEANG         GOSUB entry     >66         1
779F 57B0           2037              BR  CLEANF         FOR entry       >67         1
77A1 57BA           2038              BR  CLEANU         UDF entry       >68         1
77A3 57A7           2039              BR  CLEANE         ERROR entry     >69         1
77A5 57B4           2040              BR  CLEANS         SUB entry       >6A         1
                    2041  *
77A7 06A014         2042  CLEANE   CALL SQUISH          ERROR Entry - squish it out 1
77AA A75200         2043  CLEANG   DSUB 8,@FAC8         Go down 1 entry             1
77AD 08
77AE 5785           2044           BR CLEAN1            Go on to next entry         1
                    2045  *        Jump always
77B0 A75200         2046  CLEANF   DSUB  16,@FAC8       Keep it around but get below 1
77B3 10
77B4 A75200         2047  CLEANS   DSUB  16,@FAC8       16 bytes further down       1
77B7 10
77B8 5785           2048           BR CLEAN1            FOR or SUB entry            1
                    2049  *        Jump always
77BA 874E           2050  CLEANU   DCLR @FAC4           Cause delink to work right  1
77BC 06A010         2051           CALL DELINK          Delink the symbol table entry1
77BF 57AA           2052           BR   CLEANG                                      1
                    2053         $END IF
77C1 00             2054         RTN
```

```
                2056  ************************************************************
                2057  *       Subroutine to convert numeric constant to integer
                2058  ************************************************************
77C2 874A       2059  CSINT   DCLR  @FAC                  Start with clean FAC
77C4 0F79       2060  CSINT2  XML   PGMCHR
77C6 A64230     2061          S     >30,@CHAT             Subtract ASCII value for "0"
77C9 CA420A     2062          $IF @CHAT .L. 10 THEN       Valid numeric
77CC 77E6
77CE AB4A00     2063            DMUL 10,@FAC              Multiply previous result
77D1 0A
77D2 8F4A57     2064            $IF @FAC .DNE. 0 GOTO RETSET Overflow ?????
77D5 EA
77D6 BC4B42     2065            ST    @CHAT,@FAC1         Get result back down
77D9 A14A4C     2066            DADD @FAC2,@FAC           Add current digit
77DC 0C77EA     2067            $IF .CARRY. GOTO RETSET If >65535
77DF D24A00     2068            $IF @FAC .LT. 0 GOTO RETSET Integer > 32767
77E2 57EA
77E4 57C4       2069            BR    CSINT2              And loop until done
                2070          $END IF
77E6 A24230     2071          A     >30,@CHAT
77E9 00         2072          RTN                         Also used somewhere else
77EA D40000     2073  RETSET CEQ   @0,@0
77ED 01         2074          RTNC
```

```
                    2076          GROM 5
                    2077          ORG  >14E0
                    2078  ********************************************************
                    2079  *                    BASIC KEYWORD TABLE                *
                    2080  *             THE TOKEN IS ITS LEFT BINDING POWER.      *
                    2081  ********************************************************
                    2082  KEYTAB DATA #CHAR1,#CHAR2,#CHAR3,#CHAR4;
                    2083                #CHAR5,#CHAR6,#CHAR7,#CHAR8;
B4E0 B4F4B5         2084                #CHAR9,#CHARA
B4E3 15B52E
B4E6 B5C7B6
B4E9 27B676
B4EC B6C4B6
B4EF FDB72B
B4F2 B740
B4F4 2183           2085  CHAR1   DATA :!:,TREM$          ! (TAIL REMARK)
B4F6 23FD           2086          DATA :#:,NUMBE$         #
B4F8 26B8           2087          DATA :&:,CONC$          & (CONCATENATE)
B4FA 28B7           2088          DATA :(:,LPAR$          (
B4FC 29B6           2089          DATA :):,RPAR$          )
B4FE 2AC3           2090          DATA :*:,MULT$          *
B500 2BC1           2091          DATA :+:,PLUS$          +
B502 2CB3           2092          DATA :,:,COMMA$         ,
B504 2DC2           2093          DATA :-:,MINUS$         -
B506 2FC4           2094          DATA :/:,DIVI$          /
B508 3AB5           2095          DATA ::::,COLON$        :
B50A 3BB4           2096          DATA :;:,SEMIC$         ;
B50C 3CBF           2097          DATA :<:,LESS$          <
B50E 3DBE           2098          DATA :=:,EQUAL$         =
B510 3EC0           2099          DATA :>:,GREAT$         >
B512 5EC5           2100          DATA :^:,CIRCU$         ^
B514 FF             2101          DATA >FF
B515 3A3A82         2102  CHAR2   DATA ::::::,SSEP$       ::
B518 4154F0         2103          DATA :AT:,AT$
B51B 474F85         2104          DATA :GO:,GO$
B51E 494684         2105          DATA :IF:,IF$
B521 4F4E9B         2106          DATA :ON:,ON$
B524 4F52BA         2107          DATA :OR:,OR$
B527 5049DD         2108          DATA :PI:,PI$
B52A 544FB1         2109          DATA :TO:,TO$
B52D FF             2110          DATA >FF
B52E 414253         2111  CHAR3   DATA :ABS:,ABS$
B531 CB
B532 414C4C         2112          DATA :ALL:,ALL$
B535 EC
B536 414E44         2113          DATA :AND:,AND$
B539 BB
B53A 415343         2114          DATA :ASC:,ASC$
B53D DC
B53E 41544E         2115          DATA :ATN:,ATN$
B541 CC
B542 425945         2116          DATA :BYE:,>03
B545 03
B546 434F4E         2117          DATA :CON:,>01     ABBREVIATION OF 'CONTINUE'
B549 01
```

```
B54A 434F53  2118          DATA :COS:,COS$
B54D CD
B54E 444546  2119          DATA :DEF:,DEF$
B551 89
B552 44494D  2120          DATA :DIM:,DIM$
B555 8A
B556 454E44  2121          DATA :END:,END$
B559 8B
B55A 454F46  2122          DATA :EOF:,EOF$
B55D CA
B55E 455850  2123          DATA :EXP:,EXP$$
B561 CE
B562 464F52  2124          DATA :FOR:,FOR$
B565 8C
B566 494E54  2125          DATA :INT:,INT$
B569 CF
B56A 4C454E  2126          DATA :LEN:,LEN$
B56D D5
B56E 4C4554  2127          DATA :LET:,LET$
B571 8D
B572 4C4F47  2128          DATA :LOG:,LOG$
B575 D0
B576 4D4158  2129          DATA :MAX:,MAX$
B579 DF
B57A 4D494E  2130          DATA :MIN:,MIN$
B57D E0
B57E 4E4557  2131          DATA :NEW:,>00
B581 00
B582 4E4F54  2132          DATA :NOT:,NOT$
B585 BD
B586 4E554D  2133          DATA :NUM:,>04     ABBREVIATION OF 'NUMBER'
B589 04
B58A 4F4C44  2134          DATA :OLD:,>05
B58D 05
B58E 504F53  2135          DATA :POS:,POS$
B591 D9
B592 524543  2136          DATA :REC:,REC$
B595 DE
B596 52454D  2137          DATA :REM:,REM$
B599 9A
B59A 524553  2138          DATA :RES:,>06     ABBREVIATION OF 'RESEQUENCE'
B59D 06
B59E 524E44  2139          DATA :RND:,RND$
B5A1 D7
B5A2 52554E  2140          DATA :RUN:,RUN$
B5A5 A9
B5A6 53474E  2141          DATA :SGN:,SGN$$
B5A9 D1
B5AA 53494E  2142          DATA :SIN:,SIN$
B5AD D2
B5AE 535152  2143          DATA :SQR:,SQR$
B5B1 D3
B5B2 535542  2144          DATA :SUB:,SUB$
B5B5 A1
B5B6 544142  2145          DATA :TAB:,TAB$
```

```
B5B9 FC
B5BA 54414E  2146          DATA :TAN:,TAN$
B5BD D4
B5BE 56414C  2147          DATA :VAL:,VAL$
B5C1 DA
B5C2 584F52  2148          DATA :XOR:,XOR$
B5C5 BC
B5C6 FF      2149          DATA >FF
B5C7 424153  2150  CHAR4   DATA :BASE:,BASE$
B5CA 45F1
B5CC 424545  2151          DATA :BEEP:,BEEP$
B5CF 50EE
B5D1 43414C  2152          DATA :CALL:,CALL$
B5D4 4C9D
B5D6 434852  2153          DATA :CHR$:,CHR$$
B5D9 24D6
B5DB 444154  2154          DATA :DATA:,DATA$
B5DE 4193
B5E0 454C53  2155          DATA :ELSE:,ELSE$
B5E3 4581
B5E5 474F54  2156          DATA :GOTO:,GOTO$
B5E8 4F86
B5EA 4C4953  2157          DATA :LIST:,>02
B5ED 5402
B5EF 4E4558  2158          DATA :NEXT:,NEXT$
B5F2 5496
B5F4 4F5045  2159          DATA :OPEN:,OPEN$
B5F7 4E9F
B5F9 524541  2160          DATA :READ:,READ$
B5FC 4497
B5FE 525054  2161          DATA :RPT$:,RPT$$
B601 24E1
B603 534156  2162          DATA :SAVE:,>07
B606 4507
B608 534547  2163          DATA :SEG$:,SEG$$
B60B 24D8
B60D 53495A  2164          DATA :SIZE:,SIZE$
B610 45EB
B612 535445  2165          DATA :STEP:,STEP$
B615 50B2
B617 53544F  2166          DATA :STOP:,STOP$
B61A 5098
B61C 535452  2167          DATA :STR$:,STR$$
B61F 24DB
B621 544845  2168          DATA :THEN:,THEN$
B624 4EB0
B626 FF      2169          DATA >FF
B627 425245  2170  CHAR5   DATA :BREAK:,BREAK$
B62A 414B8E
B62D 434C4F  2171          DATA :CLOSE:,CLOSE$
B630 5345A0
B633 444947  2172          DATA :DIGIT:,DIGIT$
B636 4954E9
B639 455241  2173          DATA :ERASE:,ERASE$
B63C 5345EF
```

```
B63F 455252  2174          DATA :ERROR:,ERROR$
B642 4F52A5
B645 464958  2175          DATA :FIXED:,FIXED$
B648 4544FA
B64B 474F53  2176          DATA :GOSUB:,GOSUB$
B64E 554287
B651 494D41  2177          DATA :IMAGE:,IMAGE$
B654 4745A3
B657 494E50  2178          DATA :INPUT:,INPUT$
B65A 555492
B65D 4D4552  2179          DATA :MERGE:,>08
B660 474508
B663 505249  2180          DATA :PRINT:,PRINT$
B666 4E549C
B669 545241  2181          DATA :TRACE:,TRACE$
B66C 434590
B66F 555349  2182          DATA :USING:,USING$
B672 4E47ED
B675 FF      2183          DATA >FF
B676 414343  2184  CHAR6   DATA :ACCEPT:,ACCEP$
B679 455054
B67C A4
B67D 415050  2185          DATA :APPEND:,APPEN$
B680 454E44
B683 F9
B684 44454C  2186          DATA :DELETE:,DELET$
B687 455445
B68A 99
B68B 4C494E  2187          DATA :LINPUT:,LINPU$
B68E 505554
B691 AA
B692 4E554D  2188          DATA :NUMBER:,>04
B695 424552
B698 04
B699 4F5054  2189          DATA :OPTION:,OPTIO$
B69C 494F4E
B69F 9E
B6A0 4F5554  2190          DATA :OUTPUT:,OUTPU$
B6A3 505554
B6A6 F7
B6A7 524554  2191          DATA :RETURN:,RETUR$
B6AA 55524E
B6AD 88
B6AE 535542  2192          DATA :SUBEND:,SUBND$
B6B1 454E44
B6B4 A8
B6B5 55414C  2193          DATA :UALPHA:,UALPH$
B6B8 504841
B6BB EA
B6BC 555044  2194          DATA :UPDATE:,UPDAT$
B6BF 415445
B6C2 F8
B6C3 FF      2195          DATA >FF
B6C4 444953  2196  CHAR7   DATA :DISPLAY:,DISPL$
B6C7 504C41
```

```
B6CA 59A2
B6CC 4E554D  2197          DATA :NUMERIC:,NUMER$
B6CF 455249
B6D2 43E8
B6D4 524553  2198          DATA :RESTORE:,RESTO$
B6D7 544F52
B6DA 4594
B6DC 535542  2199          DATA :SUBEXIT:,SUBXT$
B6DF 455849
B6E2 54A7
B6E4 554E42  2200          DATA :UNBREAK:,UNBRE$
B6E7 524541
B6EA 4B8F
B6EC 554E54  2201          DATA :UNTRACE:,UNTRA$
B6EF 524143
B6F2 4591
B6F4 574152  2202          DATA :WARNING:,WARN$
B6F7 4E494E
B6FA 47A6
B6FC FF      2203          DATA >FF
B6FD 434F4E  2204  CHAR8   DATA :CONTINUE:,>01
B700 54494E
B703 554501
B706 494E54  2205          DATA :INTERNAL:,INTER$
B709 45524E
B70C 414CF5
B70F 52454C  2206          DATA :RELATIVE:,RELAT$
B712 415449
B715 5645F4
B718 56414C  2207          DATA :VALIDATE:,VALID$
B71B 494441
B71E 5445FE
B721 564152  2208          DATA :VARIABLE:,VARIA$
B724 494142
B727 4C45F3
B72A FF      2209          DATA >FF
B72B 504552  2210  CHAR9   DATA :PERMANENT:,PERMA$
B72E 4D414E
B731 454E54
B734 FB
B735 52414E  2211          DATA :RANDOMIZE:,RANDO$
B738 444F4D
B73B 495A45
B73E 95
B73F FF      2212          DATA >FF
B740 524553  2213  CHARA   DATA :RESEQUENCE:,>06
B743 455155
B746 454E43
B749 4506
B74B 534551  2214          DATA :SEQUENTIAL:,SEQUE$
B74E 55454E
B751 544941
B754 4CF6
B756 FF      2215          DATA >FF
```

```
                    2217  ************************************************************
                    2218  *
                    2219  *       ERRTAB - Error table  containing  all of the
                    2220  *            error messages, error numbers  and  the
                    2221  *            severity  code  for  each  error.   The
                    2222  *            error call  number is listed at the far
                    2223  *            right.  This is the data byte that must
                    2224  *            follow the  CALL ERR$$ or  CALL WARN$$.
                    2225  *            Messages  with  severity  of  zero  are
                    2226  *            system messages and not error messages.
                    2227  *
                    2228  *       Message,Error#,Severity                          Call #
                    2229  ************************************************************
                    2230  ERRTAB
B757 604000         2231          DATA #MSGFST,0,0        * READY *                    0
B75A 00
B75B 604800         2232          DATA #MSGBRK,0,0        * BREAKPOINT                 1
B75E 00
                    2233  *
B75F 60650A         2234          DATA #MSG10,10,1        * NUMERIC OVERFLOW           2
B762 01
B763 60760E         2235          DATA #MSG14,14,9        * SYNTAX ERROR               3
B766 09
B767 608310         2236          DATA #MSG16,16,9        * ILLEGAL AFTER SUBPROGRAM   4
B76A 09
B76B 609C11         2237          DATA #MSG17,17,9        * UNMATCHED QUOTES           5
B76E 09
B76F 60AD13         2238          DATA #MSG19,19,9        * NAME TOO LONG              6
B772 09
B773 60BB18         2239          DATA #MSG24,24,9        * STRING-NUM MISMATCH        7
B776 09
B777 60D219         2240          DATA #MSG25,25,9        * OPTION BASE ERROR          8
B77A 09
B77B 60E41C         2241          DATA #MSG28,28,9        * IMPROPERLY USED NAME       9
B77E 09
B77F 611024         2242          DATA #MSG36,36,9        * IMAGE ERROR               10
B782 09
B783 611C27         2243          DATA #MSG39,39,9        * MEMORY FULL               11
B786 09
B787 612828         2244          DATA #MSG40,40,9        * STACK OVERFLOW            12
B78A 09
B78B 61372B         2245          DATA #MSG43,43,9        * NEXT WITHOUT FOR          13
B78E 09
B78F 61482C         2246          DATA #MSG44,44,9        * FOR-NEXT NESTING          14
B792 09
B793 61592F         2247          DATA #MSG47,47,9        * MUST BE IN SUBPROGRAM     15
B796 09
B797 616F30         2248          DATA #MSG48,48,9        * RECURSIVE SUB PROG CALL   16
B79A 09
B79B 618931         2249          DATA #MSG49,49,9        * MISSING SUBEND            17
B79E 09
B79F 619833         2250          DATA #MSG51,51,9        * RETURN WITHOUT GOSUB      18
B7A2 09
B7A3 61AD36         2251          DATA #MSG54,54,1        * STRING TRUNCATED          19
B7A6 01
```

```
B7A7 61BE39  2252          DATA #MSG57,57,9       * BAD SUBSCRIPT                20
B7AA 09
B7AB 632438  2253          DATA #MSG56,56,9       * SPEECH STRING TOO LONG       21
B7AE 09
B7AF 61CC3C  2254          DATA #MSG60,60,9       * LINE NOT FOUND               22
B7B2 09
B7B3 61DB3D  2255          DATA #MSG61,61,9       * BAD LINE NUMBER              23
B7B6 09
B7B7 62C53E  2256          DATA #MSG62,62,9       * LINE TOO LONG                24
B7BA 09
B7BB 61EB43  2257          DATA #MSG67,67,9       * CAN'T CONTINUE               25
B7BE 09
B7BF 61FA45  2258          DATA #MSG69,69,9       * COMMAND ILLEGAL IN PROG      26
B7C2 09
B7C3 621546  2259          DATA #MSG70,70,9       * ONLY LEGAL IN A PROG         27
B7C6 09
B7C7 622D4A  2260          DATA #MSG74,74,9       * BAD ARGUMENT                 28
B7CA 09
B7CB 623A4E  2261          DATA #MSG78,78,1       * NO PROGRAM PRESENT          (29)
B7CE 01
B7CF 624D4F  2262          DATA #MSG79,79,9       * BAD VALUE                    30
B7D2 09
B7D3 625751  2263          DATA #MSG81,81,9       * INCORRECT ARGUMENT LIST      31
B7D6 09
B7D7 626F53  2264          DATA #MSG83,83,1       * INPUT ERROR                 (32)
B7DA 01
B7DB 627B54  2265          DATA #MSG84,84,9       * DATA ERROR                   33
B7DE 09
B7DF 629B6D  2266          DATA #MSG109,109,9     * FILE ERROR                   34
B7E2 09
B7E3 62A682  2267          DATA #MSG130,130,1     * I/O ERROR (WARNING)         (35)
B7E6 01
B7E7 62A682  2268          DATA #MSG130,130,9     * I/O ERROR                    36
B7EA 09
B7EB 62B087  2269          DATA #MSG135,135,9     * SUBPROGRAM NOT FOUND         37
B7EE 09
B7EF 61CC3C  2270          DATA #MSG60,60,1       * LINE NOT FOUND (WARNING)    (38)
B7F2 01
B7F3 628661  2271          DATA #MSG97,97,9       * PROTECTION VIOLATION         39
B7F6 09
B7F7 60F914  2272          DATA #MSG34,20,9       * UNRECOGNIZED CHARACTER       40
B7FA 09
             2273          END

ERRORS=     0

LENGTH= 4249 (>1099)

491 SYMBOLS USED
```

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

ABS$    00CB   352      2111
ACCEP$  00A4   311      2184
ACCUM   0002   106      1761 1762 1763 1764
ALL$    00EC   381      2112
AND$    00BB   334      2113
APPEN$  00F9   394      2185
ARG     005C   180       181  182  183  184  185  186  187  188  189  190
                         191  547  548  549  550  553  554  554  555  598
                         599  599  600  605  612  618  622  624  644  645
                         646  647  647  648  649  650  650  799  800  801
                         831  832  833  834  897  904  909  914 1003 1162
                        1164 1167 1661 1673 1736 1803
ARG1    005D   181       551  552
ARG11   0067   189      1160 1160 1163 1166 1176 1178 1186 1189
ARG15   006B   190      1793 1796 1879
ARG16   006C   191      1614 1615 1912
ARG2    005E   182       446  590  639  653  661  679  837  838  840  841
                         972  983  985  987  997 1006 1014 1035 1036 1117
                        1167 1169
ARG3    005F   183      1119 1165 1166
ARG4    0060   184       453  542  578  659  995  996  997 1000 1002 1006
                        1010
ARG5    0061   185       455  461  462  463  464  478  491  521  543  548
                         549  549  550  645  658  661  665  666  667  669
                         670  680 1159 1168 1179 1190
ARG6    0062   186       998 1005 1007
ARG7    0063   187       445  520  597
ARG8    0064   188       471  536  633  675  686
ASC     A00A    31       854  937
ASC$    00DC   369      2114
AT$     00F0   385      2103
ATN$    00CC   353      2115
BACK    0008   248       527
BASE    0043   156      1250 1298 1303 1683 1752
BASE$   00F1   386      1293 2150
BEEP$   00EE   383      2151
BREAK   0002   243       497
BREAK$  008E   289      2170
BTOP15  6CF8   727       499
BUFEND  038E   208       514
BUFLEV  0046   159
BUFSRT  038C   207       513
BYTE    000C   114       115
CALENT  723C  1369      1269
CALIST  000A   113      1217 1422 1424 1436 1439
CALL$   009D   304      2152
CALL20  7285  1411      1416
CALLS   726E  1402      1264
CB      0008     1
CCPADD  0008   112      1038 1203 1204 1205
CCPPTR  0006   109      1199 1206
CHAR    0004     1
CHAR1   B4F4  2085      2082
CHAR2   B515  2102      2082
CHAR3   B52E  2111      2082
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
CHAR4	B5C7	2150	2082
CHAR5	B627	2170	2082
CHAR6	B676	2184	2082
CHAR7	B6C4	2196	2082
CHAR8	B6FD	2204	2082
CHAR9	B72B	2210	2082
CHARA	B740	2213	2082
CHAT	0042	155	853 855 1043 1045 1047 1054 1055 1058 1059 1061 1062 1067 1068 1070 1078 1082 1083 1087 1088 1096 1100 1115 1117 1119 1131 1176 1197 1204 1207 1284 1299 1301 1342 1343 1344 1382 1385 1386 1391 1406 1409 1410 1413 1468 1469 1491 1560 1582 1601 1611 1613 1621 1688 1689 1697 1702 1704 1713 1717 1718 1975 2061 2062 2065 2071
CHKEND	725D	1385	408 1307 1340 1515 1566 1966
CHR$$	00D6	363	2153
CHRTAB	601C	37	825
CHRTN	000D	223	585
CHSAV	000E	116	1635 1671 1739 1955
CIRCU$	00C5	344	2100
CLEAN	7782	2026	725 789 2032
CLEAN1	7785	2027	2044 2048
CLEANE	77A7	2042	2039
CLEANF	77B0	2046	2037
CLEANG	77AA	2043	2036 2052
CLEANS	77B4	2047	2040
CLEANU	77BA	2050	2038
CLOSE$	00A0	307	2171
CLRLN	0007	247	570
CLSALL	8012	91	774
COLON$	00B5	328	1055 1056 2095
COMMA$	00B3	326	1058 1141 1284 1562 1688 1704 2092
CONC$	00B8	331	2087
CONSCN	7242	1371	1308 1362 1367 1442 1571
CONTIN	0081	265	816
COS$	00CD	354	2118
CPUBAS	A040	23	1851
CRNBUF	0820	212	761 762 1222 1328 1329 1334 1335 1338 1355 1356 1357 1358 1361
CRNEND	08BE	213	
CSINT	77C2	2059	1680
CSINT2	77C4	2060	2069
CURINC	000E	117	760
CURLIN	0014	121	760
CURSOR	007E	257	472 490
CVRTLN	7092	1158	1175 1185
DATA	0034	149	
DATA$	0093	294	2154
DDD1	0054	177	1479
DEF$	0089	284	2119
DELET$	0099	300	2186
DELINK	A010	33	2051
DFLTLM	0006	108	1218 1706 1707 1725 1726
DIGIT	0001	225	612
DIGIT$	00E9	378	2172

SYMBOL	VALUE	DEF	REFERENCE TABLE
DIM$	008A	285	2120
DISO	70C1	1185	410 839 842 901 923 975
DISP$1	70C4	1186	1191
DISPL$	00A2	309	2196
DISPLA	0009	1	
DIVI$	00C4	343	2094
DLETE	0003	244	541
DOLLAR	0024	255	1616
DOWN	000A	250	587
DSPCHR	70D9	1199	1046 1069 1080 1101 1104 1135 1177
DSPQUO	70D6	1196	1089 1108
DSRFLG	0017	131	1203
EDGECH	007F	258	463 544 558 572 640 648 666
EEE1	0058	179	815 987 1003 1005 1007 1476 1477 1485
ELSE$	0081	276	2155
END$	008B	286	2121
ENDSCR	02FE	204	
ENLN	0032	148	1000 1238
ENT01	7410	1609	1613
ENT08	7444	1634	1622
ENT09	7446	1635	412
ENT10	745A	1648	1740
ENT16	7475	1661	1638 1640
ENT22	7485	1671	1621
ENT24	7495	1677	1688
ENT28	74BE	1694	1675
ENT29	74C1	1696	1700 1705 1708 1715 1721 1724
ENT30	74F9	1714	1717
ENT34	750C	1723	1702
ENT40	7514	1731	1689
ENT44	7530	1741	1738
ENT50	7552	1757	1766
ENT53	756C	1766	1755
ENT60	759B	1783	1665 1742
ENT61	759F	1790	1664 1780
ENT62	75C5	1805	1791 1793
ENT62A	75EE	1823	1811
ENT62B	760A	1837	1832
ENT62C	7606	1833	1831
ENT63	7625	1847	1821
ENT63A	763A	1856	1824 1834
ENT65	7641	1864	1853
ENT67	7669	1879	1876
ENT68	76DC	1919	1926
ENT69	76F6	1930	1897 1899 1909 1917 1920
ENT69B	772B	1949	1913
ENT69C	7713	1938	1934
ENT69D	771D	1945	1931
ENTER	7400	1600	411 1283 1315
ENTERW	7403	1601	1322
ENTERX	7408	1603	1351 1370 1524
ENTXFL	0007	235	1319 1350 1369 1523 1622 1649 1952
EOF$	00CA	351	2122
EQUAL$	00BE	337	1327 1366 2098
ERASE$	00EF	384	2173

SYMBOL	VALUE	DEF	REFERENCE TABLE
ERPNT5	6D9E	829	718 892 945
ERPRNT	6D91	825	772 802 805 894 947
ERR$$	6CFB	759	414 1031 1376 1383 1983 1986 1989 1992 1995 1998 2001 2004 2007 2010 2013
ERR$$4	6CF1	725	776 783
ERR$$5	6CF4	726	807
ERR$$R	6D24	772	784
ERR$1	6D3B	782	780
ERRBA	777A	2010	1699 1712
ERRBV	777E	2013	1681 1683
ERRCOD	0022	140	703 759
ERRFNN	776E	2001	1272 1455 1583
ERRIBS	7756	1983	1276
ERRLN	038A	217	781 811
ERRMEM	7766	1995	1275 1474 1489 1527 1763 1768 1779 1799 1820 1843 1851 1858
ERRMS	7776	2007	1273 1457
ERRMUV	7762	1992	1649 1652 1656
ERRNTL	775A	1986	1406 1468 1610
ERRNWF	776A	1998	1271
ERROBE	775E	1989	1290 1302
ERROLP	724E	1376	1270
ERROR$	00A5	312	2174
ERRPV	6F88	1031	
ERRSNS	7772	2004	1580 1584
ERRSYN	7259	1383	1274 1281 1309 1374 1567 1585 1601 1623 1687 1690 1703 1709 1967
ERRTAB	B757	2230	707 767
EXEC	A004	30	1233
EXEC6D	A00E	32	501
EXP$	0076	193	704 705 706 707 708 764 765 766 767 768 853 855
EXP$$	00CE	355	2123
EXTRAM	002E	146	796 810 811 814 1035 1040 1238 1239
FAC	004A	161	162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 515 516 518 708 768 771 790 1056 1084 1096 1097 1105 1126 1139 1140 1141 1142 1143 1381 1382 1408 1430 1434 1435 1472 1475 1478 1487 1490 1500 1506 1525 1528 1529 1532 1537 1538 1540 1543 1545 1546 1547 1547 1550 1551 1552 1553 1603 1619 1654 1661 1673 1682 1686 1736 1748 1750 1753 1754 1757 1759 1760 1762 1774 1777 1778 1797 1818 1841 1844 1845 1846 1849 1856 1864 1866 1869 1874 1885 2059 2063 2064 2066 2068
FAC1	004B	162	794 1435 1469 1491 1683 1685 1749 1758 1775 1776 1879 2065
FAC10	0054	170	773 792 803 831 833 835 844 954 955 957 961 962 965 972 1066 1207
FAC12	0056	171	790 888 889 896 952 957 959 960 968 1502
FAC13	0057	172	890 944
FAC14	0058	173	1610 2028 2029 2030 2035
FAC15	0059	174	1409 1429 1603 1609 1610 1611 1616 1619 1620
FAC16	005A	175	953 955 956 956 957 1410 1415
FAC17	005B	176	1408 1413 1414

SYMBOL	VALUE	DEF	REFERENCE TABLE
FAC2	004C	163	791 1041 1065 1067 1070 1074 1099 1107 1109 1144 1436 1492 1540 1541 1542 1848 1849 1850 1851 1852 1880 2066
FAC3	004D	164	793 1044 1045 1047 1053 1064 1074 1146
FAC4	004E	165	797 1437 1470 1485 1881 2050
FAC5	004F	166	1137
FAC6	0050	167	619 625 796 1493 1823 1826 1827 1833 1900 1936
FAC7	0051	168	1805 1814 1894 1932
FAC8	0052	169	873 874 875 876 880 881 904 906 907 908 909 912 913 914 915 917 928 929 930 931 953 965 966 966 967 1131 1136 1430 1501 1809 1813 1826 1872 1873 1874 2026 2027 2028 2043 2046 2047
FFF1	0056	178	1478
FIGURE	0002	1	
FIXED$	00FA	395	2175
FLAG	0045	158	498 508 711 724 1030 1228 1320 1323 1638 1887
FNCFLG	0002	230	1623 1703 1824 1868 1911
FNDLNE	6F53	994	898 921
FOR$	008C	287	2124
FORNET	0017	132	1455 1583
FORW	0009	249	531
FREPTR	0040	154	1243 1244 1248 1251 1337 1339 1475 1476 1503 1504 1556 1800 1801 1882 1883 1885 1886 1892 1896 1904 1912 1921 1923 1925 1946 1947 1947 1949 1950
GETLN2	6F49	982	973 1013
GO$	0085	280	2104
GOSUB$	0087	282	2176
GOTO$	0086	281	2156
GREAT$	00C0	339	2099
GRSUB2	802C	38	812 984 1001
GRSUB3	802E	39	
GVWITE	008B	269	1480
IF$	0084	279	2105
IFFLAG	0006	234	1281 1374 1585
IMAGE$	00A3	310	2177
IMPIF	7245	1374	1288 1312 1454
IMPILL	724A	1375	1579
INPUT$	0092	293	2178
INSRT	0004	245	535
INT$	00CF	356	2125
INTER$	00F5	390	2205
IO	0084	268	1127 1938
KEYBD	0074	99	830
KEYTAB	B4E0	2082	1126
LEN$	00D5	362	2126
LESS$	00BF	338	2097
LET$	008D	288	2127
LINPU$	00AA	317	2187
LINUM	0012	119	1253 1254
LLI$12	6FA7	1042	1147
LLI$15	6FBA	1053	1085
LLI$16	6FC6	1056	1054
LLI$17	6FEB	1067	1056
LLIS$0	6FA3	1041	1141 1142

SYMBOL	VALUE	DEF	REFERENCE TABLE
LLIS$1	708E	1146	1107 1110 1121
LLIS$2	6FF7	1074	1059 1062
LLIS$3	7048	1114	1088
LLIS$6	7063	1131	1138
LLIS$7	708C	1144	1139 1140
LLIS$9	70AF	1170	1043 1082
LLIST	6F83	1029	406
LN$	00C9	350	1115
LNBUF	0036	150	
LOG$	00D0	357	2128
LPAR$	00B7	330	1520 1621 1718 2088
LSUBP	0048	160	1230
M$EDIT	6010	26	28 36 37 44 45 46 47 48 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 87 88 89
M$EXEC	A000	25	30 31 32 33 34
M$FLMG	8000	27	38 39 91 92
M$MSG	6038	28	
MAX$	00DF	372	2129
MEMCHK	0072	266	1473 1488 1526 1798 1819 1842 1857
MIN$	00E0	373	2130
MINUS$	00C2	341	2093
MSG10	6065	50	2234
MSG109	629B	83	2266
MSG130	62A6	84	792 835 2267 2268
MSG135	62B0	85	2269
MSG14	6076	51	2235
MSG16	6083	52	2236
MSG17	609C	53	2237
MSG19	60AD	54	2238
MSG24	60BB	55	2239
MSG25	60D2	56	2240
MSG28	60E4	57	2241
MSG34	60F9	58	2272
MSG36	6110	59	2242
MSG39	611C	60	803 2243
MSG40	6128	61	2244
MSG43	6137	62	2245
MSG44	6148	63	2246
MSG47	6159	64	2247
MSG48	616F	65	2248
MSG49	6189	66	2249
MSG51	6198	67	2250
MSG54	61AD	68	2251
MSG56	6324	89	2253
MSG57	61BE	69	2252
MSG60	61CC	70	2254 2270
MSG61	61DB	71	2255
MSG62	62C5	72	2256
MSG67	61EB	73	2257
MSG69	61FA	74	2258
MSG70	6215	75	2259
MSG74	622D	76	2260

SYMBOL	VALUE	DEF	REFERENCE TABLE
MSG78	623A	77	2261
MSG79	624D	78	2262
MSG81	6257	79	2263
MSG83	626F	80	2264
MSG84	627B	81	2265
MSG97	6286	82	2271
MSGBRK	6048	46	2232
MSGCF	6319	88	920 935 964
MSGCIS	630A	87	910
MSGERR	6038	44	806
MSGFST	6040	45	773 844 2231
MSGTA	6053	47	
MSGWRN	605C	48	713
MULT$	00C3	342	2090
MVUP	000B	251	586
NEXT$	0096	297	2158
NLNADD	02E2	203	509 713 715 806 827 828 849 899 900 910 920 922 935 936 951 952 964 968 974
NMLEN	0014	120	1662 1754 1761 1764 1767 1769 1771 1773 1778 1783 1815 1816 1818 1837 1838 1839 1841 1856 1882 1895 1901 1924 1933 1937 1941 1945 1947
NMPTR	000C	115	1604 1605 1801 1802 1803 1881
NOT$	00BD	336	1061 2132
NUM$	00C8	348	1087 1343
NUMBE$	00FD	398	1143 2086
NUMBR	0000	224	605
NUMCO$	00C8	349	1062 1295 1679 1717
NUMER$	00E8	377	2197
OFFSET	0060	256	257 258 561 564 573 591 657 827 959 1187
ON$	009B	302	2106
OPEN$	009F	306	2159
OPTFLG	0001	229	1290 1304 1877
OPTIO$	009E	305	2189
OR$	00BA	333	1059 2107
OUTL$1	70B3	1176	1180
OUTLN	70B0	1174	1037 1120
OUTPU$	00F7	392	2190
OUTREC	801A	92	1034 1200
PABPTR	0004	107	793 837 840
PERMA$	00FB	396	2210
PGMCHR	0079	264	1042 1081 1095 1098 1116 1118 1223 1305 1347 1403 1412 1441 1465 1514 1565 1612 1677 1958 1965 1973 2060
PGMERR	7743	1965	1289 1382 1600 1696
PGMPTR	002C	145	815 897 1040 1222 1407 1470 1479 1500 1501 1604 1614 1634 1635 1641 1671 1739 1955 1976
PI$	00DD	370	2108
PLUS$	00C1	340	2091
POP	0005	1	
POS$	00D9	366	2135
PRESCN	70F5	1216	404
PRGFLG	0044	157	499 507 709 722 780 804 846 889 896 934 949 950 1221 1375 1791
PRINT$	009C	303	2180
PRTNFN	00CE	200	628

SYMBOL	VALUE	DEF	REFERENCE TABLE
QUOTE	0022	254	1100 1197
RAM	0001	1	463 478 491 500 513 514 518 518 544 549
			549 554 554 558 561 564 572 573 591 598
			599 618 619 622 640 647 647 648 650 650
			658 666 713 726 762 781 793 806 811 827
			833 837 840 851 875 876 880 899 904 908
			909 910 913 914 917 920 928 930 935 951
			953 955 956 957 957 959 964 966 972 1035
			1040 1186 1187 1203 1204 1232 1317 1328 1328 1329
			1335 1337 1356 1356 1357 1361 1459 1506 1510 1511
			1512 1512 1528 1532 1532 1534 1535 1536 1540 1543
			1545 1546 1547 1550 1551 1552 1570 1586 1587 1654
			1803 1885 1889 1896 1905 1912 1921 1923 1946 1947
			1947 2028
RAMFLG	0089	196	1426
RAMFRE	0086	195	1246 1247 1848 1852 1896 1905 1940 1942
RAMTOP	0084	194	1242 1330 1471 1790 1806 1893 1930
RANDO$	0095	296	2211
RBACK	6A9E	460	527
READ$	0097	298	2160
READ$1	6AB4	473	485 507 587 630 663 672 681
READ$2	6AC6	489	483
READ$3	6B15	519	517
READ$4	6C3A	643	639
READ00	6AAF	470	416 522 561 565
READ05	6C6C	660	687
READL1	6A98	452	415
READL2	6A9B	454	579
READLN	6A8E	444	407
REC$	00DE	371	2136
RECALL	0006	246	506
RECBUF	08C0	214	518
RECLEN	0007	110	1199
RELAT$	00F4	389	2206
REM$	009A	301	1140 2137
REMODE	0000	228	1456 1464 1584 1588
RESTO$	0094	295	2198
RETNOS	6F48	979	722 724 950
RETSET	77EA	2073	2064 2067 2068
RETUR$	0088	283	2191
RFORW	6CAC	685	531
RKEY	0075	100	493 497 506 527 531 535 541 570 585 586
			587 601 602 603 606 607 608 609 614 615
			622 657 658
RND$	00D7	364	2139
ROM	000A	1	708 713 768 806 831 833 910 920 935 964
			1131
RPAR$	00B6	329	1325 1560 1689 1702 2089
RPT$$	00E1	374	2161
RSTK	0088	197	769
RTNSET	6F45	978	940 949
RUN$	00A9	316	2140
SAFLG	0005	233	1569 1699 1712 1831 1897 1899 1909 1953
SAVEVP	0388	218	726 1232
SCAN	711D	1238	1227

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

SCAN10  714E   1259     1225
SCAN20  71A4   1304     1300
SCAN22  71A7   1305     1590
SCAN25  71A9   1307     1285
SCAN35  71F9   1340     1348 1355 1359
SCAN40  7214   1350     1342
SCAN45  720D   1345     1343
SCAN50  7230   1361     1341
SCAN55  7236   1365     1318
SCAN67  72C1   1441     1425
SCAN86  7355   1522     1563
SCAN88  7386   1539     1548
SCAN90  73CE   1568     1516
SCANRT  726D   1392     1261 1375
SCHSYM  007D    261     1423 1639 1737
SCNDEF  71B0   1312     1262
SCNDIM  7171   1281     1263
SCNOPT  7180   1288     1265
SCNSMT  008F    270     1257 1259 1371
SCROLL  0083    267      511  673  712  714  826  848  858  902  911  924
                         938  963  976 1224
SEG$$   00D8    365     2163
SEMIC$  00B4    327     2096
SEQUE$  00F6    391     2214
SGN$$   00D1    358     2141
SIN$    00D2    359     2142
SIZE$   00EB    380     2164
SKPSTR  774B   1973     1345 1714
SMTSRT  001E    138      500  797
SPGMPT  0382    206      500
SPRITE  0006      1
SQR$    00D3    360     2143
SQUISH  A014     34     2042
SREF    001C    137
SSEP$   0082    277     1054 2102
SSTEMP  039A    210     1510 1511 1512 1528 1535 1551 1570 1586
SSTMP2  039C    211     1512 1534 1550
STACK   0072     97     1672 1687 1708 1731 1735 1746 1747 1766 1808 1871
                        1917 1918 1919 1920 1921 1922 1923
STEP$   00B2    325     2165
STKMAX  00BD    199     1687 1708
STKMIN  00AF    198     1636 1637 1672 1732 1766 1917 1918
STLN    0030    147      995 1243 1246 1253
STOP$   0098    299     2166
STR$$   00DB    368     2167
STREND  001A    136      801 1252
STRFLG  0004    232     1617 1770 1832 1865 1931
STRIN$  00C7    346     1088 1344 1713
STRSP   0018    135     1251 1252
STVSPT  0024    141      874  907 2027
SUB$    00A1    308     2144
SUBFLG  0003    231     1457 1580 1888
SUBND$  00A8    315     1582 2192
SUBNDS  73D8   1578     1267 1268
SUBS    72C5   1453     1266
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
SUBSTK	0073	98	769
SUBTAB	003A	152	1241 1492 1504 1505 1506 1510 1533 1542
SUBXT$	00A7	314	2199
SYMBOL	0376	205	1889
SYMTAB	003E	153	761 762 1240 1317 1328 1333 1337 1338 1355 1356 1357 1358 1361 1422 1424 1459 1462 1529 1532 1532 1536 1537 1555 1587 1880 1886 1889 1949
SYNCHK	7252	1381	1292 1294 1296 1324 1326 1365 1404 1466 1519 1561 1678
TAB$	00FC	397	2145
TABLE	000B	1	
TABSAV	0392	209	1459
TAN$	00D4	361	2146
THEN$	00B0	323	2168
TIMER	0079	101	479 484
TO$	00B1	324	2109
TONE2	0036	95	629 662 845
TOP	0003	1	
TOPL15	6012	36	727
TOPSTK	0010	118	1637 1656 1735 1746 1775 1808 1871 1920
TRAC05	6E29	887	875 878
TRAC09	6E52	903	914 925
TRAC10	6E56	905	928
TRAC12	6E75	912	917
TRAC50	6ED8	944	880
TRAC51	6EE9	950	930
TRAC55	6EF6	953	969
TRACBK	6DFF	872	716 782
TRACE$	0090	291	2181
TREM$	0083	278	1139 1386 2085
UALPH	0002	226	600
UALPH$	00EA	379	2193
UNBRE$	008F	290	2200
UNGST$	00C8	347	348 349 1405 1467
UNTRA$	0091	292	2201
UPDAT$	00F8	393	2194
USING$	00ED	382	1142 2182
VAL$	00DA	367	2147
VALI$1	6BFD	613	610
VALI$2	6C07	617	601 614
VALI$3	6C11	621	626
VALI$4	6C17	623	620
VALI$9	6C27	632	602 603 606 607 608 609 615 622
VALID$	00FE	399	2207
VALIDL	03B2	220	619
VALIDP	03B0	219	598 618
VAR0	0000	104	1329 1333 1334 1335 1407 1428 1437 1535 1536 1586 1587 1654 1655 1656 1694 1705 1719 1723 1731 1732 1733 1734 1795 1796 1797 1800 1803 1815 1837 1844 1895 1901 1902 1904 1905
VARA	002A	144	447 514 515 543 544 545 547 553 557 558 559 561 562 564 572 573 575 576 577 590 591 592 639 640 641 644 653 654 669 670 672 674 676
VARC	0008	111	1751 1752 1753 1760 1777 1838 1845 1902 1974 1975

SYMBOL	VALUE	DEF	REFERENCE TABLE
			1976
VARIA$	00F3	388	2208
VARV	0001	105	472 478 490 491
VARW	0020	139	447 455 461 509 512 513 516 518 521 576
			678 715 828 833 834 836 847 849 851 852
			900 922 936 974 1038 1039 1186 1187 1188 1201
VDP	000C	1	
VEL	0007	1	
VPOP	0078	263	2031
VPUSH	0077	262	809 1427 1431 1438
VRAMVS	0958	215	929
VSPTR	006E	192	726 799 873 877 1232 1428 1439 2026
WARN$	00A6	313	2202
WARN$$	6CB0	703	413
WRN$$3	6CCE	712	709
WRN$$5	6CE5	719	717
WRNPRT	0001	239	711
WRNSTP	0002	240	724
XFLAG	0016	122	1219 1281 1290 1304 1313 1319 1350 1369 1374 1456
			1457 1463 1464 1523 1569 1580 1584 1585 1588 1617
			1622 1623 1649 1651 1657 1663 1674 1699 1703 1712
			1741 1770 1810 1824 1831 1832 1865 1868 1875 1877
			1888 1897 1899 1909 1911 1931 1951 1952 1953
XOR$	00BC	335	2148
XPT	000E	1	719
YPT	000D	1	