```
                    1          TITLE FLMGR359
                    2  ********************************************************************
                    3  *                                                                  *
                    4  *       FFFFFF    L          M      M     GGGGG     RRRRRR       *
                    5  *       F         L          MM    MM    G     G    R      R      *
                    6  *       F         L          M M M M    G          R      R      *
                    7  *       FFFF      L          M  M  M    G    GG    RRRRRR       *
                    8  *       F         L          M      M    G     G    R    R       *
                    9  *       F         L          M      M    G     G    R     R      *
                   10  *       F         LLLLLLL    M      M     GGGGG     R      R     *
                   11  *                                                                  *
                   12  *                                                                  *
                   13  *                 33333      555555      99999                    *
                   14  *               3     3     5          9     9                   *
                   15  *                     3     5          9     9                   *
                   16  *                  333       5555       999999                   *
                   17  *                    3           5         9                      *
                   18  *               3    3     5     5        9                       *
                   19  *                33333       5555        9                       *
                   20  *                                                                  *
                   21  ********************************************************************
A040               22  CPUBAS EQU  >A040          Expansion RAM base
                   23  ********************************************************************
A000               24  M$EXEC EQU  >A000          Module EXEC branch table address
6010               25  M$EDIT EQU  >6010          Module EDIT branch table address
6A70               26  M$PSCN EQU  >6A70          Module PSCAN branch table address
                   27  ********************************************************************
                   28  *                XML instruction equates
                   29  *
0071               30  GETSTR EQU  >71            SYSTEM GET-STRING
0072               31  MEMCHK EQU  >72            MEMORY CHECK FOR PABs
0073               32  CNS    EQU  >73            CONVERT NUMBER TO STRING
0074               33  PARSE  EQU  >74            PARSE A VALUE
0075               34  CONT   EQU  >75            CONTINUE PARSING
0077               35  VPUSH  EQU  >77            PUSH TO V-STACK
0078               36  VPOP   EQU  >78            POP FROM V-STACK
0079               37  PGMCHR EQU  >79            GET PROGRAM CHARACTER
007A               38  SYM    EQU  >7A            FIND SYMBOL ENTRY
007B               39  SMB    EQU  >7B            ALSO FOR ARRAYS
007C               40  ASSGNV EQU  >7C            ASSIGN VARIABLE
007E               41  SPEED  EQU  >7E            SPEED-UP XML
0000               42  SYNCHK EQU  0              SYNCHK XML selector
0003               43  SEETWO EQU  3              SEETWO XML selector
007F               44  CRUNCH EQU  >7F            CRUNCH INPUT LINE
0080               45  CIF    EQU  >80            CONVERT INTEGER TO FLOATING POINT
0083               46  SCROLL EQU  >83            SCROLL THE SCREEN
0084               47  IO     EQU  >84            I/O UTILITIES
0088               48  MVDN   EQU  >88            MOVE DATA IN VDP/ERAM
                   49  FILSPC EQU  1                   Fill-space utility
0002               50  CSTRIN EQU  2                   Copy-string utility
008A               51  VGWITE EQU  >8A            WRITE DATA FROM VDP RAM TO ERAM
008B               52  GVWITE EQU  >8B            WRITE DATA FROM ERAM TO VDP RAM
008C               53  GREAD1 EQU  >8C            READ DATA FROM ERAM
0086               54  GWRITE EQU  >86            WRITE DATA TO ERAM
                   55
```

```
0010    56  CALDSR  EQU  >10          CALL DEVICE SERVICE ROUTINE
0012    57  CFI     EQU  >12          CONVERT TO TWO BYTE INTEGER
0034    58  TONE1   EQU  >34          ATTRACTIVE BEEP
            59
6012    60  TOPL15  EQU  M$EDIT+>02   RETURN FROM "OLD" OR "SAVE"
6014    61  INITPG  EQU  M$EDIT+>04   Initialize program space
601A    62  TOPL10  EQU  M$EDIT+>0A   Return to main and re-init.
6022    63  KILSYM  EQU  M$EDIT+>12   KILL SYMBOL TABLE ROUTINE
602E    64  AUTO1   EQU  M$EDIT+>1E   Get args. for LIST command
6053    65  MSGTA   EQU  M$EDIT+>43   Message "try again"
6030    66  TOPL02  EQU  M$EDIT+>20   RTN address for failing in AUTOLD
6032    67  EDITLN  EQU  M$EDIT+>22   Edit a ln into the program
6036    68  GWSUB   EQU  M$EDIT+>26   Write a few bytes of data to V/ERAM
            69
6A74    70  LLIST   EQU  M$PSCN+>04   List a line
6A76    71  READLN  EQU  M$PSCN+>06   Read a line from keyboard
6A82    72  WARN$$  EQU  M$PSCN+>12   WARNING MESSAGE ROUTINE
6A84    73  ERR$$   EQU  M$PSCN+>14   ERROR MESSAGE ROUTINE
6A86    74  READL1  EQU  M$PSCN+>16   Read a line from keyboard
            75
A002    76  LITSO5  EQU  M$EXEC+>02   Literal string common code
A006    77  LINE    EQU  M$EXEC+>06   GET LINE NUMBER ROUTINE
A008    78  DATAST  EQU  M$EXEC+>08   SEARCH FOR NEXT "DATA" STATEMENT
A012    79  CONVER  EQU  M$EXEC+>12   CONVERT WITH WARNING
A016    80  VALCD   EQU  M$EXEC+>16   CONVERT STRING TO NUMBER
A020    81  UBSUB   EQU  M$EXEC+>20   CLEAR ALL BKPT IN LN # TABLE
```

```
                    *       Temporary workspace variables in file-manager
                    *
0000                VARO    EQU  >00
0002                MNUM    EQU  >02           Usually a counter
0004                PABPTR  EQU  >04           Pointer to current PAB
0006                VARY2   EQU  >06           Use in MVDN only
0008                CCC1    EQU  >08
000C                BBB1    EQU  >0C
                    *       P A B   o f f s e t s
0002                FIL      EQU  2             File number within BASIC (0-255)
0003                OFS      EQU  3             Offset within record
0004                COD      EQU  4             I/O opcode
0005                FLG      EQU  5             I/O mode flag byte
0006                BUF      EQU  6             Start of data buffer
0008                LEN      EQU  8             Record length
0009                CNT      EQU  9             Character count
000A                RNM      EQU  10            Record number
000C                SCR      EQU  12            Screen base offset
000D                NLEN     EQU  13            Length of file descriptor
                    *
0006                CCPPTR  EQU  >06           Pointer to current column (base 1)
0007                RECLEN  EQU  >07           Length of current record (maximum)
0008                CCPADR  EQU  >08           Actual buffer address of column
0008                VARC    EQU  >08
000A                STADDR  EQU  >0A           Start address - usually for copying
000A                RAMPTR  EQU  >0A           Pointer for crunching INPUT stmt
000C                BYTE    EQU  >0C           String length for GETSTR
000E                VAR4    EQU  >0E
000E                CURINC  EQU  >0E
0010                VAR5    EQU  >10
0011                VAR6    EQU  >11
0014                CURLIN  EQU  >14           Starting line number for LIST
0017                FNUM    EQU  >17           Current file number for search
0017                DSRFLG  EQU  FNUM          Internal = 60, External = 0
0017                OPTFLG  EQU  FNUM          Option flag byte during OPEN
                    *       SUBDIVIDED INTO :
0000                MFLAG    EQU  0             Bit 0 - Mode flag (I/O mode)
0001                DFLAG    EQU  1             Bit 1 - DISPLAY/INTERNAL
0002                PFLAG    EQU  2             Bit 2 - PERMANENT
0003                SFLAG    EQU  3             Bit 3 - SEQUENTIAL/RELATIVE
0004                RFLAG    EQU  4             Bit 4 - FIXED/VARIABLE length
                    *
                    ********************************************************
                    *               Permanent workspace variables
0018                STRSP   EQU  >18           Start of string space pointer
001A                STREND  EQU  >1A           End of string space pointer
001C                SREF    EQU  >1C           Temporary string reference
001E                SMTSRT  EQU  >1E           Beginning of current statement
0020                VARW    EQU  >20           Usually for cursor position
0022                ERRCOD  EQU  >22           Return vector from 9900 code
0024                STVSPT  EQU  >24           Base of value stack
0026                RTNG    EQU  >26           Return address from 9900 code
0028                NUDTAB  EQU  >28           Start of NUD table
002A                VARA    EQU  >2A           End of input string
002C                PGMPTR  EQU  >2C           Program token pointer
```

```
002E    138  EXTRAM EQU  >2E      Line number table pointer
0030    139  STLN   EQU  >30      Line number table limit pointer
0032    140  ENLN   EQU  >32      "      "      "     "      "
0034    141  DATA   EQU  >34      Pointer to current DATA stmt
0036    142  LNBUF  EQU  >36      Line table pointer for READ
        143  *      EQU  >38
003A    144  SUBTAB EQU  >3A      Subprogram symbol table
003C    145  IOSTRT EQU  >3C      Start of I/O chain in memory
003E    146  SYMTAB EQU  >3E      Start of symbol table
0040    147  FREPTR EQU  >40      First free data-byte in memory
0042    148  CHAT   EQU  >42      Current program character
0043    149  BASE   EQU  >43
0044    150  PRGFLG EQU  >44      Program/Imperative flag
0045    151  FLAG   EQU  >45      General flag
0046    152  BUFLEV EQU  >46      Crunch-buffer destruction level
0048    153  LSUBP  EQU  >48      Last subprogram block on the stack
004A    154  FAC    EQU  >4A      Floating point accumulator #1
004B    155  FAC1   EQU  FAC+1
004C    156  FAC2   EQU  FAC+2
004D    157  FAC3   EQU  FAC+3
004E    158  FAC4   EQU  FAC+4
004F    159  FAC5   EQU  FAC+5
0050    160  FAC6   EQU  FAC+6
0051    161  FAC7   EQU  FAC+7
0052    162  FAC8   EQU  FAC+8
0053    163  FAC9   EQU  FAC+9
0054    164  FAC10  EQU  FAC+10
0055    165  FAC11  EQU  FAC+11
0056    166  FAC12  EQU  FAC+12
0057    167  FAC13  EQU  FAC+13
0058    168  FAC14  EQU  FAC+14
004C    169  AAA    EQU  FAC+2
004E    170  CCC    EQU  FAC+4
0050    171  BBB    EQU  FAC+6
0054    172  DDD1   EQU  FAC+10
0056    173  FFF1   EQU  FAC+12
0058    174  EEE1   EQU  FAC+14
005C    175  ARG    EQU  >5C      Floating point accumulator #2
005D    176  ARG1   EQU  ARG+1
005E    177  ARG2   EQU  ARG+2
0062    178  ARG6   EQU  ARG+6
0063    179  ARG7   EQU  ARG+7
0066    180  TEMP5  EQU  >66      TEMP FOR LITS05
006E    181  VSPTR  EQU  >6E      Value stack pointer
0073    182  SUBSTK EQU  >73      Subroutine stack pointer
0075    183  RKEY   EQU  >75      Key code of last scan
0075    184  SIGN$  EQU  >75
0076    185  EXP$   EQU  >76
0084    186  RAMTOP EQU  >84
0086    187  RAMFRE EQU  >86
0089    188  RAMFLG EQU  >89      Indication of ERAM existing
0088    189  RSTK   EQU  >88      STARTS AT >8A
00CE    190  PRTNFN EQU  >CE      SOUND - previous tone finished
```

```
                    192  *
                    193  *              B A S I C   T O K E N   T A B L E
                    194  *
                    195  *       EQU  >80          SPARE
0081                196  ELSE$    EQU  >81          "ELSE"
0082                197  SSEP$    EQU  >82          "::"
0083                198  TREM$    EQU  >83          "!"
0084                199  IF$      EQU  >84          "IF"
0085                200  GO$      EQU  >85          "GO"
0086                201  GOTO$    EQU  >86          "GOTO"
0087                202  GOSUB$   EQU  >87          "GOSUB"
0088                203  RETUR$   EQU  >88          "RETURN"
0089                204  DEF$     EQU  >89          "DEF"
008A                205  DIM$     EQU  >8A          "DIM"
008B                206  END$     EQU  >8B          "END"
008C                207  FOR$     EQU  >8C          "FOR"
008D                208  LET$     EQU  >8D          "LET"
008E                209  BREAK$   EQU  >8E          "BREAK"
008F                210  UNBRE$   EQU  >8F          "UNBREAK"
0090                211  TRACE$   EQU  >90          "TRACE"
0091                212  UNTRA$   EQU  >91          "UNTRACE"
0092                213  INPUT$   EQU  >92          "INPUT"
0093                214  DATA$    EQU  >93          "DATA"
0094                215  RESTO$   EQU  >94          "RESTORE"
0095                216  RANDO$   EQU  >95          "RANDOMIZE"
0096                217  NEXT$    EQU  >96          "NEXT"
0097                218  READ$    EQU  >97          "READ"
0098                219  STOP$    EQU  >98          "STOP"
0099                220  DELET$   EQU  >99          "DELETE"
009A                221  REM$     EQU  >9A          "REM"
009B                222  ON$      EQU  >9B          "ON"
009C                223  PRINT$   EQU  >9C          "PRINT"
009D                224  CALL$    EQU  >9D          "CALL"
009E                225  OPTIO$   EQU  >9E          "OPTION"
009F                226  OPEN$    EQU  >9F          "OPEN"
00A0                227  CLOSE$   EQU  >A0          "CLOSE"
00A1                228  SUB$     EQU  >A1          "SUB"
00A2                229  DISPL$   EQU  >A2          "DISPLAY"
00A3                230  IMAGE$   EQU  >A3          "IMAGE"
00A4                231  ACCEP$   EQU  >A4          "ACCEPT"
                    232  *        EQU  >A5          "REAL"          FUTURE
                    233  *        EQU  >A6          "INTEGER"       FUTURE
                    234  *        EQU  >A7          "SCRATCH"       FUTURE
                    235  *        EQU  >A8          "ACCEPT"
                    236  *        EQU  >A9          "IMAGE"
                    237  *        EQU  >AA          SPARES
                    238  *        EQU  >AB
                    239  *        EQU  >AC
                    240  *        EQU  >AD
                    241  *        EQU  >AE
                    242  *        EQU  >AF
00B0                243  THEN$    EQU  >B0          "THEN"
00B1                244  TO$      EQU  >B1          "TO"
00B2                245  STEP$    EQU  >B2          "STEP"
00B3                246  COMMA$   EQU  >B3          ","
```

```
00B4    247  SEMIC$ EQU  >B4          ";"
00B5    248  COLON$ EQU  >B5          ":"
00B6    249  RPAR$  EQU  >B6          ")"
00B7    250  LPAR$  EQU  >B7          "("
        251  *      EQU  >B8          "&"
        252  *      EQU  >B9          SPARE
        253  *      EQU  >BA          "OR"
        254  *      EQU  >BB          "AND"
        255  *      EQU  >BC          "XOR"
        256  *      EQU  >BD          "NOT"
00BE    257  EQUAL$ EQU  >BE          "="
00BF    258  LESS$  EQU  >BF          "<"
00C0    259  GREAT$ EQU  >C0          ">"
00C1    260  PLUS$  EQU  >C1          "+"
00C2    261  MINUS$ EQU  >C2          "-"
00C3    262  MULT$  EQU  >C3          "*"
00C4    263  DIVI$  EQU  >C4          "/"
00C5    264  CIRCU$ EQU  >C5          "^"
        265  *      EQU  >C6          SPARE
00C7    266  STRIN$ EQU  >C7          QUOTED STRING
00C8    267  UNQST$ EQU  >C8          UNQUOTED STRING
00C8    268  NUM$   EQU  UNQST$       ALSO NUMERICAL STRING
00C9    269  LN$    EQU  >C9          LINE NUMBER
        270  *      EQU  >CA          SPARE
00CB    271  ABS$   EQU  >CB          "ABS"
00CC    272  ATN$   EQU  >CC          "ATN"
00CD    273  COS$   EQU  >CD          "COS"
00CE    274  EXP$$  EQU  >CE          "EXP"
00CF    275  INT$   EQU  >CF          "INT"
00D0    276  LOG$   EQU  >D0          "LOG"
00D1    277  SGN$$  EQU  >D1          "SGN"
00D2    278  SIN$   EQU  >D2          "SIN"
00D3    279  SQR$   EQU  >D3          "SQR"
00D4    280  TAN$   EQU  >D4          "TAN"
00D5    281  LEN$   EQU  >D5          "LEN"
00D6    282  CHR$$  EQU  >D6          "CHR$"
00D7    283  RND$   EQU  >D7          "RND"
        284  *      EQU  >D8          "SEG$"
        285  *      EQU  >D9          "POS"
        286  *      EQU  >DA          "VAL"
        287  *      EQU  >DB          "STR$"
        288  *      EQU  >DC          "ASC"
        289  *      EQU  >DD          "PI"
00DE    290  REC$   EQU  >DE          "REC"
        291  ********************************************************************
00E8    292  NUMER$ EQU  >E8          "NUMERIC"
00E9    293  DIGIT$ EQU  >E9          "DIGIT"
00EA    294  UALPH$ EQU  >EA          "UALPHA"
00EB    295  SIZE$  EQU  >EB          "SIZE"
00EC    296  ALL$   EQU  >EC          "ALL"
00ED    297  USING$ EQU  >ED          "USING"
00EE    298  BEEP$  EQU  >EE          "BEEP"
00EF    299  ERASE$ EQU  >EF          "ERASE"
00F0    300  AT$    EQU  >F0          "AT"
00F1    301  BASE$  EQU  >F1          "BASE"
```

```
                302  *       EQU  >F2          "TEMPORARY"    FUTURE
00F3            303  VARIA$  EQU  >F3          "VARIABLE"     FUTURE
00F4            304  RELAT$  EQU  >F4          "RELATIVE"     FUTURE
00F5            305  INTER$  EQU  >F5          "INTERNAL"     FUTURE
00F6            306  SEQUE$  EQU  >F6          "SEQUENTIAL"
00F7            307  OUTPU$  EQU  >F7          "OUTPUT"
00F8            308  UPDAT$  EQU  >F8          "UPDATE"
00F9            309  APPEN$  EQU  >F9          "APPEND"
00FA            310  FIXED$  EQU  >FA          "FIXED"
00FB            311  PERMA$  EQU  >FB          "PERMANENT"
00FC            312  TAB$    EQU  >FC          "TAB"
00FD            313  NUMBE$  EQU  >FD          "#"
00FE            314  VALID$  EQU  >FE          VALIDATE
                315  *       EQU  >FF          ILLEGAL VALUE
```

```
                    317  *
                    318  *          A S C I I   D E F I N I T I O N S
                    319  *
0002                320  BREAK  EQU  >02          Keyboard break key
0020                321  SPACE  EQU  ' '         " " - SPACE
002D                322  MINUS  EQU  '-'         "-" - MINUS
                    323  *
                    324  *        P R O P E R T Y   D E F I N I T I O N S
                    325  *
000E                326  PABLEN EQU  14          PAB LENGTH
0820                327  CRNBUF EQU  >820        LOCATION OF CRUNCH BUFFER
0958                328  VRAMVS EQU  >958        V-STACK START ADDRESS
0065                329  STRVAL EQU  >65         VALUE IN ACCU IS STRING VALUE
001C                330  VWIDTH EQU  28          VIDEO SCREEN WIDTH
0060                331  OFFSET EQU  >60         OFFSET FOR VIDEO TABLES
02E0                332  SCRNBS EQU  >2E0        SCREEN BASE ADDRESS FOR LAST LINE
0020                333  BKGD   EQU  ' '         BACKGROUND CHARACTER
007F                334  EDGECH EQU  >1F+OFFSET  EDGE CHARACTER
0390                335  CSNTMP EQU  >0390       CSN TEMPORARY FOR FAC12
03BA                336  CSNTP1 EQU  >03BA       CSN TEMPORARY FOR FAC10
0394                337  AUTTMP EQU  >0394       AUTOLD TEMPORARY INSIDE ERR$2
039E                338  MRGPAB EQU  >039E       MERGED TEMPORARY FOR PAB PTR
03AA                339  INPUTP EQU  >03AA       INPUT TEMPORARY FOR PTR TO PROMPT
03AC                340  ACCVRW EQU  >03AC       ACCEPT TEMPORARY FOR @VARW,@VARA TC
03AE                341  ACCVRA EQU  >03AE       TRY AGAIN
03B0                342  VALIDP EQU  >03B0       PTR TO STANDARD STRING IN VALIDATE
03B2                343  VALIDL EQU  >03B2       LENGTH OF STANDARD STRING IN VALID
03B4                344  SIZCCP EQU  >03B4       SIZE TEMPORARY FOR CCPADR
03B6                345  SIZREC EQU  >03B6       SIZE TEMPORARY FOR RECLEN
                    346  *                       Also use as temporary in RELOCA
03B7                347  ACCTRY EQU  >03B7       ACCEPT "TRY AGAIN" FLAG
03B8                348  SIZXPT EQU  >03B8       Save XPT in SIZE when "try again"
03B9                349  SAPROT EQU  >03B9       PROTECTION flag in SAVE
03BC                350  OLDTOP EQU  >03BC       Old top of memory for RELOCA
03BC                351  CPTEMP EQU  >03BC       CCPPTR,RECLEN temp in INPUT
03BE                352  NEWTOP EQU  >03BE       New top of memory for RELOCA
                    353  *
                    354  *       I / O   C O D E   D E F I N I T I O N S
0000                355  C$OPEN EQU  0           OPEN code
0001                356  C$CLOS EQU  1           CLOSE code
0002                357  C$READ EQU  2           READ code
0003                358  C$WRIT EQU  3           WRITE code
0004                359  C$REST EQU  4           RESTORE/REWIND code
0005                360  C$LOAD EQU  5           LOAD code
0006                361  C$SAVE EQU  6           SAVE code
0007                362  C$DELE EQU  7           DELETE code
0008                363  C$SCR  EQU  8           SCRATCH code
0009                364  C$STAT EQU  9           STATUS code
                    365  *
                    366  *       C O N N E C T I O N   T O   R E S T
                    367  *
                    368          GROM 4
                    369          ORG  0
8000 4257           370          BR   DISPL1       "DISPLAY" ROUTINE
8002 417C           371          BR   DELET        "DELETE" ROUTINE
```

```
8004 4266     372        BR   PRINT     PRINT ROUTINE
8006 45F0     373        BR   INPUT     INPUT ROUTINE (NOT YET IMPLEMENTED)
8008 4032     374        BR   OPEN      OPEN ROUTINE
800A 4196     375        BR   CLOSE     CLOSE ROUTINE
800C 41FC     376        BR   RESTOR    RESTORE ROUTINE
800E 4B3A     377        BR   READ      READ ROUTINE
8010 5308     378        BR   GETDAT    GET DATA FROM ERAM/VDP (NOT USED)
8012 41F4     379        BR   CLSALL    CLOSE ALL OPEN FILES (SUBROUTINE)
8014 4DB8     380        BR   SAVE      PROGRAM SAVE ROUTINE
8016 4BC4     381        BR   OLD       PROGRAM LOAD ROUTINE
8018 5068     382        BR   LIST      LIST routine
801A 56A5     383        BR   OUTREC    Output record routine
801C 51DE     384        BR   EOF       End of file routine
801E 496B     385        BR   ACCEPT    "ACCEPT" routine
8020 4B96     386        BR   SRDATA    Search "DATA$" routine
8022 51CA     387        BR   SUBREC    RECORD routine
8024 55AD     388        BR   CHKEND    Check EOS
8026 4BCA     389        BR   OLD1      A subroutine for LOAD
8028 5002     390        BR   MERGE     Merge a program
802A 5179     391        BR   GRMLST    List a line out of the ERAM
802C 51A4     392        BR   GRSUB2    Read 2 bytes of data from ERAM/VDP
802E 51BC     393        BR   GRSUB3    Read 2 bytes of data from ERAM/VDP
              394   *                   with resetting possible breakpt
8030 4841     395        BR   LINPUT         LINPUT statement
```

```
                    397  ********************************************************
                    398  *              " O P E N "   STATEMENT HANDLER          *
                    399  *                                                       *
                    400  *      Handle the BASIC OPEN statement.  A legal syntax can*
                    401  *      only be something like                           *
                    402  *                                                       *
                    403  *           OPEN #{exp}:{string-exp}[,{open-option}]*   *
                    404  *                                                       *
                    405  *      in which {open-option} is any of the following   *
                    406  *                                                       *
                    407  *           DISPLAY, INPUT, VARIABLE, RELATIVE, INTERNAL*
                    408  *           SEQUENTIAL, OUTPUT, UPDATE, APPEND, FIXED or*
                    409  *           PERMANENT                                   *
                    410  *                                                       *
                    411  *      Each keyword can only be used once, which is being*
                    412  *      checked with an OPTFLG-bit.  For each specific   *
                    413  *      option please refer to the related routine.      *
                    414  *                                                       *
                    415  *      Scanning stops as soon as no next field starting *
                    416  *      with a comma can be found.                       *
                    417  *                                                       *
                    418  *      NOTE : After the actual DSR OPEN has been performed,*
                    419  *             the length of the record, whether VARIABLE or*
                    420  *             FIXED, has to be non-zero.  A zero length *
                    421  *             will cause an INCORRECT STATEMENT error.  *
                    422  ********************************************************
                    423  OPEN
8032 069355         424    CALL CHKFN              See if we specified any file
8035 77D7           425    BS   ERRFE              Definitely not...no # or #0 or
8037 069371         426    CALL CHKCON             Check and search given filenumber
803A 77D7           427    BS   ERRFE              *** FILE NUMBER EXISTS ***
                    428    $
                    429    $    *** ERROR IF NOT STOPPED ON COLON
                    430    $
803C 0F7E           431    XML  SPEED              Must be at a
803E 00             432    DATA SYNCHK               colon or else
803F B5             433    DATA COLON$                  its an error
8040 06965A         434    CALL PARFN              Parse filename and create PAB
8043 932C           435    DDEC @PGMPTR            Backup pgm pointer for next token
                    436  OPTION
8045 0F79           437    XML  PGMCHR             Get next program character
                    438    $
                    439    $    Next field should start with a comma
                    440    $
                    441  OPTI$0
8047 D642B3         442    $IF @CHAT .NE. COMMA$ GOTO CHECK
804A 410D
                    443    $
                    444    $    Enter HERE after comma exit in "SEQUENTIAL"
                    445    $
                    446  OPTI$1
804C 0F79           447    XML  PGMCHR             Next token please.....
                    448    $
                    449    $    Treat DISPLAY and INPUT as special cases
                    450    $
```

```
804E D642A2   451     $IF @CHAT .EQ. DISPL$ GOTO OPT$6
8051 60EA
8053 D64292   452     $IF @CHAT .EQ. INPUT$ GOTO OPT$7
8056 60F4
8058 A642F3   453     SUB  VARIA$,@CHAT       Reduce keyword offset to 0
805B CA4209   454     $IF @CHAT .HE. 9 GOTO OPERR Keyword too high
805E 6106
8060 8A42     455     CASE @CHAT              JUST IN CASE
8062 40BE     456       BR   OPT$01           Option VARIABLE
8064 407C     457       BR   OPT$02                  RELATIVE
8066 40E5     458       BR   OPT$03                  INTERNAL
8068 4081     459       BR   OPT$1                   SEQUENTIAL
806A 40A8     460       BR   OPT$2                   OUTPUT
806C 40AD     461       BR   OPT$3                   UPDATE
806E 40B7     462       BR   OPT$4                   APPEND
8070 40C3     463       BR   OPT$5                   FIXED
              464   *   BR   OPT$0                   PERMANENT
              465     $
              466     $          C A S E   0   -   " P E R M A N E N T "
              467     $
              468     $     Only check for multiple usage.  Since PERMANENT
              469     $     is the default, we might as well ignore it.....
              470     $
              471   OPT$0
8072 DA1704   472     $IF .BIT(PFLAG) @OPTFLG .NE. 0 GOTO OPERR
8075 4106
8077 B61704   473     SB   @OPTFLG,PFLAG      Not used....use now
807A 4045     474     BR   OPTION             Treat as simple default
              475     $
              476     $          C A S E   2   -   " R E L A T I V E "
              477     $
              478     $     Select relative record file in PAB and fall through
              479     $     in SEQUENTIAL code for multiple usage check.  Also
              480     $     handle initial file-size there.
              481     $
              482   OPT$02
807C B6E005   483     SB   RAM(FLG(PABPTR)),0     Indicate RELATIVE RECORD
807F 0401
              484     $
              485     $          C A S E   4   -   " S E Q U E N T I A L "
              486     $
              487     $     Check for multiple usage.  Remainder of syntax
              488     $     demands that we have something like
              489     $
              490     $          [{numeric expression}],.......
              491     $
              492     $     In case only a comma is found, we use the default.
              493     $     Everything else has to be evaluated as a numeric
              494     $     expression, convertable to a 16-bit integer value.
              495     $
              496   OPT$1
8081 DA1708   497     $IF .BIT(SFLAG) @OPTFLG .NE. 0 GOTO OPERR
8084 4106
8086 B61708   498     SB   @OPTFLG,SFLAG      First time usage -> o.k.
8089 0F79     499     XML  PGMCHR             Check next token for default
```

```
                    500     $
                    501     $     Comma means default has been used
                    502     $
808B D642B3         503     $IF @CHAT .EQ. COMMA$ GOTO OPTI$1
808E 604C
8090 0695AD         504     CALL CHKEND             Check for end of statement
8093 610D           505     BS   CHECK
8095 06809F         506     CALL CHKPAR             Perform combined checking & parsing
8098 BDE00A         507     DST  @FAC,RAM(RNM(PABPTR))  Non-zero result
809B 044A
809D 4047           508     BR   OPTI$0             Scan other options
                    509     $
                    510     $     Parse and check a numeric argument in here....
                    511     $
                    512   CHKPAR
809F 0F74           513     XML  PARSE              If not...parse up to comma
80A1 B3             514     DATA COMMA$
80A2 06935C         515     CALL CHKCNV             Check and convert to integer
80A5 6106           516     BS   OPERR             Oops....someone made a mistake here
80A7 00             517     RTN                     Return to caller
                    518     $
                    519     $          C A S E   5   -   " O U T P U T "
                    520     $
                    521     $     Select mode code "01" and check for multiple
                    522     $     usage.  Use MFLAG bit in OPTFLG for checking.
                    523     $
                    524   OPT$2
80A8 B6E005         525     OR   >2,RAM(FLG(PABPTR)) Mode code = 01
80AB 0402
                    526     $
                    527     $          C A S E   6   -   " U P D A T E "
                    528     $
                    529     $     Default....Check for multiple usage only...
                    530     $
                    531   OPT$3
                    532     $
                    533     $     Test for previous usage of any mode setting
                    534     $
80AD DA1701         535     $IF .BIT(MFLAG) @OPTFLG .NE. 0 GOTO OPERR
80B0 4106
80B2 B61701         536     SB   @OPTFLG,MFLAG      If not...set "MODE USED" bit
80B5 4045           537     BR   OPTION             Continue option scan
                    538     $
                    539     $          C A S E   7   -   " A P P E N D "
                    540     $
                    541     $     Mode code "11" indicates APPEND mode.
                    542     $
                    543   OPT$4
80B7 B6E005         544     OR   >6,RAM(FLG(PABPTR)) Mode code = 11
80BA 0406
80BC 40AD           545     BR   OPT$3
                    546     $
                    547     $          C A S E   1   -   " V A R I A B L E "
                    548     $
                    549     $     Change record type to VARIABLE and continue as F
```

```
                    550     $
                    551   OPT$01
80BE B6E005         552     SB    RAM(FLG(PABPTR)),4      Indicate variable length mode
80C1 0410
                    553     $
                    554     $          C A S E   8   -   " F I X E D "
                    555     $
                    556     $     FIXED is default.  Don't change anything, unless
                    557     $     argument is given.  In this case evaluate as numeric
                    558     $     expression and check for 8-bit integer range.....
                    559     $     This routine is also used for VARIABLE !!!!!
                    560     $
                    561   OPT$5
80C3 0F79           562     XML   PGMCHR                 Get next character
80C5 D642B3         563     $IF @CHAT .NE. COMMA$ THEN Could be some argument          1
80C8 60DB
80CA 0695AD         564         CALL CHKEND             Could also be end of stmt      1
80CD 60DB           565         BS   OPT$55             It is an EOS                   1
80CF 06809F         566         CALL CHKPAR             Check & parse expression       1
                    567         $
                    568         $ Check for byte overflow (records can only be up
                    569         $ to 255 bytes in length).....
                    570         $
80D2 8E4A41         571         $IF @FAC .NE. 0 GOTO OPERR                             1
80D5 06
80D6 BCE008         572         ST   @FAC1,RAM(LEN(PABPTR))  Select non-zero rec.-sizi
80D9 044B
                    573     $END IF
                    574   OPT$55
80DB DA1710         575     $IF .BIT(RFLAG) @OPTFLG .NE. 0 GOTO OPERR
80DE 4106
80E0 B61710         576     SB    @OPTFLG,RFLAG     Prevent too much usage of modes
80E3 4047           577     BR    OPTI$0            Continue option scan
                    578     $
                    579     $          C A S E   3   -   " I N T E R N A L "
                    580     $
                    581     $     Select INTERNAL file type and continue in DISPLAY....
                    582     $
                    583   OPT$03
80E5 B6E005         584     SB    RAM(FLG(PABPTR)),3      Select INTERNAL type
80E8 0408
                    585     $
                    586     $          C A S E   9   -   " D I S P L A Y "
                    587     $
                    588     $     Default.  Only check for mutiple usage of either
                    589     $     DISPLAY or INTERNAL.......
                    590     $
                    591   OPT$6
80EA DA1702         592     $IF .BIT(DFLAG) @OPTFLG .NE. 0 GOTO OPERR
80ED 4106
80EF B61702         593     SB    @OPTFLG,DFLAG     Else set "DISPLAY/INTERNAL" flag
80F2 4045           594     BR    OPTION            Continue....DISPLAY is default
                    595     $
                    596     $          C A S E   1 0   -   " I N P U T "
                    597     $
```

```
                    593     $     Same as any other I/O type definition.  Mode code
                    599     $     "10"....continue in OPT$3
                    600     $
                    601   OPT$7
80F4 B6E005         602     OR    >4,RAM(FLG(PABPTR)) Mode code = 10
80F7 0404
80F9 40AD           603     BR    OPT$3
                    604     $
                    605     $     CLRFRE deallocates previously allocated (parts of)
                    606     $     PAB's and returns with an error message
                    607     $
                    608   CLRFRE
80FB B602           609     CLR   @MNUM                 Undo any allocation
80FD BC03E0         610     ST    RAM(OFS(PABPTR)),@MNUM+1 We need the length for that
8100 0304
                    611   *     RAM(OFS(PABPTR)) Was set up in PARFN routine
8102 A14002         612     DADD  @MNUM,@FREPTR         Update the first free word
8105 00             613     RTN                         And return
                    614     $
                    615   OPERR
8106 0680FB         616     CALL  CLRFRE                First undo the allocation
                    617   ERRSYN
8109 066A84         618     CALL  ERR$$                 Then give an error
810C 03             619     DATA  3                     * SYNTAX ERROR
                    620     $
                    621     $     Continue with CHECK to complete the actual OPEN
                    622     $
                    623   CHECK
810D 0695AD         624     CALL  CHKEND                Check EOS
8110 4106           625     BR    OPERR                 Not EOS : SYNTAX ERROR
                    626     $
                    627     $     If the user hasn't specified VARIABLE or FIXED,
                    628     $     the default specification depends on the file
                    629     $     type.  Change current default (=VARIABLE) to
                    630     $     FIXED for RELATIVE files.
                    631     $
8112 DAE005         632     $IF . BIT0 RAM(FLG(PABPTR)) .EQ. 1 THEN RELATIVE RECORD   1
8115 040161
8118 27
8119 DAE005         633       $IF . BIT4 RAM(FLG(PABPTR)) .EQ. 1 THEN  VARIABLE mode  2
811C 041061
811F 25
                    634   FIL$$
8120 0680FB         635         CALL CLRFRE             Undo the PAB allocation
8123 57D7           636         BR   ERRFE              FILE ERROR
                    637       $END IF
8125 4131           638     $SELSE                      Sequential file...check rec. mode 1
8127 DA1710         639       $IF . BIT(RFLAG) @OPTFLG .EQ. 0 THEN No definition yet  2
812A 4131
812C B6E005         640         SB   RAM(FLG(PABPTR)),4 Force VARIABLE mode             2
812F 0410
                    641       $END IF
                    642     $END IF
8131 06976B         643     CALL  CDSR                  Call the DSR...return with error
8134 5794           644     BR    ERR$2B                  indication in COND.....
```

```
8138 87E00A   645    DCLR RAM(RNM(PABPTR))  Make sure we start with record 0
8139 04
              646    $
              647    $     Check for undefined record length...The record
              648    $     length for any type might be defined by the DSR.
              649    $
813A 8EE008   650    $IF RAM(LEN(PABPTR)) .EQ. 0 GOTO FIL$$
813D 046120
8140 BC03E0   651    ST   RAM(LEN(PABPTR)),@MNUM+1 Get record length
8143 0804
8145 8602     652    CLR  @MNUM                Create two byte result and allocate
8147 86E003   653    CLR  RAM(OFS(PABPTR))     - remove offset for later use
814A 04
814B BD4A02   654    DST  @MNUM,@FAC           - prepare for space claim
              655    $
              656    $     Check for special case . no PAB's yet
              657    $
814E 8F3C41   658    $IF @IOSTRT .DEQ. 0 THEN                                    1
8151 57
8152 BD3C04   659      DST  @PABPTR,@IOSTRT Simply enter the first pointer       1
8155 4169     660    $SELSE                                                      1
8157 BD0A3C   661      DST  @IOSTRT,@STADDR Search for the end of the chain     1
              662      $
815A 8FB00A   663      $WHILE RAM(@STADDR) .DNE. 0                             2
815D 6165
815F BD0AB0   664        DST  RAM(@STADDR),@STADDR Keep on deferring           2
8162 0A
8163 415A     665      $SEND WHILE                                             1
8165 BDB00A   666      DST  @PABPTR,RAM(@STADDR) Update last chain link        1
8168 04
              667    $SEND IF
8169 BDE006   668    DST  @PABPTR,RAM(BUF(PABPTR)) Set empty buffer first
816C 0404
816E 0F72     669    XML  MEMCHK               Check memory overflow&string space
8170 77C7     670    BS   ERRMEM               * MEMORY FULL
8172 A54002   671    DSUB @MNUM,@FREPTR        Compute buffer entry address
8175 A5E006   672    DSUB @MNUM,RAM(BUF(PABPTR)) Correct buffer address in PAB
8178 0402
817A 0F75     673    XML  CONT                 Return to the parser
```

```
                    675 ********************************************************************
                    676 *              " D E L E T E "   R O U T I N E                     *
                    677 *                                                                  *
                    678 *       Use file # 0 for this operation...  Parse the file         *
                    679 *       name string-expression as usual, and delete the PAB        *
                    680 *       before actually calling the DSR.......                     *
                    681 ********************************************************************
                    682 DELET
817C 8617           683     CLR   @FNUM                 Create file #0 - non-existing
817E 06965A         684     CALL  PARFN                 Handle as normal PAB OPEN
8181 0695AD         685     CALL  CHKEND                Check EOS first
8184 4106           686     BR    OPERR                 Not EOS : go undo PAB allocation
                    687 *                               and print SYNTAX ERROR
8186 8602           688     CLR   @MNUM                 Delete PAB again before calling DSR
8188 BC03E0         689     ST    RAM(OFS(PABPTR)),@MNUM+1 Create double byte PAB len
818B 0304
818D A14002         690     DADD  @MNUM,@FREPTR         Update free word pointer
8190 06975E         691     CALL  IOCALL                Perform I/O call for actual delete
8193 07             692     DATA  C$DELE
8194 0F75           693     XML   CONT                  Check for Junk on End in CONT
```

```
                    695  ********************************************************************
                    696  *              " C L O S E "   R O U T I N E                     *
                    697  *                                                                  *
                    698  *      Syntax could be                                             *
                    699  *                                                                  *
                    700  *      CLOSE #{num exp}  or  CLOSE #{num exp}:DELETE               *
                    701  *                                                                  *
                    702  *      Possibly output pending records before                      *
                    703  *      closing or deleting the file.....                           *
                    704  ********************************************************************
                    705  CLOSE
8196 069355         706    CALL CHKFN                Check for "no #" / "#0" cases
8199 77D7           707    BS   ERRFE                Not for "CLOSE" you don't
819B 069371         708    CALL CHKCON               Check file number etc...
819E 57D7           709    BR   ERRFE                *** FILE NUMBER NOT IN SYSTEM
81A0 069390         710    CALL OUTEOF               Output pending records
81A3 BEE004         711    ST   C$CLOS,RAM(COD(PABPTR)) Default to CLOSE I/O code
81A6 0401
81A8 D642B5         712    $IF @CHAT .EQ. COLON$ THEN  Check for ":DELETE" spec.     1
81AB 41B8
81AD 0F79           713      XML  PGMCHR             Request next input token        1
81AF 0F7E           714      XML  SPEED              Must be at a                    1
81B1 00             715      DATA SYNCHK                'DELETE' else                1
81B2 99             716      DATA DELET$                   its an error              1
81B3 BEE004         717      ST   C$DELE,RAM(COD(PABPTR)) Change CLOSE to DELETE    1
81B6 0407
                    718    $END IF
81B8 0695AD         719    CALL CHKEND               EOS?
81BB 4109           720    BR   ERRSYN               NO:SYNTAX ERROR
81BD 06976B         721    CALL CDSR                 Call DSR with whatever we have
81C0 41C7           722    BR   CLOS$1               Reset means error.............
81C2 0693A6         723    CALL DELPAB               Delete PAB and data-buffer
81C5 0F75           724    XML  CONT                 Return to parser routine
                    725  CLOS$1
81C7 BD5CE0         726    DST  RAM(4(PABPTR)),@ARG    Save error code for message
81CA 0404
81CC 0693A6         727    CALL DELPAB               Now delete the PAB
81CF BD0440         728    DST  @FREPTR,@PABPTR      Store error-code in free memory
81D2 A70400         729    DSUB 6,@PABPTR            Create standard size PAB
81D5 06
81D6 BDE004         730    DST  @ARG,RAM(4(PABPTR))  Copy error-code
81D9 045C
81DB 57BB           731    BR   ERRIO                Exit to error-routine
```

```
                    733  ************************************************************
                    734  *        " C L O S E   A L L "   R O U T I N E             *
                    735  *                                                          *
                    736  *      CLOSE all the existing PABs...ignore errors         *
                    737  *                                                          *
                    738  *      NOTE: "CLSLBL" is used in the I/O error routine     *
                    739  *            to determine if a warning should be given     *
                    740  *            rather than an error.....                     *
                    741  ************************************************************
                    742    $REPEAT
81DD BD04B0         743      DST  RAM(@PABPTR),@PABPTR                              1
81E0 04
                    744  CLSA$0
81E1 8FB004         745    $UNTIL RAM(@PABPTR) .DEQ. 0 Find last PAB in chain
81E4 41DD
81E6 069390         746    CALL OUTEOF               Take care of pending records
                    747  CLSLBL
81E9 BEE004         748    ST   C$CLOS,RAM(COD(PABPTR)) Select CLOSE code
81EC 0401
81EE 06976B         749    CALL CDSR                 CLOSE to DSR routine
81F1 0693A6         750    CALL DELPAB               Delete PAB - ignore CLOSE errors
                    751  CLSALL
81F4 BD043C         752    DST  @IOSTRT,@PABPTR      Start at beginning of chain
81F7 8F3C41         753    $IF @IOSTRT .DNE. 0 GOTO CLSA$0  Continue until done
81FA E1
81FB 00             754    RTN                       And return
```

```
                    756 ********************************************************************
                    757 *              " R E S T O R E "   R O U T I N E                   *
                    758 *                                                                  *
                    759 *       RESTORE can have any of four forms :                       *
                    760 *                                                                  *
                    761 *       RESTORE                    Restore to first DATA           *
                    762 *       RESTORE 20                 Restore DATA pointer            *
                    763 *       RESTORE #1                 Rewind file number 1            *
                    764 *       RESTORE #1, REC 2          Position file 1 at rec 2        *
                    765 *                                                                  *
                    766 ********************************************************************
                    767 RESTOR
81FC 874A           768   DCLR @FAC                  Assume simple RESTORE
81FE D642FD         769   $IF @CHAT .NE. NUMBE$ GOTO OLDCD
8201 421F
8203 069355         770   CALL CHKFN                 Check for #<filenumber>
8206 8F4A62         771   $IF @FAC .DEQ. 0 GOTO OLDC$0 Found equivalent of #0
8209 27
820A 069371         772   CALL CHKCON                Check and decode file #
820D 57D7           773   BR   ERRFE                 Give error if file not there
820F 069390         774   CALL OUTEOF                Output pending record
8212 87E00A         775   DCLR RAM(RNM(PABPTR))      Initialize to record 0
8215 04
8216 0694C2         776   CALL PARREC                Parse possible record clause
8219 06975E         777   CALL IOCALL                Call DSR routine with
821C 04             778   DATA C$REST                  RESTORE I/O code
821D 0F75           779   XML  CONT                  Return if no error found
                    780   $
                    781   $    Following code is for handling RESTORE to
                    782   $    line number within program
                    783   $
                    784 OLDCD
821F 0695AD         785     CALL  CHKEND             Check for start with end
8222 6227           786     BS    OLDC$0             If we have anything else...
8224 06A006         787     CALL LINE                  in FAC (double)
                    788 OLDC$0
8227 D53032         789   $IF @STLN .DEQ. @ENLN THEN                                     1
822A 4233
                    790 WRNNPP
822C 066A82         791     CALL WARN$$              * NO PROGRAM PRESENT *              1
822F 1D             792     DATA 29                                                      1
8230 056012         793     B  TOPL15                Go back to toplevel                 1
                    794   $END IF
8233 BD3632         795   DST  @ENLN,@LNBUF          Start at beginning of program
8236 A73600         796   DSUB 3,@LNBUF              Backup for first line number
8239 03
                    797   $
                    798   $    Check against given line number
                    799   $
                    800 OLDC$1
823A 0691BC         801   CALL GRSUB3                Read 2 bytes of ln ptr from ln #
                    802 *                            table which is in ERAM/VDP
823D 36             803   DATA LNBUF                 Source addr. on ERAM/VDP
                    804 *                            @EEE1:Destination addr. on CPU
823E C54A58         805   $IF @FAC .DH. @EEE1 THEN Try to get something higher           1
```

```
8241 424E
8243 D53630    806      $IF @LNBUF .DEQ. @STLN GOTO ERRDAT Last line in prog   1
8246 77D3
8248 A73600    807      DSUB 4,@LNBUF          Get next entry in line # table  1
8248 04
824C 423A      808      BR OLDC$1              Try again with next line        1
               809    $END IF
824E A33600    810    DADD 3,@LNBUF            Undo subtraction
8251 03
8252 06A008    811    CALL DATAST              Setup pointers for READ
8255 0F75      812    XML  CONT                Continue PARSE
```

```
                        814  ********************************************************************
                        815  *                  D I S P L A Y    R O U T I N E                  *
                        816  *                                                                  *
                        817  *       DISPLAY handles all random screen access stuff,.....*
                        818  *       the AT-clause, and the BEEP, ERASE ALL and SIZE            *
                        819  *       clause.                                                    *
                        820  ********************************************************************
                        821  DISPL1
8257 0694EF             822    CALL DISACC              Evaluate DISPLAY options
825A 6395               823    BS   EOLEX               EXIT directly on end-of-stmt
                        824  *
                        825  *       if anything is specified it has to be a colon
                        826  *
825C 8E0462             827    $IF @PABPTR .EQ. 0 GOTO PRIN$1 Nothing was specified
825F C4
                        828  *
                        829  *       At this point we MUST have a colon, or else
                        830  *       we error off ( SYNTAX ERROR )
                        831  *
8260 0F7E               832    XML  SPEED               Check for a colon
8262 00                 833    DATA SYNCHK                 and continue
8263 B5                 834    DATA COLON$                    if approved
8264 42C4               835    BR   PRIN$1              Continue with PRINT items
                        836  ********************************************************************
                        837  *              P R I N T   R O U T I N E                           *
                        838  *                                                                  *
                        839  *           MAIN-HANDLER FOR ALL PRINT-FUNCTIONS.                  *
                        840  ********************************************************************
                        841  PRINT
8266 06973F             842    CALL INITKB              initialize keyboard I/O
8269 D642FD             843    $IF @CHAT .EQ. NUMBE$ THEN Could still be anything          1
826C 42C4
826E 069355             844      CALL CHKFN             Check if default or open channel  1
8271 8F4A62             845      $IF @FAC .DEQ. 0 GOTO PRN$10 Default intended            1
8274 A5
8275 069371             846      CALL CHKCON            Check and convert expression      1
8278 57D7               847      BR   ERRFE             Error if PAB not in system        1
                        848      $
                        849      $  PRINT allowed in output, append or update modes
                        850      $         Not allowed input mode
                        851      $
827A DAE005             852      $IF .BIT2 RAM(FLG(PABPTR)) .EQ. 1 THEN                    2
827D 040462
8280 88
8281 DAE005             853        $IF .BIT1 RAM(FLG(PABPTR)) .EQ. 0 GOTO ERRFE            3
8284 040277
8287 D7
                        854      $END IF
8288 D6E004             855      $IF RAM(COD(PABPTR)) .EQ. C$READ THEN                     2
828B 040242
828E 93
828F 86E003             856        CLR  RAM(OFS(PABPTR))  Unpend pending INPUTs            2
8292 04
                        857      $END IF                  Next WRITE selection is for
8293 BEE004             858      ST   C$WRITE,RAM(COD(PABPTR))  uncomplete PRINTs        1
```

```
8296 0403
8298 0696F5    859      CALL PRINIT          Initialize some variables
               860  *
               861  *   Next character has to be either EOL,COMMA or COLON
               862  *
829B 0695AD    863      CALL CHKEND
829E 6395      864      BS   EOLEX           exit on end of statement
82A0 0694C2    865      CALL PARREC          Parse possible record clause
82A3 62B1      866      BS   PRIN$0          found "," but no REC clause
               867  PRN$10
82A5 0695AD    868      CALL CHKEND
82A8 6395      869      BS   EOLEX           Exit on end of statement for
               870  *                        "PRINT #0" or "PRINT file position"
82AA D642B3    871      $IF @CHAT .EQ. COMMA$ THEN
82AD 42DE
82AF 0F79      872        XML  PGMCHR        get next in line
               873  PRIN$0
82B1 8E0463    874        $IF @PABPTR .EQ. 0 GOTO USING For "PRINT #0....."
82B4 CF
               875  *                  Internal type of file ?
82B5 DAE005    876        $IF .BIT3 RAM(FLG(PABPTR)) .EQ. 1 GOTO ERRFE
82B8 040857
82BB D7
82BC 43CF      877        BR  USING          execute USING clause
               878      $END IF
82BE 0F7E      879      XML  SPEED           Must be at a
82C0 00        880      DATA SYNCHK              colon at this point
82C1 B5        881      DATA COLON$                and error off on others
82C2 42C9      882    $SELSE                 make it a short branched ELSE
               883  PRIN$1
82C4 D642ED    884      $IF @CHAT .EQ. USING$ GOTO USING
82C7 63CF
               885    $END IF                end standard initialization
               886  *
               887  *   Test standard separators
               888  *
               889  CONPRT
82C9 069627    890      CALL TSTSEP          Test separator character
82CC D642FC    891      $IF @CHAT .EQ. TAB$ GOTO PRTAB  Handle TABs
82CF 632D
               892      $
               893      $  At this point we've checked TAB and ";", ",", ":".
               894      $  The only remaining print items have to be expressions
               895      $       All expressions are being handled below.
               896      $  If the result of the expression is a numeric,
               897      $  the string is transformed into a string and
               898      $  printed.  Strings are printed "as is".
               899      $       The code for strings and converted numerics
               900      $  cannot be made common, since numerics may require
               901      $  an extra space behind the item, depending upon the
               902      $  current position in the record.
               903      $       Either way, the string is chunked up into little
               904      $  pieces if it won't fit in an empty record.
               905      $
82D1 0F74      906      XML  PARSE           Evaluate the expression
```

```
82D3 35          907       DATA COLON$
                 908       $
                 909       $  Special code for INTERNAL file handling
                 910       $       Translate numeric datums into string
                 911       $  format and indicate length 8.  Then
                 912       $  check to see if the item fits within the
                 913       $  current record.  If not, it is an error,
                 914       $  since each item has to fit.
                 915       $
82D4 0683C5      916       CALL TSTINT            Test for internal files
82D7 6303        917       BS   OTHE$1           Nope...something different
82D9 D64C65      918       $IF @FAC2 .NE. STRVAL THEN Change numerics          1
82DC 62EC
82DE BE5608      919         ST   8,@FAC12          to string length 8           1
82E1 350008      920         MOVE 8 FROM @FAC TO @ARG  save in ARG               1
82E4 5C4A
82E6 BE555C      921         ST   ARG,@FAC11        and use this as source       1
82E9 06960B      922         CALL RSTRING       Reserve some string space        1
                 923       $END IF
82EC BC5C07      924       ST   @RECLEN,@ARG    Compute remaining space to EOR
82EF A45C06      925       SUB  @CCPPTR,@ARG      for space checking
82F2 905C        926       INC  @ARG            Make it real space
82F4 C8515C      927       $IF @FAC7 .HE. @ARG GOTO ERRFE Not enough !!!!!!
82F7 77D7
                 928       $                    The = check includes length byte
82F9 BCB008      929       ST   @FAC7,RAM(@CCPADR) Prestore string length
82FC 51
82FD 9108        930       DINC @CCPADR         Update actual RAM address
82FF 9006        931       INC  @CCPPTR           and internal column pointer
8301 4308        932       BR   OTHE$0
                 933  OTHE$1
8303 D64C65      934       $IF @FAC2 .EQ. STRVAL THEN Print the string result  1
8306 430D
                 935  OTHE$0
8308 06970E      936         CALL OSTRNG        Output the string to the record. 1
830B 4328        937       $SELSE                   Numeric datum               1
830D 8655        938         CLR  @FAC11        Select standard BASIC format    1
830F 0F73        939         XML  CNS           Convert number to string        1
8311 06960B      940         CALL RSTRING       Reserve and copy string         1
8314 06970E      941         CALL OSTRNG        Output the string               1
                 942         $
                 943         $ Possibly add an extra space if we're not at the end
                 944         $ of the current record...
                 945         $
8317 C80706      946         $IF @RECLEN .HE. @CCPPTR THEN Enough space left   2
831A 4328
831C BEB008      947           ST   SPACE,RAM(@CCPADR)   Add trailing space    2
831F 20
8320 A0B008      948           ADD  @DSRFLG,RAM(@CCPADR) Take care of screen I/O 2
8323 17
8324 9108        949           DINC @CCPADR       Update current column address 2
8326 9006        950           INC  @CCPPTR         and base 1 pointer           2
                 951         $END IF
                 952       $END IF
                 953  CHKSEP
```

```
8328 069627   954       CALL TSTSEP          Check for legal delimiter
832B 4109     955       BR   ERRSYN           Illegal delimiter. SYNTAX ERR
              956  *                         Unconditional branch
              957       $
              958       $  PRTAB - Print TAB as part of PRINT command
              959       $
              960  PRTAB
832D 0683C5   961       CALL TSTINT          Watch out for INTERNAL file types
8330 57D7     962       BR   ERRFE           They can't handle TABs
8332 0F79     963       XML  PGMCHR          Skip TAB keyword
8334 D642B7   964       $IF @CHAT .NE. LPAR$ GOTO ERRSYN
8337 4109
8339 0F74     965       XML  PARSE           Parse TAB expression
833B B6       966       DATA RPAR$
833C 0694B8   967       CALL CNVDEF          Check and convert to integer
833F BC4C07   968       ST   @RECLEN,@FAC2   Set modulo number
8342 06961A   969       CALL COMMOD          Compute remainder
8345 C4064B   970       $IF @CCPPTR .H. @FAC1 THEN Position on next output rec 1
8348 434F
834A 0696A5   971         CALL OUTREC        Output current record - no pending1
834D 6328     972         BS   CHKSEP        react on SIZE block !!!          1
              973       $END IF
834F D4064B   974       $IF @CCPPTR .EQ. @FAC1 GOTO CHKSEP Stay here
8352 6328
8354 BC034B   975       ST   @FAC1,@MNUM+1   Fill with spaces
8357 0F84     976       XML  IO              O.K...go ahead...fill'r up
8359 01       977       DATA FILSPC
835A 4328     978       BR   CHKSEP          And check separator again
              979       $
              980       $  Comma is simular to TAB, except that it generates
              981       $  at least one space.  The exact number of spaces
              982       $  generated depends upon the current position within
              983       $  the record.  If the next fixed tab-position is
              984       $  outside the record, the current record is output
              985       $  and the column pointer is reset to column 1 of the
              986       $  next record.
              987       $
              988  PRTCOM
835C BC0306   989       ST   @CCPPTR,@MNUM+1 Compute initial # of spaces
835F 9203     990       DEC  @MNUM+1         Decrement for 0 origin
8361 8602     991       CLR  @MNUM           Clear high byte for double
8363 AE020E   992       DIV  14,@MNUM        TABs are 14 spaces apart
8366 9002     993       INC  @MNUM           Compute next TAB-stop
8368 AA020E   994       MUL  14,@MNUM          and actual position
836B C40703   995       $IF  @RECLEN .H. @MNUM+1 THEN Within this record      1
836E 4377
8370 9003     996         INC  @MNUM+1       Convert to real position         1
8372 0F84     997         XML  IO            Fill spaces to new location      1
8374 01       998         DATA FILSPC                                         1
8375 437A     999       $SELSE               Outside current record           1
             1000         $
             1001         $ The ":" (colon) separator is used to output the
             1002         $ current record, and proceed to position 1 of the
             1003         $ next record.
             1004         $
```

```
                    1005  PRCOL
8377 0696A5         1006        CALL OUTREC          Output the current record      1
                    1007      $END IF
                    1008      $
                    1009      $  The ";" generates the null string.  Since all print
                    1010      $  items should be separated by a separator, this one
                    1011      $  has been introduced to separate without moving to
                    1012      $  another position.  Notice that all separators join
                    1013      $  up here.
                    1014      $
                    1015  PRSEM
837A 0F79           1016        XML  PGMCHR          Skip the separator
837C 0695AD         1017      CALL CHKEND            Exit on end of line
837F 42C9           1018      BR   CONPRT            Continue if not end of line
                    1019  PRSM$1
8381 8E1763         1020      $IF @DSRFLG .EQ. 0 GOTO PREXIT for screen output continue
8384 A4
8385 DA0408         1021      $IF .BIT3 @PABPTR .EQ. 0 GOTO PREXIT check SIZE clause
8388 63A4
838A 0696A5         1022      CALL OUTREC          output current record (blank rest)
838D BC0609         1023      ST  @CCPADR+1,@CCPPTR compute correct value for CCPPTR
8390 A606E1         1024      SUB  >E1,@CCPPTR     subtract current screen base
8393 43A4           1025      BR   PREXIT          and exit from this command
                    1026      $
                    1027      $    End of line exit routine for PRINT statement
                    1028      $
                    1029  EOLEX
8395 8E1763         1030      $IF @DSRFLG .NE. 0 THEN screen I/O - remove blocks if  1
8398 A1
8399 DA0404         1031        $IF .BIT2 @PABPTR .EQ. 0 THEN    "AT" clause unused  2
839C 43A1
839E B204E7         1032          AND  >E7,@PABPTR   remove flag 3 (SIZE used)       2
                    1033        $END IF
                    1034      $END IF
83A1 0696A5         1035      CALL OUTREC          Output pending record
                    1036      $
                    1037      $    continue here if record remains pending
                    1038      $
                    1039  PREXIT
83A4 8E1743         1040      $IF @DSRFLG .EQ. 0 THEN Regular file/device I/O        1
83A7 B1
83A8 9206           1041        DEC  @CCPPTR         Back to actual offset          1
83AA BCE003         1042        ST   @CCPPTR,RAM(OFS(PABPTR)) Save for next time    1
83AD 0406
83AF 0F75           1043        XML  CONT            Continue with next stmt        1
                    1044      $END IF                End external I/o handling
                    1045  *
                    1046  *    Reset of code is for internal I/O  (VDP)
                    1047  *
83B1 DA0404         1048      $IF .BIT2 @PABPTR .EQ. 0 THEN AT( , ) Is not used     1
83B4 43BB
83B6 BC7F06         1049        ST  @CCPPTR,XPT      Save current value of pointer  1
83B9 947F           1050        INCT XPT             CCPPTR:1--28                   1
                    1051      $END IF
83BB DA0402         1052      $IF .BIT1 @PABPTR .EQ. 1 THEN Used BEEP clause       1
```

```
83BE 63C3
83C0 060034   1053      CALL TONE1              ------- BEEP ------
              1054    $END IF
83C3 0F75     1055    XML  CONT                 Continue in PARSE routine
              1056    $
              1057    $     TSTINT - test for INTERNAL type file...set COND
              1058    $                if file is NOT INTERNAL
              1059    $
              1060  TSTINT
83C5 8E1753   1061    $IF @DSRFLG .NE. 0 GOTO RTC Couldn't possibly be INTERNAL
83C8 8C
83C9 DAE005   1062    CLOG >08,RAM(FLG(PABPTR))   Set COND according to bit 3
83CC 0408
83CE 01       1063    RTNC                      Return without changing COND
```

```
                1065  *
                1066  *    P R I N T / D I S P L A Y   U S I N G   S E C T I O N
                1067  *
                1068  *       arrive here after the keyword "USING" has
                1069  *       been recognized.
                1070  *
                1071  USING
83CF 0F7E       1072    XML   SPEED
83D1 00         1073    DATA  SYNCHK              get first character of format stmt
83D2 ED         1074    DATA  USING$              after (double) checking USING
83D3 D642C9     1075    $IF @CHAT .EQ. LN$ THEN pick up the line number          1
83D6 4430
83D8 0F79       1076      XML  PGMCHR             get high address               1
83DA BC4A42     1077      ST   @CHAT,@FAC                                        1
83DD 0F79       1078      XML  PGMCHR             and low address                1
83DF BC4B42     1079      ST   @CHAT,@FAC1                                       1
83E2 0F79       1080      XML  PGMCHR             get next program character     1
83E4 BD4C2E     1081      DST  @EXTRAM,@FAC2      in SEETWO :EXTRAM value will be 1
                1082  *                           changed
83E7 0F7E       1083      XML  SPEED                                             1
83E9 03         1084      DATA SEETWO             Find the line # in the program 1
83EA C14C2E     1085      DEX   @EXTRAM,@FAC2     result in SEETWO is in EXTRAM  1
                1086  *                           and restore EXTRAM value
83ED 445E       1087      BR   USNG$1            has to match exactly           1
83EF 954C       1088      DINCT @FAC2             move up to the pointer field   1
83F1 BD5234     1089      DST   @DATA,@FAC8       save DATA pointer for READ usage 1
83F4 0691A4     1090      CALL GRSUB2             Read 2 bytes of data from ERAM/VDP1
83F7 4C         1091      DATA FAC2               @FAC2:Source address on ERAM/VDP 1
83F8 BD3458     1092      DST @EEE1,@DATA         @EEE1:Destination addr. on CPU 1
                1093  *                           Put it in @DATA
83FB BE4CA3     1094      ST    IMAGE$,@FAC2      search for an IMAGE token       1
83FE 068B99     1095      CALL  SEARCH            at the beginning of a statement 1
8401 645E       1096      BS    USNG$1            error if not found on this line 1
8403 069304     1097      CALL GETGFL             get first part of format string 1
8406 069323     1098      CALL CHKSTR             prepare data for string assignment1
8409 BD0C50     1099      DST  @FAC6,@BYTE        copy actral string length in BYTE 1
840C BD3452     1100      DST  @FAC8,@DATA        restore original DATA pointer  1
840F 0692CD     1101      CALL CTSTR              create a temporary string      1
8412 8F5064     1102      $IF @FAC6 .DNE. 0 THEN                                 2
8415 2E
8416 8E8084     1103        $IF @RAMTOP .EQ. 0 THEN data from RAM                3
8419 4423
841B 3450B0     1104          MOVE @FAC6 FROM RAM(@TEMP5) TO RAM(@SREF)          3
841E 1CB066
8421 442E       1105        $SELSE                                               3
8423 BD5650     1106          DST  @FAC6,@FFF1    FFF1 : byte count              3
8426 BD5466     1107          DST  @TEMP5,@DDD1   DDD1 : source address in ERAM  3
8429 BD581C     1108          DST  @SREF,@EEE1    EEE1 : destination address on VDP3
842C 0F8B       1109          XML  GVWITE         write data from ERAM to VDP    3
                1110        $END IF
                1111      $END IF
842E 4438       1112    $SELSE
8430 0F74       1113      XML PARSE               parse up to the ending ":"
8432 B5         1114      DATA COLON$
8433 D64C65     1115      $IF @FAC2 .NE. STRVAL GOTO USNG$1 "* IMAGE ERROR"
```

```
8436 445E
                1116    $END IF
8438 D642B5     1117    $IF @CHAT .NE. COLON$ THEN  probably no variable list    1
843B 6448
843D 0695AD     1118      CALL CHKEND             we better check that though    1
8440 4109       1119      BR   ERRSYN              something sneaky sneaked in    1
8442 8E5163     1120      $IF @FAC7 .EQ. 0 GOTO EOLEX  end of line exit          1
8445 95
8446 4463       1121    $ELSE                     look for format item           1
8448 8E5164     1122      $IF @FAC7 .EQ. 0 GOTO USNG$1 exclude null strings      1
844B 5E
844C BD5C4E     1123      DST  @FAC4, @ARG        get start address for string scan 1
844F BC5E51     1124      ST   @FAC7,@ARG2        get format string length       1
                1125  USNG$0
8452 D6B05C     1126      $IF RAM(@ARG) .NE. :#: THEN found no format item yet   2
8455 236460
8458 915C       1127        DINC @ARG             try next address               2
845A 925E       1128        DEC  @ARG2            update address                 2
845C 4452       1129        BR   USNG$0           try up to the end of the string 2
                1130  USNG$1
845E 57C3       1131        BR ERRIM              * IMAGE ERROR                  2
                1132      $END IF
                1133  *
                1134  *     Now we're sure that we have at least one legal
                1135  *     format item (anything with a "#" in it)
                1136  *
8460 BE42B3     1137      ST   COMMA$,@CHAT       fake comma separator for printout 1
                1138    $END IF
8463 0F77       1139    XML  VPUSH                current string might be tempora
8465 BD0C50     1140    DST   @FAC6,@BYTE         create a workstring for output
8468 900D       1141    INC @BYTE+1               create space for end of string ind.
846A 0C645E     1142    $IF .CARRY. GOTO USNG$1 string would be too long
846D 0F71       1143    XML  GETSTR               length should equal format string
846F BD141C     1144    DST  @SREF,@CURLIN        create a temporary string
8472 A11C50     1145    DADD @FAC6,@SREF          compute last position in string
8475 86B01C     1146    CLR  RAM(@SREF)           set end of string indicator
                1147  USNG$3
8478 BD4EE0     1148    DST  RAM(4(VSPTR)),@FAC4  Update FAC4 area in case garbage
847B 046E
847D 3450B0     1149    MOVE @FAC6 FROM RAM(@FAC4) TO RAM(@CURLIN) copy format
8480 14B04E
8483 BD4E14     1150    DST @CURLIN,@FAC4         complete preps for VPUSH
8486 BF4A00     1151    DST SREF,@FAC
8489 1C
848A 9150       1152    DINC @FAC6                include 0 in string length
848C 0F77       1153    XML VPUSH                 make the string temporary
848E BD14E0     1154    DST RAM(4(VSPTR)),@CURLIN  Update current line pointer
8491 046E
                1155  USNG$4
8493 D6B014     1156    $IF RAM(@CURLIN) .NE. :#: THEN try to locate the next for
8496 2364C3
8499 8EB014     1157      $IF RAM(@CURLIN) .NE. 0 THEN not end of string yet
849C 64A2
849E 9114       1158        DINC @CURLIN          update pointer if not found
84A0 4493       1159        BR  USNG$4            and continue searching
```

```
                    1160          $END IF
84A2 D642B3         1161          #IF @CHAT .NE. COMMA$ GOTO USNG$9 stop on last variabl1
84A5 45C3
84A7 0F78           1162          XML VPOP                restore original workstring data  1
84A9 BC0C51         1163          ST  @FAC7,@BYTE         print the current format string   1
84AC 920C           1164          DEC  @BYTE              don't count the last "0"          1
84AE BE0301         1165          ST  1,@MNUM+1           indicate direct output without ski1
84B1 069718         1166          CALL CHKR$0             copy string to output record      1
84B4 0696A5         1167          CALL OUTREC             also output current record        1
                    1168     *
                    1169     *    FAC still contains the right data, however it is
                    1170     *    easier just to copy the original string again.
                    1171     *
84B7 BD144E         1172          DST @FAC4,@CURLIN       reconstruct CURLIN                1
84BA 0F78           1173          XML VPOP                copy original string info.        1
84BC 0F77           1174          XML VPUSH               without actually removing it      1
84BE A51450         1175          DSUB @FAC6,@CURLIN      reconstruct start address         1
84C1 4478           1176          BR  USNG$3              continue for the next variable    1
                    1177        $END IF
84C3 D514E0         1178        $IF @CURLIN .DNE. RAM(4(VSPTR)) THEN avoid "#" as count 1
84C6 046E64
84C9 E9
84CA 9314           1179          DDEC @CURLIN            backup to the sign                1
84CC D6B014         1180          $IF RAM(@CURLIN) .EQ. :.: THEN used ".#####"            2
84CF 2E44DB
84D2 D514E0         1181            $IF @CURLIN .DEQ. RAM(4(VSPTR)) GOTO USN$42           2
84D5 046E64
84D8 E9
84D9 9314           1182            DDEC @CURLIN          avoid checking count bit        2
                    1183          $END IF
84DB D6B014         1184          $IF RAM(@CURLIN) .NE. :-: THEN check for minus or plus2
84DE 2D64E9
84E1 D6B014         1185            $IF RAM(@CURLIN) .NE. :+: THEN                        3
84E4 2B64E9
84E7 9114           1186              DINC  @CURLIN       it's neither, so we undo        3
                    1187            $END IF
                    1188          $END IF
                    1189        $END IF
                    1190     USN$42
                    1191     *
                    1192     *      Check for availabililty of variables
                    1193     *
84E9 D642B3         1194        $IF @CHAT .NE. COMMA$ GOTO USNG$9 exit if no more pt item
84EC 45C3
84EE 0F79           1195        XML  PGMCHR               get next expression
84F0 A514E0         1196        DSUB RAM(4(VSPTR)),@CURLIN  make CURLIN offset for garbage
84F3 046E
                    1197     *                                  collection
84F5 0F74           1198        XML  PARSE                parse up to ";" or ","
84F7 B4             1199        DATA SEMIC$
84F8 A114E0         1200        DADD RAM(4(VSPTR)),@CURLIN reconstruct new CLN after G.C.
84FB 046E
84FD 8752           1201        DCLR @FAC8                start with clean sheet for counts
84FF 8755           1202        DCLR @FAC11
8501 8657           1203        CLR  @FAC13
```

```
8503 BD0E14  1204     DST  @CURLIN,@VAR4      now start checking process
8506 D6B014  1205     $IF RAM(@CURLIN) .EQ. :.: GOTO USNG$5
8509 2E6533
850C D6B014  1206     $IF RAM(@CURLIN) .NE. :#: THEN has to be "+" or "-"      1
850F 236527
8512 D6B014  1207       $IF RAM(@CURLIN) .EQ. :-: THEN                         2
8515 2D4513
8518 B65502  1208         SB  @FAC11,1        set explict sign flag for CNS    2
             1209       $END IF
851B D6B014  1210       $IF RAM(@CURLIN) .EQ. :+: THEN                         2
851E 2B4527
8521 B65502  1211         SB  @FAC11,1        set explict sign flag for CNS    2
8524 B65504  1212         SB  @FAC11,2        set positive sign flag for CNS   2
             1213       $END IF
             1214     $END IF
8527 0685E5  1215     CALL ACCNM              accept first character plus "#"'s
852A BC5653  1216     ST @FAC9,@FAC12         set up FAC12 for CNS
852D D6B00E  1217     $IF RAM(@VAR4) .EQ. :.: THEN found decimal point         1
8530 2E4540
             1218   USNG$5
8533 8653    1219       CLR @FAC9             prepare for use as counter of no. 1
             1220   *                         of # sign after decimal point
8535 0685E5  1221       CALL ACCNM            accept some more "#"'s           1
8538 BC5753  1222       ST @FAC9,@FAC13       set up FAC13 for CNS             1
853B A05356  1223       ADD @FAC12,@FAC9      FAC9 now contains the total no. of1
             1224   *                         "#" sign ,d.p.and maybe a sign bit
853E 9253    1225       DEC @FAC9             exclude the d.p.                 1
             1226     $END IF
8540 D7B00E  1227     $IF RAM(@VAR4) .DEQ. :^^: THEN attemp to decode "^^^^    1
8543 5E5E45
8546 64
8547 950E    1228       DINCT @VAR4           update address                   1
8549 D7B00E  1229       $IF RAM(@VAR4) .DEQ. :^^: THEN                         2
854C 5E5E45
854F 62
8550 950E    1230         DINCT @VAR4         update address                   2
8552 B65508  1231         SB  @FAC11,3        set E-format bit for CNS         2
8555 D6B00E  1232         $IF RAM(@VAR4) .NE. :^: GOTO USN$55                  2
8558 5E4564
855B 910E    1233         DINC @VAR4          update end address               2
855D B65510  1234         SB  @FAC11,4        set extended E-format bit for CNS 2
8560 4564    1235         BR  USN$55                                           2
             1236       $END IF
8562 970E    1237       DDECT @VAR4           correct for previous errors      1
             1238     $END IF
             1239   *
             1240   *   At this point, CURLIN is pointing at the first
             1241   *   item of the format, VAR4 is pointing at the
             1242   *   character following the item
             1243   *
             1244   USN$55
8564 CA4C64  1245     $IF @FAC2 .L. >64 THEN detected numerical argument       1
8567 6596
8569 DA5502  1246       $IF .BIT1 @FAC11 .EQ. 1 THEN  exclude the sign count   2
856C 6570
```

```
856E 9253     1247          DEC  @FAC9          FAC9: no. of significant digits   2
              1248        $END IF
8570 DA55C8   1249        $IF .BIT3 @FAC11 .EQ. 1 THEN   If E-format is used      2
8573 657C
8575 CE530A   1250          $IF @FAC9 .GT. 10 GOTO ERRIM more than 10 significant2
8578 77C3
              1251  *                                    digits in image clause ****ERROR
857A 4581     1252        $SELSE                         If E-format is not used          2
857C CE530E   1253          $IF @FAC9 .GT. 14 GOTO ERRIM more than 14 significant2
857F 77C3
              1254  *                                    digits in image clause *****ERROR
              1255        $END IF
8581 B65501   1256        SB @FAC11,0                    set fixed format output  it for CN1
8584 0F73     1257        XML CNS                        convert no. to fixed format string1
              1258  *
              1259  *     FAC11 points to the beginning of the string after
              1260  *     supressing leading 0's, FAC12 contains the length of
              1261  *     the string
              1262  *
8586 BC5755   1263        ST   @FAC11,@FAC13     FAC13 now point to beginning of 1
              1264  *                                    the string
8589 8655     1265        CLR  @FAC11                    clear high byte                  1
858B 3455B0   1266        MOVE @FAC11 FROM *FAC13 TO RAM(@CURLIN) copy the               1
858E 149057
              1267  *                                    result string from temp
8591 BD140E   1268        DST  @VAR4,@CURLIN             move pointer behind print field  1
8594 4493     1269        BR   USNG$4                    continue after printing          1
              1270      $END IF
8596 BD540E   1271      DST  @VAR4,@FAC10                compute total length
8599 A55414   1272      DSUB @CURLIN,@FAC10
859C C45155   1273      $IF @FAC7 .H. @FAC11 THEN string exceeds limits                  1
859F 45B1
85A1 BE002A   1274        ST  :*:,@VARO                  prepare a "*****.." string       1
              1275        $REPEAT
85A4 BCB014   1276          ST  @VARO,RAM(@CURLIN) fill the remainder of field     2
85A7 00
85A8 9114     1277          DINC @CURLIN                 up to the end                    2
              1278  USN$67
85AA D5140E   1279        $UNTIL @CURLIN .DEQ. @VAR4 which is stored in VAR4       1
85AD 45A4
85AF 4493     1280        BR  USNG$4                     continue with next item          1
              1281      $END IF
85B1 8F5065   1282      $IF @FAC6 .DEQ. 0 GOTO USN$68
85B4 BE
85B5 3450B0   1283      MOVE @FAC6 FROM RAM(@FAC4) TO RAM(@CURLIN) copy result
85B8 14B04E
              1284  *                                                        string
85BB A11450   1285      DADD @FAC6,@CURLIN             and update address in string
              1286  USN$68
85BE BE0020   1287      ST  : :,@VARO                  fill remainder with spaces
85C1 45AA     1288      BR  USN$67
              1289  USNG$9
85C3 0F78     1290      XML  VPOP                      temporary string back out
85C5 BC0C15   1291      ST   @CURLIN+1,@BYTE           output up to the current pos.
85C8 A40C4F   1292      SUB  @FAC5,@BYTE              create one byte result
```

```
85C8 65D3    1293     BS    USN$95              avoid empty strings
85CD BE0301  1294     ST    1,@MNUM+1           prevent skip if field too small
85D0 069718  1295     CALL   CHKR$0             perform all normal I/O stuff
             1296   USN$95
85D3 0F78    1297     XML    VPOP               remove source format string
85D5 0695AD  1298     CALL  CHKEND              check for end of line exit
85D8 6395    1299     BS    EOLEX               take end of line exit
85DA 0F7E    1300     XML   SPEED
85DC 00      1301     DATA  SYNCHK              then it HAS to be a ";"
85DD B4      1302     DATA   SEMIC$
85DE 0695AD  1303     CALL  CHKEND              Now - must be EOS
85E1 6381    1304     BS    PRSM$1              Supressed end of record, make it
             1305   *                                          a pending record
85E3 4109    1306     BR    ERRSYN              SYNTAX ERROR
             1307   *
             1308   *       Collect string of "#"'s
             1309   *
             1310   ACCNM
             1311     $REPEAT
85E5 9053    1312       INC   @FAC9             update item count                1
85E7 910E    1313       DINC   @VAR4            and item address                 1
85E9 D6B00E  1314     $UNTIL RAM(@VAR4) .NE. :#: decode as many "#"'s as possibl
85EC 2365E5
85EF 00      1315     RTN                       return from duty
```

```
                    1317  *
                    1318  *
                    1319  ***************************************************************
                    1320  *                  I N P U T   R O U T I N E                  *
                    1321  *                                                             *
                    1322  *      First check for file or screen I/O.  If file I/O       *
                    1323  *      then check for pending output and print that.          *
                    1324  *      If screen I/O then check for input prompt.             *
                    1325  *                                                             *
                    1326  *      Next collect the INPUT variable list on the V-stack.   *
                    1327  *      Get enough input from either file or keyboard, and     *
                    1328  *      compare types with entries on V-stack.                 *
                    1329  *                                                             *
                    1330  *      After verification and approval, assign the values.    *
                    1331  ***************************************************************
                    1332  INPUT
 85F0 06973F        1333    CALL INITKB             Assume keyboard INPUT
 85F3 D642FD        1334    $IF @CHAT .EQ. NUMBE$ THEN Might be #0 or #1-255          1
 85F6 475A
 85F8 069355        1335      CALL CHKFN            Check for default #0              1
 85FB 8F4A46        1336      $IF @FAC .DEQ. 0 THEN If luno #0                        2
 85FE 03
 85FF BDA3AA        1337        DST @PGMPTR,RAM(INPUTP)  Save PGMPTR for "try again" 2
 8602 2C
 8603 91A3AA        1338        DINC RAM(INPUTP)    Pass the ":" for the "prompt" code3
                    1339  *                         handler later, ( using #0 will not
                    1340  *                         take care the prompt in INPUT)
 8606 06892D        1341        CALL INPU$2         #0 is equivalent to no #        2
 8609 4768          1342        BR   INP$2                                          2
                    1343      $END IF
 860B 0688E8        1344      CALL INSU1            Get info about file              1
                    1345      $
                    1346      $  INTERNAL files get special treatment
                    1347      $
 860E DAE005        1348      $IF .BIT3 RAM(FLG(PABPTR)) .EQ. 1 THEN INTERNAL file  2
 8611 040866
 8614 AD
 8615 9EE003        1349        $IF RAM(OFS(PABPTR)) .EQ. 0 THEN Fresh start....    3
 8618 04461E
                    1350  INTR$0
 861B 069765        1351          CALL IOCL$1       Get a new record through the DSR 3
                    1352        $END IF
 861E BC2BE0        1353        ST   RAM(OFS(PABPTR)),@VARA+1  Regain possible offset3
 8621 0304
 8623 862A          1354        CLR  @VARA          Make that a two byte constant    3
 8625 BD66E0        1355        DST  RAM(BUF(PABPTR)),@TEMP5  Get first address      3
 8628 0604
 862A A1662A        1356        DADD @VARA,@TEMP5   Compute actual address within rec.2
                    1357  INTR$1
 862D 0697E3        1358        CALL BUG01          Get the symbol table entry       3
                    1359  *  Above call fixes bug.       of the given variable
 8630 0F77          1360        XML  VPUSH          And save it on the stack         2
 8632 870C          1361        DCLR @BYTE          Assume no data available         2
 8634 C82BE0        1362        $IF @VARA+1 .L. RAM(CNT(PABPTR)) THEN Pick up data   3
 8637 090466
```

```
863A 43
863B BC0DB0  1363          ST   RAM(@TEMP5),@BYTE+1 Length byte first
863E 66
863F 9166    1364          DINC @TEMP5       Update both actual address
8641 F02B    1365          INC  @VARA+1        and offset
             1366        $END IF
8643 D64C65  1367        $IF @FAC2 .EQ. 265 THEN Has to be string variable
8646 4650
8648 BD500C  1368          DST  @BYTE,@FAC6 Set length of string
864B 0692DE  1369          CALL CTMPST       Create temporary string
864E 467E    1370        $ELSE
8650 D60D08  1371          $IF @BYTE+1 .NE. 8 GOTO ERRFE "* FILE ERROR"
8653 57D7
8655 340C4A  1372          MOVE @BYTE FROM RAM(@TEMP5) TO @FAC Copy value
8658 B066
865A 8F4A66  1373          $IF @FAC .DNE. 0 THEN Watch out for non-scaled stuf
865D 7C
865E BE5C51  1374            ST   FAC7,@ARG     Test for legal numeric
             1375            $REPEAT
8661 C690SC  1376              $IF *ARG .H. 99 GOTO ERRFE "* FILE ERROR"
8664 6377D7
8667 925C    1377              DEC  @ARG     Next digit for test
8669 D65C4B  1378            $UNTIL @ARG .EQ. FAC1
866C 4661
866E BD5C4A  1379            DST  @FAC,@ARG Copy in ARG for some testing
8671 815C    1380            DABS @ARG       Be sure we're positive
             1381  *       If first byte after expon. byte =0 : incorrect
             1382  *       normalization has occured : FILE ERROR
             1383  *       Or >99 : illegal numeric:FILE ERROR
8673 925D    1384            DEC  @ARG1      0 would cause underflow here
8675 C65D62  1385            $IF @ARG1 .H. 98 GOTO ERRFE
8678 77D7
867A 467E    1386          $ELSE
867C 874C    1387            DCLR @FAC2      Be sure FAC2 = 0 (no strings)
             1388          $END IF
             1389        $END IF
867E A166OC  1390        DADD @BYTE,@TEMP5  Update address and
8681 A02B0D  1391        ADD  @BYTE+1,@VARA+1     offset again
8684 0F7C    1392        XML  ASSGNV         Assign value to variable
8686 B6E003  1393        CLR  RAM(OFS(PABPTR))   Undo allocated offsets
8689 04
868A D642B3  1394        $IF @CHAT .EQ. COMMA$ THEN
868D 46AB
868F 0F79    1395          XML  PGMCHR       Get next text character
8691 0695AD  1396          CALL CHKEND       Check for end of statement
8694 669F    1397          BS   INTR$2       O.K. EOS is fine
8696 C82BE0  1398            $IF @VARA+1 .HE. RAM(CNT(PABPTR)) GOTO INTR$O
8699 090466
869C 1B
869D 462D    1399            BR   INTR$1     Still something left
             1400  INTR$2
869F C82BE0  1401          $IF @VARA+1 .L. RAM(CNT(PABPTR)) THEN
86A2 090466
86A5 AB
86A6 BCE003  1402            ST   @VARA+1,RAM(OFS(PABPTR)) Save value of o
```

```
86A9 0428
                1403          $END IF
                1404        $END IF
86AB 0F75       1405        XML  CONT           And CONTINUE                    2
                1406      $END IF
86AD 06928A     1407      CALL GETVAR           Collect variable list on stack  1
86B0 BD140A     1408      DST  @STADDR,@CURLIN  Save it in temp                 1
86B3 BF0AC8     1409      DST  CRNBUF,@RAMPTR   Initialize crunch buffer pointer 1
86B6 20
86B7 8607       1410      CLR  @RECLEN          Initialize field counter        1
86B9 BEE004     1411      ST   C$READ,RAM(COD(PABPTR)) Select READ operation    1
86BC 0402
86BE 8EE003     1412      $IF RAM(OFS(PABPTR)) .NE. 0 GOTO INP$31               1
86C1 0446E9
86C4 46CC       1413      BR   INP$3            Adjust for used record usage    1
                1414      $REPEAT
86C6 BEEFFF     1415        ST   COMMA$,RAM(-1(RAMPTR)) Fake legal separator    2
86C9 FF0AB3
                1416  INP$3
86CC 069765     1417        CALL IOCL$1         Get next input record           2
86CF 86E003     1418        CLR  RAM(OFS(PABPTR)) Reset offset within record    2
86D2 04
86D3 068827     1419        CALL RECENT         Compute record entry            2
86D6 BC2AE0     1420        ST   RAM(CNT(PABPTR)),@VARA Get record length       2
86D9 0904
86DB 8E2A66     1421        $WHILE @VARA .NE. 0                                 3
86DE E9
86DF A2B020     1422          ADD  OFFSET,RAM(@VARW)  Add video offset for normal3
86E2 60
86E3 9120       1423          DINC @VARW            screen-type crunch - proceed for3
86E5 922A       1424          DEC  @VARA            entire record.              3
86E7 46DB       1425        $END WHILE                                          2
                1426  INP$31
86E9 068827     1427        CALL RECENT         Compute actual record entry     2
86EC BC2BE0     1428        ST   RAM(CNT(PABPTR)),@VARA+1 Compute end of record 2
86EF 0904
86F1 862A       1429        CLR  @VARA          Make that a double byte         2
86F3 A12AE0     1430        DADD RAM(BUF(PABPTR)),@VARA  Add buffer start addr. 2
86F6 0604
86F8 932A       1431        DDEC @VARA          Point to last position in record 2
86FA 8611       1432        CLR  @VAR6          Assume no values input          2
86FC 0F7F       1433        XML  CRUNCH         Scan data fields as in DATA stmt 2
86FE 01         1434        DATA 1              Indicate input stmt crunch      2
86FF 8F2257     1435        $IF @ERRCOD .DNE. 0 GOTO  ERRINP If some crunch error2
8702 CF
8703 9011       1436          INC  @VAR6        Get correct # of fields (one off) 2
8705 A00711     1437          ADD  @VAR6,@RECLEN Update # of fields up to now   2
8708 C80710     1438      $UNTIL @RECLEN .HE. @VAR5 O.K...THAT'S ENOUGH !!!!!!!!!1
870B 46C6
870D 972C       1439      DDECT @PGMPTR         Backup program pointer          1
870F 0F79       1440      XML  PGMCHR           Re-inspect last token before EOL 1
8711 068827     1441      CALL RECENT           Precompute record entry         1
8714 86E003     1442      CLR  RAM(OFS(PABPTR))  Assume no pending record       1
8717 04
8718 D642B3     1443      $IF @CHAT .EQ. COMMA$ THEN Make record pending        2
```

```
8718 4752
871D D40710  1444        $IF @RECLEN .NE. @VAR5 THEN Enough left for pendin
8720 6752
8722 A40710  1445          SUB  @VAR5,@RECLEN Compute remaining # of fields
8725 A41107  1446          SUB  @RECLEN,@VAR6 # of fields used in last rec
             1447  INP$32
8728 D6B020  1448          $WHILE RAM(@VARW) .EQ. :":+OFFSET
872B 82473A
             1449            $REPEAT          Skip quoted strings
872E 9120    1450              DINC @VARW
8730 D6B020  1451            $UNTIL RAM(@VARW) .EQ. :":+OFFSET
8733 82472E
8736 9120    1452            DINC @VARW
8738 4728    1453          $END WHILE
             1454          $REPEAT            Search for Nth data item
873A 9120    1455            DINC @VARW       Update pointer
873C D6EFFF  1456          $UNTIL RAM(-1(VARW)) .EQ. :,:+OFFSET
873F FF208C
8742 473A
8744 9211    1457          DEC  @VAR6         Commas denote end of field
8746 4728    1458          BR   INP$32        Continue until done
8748 A520E0  1459          DSUB RAM(BUF(PABPTR)),@VARW  Compute current offset
8748 0604
874D BCE003  1460          ST   @VARW+1,RAM(OFS(PABPTR)) Store for next round
8750 0421
             1461        $END IF
             1462      $END IF
8752 BC1110  1463      ST   @VAR5,@VAR6       Copy # of variables for check
8755 BD0A14  1464      DST @CURLIN,@STADDR    Restore from temp
8758 4786    1465    $SELSE                   Now we're in for screen I/O
875A 06973F  1466      CALL INITKB            Initialize some variables for KB
875D BDA3AA  1467      DST @PGMPTR,RAM(INPUTP) Save for "try again" case
8760 2C
8761 BDA3BC  1468      DST @CCPPTR,RAM(CPTEMP) Save CCPPTR,RECLEN for "try agn
8764 06
             1469  INP$33
             1470  *                        Entry point for "try again" case
8765 068906  1471      CALL INSUB1          Put out prompt
             1472  INP$2
8768 06928A  1473      CALL GETVAR          Get variable list on V-stack
             1474  INPU$3
876B 06893B  1475      CALL INSUB2          Read from the screen
876E 8611    1476      CLR  @VAR6           Assume no values input
8770 0F7F    1477      XML  CRUNCH          Crunch the input line
8772 01      1478      DATA 1               Indicate input stmt scan
8773 BD0A14  1479      DST  @CURLIN,@STADDR Restore from temp
8776 8F2247  1480      $IF @ERRCOD .DNE. 0 GOTO WRNINP If got some crunch erro
8779 BB
877A 0F83    1481      XML  SCROLL          Scroll up after crunching
877C BE7F03  1482      ST   3,XPT           Reset XPT too - pending records
877F 9011    1483      INC  @VAR6           # fields = # commas + 1
8781 D41011  1484      $IF @VAR5 .NE. @VAR6 GOTO WRNINP # of variables wrong
8784 47BB
             1485    $END IF
             1486    $
```

```
                    1487    $     Once we're here, all information should be available
                    1488    $     After type verification for input and variables, we
                    1489    $     push all value entries on the V-stack.
                    1490    $     VAR6 = VAR5 = number of variables
                    1491    $
8786 BD1434         1492      DST   @DATA,@CURLIN       Save current DATA pointer
8789 BF3408         1493      DST   CRNBUF,@DATA        Get crunch entry
878C 20
878D BD020E         1494      DST   @VAR4,@MNUM         Get entry in V-stack before PUSH
                    1495    INPU$4
8790 A30200         1496      DADD  8,@MNUM             Point to first symbol table entry
8793 08
8794 BD06B0         1497      DST   RAM(@MNUM),@CCPPTR Get intermediate result
8797 02
8798 06930C         1498      CALL  GETRAM              Get value descriptor from RAM
879B DAB006         1499      $IF .BIT7 RAM(@CCPPTR) .EQ. 0 THEN Numerical value      1
879E 8047CF
87A1 0692EC         1500        CALL CHKNUM             Check entered value against numeri1
87A4 47B4           1501        BR   INPU$5             Found error                      1
87A6 8E1767         1502        $IF @DSRFLG .EQ. 0 GOTO INPU$6 Do not check overflow ini
87A9 D4
                    1503    *     file I/O, supply machine infinity with appropriate sign
                    1504    *     and contine
87AA 8EA3BA         1505        $IF RAM(CSNTP1) .EQ. 0 GOTO INPU$6 Watch out for over- 1
87AD 67D4
                    1506    *                                                 flow in screen I/O
87AF BD3414         1507        DST  @CURLIN,@DATA    Restore DATA pointer             1
87B2 47BF           1508        BR  WR$$5             Ask for input reenter            1
                    1509    INPU$5
87B4 8E1777         1510        $IF @DSRFLG .EQ. 0 GOTO ERRINP  FILE I/O IS FATAL      1
87B7 CF
87B8 BD3414         1511        DST  @CURLIN,@DATA    Restore DATA pointer on error    1
                    1512    WRNINP
87BB 066A82         1513        CALL WARN$$           Go here for simple warnings too  1
87BE 20             1514        DATA 32               INPUT ERROR - TRY AGAIN          1
                    1515    WR$$5
                    1516
87BF 06883A         1517        CALL SCR$              Scroll the screen and reset CCPAD1
87C2 BD2CA3         1518        DST RAM(INPUTP),@PGMPTR Restore ptr to "promp" if any  1
87C5 AA
87C6 BD06A3         1519        DST RAM(CPTEMP),@CCPPTR Restore CCPPTR,RECLEN for try a1
87C9 BC
87CA BD6E0E         1520        DST @VAR4,@VSPTR      Restore original stack ptr.      1
87CD 4765           1521        BR   INP$33                                            1
                    1522      $END IF
87CF 069323         1523      CALL CHKSTR             Check string input
87D2 67B4           1524      BS   INPU$5             ERROR...CHECK I/O TYPE
                    1525    INPU$6
87D4 06930C         1526      CALL GETRAM             Get separation character (RAM)
87D7 D601B3         1527      $IF @VAR0+1 .NE. COMMA$ THEN                           1
87DA 67E6
87DC 9211           1528        DEC  @VAR6            Has to be end of data            1
87DE 47B4           1529        BR   INPU$5           If not...ERROR                   1
87E0 8E0147         1530        $IF @VAR0+1 .NE. 0 GOTO INPU$5                         1
87E3 B4
```

```
87E4 47EA    1531    $SELSE                                                  1
87E6 9211    1532      DEC   @VAR6          Count number of value entries   1
87E8 4790    1533      BR    INPU$4         Continue                        1
             1534    $END IF
             1535    $
             1536    $     Assign cycle - assign values to variables
             1537    $     Because it rescans the program line, this code
             1538    $     can not be used for imperative statements, since
             1539    $     the crunch buffer get's destroyed on input.
             1540    $          The rescan is necessary because subscripts
             1541    $     should be evaluated AFTER all previous values
             1542    $     have been assigned, i.e.
             1543    $
             1544    $          INPUT I,A(I)   with values 2,3
             1545    $
             1546    $     Should assign value 3 to A(2) !!!!!!
             1547    $
             1548    $          No error-checking is done here, since types
             1549    $     are already validated.  We might get subscripts
             1550    $     out of range though !!!!
             1551    $
87EA BF3408  1552      DST   CRNBUF,@DATA   Prepare for input rescan
87ED 20
87EE BD2C0A  1553      DST   @STADDR,@PGMPTR  Restore token pointer for rescan
87F1 932C    1554      DDEC  @PGMPTR        Backup one token
87F3 BD6E0E  1555      DST   @VAR4,@VSPTR   Restore original stack pointer
             1556    INP$65
87F6 0F79    1557        XML   PGMCHR       Get next program characters
87F8 0695AD  1558        CALL  CHKEND       Might have , before EOS
87FB 6822    1559        BS    INPU$7
87FD 0697E3  1560        CALL  BUG01        Rescan variable name
             1561    * Above call fixes bug.    Get correct entry for arrays
8800 0F77    1562        XML   VPUSH        Save on stack for ASSGNV
8802 06930C  1563        CALL  GETRAM       Get first token of input value
8805 D64C65  1564        $IF @FAC2 .NE. STRVAL THEN Numerical case              1
8808 680F
880A 0692EC  1565          CALL CHKNUM      Check for numerical value          1
880D 6818    1566          BS   INP$67      COND should be set (valid numeric)1
             1567        $END IF
880F 069323  1568        CALL  CHKSTR       Get the correct string value
8812 BD0C50  1569        DST   @FAC6,@BYTE  Length for temporary string
8815 0692DE  1570        CALL  CTMPST       Create temporary string
             1571    INP$67
8818 0F7C    1572        XML   ASSGNV       Assign value to variable
881A 06930C  1573        CALL  GETRAM       Skip separator (already checked)
881D 0695AD  1574        CALL  CHKEND       Check for end of statement
8820 47F6    1575        BR    INP$65       Found it
             1576    INPU$7
8822 BD3414  1577      DST   @CURLIN,@DATA  Restore DATA pointer
8825 0F75    1578      XML   CONT           Continue in PARSE
             1579
             1580    RECENT
8827 BC21E0  1581      ST    RAM(OFS(PABPTR)),@VARW+1 Get record offset
882A 0304
882C 8620    1582      CLR   @VARW          Double byte value reqr'd.....
```

```
882E A120E0  1583    DADD RAM(BUF(PABPTR)),@VARW Got it.....................
8831 0604
8833 00      1584    RTN                         AND NOW, THE END IS NEAR.......
             1585
             1586  CHKRM
8834 C70802  1587    $IF @CCPADR .DH. SCRNBS+29 THEN not enough room for "?"   1
8837 FD4840
             1588  SCR$
883A 0F83    1589      XML  SCROLL               Scroll one line for "?"        1
883C BF0802  1590      DST  SCRNBS+2,@CCPADR  and update CCPADR accordingly     1
883F E2
             1591    $END IF
8840 00      1592    RTN
```

```
                    1594  ************************************************************
                    1595  *                  L I N P U T   R O U T I N E             *
                    1596  *                                                          *
                    1597  *       If file-I/O then                                   *
                    1598  *            Get file number and check it                  *
                    1599  *            Internal file not allowed                     *
                    1600  *       End if                                             *
                    1601  *       Get variable info                                  *
                    1602  *       Must be string variable                            *
                    1603  *       If file-I/O then                                   *
                    1604  *          If no-partial-record or REC clause included     *
                    1605  *            Read new record                               *
                    1606  *          End if                                          *
                    1607  *          Set up copy pointers                            *
                    1608  *       Else                                               *
                    1609  *          Call readln to read from keyboard               *
                    1610  *          Copy to crunch buffer adjusting for screen offset*
                    1611  *          Set up copy pointers                            *
                    1612  *       End if                                             *
                    1613  *       Get string of proper length                        *
                    1614  *       Move data into string                              *
                    1615  *       Assign string                                      *
                    1616  *       Done.                                              *
                    1617  ************************************************************
                    1618
  8841 06973F       1619  LINPUT CALL INITKB              Assume input from keyboard
  8844 D642FD       1620         $IF @CHAT .EQ. NUMBE$ THEN  If '#' - then device    1
- 8847 485C
  8849 069355       1621           CALL CHKFN            Check for default = 0       1
  884C 8F4A68       1622           $IF @FAC .DEQ. 0 GOTO LINP10  #0 is same as keyboa
  884F 5F
  8850 0688E8       1623           CALL INSU1            Parse the device #          1
                    1624  *        If internal file-then error
  8853 DAE005       1625           $IF .BIT3 RAM(FLG(PABPTR)) .EQ. 1 GOTO ERRFE      1
  8856 040857
  8859 D7
  885A 485F         1626         $SELSE                                              1
  885C 068906       1627           CALL INSUB1           Handle possible prompt      1
                    1628         $END IF
  885F BD0E6E       1629  LINP10 DST @VSPTR,@VAR4        Save original V-ptr
                    1630  *                             incase BREAK in READLN
  8862 0697E3       1631         CALL BUG01             Get info about the symbol
                    1632  * Above call fixes bug.       Get value pointer and type
  8865 D64C65       1633         $IF  @FAC2 .NE. STRVAL GOTO ERRMUV Must be string
  8868 57DF
  886A 0F77         1634         XML VPUSH              Push entry on the stack
  886C 8E1748       1635         $IF @DSRFLG .EQ. 0 THEN If device I/O               1
  886F AF
  8870 8EE003       1636           $IF RAM(OFS(PABPTR)) .EQ. 0 THEN  If new record   2
  8873 04487B
  8876 069765       1637             CALL IOCL$1         Read the record             2
  8879 4893         1638           $SELSE                Else partial left by INPUT  2
  887B BC0CE0       1639             ST   RAM(CNT(PABPTR)),@BYTE  Get length of rec  2
  887E 0904
  8880 BD66E0       1640             DST  RAM(BUF(PABPTR)),@TEMP5  Get addr of bu    r
```

```
8883 0604
8885 8E0C68  1641          $WHILE @BYTE .NE. 0   While chars in buffer     3
8888 93
8889 A68066  1642            SUB  OFFSET,RAM(@TEMP5) Remove INPUT's offset 3
888C 60
888D 9166    1643            DINC @TEMP5       Increment pointer          3
888F 920C    1644            DEC  @BYTE        Decrement count            3
8891 4885    1645          $SEND WHILE         Drop out directly when done 2
             1646        $END IF
8893 8666    1647        CLR  @TEMP5           Need a word value          1
8895 BC67E0  1648        ST   RAM(OFS(PABPTR)),@TEMP5+1 Restore offset    1
8898 0304
889A 860C    1649        CLR  @BYTE            Need a word value          1
889C BC0DE0  1650        ST   RAM(CNT(PABPTR)),@BYTE+1  Get the length    1
889F 0904
88A1 A50C66  1651        DSUB @TEMP5,@BYTE     Calculate length           1
88A4 A166E0  1652        DADD RAM(BUF(PABPTR)),@TEMP5 Current buffer addres1
88A7 0604
88A9 86E003  1653        CLR  RAM(OFS(PABPTR)) Read next record next time 1
88AC 04
88AD 48E1    1654      $SELSE                  Else if keyboard input     1
88AF 063933  1655        CALL INSUB2           Clear line and call READLN 1
88B2 870C    1656        DCLR @BYTE            Initialize byte counter    1
88B4 BD660A  1657        DST  @RAMPTR,@TEMP5   Initialize "crunch" pointer 1
88B7 D6B02A  1658        $IF  RAM(@VARA) .EQ. SPACE+OFFSET THEN If space  2
88BA 8048BF
88BD 932A    1659          DDEC @VARA          Don't include space on end 2
             1660        $END IF
88BF CD202A  1661        $WHILE @VARW .DLE. @VARA THEN  While not at end  2
88C2 68DC
88C4 BC00B0  1662          ST   RAM(@VARW),@VARO Get the character        2
88C7 20
88C8 D6007F  1663          $IF  @VARO .NE. EDGECH THEN  If not at edge char3
88CB 68D8
88CD A60060  1664            S    OFFSET,@VARO   Subtract screen offset   3
88D0 BCB00A  1665            ST   @VARO,RAM(@RAMPTR)  And put into crnbuf 3
88D3 00
88D4 910C    1666            DINC @BYTE         Count it                   3
88D6 910A    1667            DINC @RAMPTR       And update "crunch" pointer 3
             1668          $END IF
88D8 9120    1669          DINC @VARW           Update input pointer       2
88DA 48BF    1670        $SEND WHILE                                       1
88DC 0F83    1671        XML  SCROLL            Scroll the screen          1
88DE BE7F03  1672        ST   3,XPT             Initialize X-pointer       1
             1673      $END IF
88E1 0692DE  1674      CALL CTMPST              Create temporary string
88E4 0F7C    1675      XML  ASSGNV              Assign the value to it
88E6 0F75    1676      XML  CONT                And continue execution
```

```
                1678  *       Get file number and info about the file
                1679  INSU1
88E8 069371     1680          CALL CHKCON             Check & convert & search..
88EB 57D7       1681          BR   ERRFE              Give error if required
                1682  *
                1683  *       INPUT allowed for input and update modes
                1684  *
88ED DAE005     1685          $IF .BIT1 RAM(FLG(PABPTR)) .NE. 0 GOTO ERRFE
88F0 040257
88F3 D7
88F4 069390     1686          CALL OUTEOF             Output pending PRINT stuff
88F7 BEE004     1687          ST   C$READ,RAM(COD(PABPTR))  Ensure read operation
88FA 0402
88FC 0694C2     1688          CALL PARREC             Parse REC clause
88FF 0F7E       1689          XML  SPEED              Must be at a
8901 00         1690          DATA SYNCHK                colon else
8902 B5         1691          DATA COLON$                   its an error
8903 8617       1692          CLR  @DSRFLG            Clear keyboard input flag
8905 00         1693          RTN
```

```
                    1695  *       Parse and put out input prompt
                    1696
 8906 BD0A2C        1697  INSUB1 DST  @PGMPTR,@STADDR   Save pointer for prompt check
 8909 930A          1698         DDEC @STADDR          Backup to previous token
                    1699         $REPEAT               Go into a tight loop
 890B 06958A        1700           CALL NXTCHR         Get next program character   1
 890E 6927          1701           BS   INP$37         Detected end of stmt         1
 8910 D64285        1702         $UNTIL @CHAT .EQ. COLON$  Stop if we find a colon
 8913 490B
 8915 BD2C0A        1703         DST  @STADDR,@PGMPTR  Backup for actual prompt scan
 8918 0F79          1704         XML  PGMCHR           Jump into 1st char of prompt
 891A 0F74          1705         XML  PARSE            And try to decode string expr.
 891C B5            1706         DATA COLON$
 891D D64C65        1707         $IF @FAC2 .NE. STRVAL GOTO ERRSNM Num. prompt illegal
 8920 57BF
 8922 06970E        1708         CALL OSTRNG           Output the given prompt
 8925 4936          1709         BR   INP$39           Exit without prompt backup
 8927 BD2C0A        1710  INP$37 DST  @STADDR,@PGMPTR  Backup to beginning of line
 892A BE42B5        1711         ST   COLON$,@CHAT     Fake prompt with ":"
                    1712
 892D 068834        1713  INPU$2 CALL CHKRM            Check for room for ?
 8930 BEB008        1714         ST   :?:+OFFSET,RAM(@CCPADR) Display ?
 8933 9F
 8934 9508          1715         DINCT @CCPADR         Count it too
 8936 0F7E          1716  INP$39 XML  SPEED            Must be at a
 8938 00            1717         DATA SYNCHK            colon else
 8939 B5            1718         DATA COLON$               its an error
 893A 00            1719         RTN
```

```
                1721  *       Issue 'BEEP' and call readln to read from screen
893B 068834     1722  INSUB2  CALL  CHKRM             Check for room for answer
893E BD2008     1723          DST   @CCPADR,@VARW      Copy current cursor positi
                1724          $REPEAT
8941 BEB008     1725            ST    :  :+OFFSET,RAM(@CCPADR) Clear the remainder
8944 80
8945 9108       1726            DINC @CCPADR                 of the current line
8947 C30802     1727          $UNTIL @CCPADR .DHE. >2FE Stop if we're there
894A FE4941
894D BFA2FE     1728          DST   >7F7F,RAM(>2FE)    Replace edgechars
8950 7F7F
8952 8E80CE     1729          $IF @PRTNFN .EQ. 0 THEN If previous tone finished
8955 495A
8957 060034     1730            CALL TONE1             ------ "BEEP" ------
                1731          $END IF
895A C16E0E     1732          DEX   @VAR4,@VSPTR       Don't destroy V-stack on BREA
895D 066A76     1733          CALL  READLN             Input a line from the keyboar
8960 C16E0E     1734          DEX   @VAR4,@VSPTR       Restore V-stack pointer
8963 BD140A     1735          DST   @STADDR,@CURLIN    Save in a temp
8966 BF0A08     1736          DST   CRNBUF,@RAMPTR     Init crunch buffer pointer
8969 20
896A 00         1737          RTN
```

```
                1739  ************************************************************
                1740  *          A C C E P T   S T A T E M E N T                 *
                1741  *                                                          *
                1742  *      Accept input anywhere on the screen.  The           *
                1743  *      total number of input variables is limited to one.  *
                1744  *          On an ACCEPT AT( , ), the maximum number of     *
                1745  *      that can be accepted is up to the right margin !!!  *
                1746  *          If SIZE() is used, the maximum number is        *
                1747  *      limited to the given SIZE, or to the number of      *
                1748  *      characters remaining on the line, whichever is      *
                1749  *      the lesser.                                         *
                1750  ************************************************************
                1751  ACCEPT
896B B6A3B7     1752    CLR  RAM(ACCTRY)          Clear "try again" flag
896E 0694EF     1753    CALL DISACC               Use common code for DISPLAY/ACCEPT
8971 6109       1754    BS   ERRSYN                 COND set means end of statement
8973 BE63FF     1755    ST   >FF, @ARG7           Assume we don't have VALIDATE
                1756  *
                1757  *          V A L I D A T E   O P T I O N   H A N D L I N G
                1758  *
8976 D642FE     1759    $IF @CHAT .EQ. VALID$ THEN  Detected VALIDATE option      1
8979 49FD
897B 0F79       1760      XML  PGMCHR               Next character should start optioni
897D D642B7     1761      $IF @CHAT .NE. LPAR$ GOTO ERRSYN  "* SYNTAX ERROR "     1
8980 4109
8982 B60440     1762      SB   @PABPTR,6           Indicate usage of VALIDATE clause 1
8985 BF2A00     1763      DST  1,@VARA             Use VARA as length of option strini
8988 01
8989 8720       1764      DCLR @VARW               VARW= options used, VARW+1 = # of 1
                1765      $REPEAT                     stack entries for strings
898B 0F79       1766        XML PGMCHR             Skip separator token              2
898D CA42E8     1767        $IF @CHAT .HE. NUMER$ THEN Could be valid option      3
8990 49AA
8992 CA42EB     1768          $IF @CHAT .L. UALPH$+1 THEN It is...                  4
8995 69AA
8997 BE5C01     1769            ST   1,@ARG          Select bit 0 as number option 4
899A A642E8     1770            SUB NUMER$,@CHAT     Creat correct offset          4
899D 69A2       1771            BS   SETVW           Skip the shift stat.          4
899F E05C42     1772            SLL  @ARG,@CHAT Then select whatever option we s4
89A2 B4205C     1773  SETVW     OR   @ARG,@VARW      Remember options in VARW      4
89A5 0F79       1774            XML  PGMCHR          Get next token                4
89A7 0589C4     1775            B VLID$0             Must use a long branch here   4
                1776          $END IF
                1777        $END IF
89AA 0F74       1778        XML  PARSE               Try to decode a string expression 2
89AC B6         1779        DATA RPAR$                                             2
89AD D64C65     1780        $IF @FAC2 .NE. STRVAL GOTO ERRSNM String-number mism2
89B0 57BF
89B2 8E5169     1781        $IF @FAC7 .NE. 0 THEN    Only count non-null strings 3
89B5 C4
89B6 A02B51     1782          ADD  @FAC7,@VARA+1     Now watch out for overflow  3
89B9 0C49C0     1783          $IF .CARRY. THEN       String truncated            4
89BC 066A84     1784  ERRST     CALL ERR$$           * STRING TRUNCATED ERROR    4
89BF 13         1785            DATA 19                                          4
                1786          $END IF
```

```
89C0 0F77     1787            XML  VPUSH              Push the result for future re3
89C2 9021     1788            INC  @VARW+1         Count number of entries on st 3
              1789          $END IF
              1790  VLID$0
89C4 D6423B   1791      $UNTIL @CHAT .NE. COMMA$  Evaluate all fields            1
89C7 698B
89C9 0F7E     1792      XML  SPEED                                               1
89CB 00       1793      DATA SYNCHK                Check for ")" on end          1
89CC B6       1794      DATA RPAR$                 If not ..."* SYNTAX ERROR"    1
89CD 0694F2   1795      CALL DISP$1                Try to evaluate further optioi
89D0 6109     1796      BS   ERRSYN                Premature end of statement    1
89D2 BD0C2A   1797      DST  @VARA,@BYTE           Allocate string for character1
89D5 0F71     1798      XML   GETSTR                                             1
89D7 BD5C1C   1799      DST   @SREF,@ARG           Get start of allocated string1
89DA BCB05C   1800      ST    @VARW,RAM(@ARG)      Copy selected standard optioni
89DD 20
89DE 915C     1801      DINC  @ARG                 Leave room for standard optioi
89E0 8E2169   1802      $WHILE @VARW+1 .NE. 0      Copy all available informatio2
89E3 F3
89E4 0F78     1803        XML  VPOP                Regain stack-entry           2
89E6 3450B0   1804        MOVE @FAC6 FROM RAM(@FAC4) TO RAM(@ARG) Copy string   2
89E9 5CB04E
89EC A15C50   1805        DADD @FAC6,@ARG          Update destination address   2
89EF 9221     1806        DEC  @VARW+1             Count # of stack entries      2
89F1 49E0     1807      $SEND WHILE                and again drop out on 0 !!!!  1
89F3 BDA3B0   1808      DST  @SREF,RAM(VALIDP)     Copy start address of string i
89F6 1C
89F7 BDA3B2   1809      DST  @VARA,RAM(VALIDL)       and total string length     1
89FA 2A
89FB 8663     1810        CLR  @ARG7             Indicate VALIDATE usage for READLNi
              1811    $END IF
89FD BD2008   1812    DST  @CCPADR,@VARW         Save start address of the field
8A00 BD2A20   1813    DST  @VARW,@VARA           Set default highest address used
8A03 BD5E08   1814    DST  @CCPADR,@ARG2         Select absolute highest usable loc.
8A06 A35E01   1815    DADD  290,@ARG2            290=2+32*9 maximum of 254 character
8A09 22
8A0A C62BFC   1816    $IF @VARA+1 .H. >FC THEN Start at the end of line          1
8A0D 4A13
8A0F A35E00   1817      DADD  4, @ARG2                                           1
8A12 04
              1818    $END IF
8A13 8E046A   1819    $IF @PABPTR .NE. 0 THEN  We used some options like AT,SIZi
8A16 66
8A17 0F7E     1820      XML  SPEED                                               1
8A19 00       1821      DATA SYNCHK            Should always end on ":"          1
8A1A B5       1822      DATA COLON$                                              1
8A1B DA0402   1823      $IF .BIT1 @PABPTR .EQ. 1 THEN  Used BEEP clause          2
8A1E 6A23
8A20 060034   1824        CALL TONE1           Wake up the user                 2
              1825      $END IF                Continue with option evaluation
8A23 DA0404   1826      $IF .BIT2 @PABPTR .EQ. 1 THEN  Used AT option...SIZE!!   2
8A26 6A35
8A28 DA0408   1827        $IF .BIT3 @PABPTR .EQ. 0 THEN Use default SIZE optioni
8A2B 4A33
8A2D BE051C   1828          ST  VWIDTH,@PABPTR+1   Limit current record len      3
```

```
8A30 0695DF  1829          CALL SIZE1              Compute some intermediary dat3
             1830        $END IF
8A33 4A47    1831        BR  ACCP$1                                              2
             1832      $END IF
8A35 DA0408  1833      $IF .BIT3 @PABPTR .EQ. 1 THEN SIZE option used somewher2
8A38 6A66
             1834  *
             1835  *       We're sure now that SIZE has been used WITHOUT
             1836  *       the AT option...this means that we should
             1837  *       set XPT to point behind the SIZE field.
             1838  *       This can be done by adding the record length
             1839  *       to the current screen base address and
             1840  *       the line's screen base address
             1841  *
8A3A BC7F09  1842        ST  @CCPADR+1,@XPT  Start of with current address      2
8A3D A07F07  1843        ADD  @RECLEN,@XPT   Add in the current record length   2
8A40 A67FDF  1844        SUB  >DF,@XPT       And subtract the lower base addres2
             1845  *                        Also adjust for edge char.
8A43 BCA3B8  1846        ST @XPT,RAM(SIZXPT) Save it for "try again"case becaus2
8A46 7F
             1847  *                        in WARNING, XPT gets changed
             1848  ACCP$1
8A47 BDA3B4  1849          DST @CCPADR,RAM(SIZCCP) Save for "try again" case     2
8A4A 08
8A4B BCA3B6  1850          ST @RECLEN,RAM(SIZREC)  Save for "try again" case     2
8A4E 07
             1851  ********************************************************************
             1852  *ENTRY POINT FOR "TRY AGAIN" CASE WHEN SIZE OR ACCEPT USED
             1853  ********************************************************************
             1854  ACCP$9
8A4F DA0480  1855          $IF .BIT7 @PABPTR .EQ. 0 THEN  Blank current field    3
8A52 4A58
8A54 BEB008  1856            ST  :  :+OFFSET,RAM(@CCPADR)                       3
8A57 80
             1857          $END IF
8A58 9108    1858          DINC @CCPADR        Update screen address            2
8A5A 9207    1859          DEC  @RECLEN        Reduce count ...always at least on2
8A5C 4A4F    1860          BR   ACCP$9         Loop until at end of field       2
8A5E 9308    1861          DDEC @CCPADR        Fix end of field for maximum size 2
8A60 BD2A08  1862          DST  @CCPADR,@VARA  Set highest field address used    2
8A63 BD5E2A  1863          DST  @VARA,@ARG2    Also highest location available  2
             1864        $END IF               OK all set to go
             1865      $END IF
8A66 D6A3B7  1866      $IF RAM(ACCTRY) .EQ. 1 GOTO ACCP$7  Skip if in "try again"
8A69 016A74
8A6C BD0E6E  1867      DST @VSPTR,@VAR4        Save first entry in V-stack
8A6F 0697E3  1868      CALL BUG01              Collect the symbol designator
             1869  * Above call fixes bug.       Take care of arrays too
8A72 0F77    1870      XML  VPUSH              Save symbol table entry
             1871  ACCP$7
8A74 BDA3AC  1872      DST @VARW,RAM(ACCVRW)   Save for trying again case
8A77 20
8A78 BDA3AE  1873      DST @VARA,RAM(ACCVRA)   Save for trying again case
8A7B 2A
             1874  ********************************************************************
```

```
                    1875  *ENTRY POINT FOR "TRY AGAIN" WHEN NEITHER SIZE OR ACCEPT IS
                    1876  ************************************************************
                    1877  ACCP$5
                    1878  *  In case a CALL CLEAR or ERASE ALL or CALL HCHAR has just
                    1879  *  been processed, EDGE CHARS. are gone at the bottom line
8A7C DA040C         1880    CLOG >0C,@PABPTR        If AT/SIZE used, maximum field is 1
8A7F 4A86           1881    BR   A$1                line, so no need to worry about it
8A81 BFA2FE         1882    DST >7F7F,RAM(>2FE)     Put the EDGE CHARS. back
8A84 7F7F
                    1883  A$1
8A86 C10E6E         1884    DEX  @VSPTR,@VAR4       Don't destroy V-stack on BREAK
8A89 066A86         1885    CALL READL1             Ask for some input that can be used
8A8C C10E6E         1886    DEX  @VSPTR,@VAR4       Restore V-stack ptr
                    1887  *
                    1888  *       At this point, VARA contains the highest location
                    1889  *       used, and VARW contains the string's start address
                    1890  *
                    1891  ACCP$2
8A8F D52A20         1892    $IF @VARA .DNE. @VARW THEN Only non-empty string          1
8A92 6A9E
8A94 932A           1893      DDEC @VARA            Go to the next position         1
8A96 D6B02A         1894      $IF RAM(@VARA) .EQ. : :+OFFSET GOTO ACCP$2            1
8A99 806A8F
8A9C 912A           1895      DINC @VARA            Back to the last space          1
                    1896    $END IF
8A9E 0F78           1897    XML VPOP                Check the symbol designator is a
8AA0 0F77           1898    XML VPUSH               string or numeric variable
8AA2 D64C65         1899    $IF @FAC2 .NE. >65 THEN If numeric : empty string is n  1
8AA5 6AB2
                    1900  *                          allowed to be entered
8AA7 D5202A         1901      $IF @VARW .DEQ. @VARA THEN  If an empty string was ente2
8AAA 4AB2
8AAC 066A82         1902        CALL WARN$$           ***INPUT ERROR***             2
8AAF 20             1903        DATA 32                                             2
8AB0 4AFC           1904        BR  ACCP$8            Try again                     2
                    1905      $END IF
                    1906    $END IF
8AB2 870C           1907    DCLR  @BYTE             Compute length of input string
8AB4 BD1C20         1908    DST  @VARW,@SREF        Use SREF as temporary variable
8AB7 D51C2A         1909    $WHILE @SREF .DNE. @VARA                                  1
8ABA 6AC8
8ABC D6B01C         1910      $IF RAM(@SREF) .NE. EDGECH THEN Exclude edge character 2
8ABF 7F6AC4
8AC2 910C           1911        DINC @BYTE                                          2
                    1912      $END IF
8AC4 911C           1913      DINC @SREF            Decrement the counter           1
8AC6 4AB7           1914    $END WHILE
8AC8 0692D1         1915    CALL  CTSTRO            Create a temporary string
                    1916  ACCP$3
8ACB D5202A         1917    $WHILE @VARW .DNE. @VARA                                  1
8ACE 6AEB
8AD0 D6B020         1918      $IF RAM(@VARW) .EQ. EDGECH THEN Skip the edge character2
8AD3 7F4ADC
8AD6 A32000         1919        DADD 4, @VARW                                       2
8AD9 04
```

```
8ADA 4ACB     1920        BR  ACCP$3                                          2
              1921      $END IF
8ADC BCB01C   1922      ST RAM(@VARW),RAM(@SREF)  Copy the string             1
8ADF B020
8AE1 A6B01C   1923      SUB  OFFSET,RAM(@SREF)    Subtract the screen offset  1
8AE4 60
8AE5 9120     1924      DINC @VARW                Update pointers             1
8AE7 911C     1925      DINC @SREF                                            1
8AE9 4ACB     1926    $END WHILE                  Result can't be 0
8AEB D64C65   1927    $IF @FAC2 .NE. STRVAL THEN  Numerical variable          1
8AEE 6B2A
8AF0 BE4C65   1928      ST  STRVAL,@FAC2          Create temp string          1
8AF3 06A016   1929      CALL VALCD                Use VAL code for translation 1
8AF6 4B2A     1930      BR   ACCP$6               No error - o.k. go on       1
              1931  WRNSNM
8AF8 066A82   1932      CALL WARN$$               Error                       1
8AFB 07       1933      DATA 7                    STRING NO. MISMATCH         1
              1934  ACCP$8
8AFC DA0408   1935      $IF .BIT3 @PABPTR .EQ. 1 THEN  If SIZE is used        2
8AFF 6B0A
8B01 DA0404   1936        $IF .BIT2 @PABPTR .EQ. 0 THEN  Also AT is not used  3
8B04 4B0A
8B06 BC7FA3   1937          ST  RAM(SIZXPT),@XPT  Restore XPT: in WARNING XPT 3
8B09 B8
              1938  *                           got changed
              1939        $END IF
              1940      $END IF
8B0A BD20A3   1941      DST RAM(ACCVRW),@VARW     Restore @VARA,@VARW         1
8B0D AC
8B0E BD2AA3   1942      DST RAM(ACCVRA),@VARA                                 1
8B11 AE
8B12 BEA3B7   1943      ST 1,RAM(ACCTRY)          Set the "try again" flag    1
8B15 01
8B16 DA0408   1944      $IF .BIT3 @PABPTR .EQ. 0 THEN  If SIZE is not used    2
8B19 4B20
              1945  *   IF ACCEPT ALSO NOT USED, GOTO "TRY AGAIN" FROM HERE
8B1B DA0404   1946        $IF .BIT2 @PABPTR .EQ. 0 GOTO ACCP$5                2
8B1E 6A7C
              1947      $END IF                                               .
              1948  *   IF "EITHER SIZE OR ACCEPT IS USED" THEN
8B20 BD08A3   1949      DST RAM(SIZCCP),@CCPADR Restore CCPADR                1
8B23 B4
8B24 BC07A3   1950      ST RAM(SIZREC),@RECLEN  Restore RECLEN                1
8B27 B6
8B28 4A4F     1951      BR  ACCP$9              Go blanking the field and "try 1
              1952  *                             again"
              1953    $END IF
              1954  ACCP$6
8B2A 0F7C     1955    XML  ASSGNV               Should be o.k. now
              1956  *
              1957  *       Scroll the screen one line if no AT or SIZE
              1958  *
8B2C DA040C   1959    CLOG >C, @PABPTR          Test usage of AT and/or SIZE
8B2F 4B36     1960    BR  ACCP$4                At least one of the two used
8B31 0F83     1961    XML  SCROLL               Scroll the screen up
```

```
8B33 BE7F03   1962     ST 3,XPT                And reset XPT
              1963   ACCP$4
8B36 0F75     1964     XML  CONT
```

```
                  1966  ********************************************************************
                  1967  *          R E A D   S T A T E M E N T                             *
                  1968  *                                                                  *
                  1969  *      Assign DATA values to variables in READ-list                *
                  1970  *      one at a time.  Possibly search for new DATA                *
                  1971  *      statements in the program if the current DATA               *
                  1972  *      statement has been used.  Be careful with null              *
                  1973  *      entries.....!                                               *
                  1974  ********************************************************************
                  1975    $REPEAT
8B38 0F79         1976      XML  PGMCHR              Get character following ","          1
                  1977  READ
8B3A 0697E3       1978      CALL BUG01               Get pointers and correct entries    1
                  1979  * Above call fixes bug.       also allow for array variables
8B3D 0F77         1980      XML  VPUSH               Push on Vstack for assignment       1
8B3F 8E3477       1981      $IF @DATA .EQ. 0 GOTO ERRDAT  DATA ERROR                     1
8B42 D3
8B43 069304       1982      CALL GETGFL              Get next data item (RAM/GROM)       1
8B46 D64C65       1983      $IF @FAC2 .NE. STRVAL THEN                                   2
8B49 6B63
8B4B D601C8       1984        $IF @VAR0+1 .NE. NUM$ GOTO ERRSNM Not a numeric            2
8B4E 57BF
                  1985  *                              string-number mismatch error
8B50 06932F       1986        CALL CHKS$0            Build up string info                2
8B53 9150         1987        DINC @FAC6             Force legal delimiter on end        2
8B55 06A002       1988        CALL LITS05            Copy numeric into string space      2
8B58 BD561C       1989        DST  @SREF,@FAC12      Copy string start address           2
8B5B A11C50       1990        DADD @FAC6,@SREF       Compute end address of string       2
8B5E 931C         1991        DDEC @SREF             Back up over delimiter              2
8B60 06A012       1992        CALL CONVER.           Convert string to number            2
8B63 D5A390       1993        $IF RAM(CSNTMP) .DNE. @SREF GOTO ERRDAT WRONG !!!!!!!!2
8B66 1C57D3
8B69 4B73         1994      $SELSE                                                       2
8B6B 069323       1995        CALL CHKSTR            Check string input                  2
8B6E 77D3         1996        BS   ERRDAT           Give error on error                 2
8B70 06A002       1997        CALL LITS05            Allocate string in string space     2
                  1998      $SEND IF
8B73 0F7C         1999      XML  ASSGNV              Assign variable                     1
8B75 069304       2000      CALL GETGFL              Get next datum from DATA stmt       1
8B78 D601B3       2001      $IF @VAR0+1 .NE. COMMA$ THEN   Has to be an end of DATA2
8B7B 6B8F
8B7D 8E0157       2002        $IF @VAR0+1 .NE. 0 GOTO ERRDAT Check  for end of data2
8B80 D3
8B81 9736         2003        DDECT @LNBUF           Pointer to line # of DATA stmt      2
8B83 8634         2004        CLR   @DATA            Assume the worst - no more DATAs    2
                  2005        $  WATCH OUT FOR "DATA" STMT AT END OF PROGRAM
8B85 D53630       2006        $IF @LNBUF .DNE. @STLN THEN                                3
8B88 6B8F
8B8A 9336         2007          DDEC @LNBUF          Next line's 1st token address       3
8B8C 06A008       2008          CALL DATAST          Get next DATA statement             3
                  2009        $END IF
                  2010      $END IF
8B8F D642B3       2011    $UNTIL @CHAT .NE. COMMA$ Worry about junk in CONT
8B92 6B38
8B94 0F75         2012    XML  CONT
```

```
                    2013  *
                    2014  *       SRDATA-Search for DATA statements (DATA stat.
                    2015  *              must be the only stat. on one line)
                    2016  *       SEARCH-also uesed for searching IMAGE stat.
                    2017  *
                    2018  SRDATA
 8B96 BE4C93        2019    ST  DATA$,@FAC2          search for a DATA token
                    2020  SEARCH
 8B99 C12C34        2021    DEX  @DATA,@PGMPTR       exchange with normal PC
 8B9C C00142        2022    EX   @CHAT,@VARO+1       preserve current PGM character
 8B9F BE444B        2023    $IF @PRGFLG .EQ. 0      If imperative statement
 8BA2 B3
 8BA3 BE8084        2024      $IF @RAMTOP .NE. 0    With ERAM: text itself in ERAM
 8BA6 6BB3
 8BA8 BE8089        2025        ST  >FF,@RAMFLG     Fake RAMFLG in this case
 8BAB FF
 8BAC 0F79          2026        XML PGMCHR          Get first char. on the line
 8BAE 868089        2027        CLR  @RAMFLG        Restore it back
 8BB1 4BB5          2028        BR   SRDA$1         Skip that PGMCHR
                    2029      $END IF
                    2030    $END IF
 8BB3 0F79          2031    XML  PGMCHR             get first character on the line
                    2032  SRDA$1
 8BB5 D4424C        2033    $IF @CHAT .EQ. @FAC2 GOTO SRDA$0 search for specific token
 8BB8 6BBD
 8BBA D40000        2034    CEQ @0,@0               set COND if no DATA found
                    2035  SRDA$0
 8BBD C12C34        2036    DEX @DATA,@PGMPTR       exchanges won't  affect the CON
 8BC0 C00142        2037    EX  @CHAT,@VARO+1       situation o.k.
 8BC3 01            2038    RTNC                    return to caller with COND
```

```
                2040  ****************************************************************
                2041  *                    O L D   S T A T E M E N T                 *
                2042  *     A normal load:                                           *
                2043  *       Get a program from an external device to VDP and       *
                2044  *       reinitialize the program pointers.  Also               *
                2045  *       update the line pointer table, since the               *
                2046  *       memory size of the machine on which the program        *
                2047  *       was created doesn't have to be the same as on the      *
                2048  *       current system !!!!!! Then check if ERAM existed,      *
                2049  *       move it to ERAM if does exist (in relocated form)      *
                2050  *     Load a sequential file :                                 *
                2051  *       When program is bigger than 13.5K and ERAM exists,     *
                2052  *       maximum-length record reads are performed to           *
                2053  *       read the file and each record is copied into the       *
                2054  *       ERAM as it is read.                                    *
                2055  ****************************************************************
                2056  OLD
8BC4 068BCA     2057    CALL OLD1                   Make OLD1 a subroutine for LOAD
8BC7 056012     2058    B  TOPL15                   Go back to top level
                2059  OLD1
8BCA 069235     2060    CALL GPNAME                 Get program name & reinitialize ^s
8BCD 0F79       2061    XML  PGMCHR                 Check for EOL
8BCF BD0A04     2062    DST  @PABPTR,@STADDR        Compute memory start address
8BD2 A10AE0     2063    DADD RAM(NLEN-1(PABPTR)),@STADDR Add PAB-name length
8BD5 0C04
8BD7 A30A00     2064    DADD PABLEN-4,@STADDR                and PAB length
8BDA 0A
8BDB BDE00A     2065    DST  @>70,RAM(RNM(PABPTR))  Compute # of available bytes
8BDE 0470
8BE0 A5E00A     2066    DSUB @STADDR,RAM(RNM(PABPTR))
8BE3 040A
8BE5 91E00A     2067    DINC RAM(RNM(PABPTR))  Include current address
8BE8 04
8BE9 BDE006     2068    DST  @STADDR,RAM(BUF(PABPTR)) for copy start
8BEC 040A
8BEE BEE004     2069    ST   C$LOAD,RAM(COD(PABPTR)) Select LOAD I/O code
8BF1 0405
8BF3 069768     2070    CALL CDSR                   Call device service routine
8BF6 4C63       2071    BR   OLD$3                  Not a program file, may be a
                2072  *                             sequential file
                2073    $    STADDR still points to the info bytes
8BF8 BD02E0     2074    DST  RAM(2(STADDR)),@MNUM   First test checksum
8BFB 020A
8BFD B902E0     2075    DXOR RAM(4(STADDR)),@MNUM          which is a simple XOR
8C00 040A
8C02 D5B00A     2076    $IF  RAM(@STADDR) .DNE. @MNUM THEN Try PROTECTION option
8C05 026C15
8C08 8302       2077      DNEG @MNUM
8C0A D5B00A     2078      $IF RAM(@STADDR) .DNE. @MNUM GOTO OLDER No-ERROR
8C0D 024D2D
8C10 B64580     2079      SB @FLAG,7          Yes-Set LIST/EDIT PROTECTION flag
8C13 4C17       2080    $SELSE
8C15 8645       2081      CLR @FLAG           Otherwise clear protection.
                2082    $END IF
8C17 BD32E0     2083    DST  RAM(2(STADDR)),@ENLN   Copy new ENLN,
```

```
8C1A 020A
8C1C BD30E0   2084    DST  RAM(4(STADDR)),@STLN            STLN and
8C1F 040A
8C21 BDA3BC   2085    DST  RAM(6(STADDR)),RAM(OLDTOP)      top of memory info
8C24 E0060A
8C27 A30A00   2086    DADD 8,@STADDR          Point to program data
8C2A 08
8C2B BDA3BE   2087    DST  @>70,RAM(NEWTOP)   Set up the new top
8C2E 70
8C2F 068D36   2088    CALL RELOCA             Relocate according to @>70
              2089  *******
              2090  OLD$5
8C32 8E8084   2091    $IF @RAMTOP .EQ. 0 GOTO LRTOP$ ERAM present ?
8C35 6D32
              2092  *                               No-Go back to toplevel
              2093  *                               Yes-move from VDP to ERAM (in relo-
              2094  *                               cated form)
              2095  ********************Move to the ERAM from CPUBAS first********
8C37 BD0070   2096    DST @>70,@VARO
8C3A A50030   2097    DSUB @STLN,@VARO
8C3D 9100     2098    DINC @VARO              # of bytes to move
8C3F BD4E00   2099    DST  @VARO,@CCC         @CCC : Byte count for VGWITE
8C42 BF50A0   2100    DST CPUBAS,@BBB         @BBB : Destination addr. on ERAM
8C45 40
8C46 BD0A50   2101    DST @BBB,@STADDR        For later use as the base of curent
              2102  *                         program image in RELOCA
8C49 BD4C30   2103    DST @STLN,@AAA          @AAA : Source addr. on VDP
8C4C 0F8A     2104    XML VGWITE              Move from VDP to ERAM
8C4E BDA3BC   2105    DST @>70,RAM(OLDTOP)    Set up old memory top
8C51 70
8C52 BDA3BE   2106    DST @RAMTOP,RAM(NEWTOP) Set up new memory top
8C55 8084
8C57 068D36   2107    CALL RELOCA             Relocate the program image
              2108  OLD$7
8C5A BD8086   2109    DST @STLN,@RAMFRE       Reset the RAMFRE on ERAM
8C5D 30
8C5E 938086   2110    DDEC @RAMFRE
8C61 4D32     2111    BR  LRTOP$              Go back to toplevel
              2112  ************************************************************
              2113  *    At this point : if ERAM not exist - ERROR off          *
              2114  *      else open sequential file to load program to ERAM    *
              2115  *      through VDP RAM                                      *
              2116  ************************************************************
              2117  OLD$3
8C63 8E8084   2118    $IF @RAMTOP .EQ. 0 GOTO OLDER
8C66 6D2D
              2119  *   Set up PAB for OPEN
              2120  *   File type : Sequential file,
              2121  *   Mode of operation : Input
              2122  *   Data type : internal
              2123  *   Record type : Variable length records
              2124  *   Logical record length : 254 maximum
8C68 310009   2125    MOVE 9 FROM ROM(PAB3) TO RAM(4(PABPTR)) Build the PAB
8C6B E00404
8C6E 8D24
```

```
8C70 BD4A70  2126     DST @>70,@FAC          Compute the data buffer address
8C73 A74A00  2127     DSUB 253,@FAC
8C76 FD
8C77 BD4C4A  2128     DST @FAC,@AAA          Save it for later use in VGWITE
8C7A BDE006  2129     DST @FAC,RAM(BUF(PABPTR))
8C7D 044A
8C7F 06976B  2130     CALL CDSR               Call the device service routine
8C82 5794    2131     BR ERR$2B            Return with ERROR indication in COND.
             2132  *    Start to read in file
8C84 06975E  2133     CALL IOCALL            Read in the first record
8C87 02      2134     DATA C$READ
             2135  *    Check the control information
8C88 D6E009  2136     $IF RAM(CNT(PABPTR)) .NE. 10 GOTO OLDER 10 bytes contr.inf
8C8B 040A4D
8C8E 2D
             2137  *    >ABCD is the flag set at SAVE time indicating a program
             2138  *    file
8C8F D7B04A  2139     $IF RAM(@FAC) .DNE. >ABCD GOTO OLDER
8C92 ABCD4D
8C95 2D
8C96 954A    2140     DINCT @FAC
8C98 BD30B0  2141     DST RAM(@FAC),@STLN    Copy the new STLN
8C9B 4A
8C9C 954A    2142     DINCT @FAC
8C9E BD32B0  2143     DST RAM(@FAC),@ENLN    ENLN too
8CA1 4A
8CA2 BD0232  2144     DST @ENLN,@MNUM        Test checksum
8CA5 B90230  2145     DXOR @STLN,@MNUM
8CA8 954A    2146     DINCT @FAC
8CAA D5B04A  2147     $IF RAM(@FAC) .DNE. @MNUM THEN Try PROTECTION option     1
8CAD 026CBD
8CB0 8302    2148       DNEG @MNUM                                            1
8CB2 D5B04A  2149       $IF RAM(@FAC) .DNE. @MNUM GOTO OLDER No-ERROR         1
8CB5 024D2D
8CB8 B64580  2150       SB @FLAG,7           Yes-set LIST/EDIT PROTECTION flag 1
8CBB 4CBF    2151     $SELSE                                                  1
8CBD 8645    2152       CLR  @FLAG           Otherwise clear protection flag.  1
             2153     $END IF
8CBF 954A    2154     DINCT @FAC
             2155  *  Check is there enough memory in ERAM
8CC1 BD02B0  2156     DST RAM(@FAC),@MNUM    Get the old top of memory out
8CC4 4A
8CC5 BDA3BC  2157     DST @MNUM,RAM(OLDTOP)  For later use in RELOCA
8CC8 02
8CC9 A50230  2158     DSUB @STLN,@MNUM
8CCC 9102    2159     DINC @MNUM             Total # of bytes in program
8CCE BD0802  2160     DST @MNUM,@CCC1        For later use as the byte count
8CD1 A302A0  2161     DADD CPUBAS,@MNUM      Add the total # of bytes to CPUBAS
8CD4 40
             2162  *      Check if enough memory in ERAM
8CD5 0A6D2D  2163     $IF .GT. GOTO OLDER    Greater than >ffff case
8CD8 C50280  2164     $IF @MNUM .DH. @RAMTOP GOTO OLDER Greater than >dfff case
8CDB 846D2D
             2165  *        Move to ERAM starting from CPUBAS first,
             2166  *        then relocate according the new top of memory inERAM
```

```
                2167  OLD$$
8CDE BF50A0     2168    DST CPUBAS,@BBB         @BBB : Destination addr. in ERA
8CE1 40
                2169  *                                 for VGWITE
8CE2 BD0A50     2170    DST @BBB,@STADDR        For later use as base of the
                2171  *                         current program image in RELOCA
                2172  * DST @>70,@AAA           @AAA has been set up before
                2173  * DSUB 253,@AAA           For copy start on VDP RAM
                2174  * @CCC1 : Total # of bytes to move to ERAM, set up above
8CE5 06975E     2175    CALL IOCALL             Read in the second record
8CE8 02         2176    DATA C$READ
                2177    $REPEAT                 Read in the file and each record
                2178  *  Should be a full(maximum length 254) record at this time,
                2179  *  because program supposed to be bigger than 13.5K
8CE9 D6E009     2180      $IF RAM(CNT(PABPTR)) .NE. 254 GOTO OLDER                i
8CEC 04FE4D
8CEF 2D
8CF0 BF4E00     2181      DST 254,@CCC          @CCC:# of bytes to move           1
8CF3 FE
8CF4 0F8A       2182      XML VGWITE            Move data from VDP RAM to ERAM    1
8CF6 A35000     2183      DADD 254,@BBB         Update the destination addr.on ERA1
8CF9 FE
8CFA A70800     2184      DSUB 254,@CCC1        # of bytes left to move           1
8CFD FE
8CFE 6D16       2185      BS   OLD$9            No more bytes to move            1
8D00 06975E     2186      CALL IOCALL           Read in the file and each record i1
8D03 02         2187      DATA C$READ           copied into the ERAM as it is readi
8D04 CB0800     2188    $UNTIL @CCC1 .DL. 254   Leave the last record alone
8D07 FE6CE9
                2189  *  The record length should be the same as the # of bytes
                2190  *  left to move at this time
8D0A D4E009     2191    $IF RAM(CNT(PABPTR)) .NE. @CCC1+1 GOTO OLDER
8D0D 04094D
8D10 2D
8D11 BD4E08     2192    DST @CCC1,@CCC          Set up byte count for the last read
8D14 0F8A       2193    XML VGWITE              Move data from VDP RAM to ERAM
                2194  OLD$9
8D16 06975E     2195    CALL IOCALL             close the file
8D19 01         2196    DATA C$CLOS
8D1A BDA3BE     2197    DST @RAMTOP,RAM(NEWTOP) New top of memory
8D1D 8084
                2198  * RAM(OLDTOP) : old top of memory, set up above
                2199  * @STADDR : base of current program image in ERAM,set above
8D1F 068D36     2200    CALL RELOCA             Relocate the program
8D22 4C5A       2201    BR OLD$7                Go to set the RAMFRE and back to
                2202  *                         toplevel
                2203  PAB3
8D24 001C00     2204    DATA 0,>1C,#0,254,0,#0,OFFSET
8D27 00FE00
8D2A 000060
                2205    $
                2206    $     OLD error exit code...don't kill machine...
                2207    $
                2208  OLDER
8D2D 066014     2209    CALL INITPG             Initialize program space
```

```
8D30 5797     2210    BR   ERR$2              And take error exit
              2211  LRTOP$
8D32 066022   2212    CALL KILSYM             Release string space/symbol table
8D35 00       2213    RTN
              2214  ***************************************************************
              2215  *  RELOCATE THE PROGRAM IMAGE ACCORDING TO THE NEW TOP OF      *
              2216  *          STLN : old stln                                MEMORY  *
              2217  *          ENLN : old enln                                        *
              2218  *          RAM(OLDTOP) : old top of memory                        *
              2219  *          RAM(NEWTOP) : new top of memory                        *
              2220  *          @STADDR : current base for the old image               *
              2221  ***************************************************************
              2222  RELOCA
8D36 BDA3B4   2223    DST @PABPTR,RAM(SIZCCP) Save in temp.
8D39 04
8D3A BD02A3   2224    DST RAM(OLDTOP),@MNUM  Get the old top of memory
8D3D BC
8D3E BD04A3   2225    DST RAM(NEWTOP),@PABPTR Get the new top of memory
8D41 BE
8D42 A53202   2226    DSUB @MNUM,@ENLN        Compute ENLN relative to top (-)
8D45 A53002   2227    DSUB @MNUM,@STLN           "     STLN     "    "   "  (-)
8D48 A50A30   2228    DSUB @STLN,@STADDR      Highest memory address used
8D4B 8702     2229    DCLR @MNUM              Total # of bytes to be moved
8D4D A50230   2230    DSUB @STLN,@MNUM        STLN = - (# of bytes - 1)
8D50 9102     2231    DINC @MNUM              Take care of that one
8D52 A13204   2232    DADD @PABPTR,@ENLN      Compute new address of ENLN
8D55 A13004   2233    DADD @PABPTR,@STLN             and STLN
              2234  *     @PABPTR : destination addr., @STADDR : source addr.
8D58 BD5C02   2235    DST @MNUM,@ARG          @ARG : byte count
8D5B BD000A   2236    DST @STADDR,@VAR0       @VAR0 : source addr. for MVDN
8D5E BD1006   2237    DST @CCPPTR,@VAR5       Save in tem. (CCPPTR,VARY2 EQU >06)
8D61 BD0604   2238    DST @PABPTR,@VARY2      @VARY2 : desitination addr. for MVDN
8D64 D5A3BE   2239    $IF RAM(NEWTOP) .DEQ. @RAMTOP THEN Relocate the program  1
8D67 80844D
8D6A 6F
              2240  *                             in ERAM
8D6B 0F88     2241      XML MVDN                  Move from lower memory to higher  1
              2242  *                             memory one byte at a time
8D6D 4D7E     2243    $SELSE                                                      1
8D6F 87A3B6   2244      DCLR RAM(SIZREC)          Clear a temporary var             1
8D72 C1A3B6   2245      DEX @RAMTOP,RAM(SIZREC)   Save the RAMTOP, also fake as1
8D75 8084
              2246  *                             if ERAM not exist for MVDN in this
8D77 0F88     2247      XML  MVDN                 Move in VDP                       1
8D79 C1A3B6   2248      DEX @RAMTOP,RAM(SIZREC) Restore RAMTOP                      1
8D7C 8084
              2249    $END IF
8D7E BD0610   2250    DST @VAR5,@CCPPTR         Restore back
              2251    $
              2252    $    Update line # links according to new size
              2253    $
8D81 BD02A3   2254    DST RAM(OLDTOP),@MNUM  Old memory top
8D84 BC
8D85 A502A3   2255    DSUB RAM(NEWTOP),@MNUM Stop if sizes are same
8D88 BE
```

```
8D89 6DB3     2256    BS  RELO$1
8D8B BD0A30   2257    DST  @STLN,@STADDR      Start relocations at STLN
              2258  OLD$2
8D8E C9320A   2259    $IF @ENLN .DL. @STADDR GOTO RELO$1 and continue up to ENLN
8D91 4DB3
8D93 950A     2260    DINCT @STADDR           Skip the line #
8D95 D4A3BE   2261    $IF RAM(NEWTOP) .EQ. @RAMTOP THEN  If in ERAM          1
8D98 80844D
8D9B AB
8D9C 0691A4   2262      CALL GRSUB2           Read the link out              1
8D9F 0A       2263      DATA STADDR                                          1
8DA0 A55802   2264      DSUB @MNUM,@EEE1      Update it                      1
8DA3 066036   2265      CALL GWSUB            Write it back                  1
8DA6 0A5802   2266      DATA STADDR,EEE1,2                                   1
8DA9 4DAF     2267    $SELSE                                                 1
8DAB A5B00A   2268      DSUB @MNUM,RAM(@STADDR)     Update the link          1
8DAE 02
              2269    $END IF
8DAF 950A     2270    DINCT @STADDR           Skip the link...next line #
8DB1 4D8E     2271    BR   OLD$2              And continue until done
              2272  RELO$1
8DB3 BD04A3   2273    DST RAM(SIZCCP),@PABPTR Restore from temp.
8DB6 B4
8DB7 00       2274    RTN
```

```
                2276  ************************************************************
                2277  *              S A V E   S T A T E M E N T                 *
                2278  *                                                          *
                2279  *  SAVE "NAME",MERGE :Save in crunched form of a program   *
                2280  *      into a file one line at a time with the ln #.       *
                2281  *      File opened with sequential accessed, variable-len  *
                2282  *      records (161 max), display type & outpt mode,       *
                2283  *      move one ln # and one ln text to the crunch         *
                2284  *      buffer then write to the file one ln at a time      *
                2285  *   A normal SAVE : When ERAM not exist or the size of     *
                2286  *      the program and ln # table in ERAM can fit in VDP   *
                2287  *      (can be moved into VDP from ERAM once), then        *
                2288  *      the save statement saves a program image to an      *
                2289  *      external device, including all the information      *
                2290  *      the system needs for rebuilding the program image on*
                2291  *      a machine with a different memory size..also include*
                2292  *      is a checksum for rudimetary error checking and for *
                2293  *      PROTECTION VIOLATION                                *
                2294  *   A sequential SAVE : Maximum-length records are         *
                2295  *      performed to write the file and each record is      *
                2296  *      copied into the VDP from ERAM before it is written  *
                2297  ************************************************************
                2298  SAVE
8DB8 DA4580     2299    $IF .BIT7 @FLAG .EQ. 1 GOTO ERRPV *PROTECTION VIOLATION
8DBB 57DB
8DBD 069235     2300    CALL GPNAME             This will also close all files
                2301  *    Check SAVE "NAME",MERGE or SAVE "NAME",PROTECTED first
8DC0 86A3B9     2302    CLR RAM(SAPROT)         Clear "PROTECTED" flag
8DC3 0F79       2303    XML PGMCHR
8DC5 8E42       2304    CZ  @CHAT               EOL ?
8DC7 6E2B       2305    BS SA$1                 Yes- no need to check any option
8DC9 D642B3     2306    $IF @CHAT .NE. COMMA$ GOTO ERRSYN Has to be a comma here
8DCC 4109
8DCE D7B02C     2307    $IF RAM(@PGMPTR) .DEQ. >C805 THEN  Unquoted string with  1
8DD1 C8054D
8DD4 F4
                2308  *                       length 5 : has to be MERGE at this time
8DD5 D7E002     2309      $IF RAM(2(PGMPTR)) .DNE. :ME: GOTO ERRSYN  If not:SYNTA1
8DD8 2C4D45
8DDB 4109
8DDD D7E004     2310      $IF RAM(4(PGMPTR)) .DNE. :RG: GOTO ERRSYN         ERROR 1
8DE0 2C5247
8DE3 4109
8DE5 D6E006     2311      $IF RAM(6(PGMPTR)) .NE. :E: GOTO ERRSYN If not:SYNTAX  1
8DE8 2C4541
8DEB 09
8DEC 8EE007     2312      CZ RAM(7(PGMPTR))   Check for EOL                 ERROR 1
8DEF 2C
8DF0 4109       2313      BR ERRSYN           Not EOL: SYNTAX ERROR             1
8DF2 4F76       2314      BR SAVMG            Go to handle this option          1
                2315    $END IF
                2316  * Has to be PROTECTED option here, crunched as unquoted str.
8DF4 D7B02C     2317    $IF RAM(@PGMPTR) .DNE. >C809 GOTO ERRSYN   SYNTAX ERROR
8DF7 C80941
8DFA 09
```

```
8DFB D7E002  2318    $IF RAM(2(PGMPTR)) .DNE. :PR: GOTO ERRSYN
8DFE 2C5052
8E01 4109
8E03 D7E004  2319    $IF RAM(4(PGMPTR)) .DNE. :OT: GOTO ERRSYN
8E06 2C4F54
8E09 4109
8E0B D7E006  2320    $IF RAM(6(PGMPTR)) .DNE. :EC: GOTO ERRSYN
8E0E 2C4543
8E11 4109
8E13 D7E008  2321    $IF RAM(8(PGMPTR)) .DNE. :TE: GOTO ERRSYN
8E16 2C5445
8E19 4109
8E1B D6E00A  2322    $IF RAM(10(PGMPTR)) .NE. :D: GOTO ERRSYN
8E1E 2C4441
8E21 09
8E22 8EE00B  2323    CZ  RAM(11(PGMPTR)) Check EOL
8E25 2C
8E26 4109    2324    BR  ERRSYN              No-SYNTAX ERROR
8E28 90A3B9  2325    INC RAM(SAPROT)         Set PROTECTION flag
             2326  SA$1
             2327  ************************************************************
8E2B 8E8084  2328    $IF @RAMTOP .EQ. 0 THEN     If ERAM NOT present then     1
8E2E 4E42
             2329  ************CLEAR THE BREAKPT IN VDP ALONE TO SPEED UP ***
8E30 BD5230  2330      DST @STLN,@FAC8       End of ln # buffer                1
             2331      $REPEAT
8E33 B2B052  2332        RB  RAM(@FAC8),7    Clear the breakpt                 2
8E36 7F
8E37 A35200  2333        DADD 4,@FAC8        Move to the next one              2
8E3A 04
8E3B C55232  2334      $UNTIL @FAC8 .DH. @ENLN  Until done                       1
8E3E 4E33
8E40 4E69    2335    $SELSE                                                    1
8E42 06A020  2336      CALL UBSUB            Clear the breakpt in ERAM         1
8E45 BD0280  2337      DST  @RAMTOP,@MNUM    Top of memory in ERAM             1
8E48 84
8E49 A50230  2338      DSUB @STLN,@MNUM                                        1
8E4C 9102    2339      DINC @MNUM            # of bytes total in ERAM          1
8E4E BD0070  2340      DST @>70,@VARO        Top of memory in VDP              1
8E51 A50002  2341      DSUB @MNUM,@VARO                                        1
8E54 9100    2342      DINC @VARO                                              1
             2343  *                        Check is there enough memory in VDP
             2344  *                        to move the program text and ln #
             2345  *                        table from ERAM to VDP
8E56 0A4EB7  2346      $IF .NOT. .GT. GOTO GSAVE Not enough memory in VDP for 1
             2347  *                                                      sure
8E59 BF100A  2348      DST  VRAMVS+64+256,@VAR5 64 bytes are for savety bufferi
8E5C 98
             2349  * DSR routine gives file error when loading a program which
             2350  * has VDP maximum size and was saved from VDP to be a progra
             2351  * file on disk when ERAM not exist.
             2352  * In order to fix this problem, restrict the program memory
             2353  * to be 256 bytes less than the real space in VDP when ERAM
             2354  * not exist
8E5D C90010  2355      $IF @VARO .DL. @VAR5 GOTO GSAVE  Not enough memory        1
```

```
8E60 4EB7
                2356  *                               VDP, do a sequential file save
8E62 A71000     2357      DSUB 10,@VAR5               10 bytes for control inform.    1
8E65 0A
                2358  *                               Destination addr. on VDP for GVMOV
8E66 068F4A     2359      CALL GVMOV                  Enough memory in VDP, move it over1
                2360  *                               and do the normal save later
                2361    $END IF
                2362  *************** Without ERAM, or after GVMOV :***************
                2363  *************** do the normal save      **********************
                2364  VSAV$
8E69 BD0A40     2365    DST  @FREPTR,@STADDR    Store additional control info
8E6C 930A       2366    DDEC @STADDR            Back up some more for 2 byte save
8E6E BDB00A     2367    DST  @>70,RAM(@STADDR)  First current top of memory
8E71 70
8E72 970A       2368    DDECT @STADDR
8E74 BDB00A     2369    DST  @STLN,RAM(@STADDR) Then STLN
8E77 30
8E78 970A       2370    DDECT @STADDR
8E7A BDB00A     2371    DST  @ENLN,RAM(@STADDR) Then ENLN
8E7D 32
8E7E 970A       2372    DDECT @STADDR            Then
8E80 BDB00A     2373    DST  @STLN,RAM(@STADDR)
8E83 30
8E84 B9B00A     2374    DXOR @ENLN,RAM(@STADDR)      STLN XORed with ENLN
8E87 32
8E88 D6A3B9     2375    $IF RAM(SAPROT) .EQ. 1 THEN Check is there PROTECTED optil
8E8B 014E91
8E8E 83B00A     2376      DNEG RAM(@STADDR)     Negate the CHECKSUM to indicate   1
                2377    $END IF                 LIST/EDIT protection
8E91 BDE006     2378    DST  @STADDR,RAM(BUF(PABPTR)) Save start address in PAB
8E94 040A
8E96 930A       2379    DDEC @STADDR
8E98 BDE00A     2380    DST  @>70,RAM(RNM(PABPTR))  Compute # of bytes used
8E9B 0470
8E9D A5E00A     2381    DSUB @STADDR,RAM(RNM(PABPTR))  and store that in PAB too
8EA0 040A
8EA2 8E8084     2382    $IF @RAMTOP .NE. 0 THEN   If ERAM exists then             1
8EA5 6EAD
8EA7 BD300C     2383      DST @BBB1,@STLN       Restore the original STLN,ENLN    1
8EAA BD3208     2384      DST @CCC1,@ENLN       which points to ERAM              1
                2385    $END IF
8EAD 06975E     2386    CALL IOCALL             Call Device Service Routine for
8EB0 06         2387    DATA C$SAVE             SAVE operation
                2388  LRTOPL
8EB1 066022     2389    CALL KILSYM             Release string space/symbol table
8EB4 056012     2390    B  TOPL15               Go back to toplevel
                2391  ********************************************************************
                2392  GSAVE
                2393  *  Open the sequential file, set the PAB
                2394  *  File type : sequential file
                2395  *  Mode of operation : output
                2396  *  Data type : internal
                2397  *  Record type : variable length records
                2398  *  Logical record length : 254 maximum
```

```
8EB7 310009  2399    MOVE 9 FROM ROM(PAB3) TO RAM(4(PABPTR)) Build the PAB
8EBA E00404
8EBD 8D24
8EBF 96E005  2400    DECT RAM(FLG(PABPTR))  Put in the correct i/o mode, :>1A
8EC2 04
             2401  *  Compute the data buffer address
8EC3 BD4A70  2402    DST @>70,@FAC
8EC6 A74A00  2403    DSUB 253,@FAC
8EC9 FD
8ECA BDE006  2404    DST @FAC,RAM(BUF(PABPTR))
8ECD 044A
8ECF BD584A  2405    DST @FAC,@EEE1          Save it for later use in GVWITE
8ED2 06976B  2406    CALL CDSR               Call device service routine to open
8ED5 5794    2407    BR ERR$2B               Return with ERROR indication inCOND
             2408  *                         Put 8 bytes control info. at the
             2409  *                         beginning of the data buffer
8ED7 BFB04A  2410    DST >ABCD,RAM(@FAC)     >ABCD indentifies a program file
8EDA ABCD
8EDC 954A    2411    DINCT @FAC              when doing LOAD later
8EDE BDB04A  2412    DST @STLN,RAM(@FAC)     Save STLN in control info.
8EE1 30
8EE2 954A    2413    DINCT @FAC
8EE4 BDB04A  2414    DST @ENLN,RAM(@FAC)     ENLN too
8EE7 32
8EE8 954A    2415    DINCT @FAC
8EEA BDB04A  2416    DST @STLN,RAM(@FAC)
8EED 30
8EEE B9B04A  2417    DXOR @ENLN,RAM(@FAC)    Save the checksum
8EF1 32
8EF2 D6A3B9  2418    $IF RAM(SAPROT) .EQ. 1 THEN Check is there PROTECTED opt 1
8EF5 014EFB
8EF8 83B04A  2419      DNEG RAM(@FAC)        Negate the CHECKSUM to indicate   1
             2420    $END IF                 the LIST/EDIT protection
8EFB 954A    2421    DINCT @FAC
8EFD BDB04A  2422    DST @RAMTOP,RAM(@FAC)   Save the top of memory info.
8F00 8084
8F02 BEE009  2423    ST 10,RAM(CNT(PABPTR)) Set the character count in PAB
8F05 040A
8F07 06975E  2424    CALL IOCALL             Call device service routine
8F0A 03      2425    DATA C$WRIT             With I/O opcode : write,
             2426  *                         to save the control info. for
             2427  *                         the first record
             2428  *       Now start to use maximum-length record to write
             2429  *       the file and each record is copied into the VDP
             2430  *       from ERAM before  it is written
8F0B BD5430  2431    DST @STLN,@DDD1         Staring address on ERAM
             2432  * DST @>70,@EEE1         @EEE1 has been set up before
             2433  * DSUB 253,@EEE1         Staring address of the data buffer
             2434  *                              on VDP RAM
8F0E BD0880  2435    DST @RAMTOP,@CCC1
8F11 84
8F12 A50830  2436    DSUB @STLN,@CCC1        Total # of bytes in ERAM to move
8F15 9108    2437    DINC @CCC1
8F17 BEE009  2438    ST 254,RAM(CNT(PABPTR)) Set the character count of PAB
8F1A 04FE
```

```
                2439    $REPEAT
8F1C BF5600     2440      DST 254,@FFF1          @FFF1:Byte count                1
8F1F FE
8F20 0F8B       2441      XML GVWITE             Move data from ERAM to VDP      1
8F22 06975E     2442      CALL IOCALL            Call device service routine     1
8F25 03         2443      DATA C$WRIT                                            1
8F26 A35400     2444      DADD 254,@DDD1         Update the source addr.on ERAM  1
8F29 FE
8F2A A70800     2445      DSUB 254,@CCC1         # of bytes left to move         1
8F2D FE
8F2E 6F44       2446      BS   GSAV1            No more bytes to save           1
8F30 CB0800     2447    $UNTIL @CCC1 .DL. 254      Leave the last record alone
8F33 FE6F1C
                2448  *       Move the last @CCC1 bytes from ERAM to VDP RAM
8F36 BD5608     2449    DST @CCC1,@FFF1            @FFF1:Byte count
8F39 0F8B       2450    XML GVWITE                 Write data from ERAM to VDP RAM
8F3B BCE009     2451    ST @CCC1+1,RAM(CNT(PABPTR)) Update the character count in
8F3E 0409
8F40 06975E     2452    CALL IOCALL                Call device service routine.   PAB
8F43 03         2453    DATA  C$WRIT
                2454  GSAV1
8F44 06975E     2455    CALL IOCALL                Close the file
8F47 01         2456    DATA C$CLOS
8F48 4EB1       2457    BR   LRTOPL                Continue
                2458  ********************************************************************
                2459  ********************************************************************
                2460  * Move the program text & ln # table to VDP, and relocate
                2461  GVMOV
8F4A BD0C30     2462    DST @STLN,@BBB1            Save STLN,ENLN for later use
8F4D BD0832     2463    DST @ENLN,@CCC1
8F50 BD5430     2464    DST @STLN,@DDD1            Source addr. on ERAM
8F53 BD5810     2465    DST @VAR5,@EEE1            Destination addr. on VDP
8F56 BD0A58     2466    DST @EEE1,@STADDR          Use later for RELOCA
8F59 BD5680     2467    DST @RAMTOP,@FFF1
8F5C 84
8F5D A55630     2468    DSUB @STLN,@FFF1           # of bytes to move
8F60 9156       2469    DINC @FFF1                 @FFF1 : Byte count for GVWITE
8F62 0F8B       2470    XML  GVWITE                Move from ERAM to VDP
8F64 BDA3BC     2471    DST @RAMTOP,RAM(OLDTOP) Set up @RAMTOP for old top of memo
8F67 8084
8F69 BDA3BE     2472    DST @>70,RAM(NEWTOP)     Set up @>70 for new top of memory
8F6C 70
8F6D 068D36     2473    CALL RELOCA                Relocate the program
8F70 BD4030     2474    DST @STLN,@FREPTR          Set up @FREPTR
8F73 9340       2475    DDEC @FREPTR
8F75 00         2476    RTN
                2477  ********************************************************************
                2478  *       Save the crunched form of a program into a file            *
                2479  *       Move the ln # and text to the crunch buffer, then          *
                2480  *       write to the file one ln at a time                         *
                2481  ********************************************************************
                2482  SAVMG
                2483  * Open the file with
                2484  *    I/O opcode : OPEN
                2485  *    File type : SEQUENTIAL file
```

```
                    2486  *   Mode of operation : OUTPUT
                    2487  *   Data type : DISPLAY type data
                    2488  *   Recordtype : VARIABLE LENGTH records
                    2489  *   Data buffer address : crunch buffer address
                    2490  *   Logical record length : 163 (length of crunch buffer +
                    2491  *                           2 bytes for ln #) maximum
8F76 310009         2492      MOVE 9 FROM ROM(PAB1) TO RAM(4(PABPTR)) Build the PAB
8F79 E00404
8F7C 8FF9
8F7E 069765         2493      CALL IOCL$1               Call the DSR routine to open file
8F81 BD5032         2494      DST  @ENLN,@FAC6          Start from the first ln #
8F84 A75000         2495      DSUB >03,@FAC6            @FAC6 now points to the 1st ln #
8F87 03
                    2496      $REPEAT                   Write to the file from crunch buffr
                    2497  *                             one ln at a time
8F88 8600           2498        CLR @VARO               Make it a two byte later          1
8F8A 8E8084         2499        $IF @RAMTOP .NE. 0 THEN   If ERAM exists then             2
8F8D 6FB6
8F8F BD5450         2500          DST @FAC6,@DDD1       Write the 4 bytes (ln # and ln ptr2
                    2501  *                             from ERAM to crunch buffer
                    2502  *                             @DDD1:Source addr. on ERAM
8F92 BF5808         2503          DST CRNBUF,@EEE1      @EEE1:Destination addr. on VDP    2
8F95 20
8F96 BF5600         2504          DST 4,@FFF1           @FFF1:Byte count                  2
8F99 04
8F9A 0F8B           2505          XML GVWITE            Write data from ERAM to VDP       2
8F9C BD54A8         2506          DST RAM(CRNBUF+2),@DDD1 Ln ptr now points to len byte2
8F9F 22
8FA0 9354           2507          DDEC @DDD1            Get the length of this ln         2
                    2508  *                             @DDD1:Source addr. on ERAM
8FA2 9156           2509          DINC  @FFF1           @FFF1:Byte count, coming back from2
                    2510  *                             GVWITE above, =0.
8FA4 0F8C           2511          XML GREAD1            Read the length byte from ERAM    2
8FA6 BC0158         2512          ST @EEE1,@VARO+1      @EEE1:Destination addr. on CPU    2
8FA9 BF5808         2513          DST CRNBUF+2,@EEE1    Write the text from ERAM to 3rd   2
8FAC 22
                    2514  *                             byte of crunch buffer
                    2515  *                             @EEE1:Destination addr. on VDP
                    2516  *                             @DDD1:Source addr. on ERAM
8FAD 9154           2517          DINC @DDD1            Back to point to the text         2
8FAF BD5600         2518          DST @VARO,@FFF1       @FFF1:Byte count                  2
8FB2 0F8B           2519          XML GVWITE            Write data from ERAM to VDP       2
8FB4 4FCD           2520        $SELSE                  ERAM not exist : ln # table and   2
                    2521  *                             text in VDP
8FB6 BDA820         2522          DST RAM(@FAC6),RAM(CRNBUF) PUT THE LN # IN              2
8FB9 B050
8FBB BD4CE0         2523          DST RAM(2(FAC6)),@FAC2 Get the ln ptr out               2
8FBE 0250
8FC0 934C           2524          DDEC @FAC2            Ln ptr now points to the len byte 2
8FC2 BC01B0         2525          ST RAM(@FAC2),@VARO+1  Get the length out               2
8FC5 4C
                    2526  *             Move the text into the crunch buffer
8FC6 3400A8         2527          MOVE @VARO FROM RAM(1(FAC2)) TO RAM(CRNBUF+2)           3
8FC9 22E001
8FCC 4C
```

```
                2528        $END IF
3FCD B2A820     2529        RB RAM(CRNBUF),7      Reset possible breakpt          1
3FD0 7F
3FD1 9500       2530        DINCT @VAR0           Total len = text len + ln # len(2)1
3FD3 BCE009     2531        ST @VAR0+1,RAM(CNT(PABPTR)) Store in the character couni
3FD6 0401
3FD8 06975E     2532        CALL IOCALL           Call the device service routine toi
3FDB 03         2533        DATA C$WRIT           write                           1
3FDC A75000     2534        DSUB 4,@FAC6          Go to the next ln #             1
3FDF 04
3FE0 C95030     2535      $UNTIL @FAC6 .DL. @STLN  Finish moving all
3FE3 6F88
3FE5 BFA820     2536      DST >FFFF,RAM(CRNBUF)  Set up a EOF for the last record
3FE8 FFFF
3FEA BEE009     2537      ST 2,RAM(CNT(PABPTR)) Only write this 2 bytes
3FED 0402
3FEF 06975E     2538      CALL IOCALL           Call the device service routine to
3FF2 03         2539      DATA C$WRIT           write
3FF3 06975E     2540      CALL IOCALL           Call the device service routine to
3FF6 01         2541      DATA C$CLOS           close the file
3FF7 4EB1       2542      BR LRTOPL             Go back to top level
                2543    PAB1
3FF9 001208     2544      DATA 0,>12,#CRNBUF,163,0,#0,OFFSET
3FFC 20A300
3FFF 000060
```

```
                2546  ********************************************************************
                2547  *              M E R G E    R O U T I N E                          *
                2548  *                                                                  *
                2549  *       MERGE  load a file which is in crunched program            *
                2550  *       form into the CRNBUF one record (one ln) at a time        *
                2551  *       then take the ln # out in FAC, text length into           *
                2552  *       @CHAT, and edit it into the program.                      *
                2553  *       Identify EOF by the last record which is set up           *
                2554  *       at SAVE time                                              *
                2555  ********************************************************************
                2556  MERGE
 9002 069235    2557    CALL GPNAME                 Close all file, set up PAB
 9005 DA4580    2558    $IF .BIT7 @FLAG .EQ. 1 GOTO ERRPV Check PROTECTION VIOLATI
 9008 57DB
                2559  *  To fix the bug #06       in MERGE
 900A 0F79      2560    XML PGMCHR                  Check EOL
 900C 8E42      2561    CZ  @CHAT
 900E 4109      2562    BR  ERRSYN                    Not EOL : SYNTAX ERROR
                2563  * Open the file with
                2564  *   I/O opcode : OPEN
                2565  *   File type : SEQUENTIAL file
                2566  *   Mode of operation : INPUT
                2567  *   Data type : DISPLAY type data
                2568  *   Recordtype : VARIABLE LENGTH records
                2569  *   Data buffer addr. : crnbuf addr.
                2570  *   Logical record length : 163 maximum
 9010 310009    2571    MOVE 9 FROM ROM(PAB1) TO RAM(4(PABPTR)) Set up PAB
 9013 E00404
 9016 8FF9
 9018 94E005    2572    INCT RAM(FLG(PABPTR))   Put in correct i/o mode :>14
 901B 04
 901C 069765    2573    CALL IOCL$1             Call the device service routine to
                2574  *                         open the file
 901F 06975E    2575    CALL IOCALL             Call the device service routine to
 9022 02        2576    DATA C$READ             read
 9023 D7A820    2577    $IF RAM(CRNBUF) .DEQ. >FFFF GOTO ERR$2B If 1st rec is EOF
 9026 FFFF77
 9029 94
                2578    $REPEAT                 Read in one ln and edit it to progm
 902A 8780D6    2579      DCLR @>D6             Reset VDP timeout                    1
 902D BC42E0    2580      ST RAM(CNT(PABPTR)),@CHAT Len of this record               1
 9030 0904
 9032 9642      2581      DECT @CHAT            Text len = total len - 2 (ln # len1
                2582  *                         Put it in @CHAT for EDITLN
 9034 BD4AA8    2583      DST RAM(CRNBUF),@FAC Put the ln # in @FAC for EDITLN       1
 9037 20
 9038 8656      2584      CLR @FAC12            Make it a double byte                1
 903A BC5742    2585      ST @CHAT,@FAC13                                            1
                2586  *       Move the text up 2 bytes
 903D 3456A8    2587      MOVE @FAC12 FROM RAM(CRNBUF+2) TO RAM(CRNBUF)              1
 9040 20A822
 9043 BDA39E    2588      DST @PABPTR,RAM(MRGPAB)  SAVE PAB PTR                      1
 9046 04
 9047 066032    2589      CALL EDITLN           EDIT IT TO THE PROGRAM               1
 904A 8704      2590      DCLR @PABPTR          Clear temporary PAB pointer.         1
```

```
904C C104A3  2591       DEX  RAM(MRGPAB),@PABPTR Restore old PAB PTR          1
904F 9E
9050 06975E  2592       CALL IOCALL             CALL THE DEVICE SERVICE ROUTINE TO1
9053 02      2593       DATA  C$READ            read another record or another lin1
9054 D7A820  2594     $UNTIL RAM(CRNBUF) .DEQ. >FFFF  End on EOF
9057 FFFF50
905A 2A
             2595  MERG$1
             2596  *     Double check EOF record
905B D6E009  2597     $IF RAM(CNT(PABPTR)) .NE. 2 GOTO ERR$2B I/O ERROR
905E 040257
9061 94
9062 06975E  2598     CALL IOCALL               Call the device service routine to
9065 01      2599     DATA C$CLOS               close the file
9066 4EB1    2600     BR LRTOPL                 Go back to top level
```

```
                    2602  ********************************************************************
                    2603  *                  L I S T   R O U T I N E                         *
                    2604  *                                                                  *
                    2605  *       LIST lists a readable copy of the current program          *
                    2606  *       image to the specified device. In case no device           *
                    2607  *       is specified, the listing is copied to the screen.         *
                    2608  *                                                                  *
                    2609  *            This routine uses the fact that ERR$$ returns         *
                    2610  *       to the caller if the call has been issued in EDIT          *
                    2611  *       mode.  Therefore, ERR$2 will return to TOPL10,             *
                    2612  *       which will reinitiate the variable stuff                   *
                    2613  ********************************************************************
                    2614  LIST
 9068 DA4580        2615    $IF .BIT7 @FLAG .EQ. 1 GOTO ERRPV  PROTECTION VIOLATION ER
 906B 57DB
 906D 8714          2616    DCLR @CURLIN              Create some kind of control for
 906F 870E          2617    DCLR @CURINC                defaults
 9071 BE082D        2618    ST   MINUS,@VARC          Select "-" as separator
 9074 06602E        2619    CALL AUTO1                Pick up any available arguments
                    2620    $
                    2621    $     If either CURLIN or CURINC is non-zero, use it.
                    2622    $     For zero values replace the default (ENLN-3, STLN)
                    2623    $
 9077 8F1450        2624    $IF @CURLIN .DEQ. 0 THEN
 907A 94
 907B BD5432        2625        DST @ENLN,@DDD1       Get the first ln.'s ln. #
 907E A75400        2626        DSUB 3,@DDD1          DDD1 : Source addr on ERAM/VDP
 9081 03
 9082 0691BC        2627        CALL GRSUB3           Read the ln. # from ERAM/VDP
 9085 54            2628        DATA DDD1             @DDD1:Source addr. on ERAM/VDP
                    2629  *                           Reset possible breakpt too
 9086 BD1458        2630        DST @EEE1,@CURLIN     Use standard default
 9089 8F0E50        2631      $IF @CURINC .DEQ. 0 THEN
 908C 94
                    2632  LIST$0
 908D 0691BC        2633        CALL GRSUB3           Read last ln. # from ERAM/VDP
 9090 30            2634        DATA STLN             @STLN:Source addr. on ERAM/VDP
                    2635  *                           Reset possible breakpt too
 9091 BD0E58        2636        DST @EEE1,@CURINC     @EEE1:Destination addr. on ERAM/VD
                    2637  *                           Also default for end line
                    2638      $END IF
                    2639    $END IF
                    2640    $
                    2641    $     Now first evaluate what we've got in CURLIN
                    2642    $
 9094 8F0E50        2643    $IF @CURINC .DEQ. 0 THEN  Check for combination xxx-
 9097 A6
                    2644      $REPEAT
 9098 9320          2645        DDEC @VARW            Backup to the separation mark
 909A D6B020        2646      $UNTIL RAM(@VARW) .NE. : :+OFFSET
 909D 807098
 90A0 D6B020        2647      $IF RAM(@VARW) .EQ. :-:+OFFSET GOTO LIST$0 Select last
 90A3 8D708D
                    2648    $END IF
 90A6 C90E14        2649    $IF @CURINC .DL. @CURLIN THEN If something like LIST
```

```
90A9 70AE
90AB BD0E14  2650       DST  @CURLIN,@CURINC Replace by LIST 15-15          1
             2651     $END IF
90AE BD4A14  2652     DST  @CURLIN,@FAC      Prepare for line # search
90B1 0F7E    2653     XML  SPEED             Search the line number table
90B3 03      2654     DATA SEETWO
90B4 BD142E  2655     DST  @EXTRAM,@CURLIN   Get first real line # in CURLIN
90B7 BD4A0E  2656     DST  @CURINC,@FAC
90BA 0F7E    2657     XML  SPEED
90BC 03      2658     DATA SEETWO            Evaluate second line #
90BD 0691BC  2659     CALL GRSUB3            Read 2 bytes of data from ERAM/VDP
90C0 2E      2660     DATA EXTRAM            @EXTRAM:Source addr. on ERAM/VDP
             2661  *                         Reset possible breakpt too
90C1 C5580E  2662     $IF  @EEE1 .DH. @CURINC THEN                           1
90C4 50CA
90C6 A32E00  2663       DADD 4,@EXTRAM         Else take next lower line    1
90C9 04
             2664     $END IF
90CA BD0E2E  2665     DST  @EXTRAM,@CURINC   Which could be equal to CURLIN
90CD BD2E14  2666     DST  @CURLIN,@EXTRAM   For use below by LLIST
90D0 932C    2667     DDEC @PGMPTR           Backup to last CHAT
90D2 0F79    2668     XML  PGMCHR            Retrieve last CHAT
90D4 8E4271  2669     $IF @CHAT .NE. 0 THEN  Devicename available           1
90D7 32
90D8 0681F4  2670       CALL CLSALL            Close all files that are open 1
90DB BF6E09  2671       DST  VRAMVS,@VSPTR     Re-initialize the V-stack     1
90DE 58
90DF BD246E  2672       DST  @VSPTR,@STVSPT    And it's base                 1
90E2 0F79    2673       XML  PGMCHR          Get name length in CHAT         1
90E4 BF0409  2674       DST  VRAMVS+16,@PABPTR Get entrypoint in PAB         1
90E7 68
90E8 8617    2675       CLR  @DSRFLG         Indicate device I/O             1
90EA 310009  2676       MOVE 9,FROM ROM(PAB) TO RAM(4(PABPTR))               1
90ED E00404
90F0 9170
90F2 BF0809  2677       DST  VRAMVS+16+NLEN,@CCPADR Select start address for co1
90F5 75
90F6 BC4C42  2678       ST   @CHAT,@FAC2     Number of characters to copy    1
90F9 904C    2679       INC  @FAC2           Plus length byte                1
             2680  LIST$1
90FB BCB008  2681       ST   @CHAT,RAM(@CCPADR) Copy the bytes one by one     1
90FE 42
90FF 0F79    2682       XML  PGMCHR          Get next character              1
9101 9108    2683       DINC @CCPADR         CCPADR ends up with highest addres1
9103 924C    2684       DEC  @FAC2           Count total # of characters     1
9105 50FB    2685       BR   LIST$1                                          1
9107 069765  2686       CALL IOCL$1          Perform OPEN on DSR             1
910A 864A    2687       CLR  @FAC            Create double byte PAB length   1
910C BC07E0  2688       ST   RAM(LEN(PABPTR)),@RECLEN Get record length      1
910F 0804
9111 BC4B07  2689       ST   @RECLEN,@FAC1   Compute record length
9114 A14A08  2690       DADD @CCPADR,@FAC    Get highest address used
9117 BDE006  2691       DST  @CCPADR,RAM(BUF(PABPTR)) Store it
911A 0408
911C 8E8084  2692       $IF @RAMTOP .NE. 0 THEN  If ERAM exists then         1
```

```
911F 7128
9121 C54A70  2693        $IF @FAC . DH. @>70 GOTO ERRIO  Compare with top of
9124 77BB
             2694  *                      VDP: if higher then 'not enough room'
9126 512D    2695      $SELSE
9128 C54A30  2696        $IF @FAC . DH. @STLN GOTO ERRIO   Not enough room
912B 77BB
             2697      $END IF
912D BE0601  2698      ST    1,@CCPPTR         Clear first line in output
9130 5138    2699    $SELSE
9132 BE7F1F  2700      ST    VWIDTH+3,XPT       For common code usage
9135 06973F  2701      CALL INITKB              Reset current record length
             2702    $SEND IF
             2703    $REPEAT                   Display at least one line
9138 8E8084  2704      $IF @RAMTOP . NE. 0 THEN  If ERAM exists then
913B 7140
913D 069179  2705      CALL GRMLST             Fake it: move each ln to the
             2706      $END IF                 CRUNCH buffer from ERAM
9140 066A74  2707      CALL LLIST              List the current line
9143 03      2708      SCAN                    Test for a break key
9144 514E    2709      BR    LIST$3            No key
9146 D67502  2710      $IF @RKEY . EQ. BREAK GOTO LIST$4
9149 715F
             2711  LIST$5
914B 03      2712      SCAN
914C 514B    2713      BR LIST$5
             2714  LIST$3
914E 8E8084  2715      $IF @RAMTOP . NE. 0 THEN If ERAM exists
9151 7156
9153 BD2E58  2716        DST @FAC14,@EXTRAM Restore the @EXTRAM
             2717      $END IF
9156 A72E00  2718      DSUB 4,@EXTRAM          Pointer to next line
9159 04
915A C50E2E  2719    $UNTIL @CURINC . DH. @EXTRAM Displayed all lines in range
915D 5138
             2720  LIST$4
915F 8E1751  2721    $IF @DSRFLG . EQ. 0 THEN Device I/O -> output last rec.
9162 6D
9163 0696A5  2722      CALL OUTREC             Output the last record
9166 06975E  2723      CALL IOCALL             Close the device properly
9169 01      2724      DATA C$CLOS
916A 05601A  2725      B     TOPL10
             2726    $END IF
916D 056012  2727    B     TOPL15              Restart the variables too
             2728  *
             2729  *       PAB image used in LIST function
             2730  *
             2731  PAB
9170 001200  2732    DATA 0,>12,#0,0,0,#0,OFFSET
9173 000000
9176 000060
             2733  *  Move each line in ERAM to CRNBUF area, put ln. no. in
             2734  *  (CRNBUF), Put CRNBUF+4 in (CRNBUF+2) which is the ln. ptr
             2735  *  field, put the text itself from ERAM to (CRNBUF+4). Before
             2736  *  calling LLIST, trick it by moving CRNBUF to @EXTRAM
```

```
                2737  GRMLST
9179 0691BC     2738        CALL GRSUB3          Get ln. # from ERAM(use GREAD1)
917C 2E         2739        DATA EXTRAM          @EXTRAN : Source addr. on ERAM
                2740  *                          Reset possible break pt too
917D BDA820     2741        DST @EEE1,RAM(CRNBUF)  Put it in CRNBUF
9180 58
9181 BFA822     2742        DST CRNBUF+4,RAM(CRNBUF+2) Put CRNBUF+4 into
9184 0824
                2743  *                          the ln ptr field
9186 9554       2744        DINCT @DDD1          Get the ptr to the text from GRA
9188 0691AA     2745        CALL GRSUB4          Read the ln. pointer in (use
                2746  *                          GREAD1)
918B 9358       2747        DDEC @EEE1           Get the ptr to the lengh byte
918D 0691A4     2748        CALL GRSUB2          Read the length form ERAM, use
9190 58         2749        DATA EEE1            GREAD1,@EEE1:Source addr. on ERAM
9191 BC5758     2750        ST @EEE1,@FFF1+1     Use the length as byte count
                2751  *                          to move the text from ERAM to
                2752  *                          VDP CRNBUF+4 area
9194 BF5808     2753        DST CRNBUF+4,@EEE1   EEE1 : Destination addr. on VDP
9197 24
9198 9154       2754        DINC @DDD1           DDD1 : Source addr. on ERAM
                2755  *                          Now point to the text
919A 0F8B       2756        XML GVWITE           Move data form ERAM to VDP
919C BD582E     2757        DST @EXTRAM,@FAC14   Save for later use
919F BF2E08     2758        DST CRNBUF,@EXTRAM   Fake it
91A2 20
91A3 00         2759        RTN
```

```
                    2761  *
                    2762  *       SUBROUTINE TO READ 2 BYTES OF DATA FROM ERAM OR VDP
                    2763  *       WITH THE OPTION TO RESET THE POSSIBLE BREAKPT
                    2764  *
                    2765  GRSUB2
91A4 8856           2766          FETCH @FFF1               Fetch the source addr. on ERAM
91A6 BD5490         2767          DST *FFF1,@DDD1           or VDP
91A9 56
                    2768  *                                 @DDD1:Source addr. on ERAM/VDP
                    2769  GRSUB4
91AA 8E8084         2770          $IF @RAMTOP .NE. 0 THEN If ERAM exists                1
91AD 71B7
91AF BF5600         2771            DST 2,@FFF1             @FFF1:Byte count            1
91B2 02
91B3 0F8C           2772            XML GREAD1              Read data from ERAM to CPU  1
91B5 51BB           2773          $SELSE                    ERAM not exists             1
91B7 BD58B0         2774            DST RAM(@DDD1),@EEE1    Read data from VDP to CPU   1
91BA 54
                    2775          $END IF
91BB 00             2776          RTN
                    2777  GRSUB3
91BC 8856           2778          FETCH @FFF1               Fetch the source addr. on ERAM
91BE BD5490         2779          DST *FFF1,@DDD1           or VDP
91C1 56
                    2780  *                                 @DDD1:Source addr. on ERAM/VDP
91C2 0691AA         2781          CALL GRSUB4               Do the actual read
91C5 B3587F         2782          DAND >7FFF,@EEE1          Reset possible breakpt
91C8 FF
91C9 00             2783          RTN
```

```
                    2785  *
                    2786  *              R E C   R O U T I N E
                    2787  *
                    2788  *          REC(X) returns the current record to which file
                    2789  *          X is positioned.
                    2790  *
                    2791  SUBREC
91CA BD5C04         2792    DST  @PABPTR,@ARG       Save the current PAB^ & set new one
91CD 06921E         2793    CALL SUBEOF             Try to find the correct PAB
91D0 C15C04         2794    DEX @PABPTR,@ARG        @ARG: new PAB ^
                    2795  *                         @PABPTR : restore current PAB ^
91D3 5212           2796    BR   EOF$2              Didn't find the corresponding PAB
91D5 BD4AE0         2797    DST  RAM(RNM(ARG)),@FAC  Obtain integer record number
91D8 0A5C
91DA 0F80           2798    XML    CIF              Convert integer to floating
91DC 0F75           2799    XML   CONT                and continue
                    2800  ********************************************************
                    2801  *            E O F   R O U T I N E                     *
                    2802  *                                                      *
                    2803  *      EOF(X) returns status codes on file X.  The meaning *
                    2804  *      of the resultcodes is :                         *
                    2805  *                                                      *
                    2806  *            -1            Physical End Of File        *
                    2807  *            0             Not at End Of File yet      *
                    2808  *            +1            Logical End Of File         *
                    2809  ********************************************************
                    2810  EOF
91DE BD5C04         2811    DST @PABPTR,@ARG        Save the current PAB ^ and set the
                    2812  *                         new one in SUBEOF
91E1 06921E         2813    CALL   SUBEOF           Try to find the PAB somewhere
91E4 57D7           2814    BR     ERRFE            Can't find
91E6 BE5E09         2815    ST   C$STAT,@ARG2       Select status code without
91E9 C0E004         2816    EX   @ARG2,RAM(COD(PABPTR))  destroying original code
91EC 045E
91EE 069765         2817    CALL IOCL$1             Get the information from the DSR
91F1 C1045C         2818    DEX  @ARG,@PABPTR       Restore original PAB ^ and original
91F4 BCE004         2819    ST   @ARG2,RAM(COD(ARG))   I/O code
91F7 5C5E
91F9 BC5EE0         2820    ST   RAM(SCR(ARG)),@ARG2   And pick up STATUS
91FC 0C5C
91FE 310008         2821    MOVE 8 FROM ROM(FLOAT1) TO @FAC Get floating 1
9201 4A9216
9204 DA5E03         2822    CLOG 3,@ARG2            Test EOF bits
9207 7212           2823    BS   EOF$2              No EOF indication
9209 DA5E02         2824    $IF .BIT1 @ARG2 .EQ. 1 THEN Physical EOF
920C 7210
920E 834A           2825      DNEG @FAC             Make result -1
                    2826    $END IF
9210 0F75           2827    XML  CONT
                    2828  EOF$2
9212 874A           2829    DCLR @FAC               Create result 0
9214 0F75           2830    XML  CONT
                    2831
                    2832  FLOAT1
9216 400100         2833    DATA >40,1,0,0,0,0,0,0  Floating point +1
```

```
9219 000000
921C 0000
                2834
                2835  SUBEOF
921E D642B7     2836    $IF @CHAT .NE. LPAR$ GOTO ERRSYN   * SYNTAX ERROR
9221 4109
9223 0F74       2837      XML  PARSE              Parse up to the matching ")"
9225 FF         2838      DATA >FF
9226 06935C     2839      CALL  CHKCNV            Convert and search for PAB
9229 77CB       2840      BS    ERRBV             Avoid 0's and negatives Bad value!
922B BC6217     2841      ST    @DSRFLG,@ARG6     @DSRFLG got changed in CHKCON
922E 069371     2842      CALL CHKCON             Check and search given filenumber
9231 BC1762     2843      ST    @ARG6,@DSRFLG     @DSRFLG got changed in CHKCON
9234 01         2844      RTNC                    Condition set:file no. exists
                2845
```

```
                    2847  ********************************************************************
                    2848  * L O A D / S A V E / M E R G E  U T I L I T Y  R O U T           *
                    2849  *                                                                  *
                    2850  *       GPNAME gets program name for OLD and SAVE                  *
                    2851  *       Can also be used for future implementation of              *
                    2852  *       REPLACE statement.                                         *
                    2853  *            Also gives valuable contribution to updating          *
                    2854  *       of program pointers (VSPTR, STVSPT, FLAG, etc.)            *
                    2855  *       and creation of LOAD/SAVE PAB.                             *
                    2856  ********************************************************************
                    2857  GPNAME
9235 B24580         2858    AND >80, @FLAG          Avoid returns from ERR$$ routin
                    2859  *                         Keep the protection bit
9238 D642C7         2860    $IF @CHAT .NE. STRIN$ THEN                                      1
923B 7242
923D D642C8         2861      $IF @CHAT  NE. NUM$ GOTO ERRSYN  "* SYNTAX ERROR"             1
9240 4109
                    2862    $END IF
9242 0681F4         2863    CALL CLSALL             First close all open files
9245 066022         2864    CALL KILSYM             Kill the symbol table
9248 BF0407         2865    DST  VRAMVS+8,@PABPTR   Create PAB as low as possible
924B 60
924C 86B004         2866    CLR  RAM(@PABPTR)       Clear PAB with ripple-move
924F 350009         2867    MOVE PABLEN-5 FROM RAM(@PABPTR) TO RAM(1(PABPTR))
9252 E00104
9255 B004
9257 0F79           2868    XML  PGMCHR             Get length of file-spec.
9259 A70400         2869    DSUB 4,@PABPTR          Make it a regular PAB ^
925C 04
925D BCE00D         2870    ST   @CHAT,RAM(NLEN(PABPTR)) Copy name length to PAB
9260 0442
9262 BD0AE0         2871    DST RAM(NLEN-1(PABPTR)),@STADDR Avoid problems(bugs!)
9265 0C04
9267 8E8089         2872    $IF @RAMFLG .EQ. 0 THEN If ERAM not exist or imperative s1
926A 5275
926C 340AE0         2873      MOVE @STADDR FROM RAM(@PGMPTR) TO RAM(NLEN+1(PABPTR))       1
926F 0E04B0
9272 2C
9273 5284           2874    $SELSE                                                          1
9275 BD560A         2875      DST @STADDR,@FFF1         @FFF1:Byte count                1
9278 BD542C         2876      DST @PGMPTR,@DDD1         Source addr. on ERAM            1
927B BD5804         2877      DST @PABPTR,@EEE1                                         1
927E A35800         2878      DADD NLEN+1,@EEE1         Destination addr. on VDP        1
9281 0E
9282 0F8B           2879      XML GVWITE                Write from ERAM to VDP          1
                    2880    $END IF
9284 A12C0A         2881    DADD @STADDR,@PGMPTR    Skip the string
                    2882    $
                    2883    $    OLD and SAVE can only be imperative
                    2884    $
9287 8634           2885    CLR   @DATA             Clear DATA line
9289 00             2886    RTN                     That's all folks
```

```
                2888  ********************************************************************
                2889  *                                                                  *
                2890  *  R E A D / I N P U T   U T I L I T Y   R O U T I N E S           *
                2891  *                                                                  *
                2892  ********************************************************************
                2893  GETVAR
928A BD0A2C     2894    DST   @PGMPTR,@STADDR    Save token pointer to first chr
928D 8610       2895    CLR   @VAR5              Clear # of parsed variables
928F BD0E6E     2896    DST   @VSPTR,@VAR4       Save first entry in V-stack
                2897    $
                2898    $     Start parse cycle for INPUT statement
                2899    $
                2900  GETV$0
9292 CA4280     2901    $IF @CHAT .HE. >80 GOTO ERRSYN  Make sure of var. name.
9295 6109
9297 0F7A       2902    XML   SYM                Get correct symbol table entry
9299 8611       2903    CLR   @VAR6              Start with zero paren nesting
                2904  GETV$1
929B D642B7     2905    $IF @CHAT .EQ. LPAR$ THEN Increment counter for "("      1
929E 52A2
92A0 9011       2906      INC   @VAR6                                            1
                2907    $END IF
92A2 8E1172     2908    $IF @VAR6 .NE. 0 THEN  Watch out for final balance       1
92A5 B6
92A6 0695AD     2909      CALL  CHKEND           Check for unbalanced parenthesis 1
92A9 6109       2910      BS    ERRSYN             Somebody forget something!!!!! 1
92AB D642B6     2911      $IF @CHAT .EQ. RPAR$ THEN    Decrement for ")"         2
92AE 52B2
92B0 9211       2912        DEC   @VAR6                                          2
                2913      $END IF
92B2 0F79       2914      XML   PGMCHR           Get character following last ")"  1
92B4 529B       2915      BR    GETV$1                                           1
                2916    $END IF
92B6 0F77       2917    XML   VPUSH              Push entry to V-stack
92B8 9010       2918    INC   @VAR5              Count all pushed variables
92BA 0695AD     2919    CALL  CHKEND           Next should either be EOS or ","
92BD 72CC       2920    BS    GETV$2           Found it EOS!!!!
92BF 0F7E       2921    XML   SPEED            Must be at a
92C1 00         2922    DATA  SYNCHK                comma else
92C2 B3         2923    DATA  COMMA$                  its an error
92C3 0695AD     2924    CALL  CHKEND         Check for end of statement
92C6 5292       2925    BR    GETV$0         Haven't found it yet
92C8 8E1741     2926    $IF @DSRFLG .NE. 0 GOTO ERRSYN  Error for keyboard I/O
92CB 09
                2927  GETV$2
92CC 00         2928    RTN
                2929    $
                2930    $     Create a temporary string in memory.  BYTE
                2931    $     contains the length
                2932    $
                2933  CTSTR
92CD BF4C65     2934    DST   >6500,@FAC2        Indicate string in FAC
92D0 00
                2935  CTSTR0
92D1 BD500C     2936    DST   @BYTE,@FAC6        Copy string length in FAC6
```

```
92D4 0F71     2937     XML  GETSTR              Reserve the string
92D6 BD4E1C   2938     DST  @SREF,@FAC4         Copy start address of string
92D9 BF4A00   2939     DST  SREF,@FAC           And indicate temp. string
92DC 1C
92DD 00       2940     RTN
              2941     $
              2942     $    Create a temporary string from TEMP5.  Length
              2943     $    is given in SYTE.
              2944     $
              2945   CTMPST
92DE 0692CD   2946     CALL CTSTR               Create the temporary string
92E1 8E5172   2947     $IF @FAC7 .NE. 0 THEN                                  1
92E4 EB
92E5 340CB0   2948       MOVE @BYTE FROM RAM(@TEMP5) TO RAM(@SREF)            1
92E8 1CB066
              2949     $END IF                  non-empty
92EB 00       2950     RTN
              2951   *
              2952   *      CHKNUM - Check for numeric argument
              2953   *
              2954   CHKNUM
92EC D601C8   2955     $IF @VAR0+1 .EQ. NUM$ THEN                             1
92EF 5303
92F1 06930C   2956       CALL GETRAM            Get string length             1
92F4 BD5634   2957       DST  @DATA,@FAC12      Store entry for conversion    1
92F7 8600     2958       CLR  @VAR0             Prepare for double action     1
92F9 A13400   2959       DADD @VAR0,@DATA       Get end of data field         1
92FC 06A012   2960       CALL CONVER            Convert data to FAC #         1
              2961       $
              2962       $  Conversion should also end at end of field
              2963       $
92FF D5A390   2964       DCEQ @DATA,RAM(CSNTMP)    Set COND according to equalit1
9302 34
              2965     $END IF
9303 01       2966     RTNC                     Back to the caller
              2967   *
              2968   *      Get data from ERAM or VDP RAM
              2969   *
              2970   GETGFL
9304 BC4D80   2971     ST   @RAMTOP,@FAC3       Select target memory
9307 84
              2972   GETDAT
9308 8E4D53   2973     $IF @FAC3 .EQ. 0 THEN Get everything from RAM          1
930B 14
              2974   GETRAM
930C BC01B0   2975       ST   RAM(@DATA),@VAR0+1   Get data in VAR0+1         1
930F 34
9310 864D     2976       CLR  @FAC3             Be sure FAC3 = 0 !!!!         1
9312 5320     2977     $SELSE                                                 1
9314 BF5600   2978       DST 1,@FFF1            FFF1 : byte count             1
9317 01
9318 3D5434   2979       DST @DATA,@DDD1        DDD1 : source addr on ERAM    1
931B 0F8C     2980       XML GREAD1             Read data from ERAM           1
931D BC0158   2981       ST  @EEE1,@VAR0+1      EEE1 : Destination addr on CPU 1
              2982     $END IF
```

```
9320 9134    2983    DINC @DATA              Go to next datum for next time
9322 00      2984    RTN
             2985  *
             2986  *
             2987  *
             2988  CHKSTR
9323 8750    2989    DCLR @FAC6              Assume we'll have an empty string
9325 D601C7  2990    $IF @VAR0+1 .NE. STRIN$ THEN
9328 732F
932A D601C8  2991      $IF @VAR0+1 .NE. NUM$ GOTO EMPSTR  See.......
932D 533E
             2992    $END IF
             2993  CHKS$0
932F 069308  2994    CALL GETDAT             Next datum is length byte
9332 8650    2995    CLR  @FAC6              Be sure high byte = 0 !!!!!!!
9334 BC5101  2996    ST   @VAR0+1,@FAC7      Prepare FAC for string assignment
9337 BD6634  2997    DST  @DATA,@TEMP5       Save string address for assignment
933A A13450  2998    DADD @FAC6,@DATA        Update DATA ^ for end of field
933D 00      2999    RTN
             3000    $
             3001    $     Empty strings are handled below
             3002    $
             3003  EMPSTR
933E D601B3  3004    $IF @VAR0+1 .NE. COMMA$ THEN
9341 7348
9343 06934B  3005      CALL DATEND           Check for end of data statement
9346 538C    3006      BR   RTC              Return with COND if not EOS
             3007    $END IF
9348 9334    3008    DDEC @DATA              Backup data pointer for emptie
934A 00      3009    RTN
             3010  *
             3011  DATEND
934B C04201  3012    EX   @VAR0+1,@CHAT      Exchange with CHAT for testing
934E 0695AD  3013    CALL  CHKEND            Check for EOS (=EOL or ":")
9351 C04201  3014    EX    @VAR0+1,@CHAT     Restore original situation
9354 01      3015    RTNC
```

```
                    3017  ********************************************************************
                    3018  *            O P E N / C L O S E / R E S T O R E                   *
                    3019  *              U T I L I T Y   R O U T I N E S                     *
                    3020  *                                                                  *
                    3021  *       CHKFN - Check for token = "#" and collect and check        *
                    3022  *       filenumber.  Also convert filenumber to (two byte)         *
                    3023  *       integer and check for range 0<X<256.                      *
                    3024  ********************************************************************
                    3025  CHKFN
9355 0F7E           3026    XML   SPEED                Must be at a
9357 00             3027    DATA  SYNCHK                  '#' else
9358 FD             3028    DATA  NUMBE$                     its an error
9359 0F74           3029    XML   PARSE                Parse argument up to ":"
935B B5             3030    DATA  COLON$
                    3031    $
                    3032    $     Code to check for negative or zero result in
                    3033    $     floating point accu.  If not...convert to
                    3034    $     integer and return two byte integer in FAC
                    3035    $
                    3036  CHKCNV
935C D64C65         3037    $IF @FAC2 .EQ. STRVAL GOTO ERRSNM String/number mismatch
935F 77BF
9361 8654           3038    CLR   @FAC10                Clear error-code byte
9363 0F12           3039    XML   CFI                   Convert to two byte integer
9365 8E5457         3040    $IF @FAC10 .NE. 0 GOTO ERRBV      BAD VALUE ERROR
9368 CB
9369 DA4A80         3041    $IF .BIT7 @FAC .NE. 0 GOTO RTC Negative result
936C 538C
936E 8F4A           3042    DCZ   @FAC                  And return with COND set/reset
9370 01             3043    RTNC
                    3044    $
                    3045  CHKCON
9371 BC174B         3046    ST    @FAC1,@FNUM           Move result into FNUM
                    3047    $
                    3048    $     Check for high byte not zero ( >255 )
                    3049
9374 8E4A57         3050    $IF @FAC .NE. 0 GOTO ERRBV    Bad value error.
9377 CB
                    3051
                    3052    $     Search routine - Search for a given file number
                    3053    $     in the chain of allocated PABs.
                    3054    $     IOSTRT contains the start of the PAB-chain
                    3055    $
9378 BD043C         3056    DST   @IOSTRT,@PABPTR       Get first link in the chain
                    3057  CHKF$1
                    3058    $
                    3059    $     Check for last PAB in the chain and exit if found
                    3060    $
937B 8F0473         3061    $IF @PABPTR .DNE. 0 THEN  Check if file # is correct      1
937E 8F
937F D4E002         3062      $IF RAM(FIL(PABPTR)) .EQ. @FNUM GOTO RTC                1
9382 041773
9385 8C
9386 BD04B0         3063      DST  RAM(@PABPTR),@PABPTR     Try the next PAB        1
9389 04
```

```
938A 5373     3064       BR    CHKF$1                                    1
              3065  RTC
938C D40000   3066       CEQ   @0,@0           Force COND to "SET"        1
              3067     $END IF
938F 01       3068     RTNC                    Exit with no COND change
              3069
              3070  ********************************************************
              3071  *       OUTEOF outputs the last record if this         *
              3072  *       record is non-empty, and if the PAB is         *
              3073  *       open for non-input mode (UPDATE, APPEND        *
              3074  *       or OUTPUT).                                    *
              3075  ********************************************************
              3076  OUTEOF
9390 8617     3077     CLR   @DSRFLG
9392 D6E004   3078     $IF RAM(COD(PABPTR)) .EQ. C$WRITE THEN Non-input mode   1
9395 040353
9398 A5
9399 8EE003   3079       $IF RAM(OFS(PABPTR)) .NE. 0 THEN Non-empty record     2
939C 0473A5
939F 0696F5   3080         CALL PRINIT         Initiate for output           2
93A2 0696A5   3081         CALL OUTREC         Output and remove pending cond. 2
              3082       $END IF
              3083     $END IF
93A5 00       3084     RTN                     Return to whoever called
```

```
                    3086  ************************************************************
                    3087  *                                                          *
                    3088  *       DELPAB routine - delete a given PAB from the chain *
                    3089  *       under the assumption that the PAB exists           *
                    3090  *                                                          *
                    3091  ************************************************************
                    3092  DELPAB
                    3093    $
                    3094    $     First compute start and end address for block move
                    3095    $
93A6 BD0AE0         3096    DST   RAM(BUF(PABPTR)),@STADDR Get lowest used address
93A9 0604
93AB 930A           3097    DDEC @STADDR              Make that an address following PAB
93AD 8608           3098    CLR  @CCPADR              Get highest address in CCPADR(2)
93AF BC09E0         3099    ST   RAM(NLEN(PABPTR)),@CCPADR+1  complete the two bytes
93B2 0D04
93B4 A2090D         3100    ADD  PABLEN-1,@CCPADR+1   Add PAB length - 1
93B7 A10804         3101    DADD @PABPTR,@CCPADR      Compute actual address within RAM
93BA D53C04         3102    $IF @IOSTRT .DNE. @PABPTR THEN Watch out for first PAB    1
93BD 73E6
93BF BD023C         3103      DST  @IOSTRT,@MNUM      Figure out where link to PAB is   1
93C2 D5B002         3104      $WHILE RAM(@MNUM) .DNE. @PABPTR Continue while not foun2
93C5 0473CE
93C8 BD02B0         3105        DST  RAM(@MNUM),@MNUM   Defer to next link in chain   2
93CB 02
93CC 53C2           3106      $SEND WHILE             Short end for code-savings        1
93CE BDB002         3107      DST  RAM(@PABPTR),RAM(@MNUM) Copy link over deleted PAB1
93D1 B004
93D3 8FB002         3108      $IF RAM(@MNUM) .DNE. 0 THEN Adjust link only if not zer2
93D6 73E0
93D8 A1B002         3109        DADD @CCPADR,RAM(@MNUM) Add deleted # of bytes for   2
93DB 08
93DC A5B002         3110        DSUB @STADDR,RAM(@MNUM)    link correction          2
93DF 0A
                    3111      $END IF
93E0 BD04B0         3112      DST  RAM(@MNUM),@PABPTR   Get new PABPTR                  1
93E3 02
93E4 53F7           3113    $SELSE                                                      1
93E6 BD3CB0         3114      DST  RAM(@PABPTR),@IOSTRT Update first link               1
93E9 04
93EA 8F3C73         3115      $IF @IOSTRT .DNE. 0 THEN  Only adjust if not last link 2
93ED F4
93EE A13C08         3116        DADD @CCPADR,@IOSTRT    Add deleted # of bytes        2
93F1 A53C0A         3117        DSUB @STADDR,@IOSTRT                                  2
                    3118      $END IF
93F4 BD043C         3119      DST  @IOSTRT,@PABPTR      Get new PABPTR                  1
                    3120    $SEND IF
                    3121    $
                    3122    $     Move the bytes below the deleted block up in
                    3123    $     memory. This includes both variables and PABs
                    3124    $
93F7 BD020A         3125    DST  @STADDR,@MNUM        Get # of bytes to move
93FA A50240         3126    DSUB @FREPTR,@MNUM
93FD BD0608         3127    DST  @CCPADR,@CCPPTR      Save destination address
9400 8F0274         3128    $WHILE @MNUM .DNE. 0                                      1
```

```
9403 11
9404 BCB008  3129       ST   RAM(@STADDR),RAM(@CCPADR) Move byte-by-byte      1
9407 B00A
9409 930A    3130       DDEC @STADDR           Update source                  1
940B 9308    3131       DDEC @CCPADR                 and destination pointers 1
940D 9302    3132       DDEC @MNUM             Also update counter value      1
940F 5400    3133     $SEND WHILE              End WHILE loop (Also ends on 0)
9411 A5080A  3134     DSUB @STADDR,@CCPADR     Compute # of bytes of old PAB
9414 8F0474  3135     $IF @PABPTR .DNE. 0 THEN    Avoid trouble with last PAB  1
9417 31
9418 8FB004  3136       $WHILE RAM(@PABPTR) .DNE. 0 Ad infinitum (or fundum)   2
941B 742C
941D A1B004  3137         DADD @CCPADR,RAM(@PABPTR)  Adjust link to next PAB   2
9420 08
9421 A1E006  3138         DADD @CCPADR,RAM(BUF(PABPTR)) Update the buffer link 2
9424 0408
9426 BD04B0  3139         DST  RAM(@PABPTR),@PABPTR Get next link in chain     2
9429 04
942A 5418    3140       $SEND WHILE                                            1
942C A1E006  3141       DADD @CCPADR,RAM(BUF(PABPTR))  Update buffer link      1
942F 0408
             3142     $END IF
             3143     $    Adjust symbol table links
9431 8F3E74  3144     $IF @SYMTAB .DNE. 0 THEN                                 1
9434 B4
9435 D13E06  3145       $IF @SYMTAB .DLT. @CCPPTR THEN Only updata lower links 2
9438 74B4
943A A13E08  3146         DADD @CCPADR,@SYMTAB Get symbol table pointer back   2
943D 8D043E  3147         DST  @SYMTAB,@PABPTR Get pointer for update          2
             3148   DELP$1
9440 8E8084  3149         $IF @RAMTOP .NE. 0 GOTO DELP$2                       2
9443 544C
9445 D1E004  3150         $IF RAM(4(PABPTR)) .DLT. @STLN THEN If imperative    3
9448 043074
944B 51
944C A1E004  3151   DELP$2  DADD @CCPADR,RAM(4(PABPTR))  Adjust name pointer   3
944F 0408
             3152         $END IF
9451 D2B004  3153         $IF RAM(@PABPTR) .LT. 0 THEN If string-fix bkptrs    3
9454 00749B
9457 BE4A07  3154           ST   >07,@FAC          Mask to get # of dims       3
945A B04AB0  3155           AND  RAM(@PABPTR),@FAC Get # of dims               3
945D 04
945E BD4C04  3156           DST  @PABPTR,@FAC2     Ptr to 1st dim max          3
9461 A34C00  3157           DADD 6,@FAC2             or string pointer         3
9464 06
9465 BF5000  3158           DST  1,@FAC6           Number of pointers to change 3
9468 01
9469 864E    3159           CLR  @FAC4             For 2-byte use of option base3
946B 8E4A74  3160           $WHILE @FAC .NE. 0     While more dimensions       4
946E 83
946F BE4F01  3161               ST   1,@FAC5       Assume option base 0        4
9472 A44F43  3162               SUB  @BASE,@FAC5   But correct if base 1       4
9475 A14EB0  3163               DADD RAM(@FAC2),@FAC4   Get dim maximum         4
9478 4C
```

```
9479 A94E50  3164              DMUL @FAC6,@FAC4   Mpy it in                    4
947C 924A    3165              DEC  @FAC          Next dim                    4
947E 954C    3166              DINCT @FAC2                                    4
9480 059468  3167            $END WHILE                                       3
             3168  *
             3169  *       FAC2 now points at the 1st string pointer
             3170  *       FAC6 contains the # of ptrs that need to be changed
             3171  *
9483 8F5074  3172           $WHILE @FAC6 .DNE. 0  While pointers to change     4
9486 98
9487 BD4AB0  3173              DST   RAM(@FAC2),@FAC   Get ptr to string       4
948A 4C
948B 8F4A74  3174              $IF @FAC .DNE. 0 THEN   If string in non-null   5
948E 95
948F BDEFFF  3175                DST  @FAC2,RAM(-3(FAC))  Fix backpointer      5
9492 FD4A4C
             3176              $END IF
9495 954C    3177              DINCT @FAC2          Point to next pointer      4
9497 9350    3178              DDEC  @FAC6          One less ptr to change     4
9499 5483    3179            $END WHILE                                       3
             3180           $END IF
949B 8FE002  3181           $IF RAM(2(PABPTR)) .DNE. 0 THEN                   3
949E 0474B4
94A1 D1E002  3182             $IF RAM(2(PABPTR)) .DLT. @CCPPTR THEN           4
94A4 040674
94A7 B4
94A8 A1E002  3183               DADD @CCPADR,RAM(2(PABPTR)) Adjust next value lin4
94AB 0408
94AD BD04E0  3184               DST  RAM(2(PABPTR)),@PABPTR    Next entry      4
94B0 0204
94B2 5440    3185               BR    DELP$1                                   4
             3186             $END IF
             3187           $END IF
             3188         $END IF
             3189       $END IF
94B4 A14008  3190       DADD @CCPADR,@FREPTR   Update free word pointer
94B7 00      3191       RTN
```

```
                3193  ********************************************************
                3194  *       CNVDEF - Convert to 2-byte integer and default to*
                3195  *                  1 on negative or 0.....               *
                3196  ********************************************************
                3197  CNVDEF
948B 06935C     3198    CALL CHKCNV             Check and convert
948B 54C1       3199    BR   CNVD$0
948D BF4A00     3200    DST  1,@FAC             Default to 1 for minus and 0
94C0 01
                3201  CNVD$0
94C1 00         3202    RTN                     And return without COND set
```

```
                    3204  ***************************************************************
                    3205  *       PARREC parses a possible REC clause in INPUT,         *
                    3206  *       PRINT or RESTORE.  In case a comma is detected        *
                    3207  *       without a REC clause following it, the COND           *
                    3208  *       is set upon return.  In case a REC clause is          *
                    3209  *       specified for a file opened for SEQUENTIAL access,    *
                    3210  *       a * FILE ERROR is given.                              *
                    3211  ***************************************************************
                    3212  PARREC
94C2 D642B3         3213    $IF @CHAT .EQ. COMMA$ THEN Only check if we have a ","    1
94C5 54EE
94C7 0F79           3214      XML  PGMCHR             Check next token for REC        1
94C9 D642DE         3215      $IF @CHAT .NE. REC$ GOTO RTC May be a USING clause      1
94CC 538C
94CE DAE005         3216      $IF .BITO RAM(FLG(PABPTR)) .NE. 1 GOTO ERRFE            1
94D1 040177
94D4 D7
94D5 0F79           3217      XML  PGMCHR             Get first character of expression 1
94D7 069390         3218      CALL OUTEOF             Output possible pending output  1
94DA 86E003         3219      CLR  RAM(OFS(PABPTR))  Clear record offset              1
94DD 04
94DE 0F74           3220      XML  PARSE              Translate the expression in REC 1
94E0 B5             3221      DATA COLON$                                             1
94E1 06935C         3222      CALL CHKCNV             Check numeric and convert to    1
94E4 D24A00         3223      $IF @FAC .LT. 0 GOTO ERRBV  2 byte integer. Bad va      1
94E7 57CB
94E9 BDE00A         3224      DST  @FAC,RAM(RNM(PABPTR)) Store actual record number   1
94EC 044A
                    3225    $END IF
94EE 00             3226    RTN
```

```
                    3228  ********************************************************************
                    3229  *                                                                  *
                    3230  *      D I S P L A Y / A C C E P T   U T I L I T I E S             *
                    3231  *                                                                  *
                    3232  ********************************************************************
                    3233  DISACC
94EF 06973F         3234    CALL INITKB             PABPTR is used as flag (no DSR I/O)
                    3235  DISP$1
94F2 D642EF         3236    $IF @CHAT .EQ. ERASE$ THEN check for ERASE ALL                  1
94F5 5518
94F7 DA0401         3237      $IF .BITO @PABPTR .EQ. 1 GOTO ERRSYN already used once        1
94FA 4109
94FC 0F79           3238      XML  PGMCHR           check next token for ALL                 1
94FE 0F7E           3239      XML  SPEED                                                    1
9500 00             3240      DATA SYNCHK           has to be ALL                           1
9501 EC             3241      DATA ALL$                                                     1
9502 0780           3242      ALL  BKGD+OFFSET       clear screen to background color       1
9504 BE7F03         3243      ST   3,XPT             Reset pending output pointer           1
9507 DA0404         3244      $IF .BIT2 @PABPTR .EQ. 0 THEN Didn't use AT yet               2
950A 5513
950C BE0601         3245        ST   1,@CCPPTR      Reset column pointer                    2
950F BF0802         3246        DST  SCRNBS+2,@CCPADR      and screen base address      2
9512 E2
                    3247      $END IF
9513 B60401         3248      SB   @PABPTR,0        Set "ERASE USED" flag                   1
9516 54F2           3249      BR   DISP$1           Try next token                          1
                    3250    $END IF
9518 D642EE         3251    $IF @CHAT .EQ. BEEP$ THEN delay action for BEEP                 1
951B 5529
951D DA0402         3252      $IF .BIT1 @PABPTR .EQ. 1 GOTO ERRSYN use it only once       1
9520 4109
9522 B60402         3253      SB  @PABPTR,1         No syntax error detected here           1
9525 0F79           3254      XML PGMCHR            Evaluate next token                     1
9527 54F2           3255      BR  DISP$1            Get set for second pass                 1
                    3256    $END IF
9529 D642F0         3257    $IF @CHAT .EQ. AT$ THEN Generate "AT" clause                    1
952C 5571
952E DA0404         3258      $IF .BIT2 @PABPTR .EQ. 1 GOTO ERRSYN Second usage not         1
9531 4109
9533 0F79           3259      XML  PGMCHR                              allowed...           1
9535 0F7E           3260      XML  SPEED                                                    1
9537 00             3261      DATA SYNCHK           Skip left parenthesis                   1
9538 B7             3262      DATA LPAR$                                                    1
9539 0F74           3263      XML  PARSE            Now parse any expression                1
953B B3             3264      DATA COMMA$                                                   1
953C 0F7E           3265      XML  SPEED                                                    1
953E 00             3266      DATA SYNCHK           Check for "," and skip it               1
953F B3             3267      DATA COMMA$                                                   1
9540 0694B8         3268      CALL CNVDEF           Convert to 2 byte numeric               1
9543 BE4C18         3269      ST   24,@FAC2         Convert modulo 24 (# screenlines)       1
9546 06961A         3270      CALL COMMOD           Compute remainder                       1
9549 924B           3271      DEC  @FAC1            Convert back to 0 (range was 1-24)      1
954B AA4B20         3272      MUL  32,@FAC1         Convert to line base address            1
954E BD084B         3273      DST  @FAC1,@CCPADR    And replace CCPADR                      1
9551 0F74           3274      XML  PARSE            Parse column expression                 1
```

```
9552 B6       3275       DATA RPAR$                                                    1
9554 0F7E     3276       XML  SPEED                                                    1
9556 00       3277       DATA SYNCHK           Check for ")"at end                     1
9557 B6       3278       DATA RPAR$                                                    1
9558 0694B8   3279       CALL CNVDEF           Again convert to two byte int.          1
955B BE4C1C   3280       ST   VWIDTH,@FAC2     Convert modulo video width              1
955E 06961A   3281       CALL COMMOD           Compute remainder                       1
9561 BC064B   3282       ST    @FAC1,@CCPPTR   Select current column                  1
9564 A1084A   3283       DADD @FAC,@CCPADR     Compute full address                    1
9567 9108     3284       DINC @CCPADR          Adjust for column 0 (offset-1)          1
9569 B60404   3285       SB   @PABPTR,2        Set "AT-CLAUSE" used flag               1
956C B60420   3286       SB   @PABPTR,5        Set "NON-STANDARD SCREEN ADDRESS"       1
956F 54F2     3287       BR   DISP$1           Continue for next item                  1
              3288     $END IF
9571 D642EB   3289     $IF @CHAT .EQ. SIZE$ THEN "SIZE" clause                         1
9574 55A0
9576 DA0408   3290       $IF .BIT3 @PABPTR .EQ. 1 GOTO ERRSYN   Only use once          1
9579 4109
957B 0F79     3291       XML  PGMCHR           Get chr following the SIZE              1
957D D642B7   3292       $IF @CHAT .NE. LPAR$ GOTO ERRSYN  has to open ("(")           1
9580 4109
9582 0F74     3293       XML  PARSE            And close again (")")                   1
9584 FE       3294       DATA VALID$                                                   1
9585 D24A00   3295       $IF @FAC .LT. 0 THEN Change to positive argument              2
9588 758F
958A 834A     3296         DNEG @FAC           For ACCEPT stmt with - size             2
958C B60480   3297         SB   @PABPTR,7      indicate in highest bit                 2
              3298       $END IF
958F 06935C   3299       CALL CHKCNV                                                   1
9592 77CB     3300       BS    ERRBV           "* BAD VALUE"                           1
9594 8E4A57   3301       $IF @FAC .NE. 0 GOTO  ERRBV  also for args > 255              1
9597 CB
9598 BC054B   3302       ST   @FAC1,@PABPTR+1      copy to PABPTR (always unused)      1
959B B60408   3303       SB   @PABPTR,3        prevent further use                     1
959E 54F2     3304       BR   DISP$1           and go on                               1
              3305     $END IF
95A0 D642FE   3306     $IF @CHAT .NE. VALID$ THEN exclude VALIDATE option              1
95A3 75B9
              3307  *
              3308  *       Start evaluating ERASE clause here
              3309  *
95A5 DA0408   3310       $IF .BIT3 @PABPTR .EQ. 1 THEN                                 2
95A8 75AD
95AA 0695DF   3311         CALL SIZE1          evaluate field defined in SIZE          2
              3312       $END IF
              3313  *
              3314  *    if it's no DISPLAY keyword ( AT, SIZE, BEEP or USING)
              3315  *    it has to be a print separator or colon ":".
              3316  *    if anything is sprcified it has to be a colon or
              3317  *    end of line...for end-of-line output current record
              3318  *
              3319  *
              3320  *    check for end of statement
              3321  *
              3322  CHKEND
```

```
95AD DA4280  3323       $IF .BIT7 @CHAT .EQ. 1 THEN                    2
95B0 75B7
95B2 CA4284  3324         $IF @CHAT  .L. TREM$+1 GOTO RTC              2
95B5 538C
             3325       $END IF
95B7 8E42    3326       CZ  @CHAT              set COND according to CHAT  1
             3327     $END IF
95B9 01      3328     RTNC
             3329
             3330  ***************************************************************
             3331  *       NXTCHR - Get next program character - skip            *
             3332  *                all strings, numerics and line refs...       *
             3333  ***************************************************************
             3334  NXTCHR
95BA 0695AD  3335     CALL CHKEND            Check for end of stmts
95BD 738C    3336     BS   RTC               Avoid end of stmt
95BF D642C7  3337     $IF @CHAT  EQ. STRIN$ GOTO NXTC$0 Skip all strings
95C2 75C9
95C4 D642C8  3338     $IF @CHAT .EQ. NUM$ THEN   and numerics/unquoted strings 1
95C7 55D5
             3339  NXTC$0
95C9 0F79    3340       XML  PGMCHR          Get string length              1
95CB BC4B42  3341       ST   @CHAT,@FAC1     Make that a double please....  1
95CE 864A    3342       CLR  @FAC            Hic.....Oops, sorry            1
95D0 A12C4A  3343       DADD @FAC,@PGMPTR    Back to the serious stuff      1
95D3 55DC    3344     $SELSE                                                 1
95D5 D642C9  3345       $IF @CHAT .EQ. LN$ THEN Line # = skip 2 tokens       2
95D8 55DC
95DA 952C    3346         DINCT @PGMPTR      <----- That's the skip         2
             3347       $END IF
             3348     $SEND IF
95DC 0F79    3349     XML  PGMCHR            Get the next token
95DE 00      3350     RTN
```

```
                    3352  ********************************************************************
                    3353  *     P R I N T / D I S P L A Y   U T I L I T I E S                *
                    3354  *                                                                  *
                    3355  *     Use the parameters specified in SIZE for  further            *
                    3356  *     evaluation of the limited field length                       *
                    3357  ********************************************************************
                    3358  SIZE1
95DF DA0404         3359    $IF .BIT2 @PABPTR .EQ. 0 THEN no "AT" clause used                1
95E2 55FC
95E4 D60601         3360      $IF @CCPPTR .NE. 1 THEN might have to print current lin2
95E7 75FC
95E9 BC4A06         3361        ST   @CCPPTR,@FAC  compute final position after size 2
95EC A04A05         3362        ADD  @PABPTR+1,@FAC in FAC and compare with record   2
95EF 924A           3363        DEC  @FAC                                            2
95F1 C44A07         3364        $IF @FAC .H. @RECLEN THEN size clause too long       3
95F4 55FC
                    3365  *
                    3366  *     We can't get here for AT( , ) output, since
                    3367  *     right margin is limited there
                    3368  *
95F6 0696A5         3369          CALL OUTREC       advance to next line               3
95F9 0696CC         3370          CALL SCRO         sccroll the screeen                3
                    3371        $END IF
                    3372      $END IF
                    3373    $END IF
95FC A40706         3374    SUB  @CCPPTR,@RECLEN    limit field size to available
95FF 9007           3375    INC  @RECLEN             space...including current position
9601 C40705         3376    $IF @RECLEN .NOT. .H. @PABPTR+1 GOTO INIT$1
9604 56C8
9606 BC0705         3377    ST   @PABPTR+1,@RECLEN only accept if available
9609 56C8           3378    BR   INIT$1             reinitialize CCPPTR
                    3379  *
                    3380  *     Copy (converted) numerical datum in string
                    3381  *
                    3382  RSTRING
960B BC0D56         3383    ST   @FAC12,@BYTE+1     Get actual string length
960E 860C           3384    CLR  @BYTE              Create double byte value
9610 0692CD         3385    CALL CTSTR              Create a temporary string
9613 340CB0         3386    MOVE @BYTE FROM *FAC11 TO RAM(@SREF) Copy value string
9616 1C9055
9619 00             3387    RTN
                    3388    $
                    3389    $     COMMOD - Compute FAC module FAC2
                    3390    $
                    3391  COMMOD
961A AC4A4C         3392    DIV  @FAC2,@FAC         Compute remainder
961D 8E4B56         3393    $IF @FAC1 .EQ. 0 THEN Avoid zero remainders
9620 24
9621 BC4B4C         3394      ST   @FAC2,@FAC1      Assume maximum remainder
                    3395    $END IF
9624 864A           3396    CLR  @FAC               Clear upper byte
9626 00             3397    RTN
                    3398    $
                    3399    $     TSTSEP tests for separator in print and
                    3400    $     branches to the correct evaluation routine.
```

```
                    3401    $     if no separator is found, simple return.
                    3402    $
                    3403  TSTSEP
                    3404    $     Test case end of line
 9627 0695AD        3405      CALL CHKEND
 962A 5631          3406      BR   TSTS$0
 962C BF9073        3407      DST  #EOLEX,*SUBSTK  Replace returnaddress with EOLEX
 962F 8395
                    3408  TSTS$0
 9631 CA42B3        3409    $IF @CHAT .L. COMMA$ GOTO TSTS$1
 9634 5659
 9636 C642B5        3410    $IF @CHAT .H. COLON$ GOTO TSTS$1
 9639 7659
 963B BF9073        3411    DST  #PRSEM,*SUBSTK     Expect it to be a ";"
 963E 837A
 9640 0683C5        3412    CALL TSTINT             Test for INTERNAL files
 9643 5659          3413    BR   TSTS$1             Treat all separators as ";"
 9645 D642B3        3414    $IF @CHAT .EQ. COMMA$ THEN                                1
 9648 564F
 964A BF9073        3415      DST  #PRTCOM,*SUBSTK                                    1
 964D 835C
                    3416    $END IF
 964F D642B5        3417    $IF @CHAT .EQ. COLON$ THEN                                1
 9652 5659
 9654 BF9073        3418      DST  #PRCOL,*SUBSTK                                     1
 9657 8377
                    3419    $END IF
                    3420  TSTS$1
 9659 00            3421    RTN
                    3422  *
                    3423  *       PARFN - Parse string expression and create PAB
                    3424  *       automatically continue in CSTRIN for copy
                    3425  *       string to PAB
                    3426  *            Exit on non-string values
                    3427  *
                    3428  PARFN
                    3429    $
                    3430    $     First evaluate string expression
                    3431    $
 965A 0F74          3432    XML  PARSE              Parse up to next comma
 965C B3            3433    DATA COMMA$
 965D D64C65        3434    $IF @FAC2 .NE. STRVAL GOTO ERRSNM Check for "STRING"
 9660 57BF
 9662 BD0250        3435    DST  @FAC6,@MNUM        Copy length byte in MNUM
 9665 A2030E        3436    ADD  PABLEN,@MNUM+1     Account for PAB length + control
 9668 0F77          3437    XML  VPUSH              Save start of string somewhere
 966A BD4A02        3438    DST  @MNUM,@FAC         Setup for MEMCHK - check for memory
 966D 0F72          3439    XML  MEMCHK                                      overflow
 966F 77C7          3440    BS   ERRMEM             * MEMORY FULL
 9671 0F78          3441    XML  VPOP               Restore all FAC information again
 9673 A54002        3442    DSUB @MNUM,@FREPTR      Update free word pointer
 9676 BD0440        3443    DST  @FREPTR,@PABPTR    Assign PAB entry address
 9679 9104          3444    DINC @PABPTR            Correct for byte within PAB
 967B 86B004        3445    CLR  RAM(@PABPTR)       Clear PAB plus control info
 967E 35000D        3446    MOVE PABLEN-1 FROM RAM(@PABPTR) TO RAM(1(PABPTR)) Rip
```

```
9681 EC0104
9684 B004
9686 BCE003  3447     ST    @MNUM+1,RAM(OFS(PABPTR)) Save length of PAB
9689 0403
968B BC0251  3448     ST    @FAC7,@MNUM        Compute # of bytes in name
968E BCE00D  3449     ST    @FAC7,RAM(NLEN(PABPTR)) Store name length
9691 0451
9693 BCE002  3450     ST    @FNUM,RAM(FIL(PABPTR)) Copy file number in PAB
9696 0417
9698 BD0804  3451     DST   @PABPTR,@CCPADR    Get start address for string dest.
969B A30800  3452     DADD  NLEN+1,@CCPADR     Add offset to actual start address
969E 0E
             3453     $
             3454     $     TRICKY - OPTFLG also resets offset added in CSTRIN
             3455     $
969F 8617    3456     CLR   @OPTFLG            Clear all option flags
96A1 0F84    3457     XML   IO                 CSTRIN I/O utility
96A3 02      3458     DATA  CSTRIN
96A4 00      3459     RTN
             3460
             3461  *>>>>>REMOVED 2/7/80 BECAUSE OF XML.  SRH
             3462  *CSTRIN
             3463  * $WHILE @MNUM .NE. 0 THEN Watch out for empty names
             3464  *    ST    @DSRFLG,RAM(@CCPADR) Store offset
             3465  *    ADD   RAM(@FAC4),RAM(@CCPADR) Add in character
             3466  *    DINC  @FAC4           Get next address for input
             3467  *    DINC  @CCPADR            and for output
             3468  *    DEC   @MNUM           Decrement counter
             3469  * $SEND WHILE
             3470  * RTN
```

```
                    3472  ********************************************************************
                    3473  *                          O U T R E C                             *
                    3474  *       OUTREC and INITRC are used to output a record to           *
                    3475  *       either screen or external I/O devices, and to              *
                    3476  *       initiate pointers for further I/O                          *
                    3477  ********************************************************************
                    3478  OUTREC
 96A5 BC0307        3479    ST  @RECLEN,@MNUM+1          Compute number of character
 96A8 9003          3480    INC @MNUM+1                    positions we should fill
 96AA 8E1776        3481    $IF @DSRFLG .NE. 0 THEN      Screen I/O                      1
 96AD D3
 96AE 0F84          3482      XML   IO                 Fill the remainder of the record  1
 96B0 01            3483      DATA  FILSPC               with appropriate fillers        1
 96B1 DA0408        3484      $IF .BIT3 @PABPTR .EQ. 1 GOTO RTC block output on size     1
 96B4 53BC
 96B6 DA0404        3485      $IF .BIT2 @PABPTR .EQ. 1 THEN "AT CLAUSE USED"             2
 96B9 76CC
                    3486  *
                    3487  *          Next test for xing the end of screen
                    3488  *
 96B8 A30800        3489        DADD 4, @CCPADR                                          2
 96BE 04
 96BF CA0803        3490        $IF @CCPADR .L. 3 GOTO INIT$1                            2
 96C2 56C8
 96C4 BF0800        3491        DST 2,@CCPADR        restart at upper left hand corner   2
 96C7 02
                    3492  INIT$1
 96C8 BE0601        3493        ST 1,@CCPPTR         reset current column pointer        2
 96CB 00            3494        RTN                                                      2
                    3495      $END IF
                    3496  SCRO
 96CC 0F83          3497      XML  SCROLL              scroll the screen one line        1
 96CE BE0601        3498      ST   1,@CCPPTR           reinitialize CCPPTR               1
 96D1 5754          3499      BR  INTKBO               and reinitialize                  1
                    3500    $END IF                    provide external I/O stuff
                    3501    $  This is also entry for last record output
 96D3 DAE005        3502    $IF .BIT4 RAM(FLG(PABPTR)) .EQ. 0 THEN FIXED records       1
 96D6 041056
 96D9 E2
 96DA BC0307        3503      ST   @RECLEN,@MNUM+1 Ready for space filling              1
 96DD 9003          3504      INC  @MNUM+1         Move to first position outside rec   1
 96DF 0F84          3505      XML  IO              And do it up to end of record        1
 96E1 01            3506      DATA FILSPC                                               1
                    3507    $END IF
 96E2 9206          3508    DEC  @CCPPTR           Update last character position
 96E4 BCE009        3509    ST   @CCPPTR,RAM(CNT(PABPTR)) Store # of characters
 96E7 0406
 96E9 86E003        3510    CLR  RAM(OFS(PABPTR))  Undo pending record offsets
 96EC 04
 96ED 06975E        3511    CALL IOCALL            Call DSR
 96F0 03            3512    DATA C$WRIT                  for WRITE mode
                    3513    $
 96F1 8609          3514    CLR  @CCPADR+1         Get address at bufferstart
 96F3 5701          3515    BR   PR$$O
                    3516  *
```

```
                    3517  *       PRINIT initializes the variables CCPADR, CCPPTR
                    3518  *       RECLEN and DSRFLG, for a given PABPTR.
                    3519  *
                    3520  PRINIT
96F5 8617           3521    CLR   @DSRFLG              Indicate external I/O in DSRFLG
96F7 BC07E0         3522    ST    RAM(LEN(PABPTR)),@RECLEN  Pick up record length
96FA 0804
96FC BC09E0         3523    ST    RAM(OFS(PABPTR)),@CCPADR+1 Get offset in record
96FF 0304
                    3524  PR$$0
9701 BC0609         3525    ST    @CCPADR+1,@CCPPTR Compute columnar position
9704 9006           3526    INC   @CCPPTR             And convert from offset
9706 8608           3527    CLR   @CCPADR             Clear upper byte
9708 A108E0         3528    DADD  RAM(BUF(PABPTR)),@CCPADR Compute actual address
970B 0604
970D 00             3529    RTN
```

```
*>>>>>>Removed 2/7/80 because XML to do it. SRH
*
*****************************************************************
*      FILSPC - Fill a record with spaces, starting from        *
*      the current position at CCPADR to the position           *
*      indicated in @MNUM+1 (1 byte)                            *
*****************************************************************
*FILSPC
*  $IF @MNUM+1 .EQ. 0 THEN      In case record length = 255
*                               (255+1, overflow to be 0)
*    $IF @CCPPTR .NE. 0 THEN
*      B FILS$2                 Skip the following "compare "
*    $SELSE
*      B FILS$3                 Just RTN
*    $END IF
*  $END IF
*  $IF @MNUM+1 .H. @CCPPTR THEN Make sure difference >=1
*FILS$2
*   SUB  @CCPPTR,@MNUM+1 Compute the # of bytes to move
*   ADD  @MNUM+1,@CCPPTR Actually move pointer ahead
*   ST   @DSRFLG,@MNUM   Assume zero filling
*   CALL TSTINT          Which would be correct for
*   BR   FILS$1            INTERNAL type files
*   ADD  SPACE,@MNUM     Make that space filling
*FILS$1
*   ST   @MNUM,RAM(@CCPADR) Fill with fillers
*   DINC @CCPADR         Next address to be treated
*   DEC  @MNUM+1         Count that space too
*   BR   FILS$1          One space is enough !!
* $END IF
*FILS$3
* RTN                    Return after completing move
```

```
                    3564  ********************************************************************
                    3565  *       CSTRNG - Copy the value of the string expression           *
                    3566  *       to the screen                                              *
                    3567  ********************************************************************
                    3568  CSTRNG
970E BC0C51         3569    ST    @FAC7,@BYTE          Pick up the string length
9711 8E0C77         3570    $WHILE @BYTE .NE. 0        Output as many records as required
9714 3E
                    3571      $
                    3572      $   CHKREC check available space in current record.
                    3573      $   If the string to be output is too long, it
                    3574      $   is chunked up into digestable pieces.  If the
                    3575      $   current record is partly filled up, it is
                    3576      $   output before any chunking is done.
                    3577      $
                    3578  CHKREC
9715 BC0306         3579      ST    @CCPPTR,@MNUM+1 Use MNUM for current offset ind.
                    3580  CHKR$0
9718 BC0207         3581      ST    @RECLEN,@MNUM    Compute remaining area
971B A40206         3582      SUB   @CCPPTR,@MNUM      between column and end
971E 9002           3583      INC   @MNUM            Also count current column
9720 C8020C         3584      $IF @MNUM .L. @BYTE THEN Won't fit in current record
9723 7730
9725 D60301         3585        $IF @MNUM+1 .EQ. 1 GOTO CHKR$1 Unused record
9728 7733
972A 0696A5         3586        CALL OUTREC          Output whatever we have
972D 5715           3587        BR    CHKREC         And try again
972F 00             3588        RTN
                    3589      $END IF
9730 BC020C         3590      ST    @BYTE,@MNUM      Use actual count if fit
                    3591  CHKR$1
9733 A40C02         3592      SUB   @MNUM,@BYTE      Update remaining chars count
9736 A00602         3593      ADD   @MNUM,@CCPPTR    Also new column pointer
9739 0F84           3594      XML   IO               Copy string to output
973B 02             3595      DATA CSTRIN
973C 5711           3596    $END WHILE               Continue as long as needed
973E 00             3597    RTN
                    3598  ********************************************************************
                    3599  *       INITKB - Initialize the variables needed                   *
                    3600  *       for keyboard output                                        *
                    3601  ********************************************************************
                    3602  INITKB
973F 8604           3603    CLR   @PABPTR            Don't use any DISPLAY options
9741 BE1760         3604    ST    OFFSET,@DSRFLG     Load for correction of screen chrs
9744 BE0601         3605    ST    1,@CCPPTR          Assume un-initialized XPT
9747 C67F02         3606    $IF XPT .H. 2 THEN      *** Patch for un-initialized XPT
974A 5751
974C BC067F         3607      ST    XPT,@CCPPTR      Initialize CCPPTR
974F 9606           3608      DECT @CCPPTR           Correct for incorrect XPT offset
                    3609    $END IF
9751 BE071C         3610    ST    VWIDTH,@RECLEN     Get video screen width
                    3611  INTKB0
9754 BC0906         3612    ST    @CCPPTR,@CCPADR+1  Initialize screen address
9757 8608           3613    CLR   @CCPADR            Clear upper byte CCPADR
9759 A30802         3614    DADD SCRNBS+1,@CCPADR    Add start-address plus compensate
```

```
975C E1
975D 00          3615    RTN
                 3616  *
                 3617  IOCALL
975E 8856        3618    FETCH @FAC12            I/O code to FAC12 (BUG!!!!)
9760 BCE004      3619    ST    @FAC12,RAM(COD(PABPTR)) Pick up the I/O code
9763 0456
                 3620  IOCL$1
9765 069763      3621    CALL CDSR               Call the DSR routine
9768 5797        3622    BR   ERR$2              Give I/O error on error
976A 00          3623    RTN                     Or else return
                 3624  *
                 3625  *       DSR CALL ROUTINE - NORMAL ENTRY
                 3626  *
                 3627  CDSR
976B BEE00C      3628    ST    OFFSET,RAM(SCR(PABPTR)) Always set screen offset
976E 0460
9770 35001E      3629    MOVE 30 FROM @FAC TO RAM(>3C0) Save FAC area
9773 A3C04A
9776 BD5604      3630    DST  @PABPTR,@FAC12     Get PAB pointer in FAC
9779 A35600      3631    DADD NLEN,@FAC12        Compute name length entry
977C 0D
977D B2E005      3632    AND >1F,RAM(FLG(PABPTR)) Clear error bits for ON ERROR
9780 041F
                 3633  *                              time, I/O process can still be continued
9782 060010      3634    CALL CALDSR             Call actual DSR link routine
9785 08          3635    DATA 8
9786 35001E      3636    MOVE 30 FROM RAM(>3C0) TO @FAC
9789 4AA3C0
                 3637  *            MOVE does not affect status
978C 7793        3638    BS   CDSR$0             ERROR = ERROR = ERROR = .....
978E DAE005      3639    CLOG >E0,RAM(FLG(PABPTR)) Set COND if no error
9791 04E0
                 3640  CDSR$0
9793 01          3641    RTNC
```

```
                    3643  *
                    3644  *                E R R O R   M E S S A G E S
                    3645  *
                    3646  ERR$2B
9794 0680FB         3647    CALL CLRFRE             Undo allocation of PAB
                    3648  ERR$2
                    3649  * First check is it error coming from AUTOLD
                    3650  * If it is  then do not print the error message
                    3651  * and go back to TOPL02
9797 310002         3652    MOVE 2 FROM ROM(#TOPL02) TO RAM(AUTTMP)
979A A39460
979D 30
979E D5808A         3653    $IF @RSTK+2 .DEQ. RAM(AUTTMP) THEN                    1
97A1 A39457
97A4 A9
97A5 BE738A         3654      ST RSTK+2,@SUBSTK                                   1
97A8 00             3655      RTN                                                 1
                    3656    $END IF
                    3657  ********************************************************
                    3658  *       Next code is to avoid recursion of errors in   *
                    3659  *       CLSALL routine.  If this entry is taken from   *
                    3660  *       CLSALL, the stack will contain CLSLBL as a     *
                    3661  *       returnaddress in the third level.              *
                    3662  ********************************************************
97A9 A67304         3663    SUB  4,@SUBSTK          Back down two levels
97AC D79073         3664    $IF *SUBSTK .DEQ. CLSLBL THEN                         1
97AF 81E957
97B2 B8
                    3665  WRNIO
97B3 066A82         3666      CALL WARN$$           Give warning to the user      1
97B6 23             3667      DATA 35               * I/O ERROR  but warning      1
97B7 00             3668      RTN                   And return to close routine   1
                    3669    $END IF
97B8 A27304         3670    ADD  4,@SUBSTK          Back up two levels for OLD/SAVE
                    3671  ERRIO
97BB 066A84         3672    CALL ERR$$
97BE 24             3673    DATA 36                 * I/O ERROR
                    3674  *
                    3675  *
                    3676  *       ERROR messages called in this file
                    3677  *
97BF 066A84         3678  ERRSNM CALL ERR$$             * STRING-NUMBER MISMATCH
97C2 07             3679         DATA 7
                    3680  *
97C3 066A84         3681  ERRIM  CALL ERR$$             * IMAGE ERROR
97C6 0A             3682         DATA 10
                    3683  *
97C7 066A84         3684  ERRMEM CALL ERR$$             * MEMORY FULL
97CA 0B             3685         DATA 11
                    3686  *
97CB 066A84         3687  ERRBV  CALL ERR$$             * BAD VALUE
97CE 1E             3688         DATA 30
                    3689  *
97CF 066A84         3690  ERRINP  CALL ERR$$
97D2 20             3691          DATA 32             * INPUT ERROR
```

```
                    3692  *
97D3 066A84         3693  ERRDAT  CALL ERR$$
97D6 21             3694          DATA 33                 * DATA ERROR"
                    3695  *
97D7 066A84         3696  ERRFE   CALL ERR$$              * FILE ERROR
97DA 22             3697          DATA 34
                    3698  *
97D8 066A84         3699  ERRPV   CALL ERR$$              * PROTECTION VIOLATION
97DE 27             3700          DATA 39
97DF 066A84         3701  ERRMUV  CALL ERR$$              * IMPROPERLY USED NAME
97E2 09             3702          DATA 9
                    3703  *
                    3704  **
                    3705  *       Other errors called in this file
                    3706  **
                    3707  * ERRSYN     * SYNTAX ERROR    DATA 3
                    3708  * ERRST      * STRING TRUNCATED ERROR     DATA 19
                    3709  * WRNNPP     * NO PROGRAM PRESENT         DATA 29
                    3710  * WRNINP     * INPUT ERROR       WARNING  DATA 32
                    3711  * ERRIO      * I/O ERROR                  DATA 36
                    3712  * WRNIO      * I/O ERROR         WARNING  DATA 36
                    3713  * WRNSNM     * STRING NO. MISMATCH WARNING DATA 7
                    3714  *
                    3715  ********************************************************
                    3716  *                                                      *
                    3717  *  The following section has been added to fix         *
                    3718  *  bugs in INPUT, ACCEPT and LINPUT statements         *
                    3719  ********************************************************
                    3720  BUG01
97E3 CA4280         3721       $IF @CHAT .HE. >80 GOTO ERRSYN  Make sure of var. name.
97E6 6109
97E8 0F7A           3722       XML  SYM                  Get the information of the
97EA 0F7B           3723       XML  SMB                        variable.
97EC 00             3724       RTN
                    3725          END
```

ERRORS= 0

LENGTH= 6125 (>17ED)

517 SYMBOLS USED

SYMBOL	VALUE	DEF	REFERENCE TABLE
A$1	8A86	1883	1881
AAA	004C	169	2103 2128
ABS$	00CB	271	
ACCEP$	00A4	231	
ACCEPT	8968	1751	385
ACCNM	85E5	1310	1215 1221
ACCP$1	8A47	1848	1831
ACCP$2	8A8F	1891	1894
ACCP$3	8ACB	1916	1920
ACCP$4	8B36	1963	1960
ACCP$5	8A7C	1877	1946
ACCP$6	8B2A	1954	1930
ACCP$7	8A74	1871	1866
ACCP$8	8AFC	1934	1904
ACCP$9	8A4F	1854	1860 1951
ACCTRY	03B7	347	1752 1866 1943
ACCVRA	03AE	341	1873 1942
ACCVRW	03AC	340	1872 1941
ALL$	00EC	296	3241
APPEN$	00F9	309	
ARG	005C	175	176 177 178 179 726 730 920 921 924 925 926 927 1123 1126 1127 1374 1376 1377 1378 1379 1380 1769 1772 1773 1799 1800 1801 1804 1805 2235 2792 2794 2797 2811 2818 2819 2820
ARG1	005D	176	1384 1385
ARG2	005E	177	1124 1128 1814 1815 1817 1863 2815 2816 2819 2820 2822 2824
ARG6	0062	178	2841 2843
ARG7	0063	179	1755 1810
ASSGNV	007C	40	1392 1572 1675 1955 1999
AT$	00F0	300	3257
ATN$	00CC	272	
AUTO1	602E	64	2619
AUTTMP	0394	337	3652 3653
BASE	0043	149	3162
BASE$	00F1	301	
BBB	0050	171	2100 2101 2168 2170 2183
BBB1	000C	90	2383 2462
BEEP$	00EE	298	3251
BKGD	0020	333	3242
BREAK	0002	320	2710
BREAK$	008E	209	
BUF	0006	96	668 672 1355 1430 1459 1583 1640 1652 2068 2129 2378 2404 2691 3096 3138 3141 3528
BUFLEV	0046	152	
BUG01	97E3	3720	1358 1560 1631 1868 1978
BYTE	000C	109	1099 1140 1141 1163 1164 1291 1292 1361 1363 1368 1371 1372 1390 1391 1569 1639 1641 1644 1649 1650 1651 1656 1666 1797 1907 1911 2936 2948 3383 3384 3386 3569 3570 3584 3590 3592
C$CLOS	0001	356	711 748 2196 2456 2541 2599 2724
C$DELE	0007	362	692 717
C$LOAD	0005	360	2069
C$OPEN	0000	355	
C$READ	0002	357	855 1411 1687 2134 2176 2187 2576 2593

SYMBOL	VALUE	DEF	REFERENCE TABLE
C$REST	0004	359	778
C$SAVE	0006	361	2387
C$SCR	0008	363	
C$STAT	0009	364	2815
C$WRIT	0003	358	858 2425 2443 2453 2533 2539 3078 3512
CALDSR	0010	56	3634
CALL$	009D	224	
CB	0008	1	
CCC	004E	170	2099 2181 2192
CCC1	0008	89	2160 2184 2188 2191 2192 2384 2435 2436 2437 2445 2447 2449 2451 2463
CCPADR	0008	105	929 930 947 948 949 1023 1587 1590 1714 1715 1723 1725 1726 1727 1812 1814 1842 1849 1856 1858 1861 1862 1949 2677 2681 2683 2690 2691 3098 3099 3100 3101 3109 3116 3127 3129 3131 3134 3137 3138 3141 3146 3151 3183 3190 3246 3273 3283 3284 3451 3452 3489 3490 3491 3514 3523 3525 3527 3528 3612 3613 3614
CCPPTR	0006	103	925 931 946 950 970 974 989 1023 1024 1041 1042 1049 1468 1497 1499 1519 2237 2250 2698 3127 3145 3182 3245 3282 3360 3361 3374 3493 3498 3508 3509 3525 3526 3579 3582 3593 3605 3607 3608 3612
CDSR	976B	3627	643 721 749 2070 2130 2406 3621
CDSR$O	9793	3640	3638
CFI	0012	57	3039
CHAR	0004	1	
CHAT	0042	148	442 451 452 453 454 455 503 563 712 769 843 871 884 891 964 1075 1077 1079 1117 1137 1161 1194 1334 1394 1443 1620 1702 1711 1759 1761 1767 1768 1770 1772 1791 2011 2022 2033 2037 2304 2306 2561 2580 2581 2585 2669 2678 2681 2836 2860 2861 2870 2901 2905 2911 3012 3014 3213 3215 3236 3251 3257 3289 3292 3306 3323 3324 3326 3337 3338 3341 3345 3409 3410 3414 3417 3721
CHECK	810D	623	442 505
CHKCNV	935C	3036	515 2839 3198 3222 3299
CHKCON	9371	3045	426 708 772 846 1680 2842
CHKEND	95AD	3322	388 504 564 624 685 719 785 863 868 1017 1118 1298 1303 1396 1558 1574 2909 2919 2924 3013 3335 3405
CHKF$1	937B	3057	3064
CHKFN	9355	3025	424 706 770 844 1335 1621
CHKNUM	92EC	2954	1500 1565
CHKPAR	809F	512	506 566
CHKR$O	9718	3580	1166 1295
CHKR$1	9733	3591	3585
CHKREC	9715	3578	3587
CHKRM	8834	1586	1713 1722
CHKS$O	932F	2993	1986
CHKSEP	8328	953	972 974 978
CHKSTR	9323	2988	1098 1523 1568 1995
CHR$$	00D6	282	
CIF	0080	45	2798
CIRCU$	00C5	264	
CLOS$1	81C7	725	722

SYMBOL	VALUE	DEF	REFERENCE TABLE
CLOSE	8196	705	375
CLOSE$	00A0	227	
CLRFRE	80FB	608	616 635 3647
CLSA$0	81E1	744	753
CLSALL	81F4	751	379 2670 2863
CLSLBL	81E9	747	3664
CNS	0073	32	939 1257
CNT	0009	98	1362 1398 1401 1420 1428 1639 1650 2136 2180 2191 2423 2438 2451 2531 2537 2580 2597 3509
CNVD$0	94C1	3201	3199
CNVDEF	94B8	3197	967 3268 3279
COD	0004	94	711 717 748 855 858 1411 1687 2069 2816 2819 3078 3619
COLON$	00B5	248	433 712 834 881 907 1114 1117 1691 1702 1706 1711 1718 1822 3030 3221 3410 3417
COMMA$	00B3	246	442 503 514 563 871 1137 1161 1194 1394 1415 1443 1527 1791 2001 2011 2306 2923 3004 3213 3264 3267 3409 3414 3433
COMMOD	961A	3391	969 3270 3281
CONPRT	82C9	889	1018
CONT	0075	34	673 693 724 779 812 1043 1055 1405 1578 1676 1964 2012 2799 2827 2830
CONVER	A012	79	1992 2960
COS$	00CD	273	
CPTEMP	03BC	351	1468 1519
CPUBAS	A040	22	2100 2161 2168
CRNBUF	0820	327	1409 1493 1552 1736 2503 2506 2513 2522 2527 2529 2536 2544 2577 2583 2587 2587 2594 2741 2742 2742 2753 2758
CRUNCH	007F	44	1433 1477
CSNTMP	0390	335	1993 2964
CSNTP1	03BA	336	1505
CSTRIN	0002	50	3458 3595
CTMPST	92DE	2945	1369 1570 1674
CTSTR	92CD	2933	1101 2946 3385
CTSTRO	92D1	2935	1915
CURINC	000E	111	2617 2631 2636 2643 2649 2650 2656 2662 2665 2719
CURLIN	0014	114	1144 1149 1150 1154 1156 1157 1158 1172 1175 1178 1179 1180 1181 1182 1184 1185 1186 1196 1200 1204 1205 1206 1207 1210 1266 1268 1272 1276 1277 1279 1283 1285 1291 1408 1464 1479 1492 1507 1511 1577 1735 2616 2624 2630 2649 2650 2652 2655 2666
DATA	0034	141	1089 1092 1100 1492 1493 1507 1511 1552 1577 1981 2004 2021 2036 2885 2957 2959 2964 2975 2979 2983 2997 2998 3008
DATA$	0093	214	2019
DATAST	A008	78	811 2008
DATEND	934B	3011	3005
DDD1	0054	172	1107 2431 2444 2464 2500 2506 2507 2517 2625 2626 2628 2744 2754 2767 2774 2779 2876 2979
DEF$	0089	204	
DELET	817C	682	371
DELET$	0099	220	716
DELP$1	9440	3148	3185
DELP$2	944C	3151	3149

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

DELPAB  93A6   3092    723  727  750
DFLAG   0001    120    592  593
DIGIT$  00E9    293
DIM$    008A    205
DISACC  94EF   3233    822 1753
DISP$1  94F2   3235   1795 3249 3255 3287 3304
DISPL$  00A2    229    451
DISPL1  8257    821    370
DISPLA  0009      1
DIVI$   00C4    263
DSRFLG  0017    116    948 1020 1030 1040 1061 1502 1510 1635 1692 2675
                      2721 2841 2843 2926 3077 3481 3521 3604
EDGECH  007F    334   1663 1910 1918
EDITLN  6032     67   2589
EEE1    0058    174    805 1092 1108 2264 2266 2405 2465 2466 2503 2512
                      2513 2630 2636 2662 2741 2747 2749 2750 2753 2774
                      2782 2877 2878 2981
ELSE$   0081    196
EMPSTR  933E   3003   2991
END$    008B    206
ENLN    0032    140    789  795 2083 2143 2144 2226 2232 2259 2334 2371
                      2374 2384 2414 2417 2463 2494 2625
EOF     91DE   2810    384
EOF$2   9212   2828   2796 2823
EOLEX   8395   1029    523  864  869 1120 1299 3407
EQUAL$  00BE    257
ERASE$  00EF    299   3236
ERR$$   6A84     73    618 1784 3672 3678 3681 3684 3687 3690 3693 3696
                      3699 3701
ERR$2   9797   3648   2210 3622
ERR$2B  9794   3646    644 2131 2407 2577 2597
ERRBV   97CB   3687   2840 3040 3050 3223 3300 3301
ERRCOD  0022    132   1435 1480
ERRDAT  97D3   3693    806 1981 1993 1996 2002
ERRFE   97D7   3696    425  427  636  707  709  773  847  853  876  927
                      962 1371 1376 1385 1625 1681 1685 2814 3216
ERRIM   97C3   3681   1131 1250 1253
ERRINP  97CF   3690   1435 1510
ERRIO   97BB   3671    731 2693 2696
ERRMEM  97C7   3684    670 3440
ERRMUV  97DF   3701   1633
ERRPV   97DB   3699   2299 2558 2615
ERRSNM  97BF   3678   1707 1780 1984 3037 3434
ERRST   89BC   1784
ERRSYN  8109    617    720  955  964 1119 1306 1754 1761 1796 2306 2309
                      2310 2311 2313 2317 2318 2319 2320 2321 2322 2324
                      2562 2836 2861 2901 2910 2926 3237 3252 3258 3290
                      3292 3721
EXP$    0076    185
EXP$$   00CE    274
EXTRAM  002E    138   1081 1085 2655 2660 2663 2665 2666 2716 2718 2719
                      2739 2757 2758
FAC     004A    154    155  156  157  158  159  160  161  162  163  164
                      165  166  167  168  169  170  171  172  173  174
                      507  571  654  768  771  805  845  920 1077 1151
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
			1336 1372 1373 1379 1622 2126 2127 2128 2129 2139
			2140 2141 2142 2143 2146 2147 2149 2154 2156 2402
			2403 2404 2405 2410 2411 2412 2413 2414 2415 2416
			2417 2419 2421 2422 2583 2652 2656 2687 2690 2693
			2696 2797 2821 2825 2829 2939 3041 3042 3050 3154
			3155 3160 3165 3173 3174 3175 3200 3223 3224 3283
			3295 3296 3301 3342 3343 3361 3362 3363 3364 3392
			3396 3438 3629 3636
FAC1	004B	155	572 970 974 975 1079 1378 2689 3046 3271 3272
			3273 3282 3302 3341 3393 3394
FAC10	0054	164	1271 1272 3038 3040
FAC11	0055	165	921 938 1202 1208 1211 1212 1231 1234 1246 1249
			1256 1263 1265 1266 1273 3386
FAC12	0056	166	919 1216 1223 1989 2584 2587 2957 3383 3618 3619
			3630 3631
FAC13	0057	167	1203 1222 1263 1266 2585
FAC14	0058	168	2716 2757
FAC2	004C	156	918 934 968 1081 1085 1088 1091 1094 1115 1245
			1367 1387 1564 1633 1707 1780 1899 1927 1928 1983
			2019 2033 2523 2524 2525 2527 2678 2679 2684 2934
			3037 3156 3157 3163 3166 3173 3175 3177 3269 3280
			3392 3394 3434
FAC3	004D	157	2971 2973 2976
FAC4	004E	158	1123 1148 1149 1150 1172 1283 1804 2938 3159 3163
			3164
FAC5	004F	159	1292 3161 3162
FAC6	0050	160	1099 1102 1104 1106 1140 1145 1149 1152 1175 1282
			1283 1285 1368 1569 1804 1805 1987 1990 2494 2495
			2500 2522 2523 2534 2535 2936 2989 2995 2998 3158
			3164 3172 3178 3435
FAC7	0051	161	927 929 1120 1122 1124 1163 1273 1374 1781 1782
			2947 2996 3448 3449 3569
FAC8	0052	162	1089 1100 1201 2330 2332 2333 2334
FAC9	0053	163	1216 1219 1222 1223 1225 1247 1250 1253 1312
FFF1	0056	173	1106 2440 2449 2467 2468 2469 2504 2509 2518 2750
			2766 2767 2771 2778 2779 2875 2978
FIGURE	0002	1	
FIL	0002	92	3062 3450
FIL$$	8120	634	650
FILSPC	0001	49	977 998 3483 3506
FIXED$	00FA	310	
FLAG	0045	151	2079 2081 2150 2152 2299 2558 2615 2858
FLG	0005	95	483 525 544 552 584 602 632 633 640 852
			853 876 1062 1348 1625 1685 2400 2572 3216 3502
			3632 3639
FLOAT1	9216	2832	2821
FNUM	0017	115	116 117 683 3046 3062 3450
FOR$	008C	207	
FREPTR	0040	147	612 671 690 728 2365 2474 2475 3126 3190 3442
			3443
GETDAT	9308	2972	378 2994
GETGFL	9304	2970	1097 1982 2000
GETRAM	930C	2974	1498 1526 1563 1573 2956
GETSTR	0071	30	1143 1798 2937
GETV$0	9292	2900	2925

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

GETV$1  929B  2904     2915
GETV$2  92CC  2927     2920
GETVAR  928A  2893     1407 1473
GO$     0085   200
GOSUB$  0087   202
GOTO$   0086   201
GPNAME  9235  2857     2060 2300 2557
GREAD1  008C    53     2511 2772 2980
GREAT$  00C0   259
GRMLST  9179  2737      391 2705
GRSUB2  91A4  2765      392 1090 2262 2748
GRSUB3  91BC  2777      393  801 2627 2633 2659 2738
GRSUB4  91AA  2769     2745 2781
GSAV1   8F44  2454     2446
GSAVE   8EB7  2392     2346 2355
GVMOV   8F4A  2461     2359
GVWITE  008B    52     1109 2441 2450 2470 2505 2519 2756 2879
GWRITE  0086    54
GWSUB   6036    68     2265
IF$     0084   199
IMAGE$  00A3   230     1094
INIT$1  96C8  3492     3376 3378 3490
INITKB  973F  3602      842 1333 1466 1619 2701 3234
INITPG  6014    61     2209
INP$2   8768  1472     1342
INP$3   86CC  1416     1413
INP$31  86E9  1426     1412
INP$32  8728  1447     1458
INP$33  8765  1469     1521
INP$37  8927  1710     1701
INP$39  8936  1716     1709
INP$65  87F6  1556     1575
INP$67  8818  1571     1566
INPU$2  892D  1713     1341
INPU$3  876B  1474
INPU$4  8790  1495     1533
INPU$5  87B4  1509     1501 1524 1529 1530
INPU$6  87D4  1525     1502 1505
INPU$7  8822  1576     1559
INPUT   85F0  1332      373
INPUT$  0092   213      452
INPUTP  03AA   339     1337 1338 1467 1518
INSU1   88E8  1679     1344 1623
INSUB1  8906  1697     1471 1627
INSUB2  893B  1722     1475 1655
INT$    00CF   275
INTER$  00F5   305
INTKBO  9754  3611     3499
INTR$0  861B  1350     1398
INTR$1  862D  1357     1399
INTR$2  869F  1400     1397
IO      0084    47      976  997 3457 3482 3505 3594
IOCALL  975E  3617      691  777 2133 2175 2186 2195 2386 2424 2442 2452
                       2455 2532 2538 2540 2575 2592 2598 2723 3511
IOCL$1  9765  3620     1351 1417 1637 2493 2573 2686 2817
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
IOSTRT	003C	145	658 659 661 752 753 3056 3102 3103 3114 3115 3116 3117 3119
KILSYM	6022	63	2212 2389 2864
LEN	0008	97	572 650 651 2688 3522
LEN$	00D5	281	
LESS$	00BF	258	
LET$	008D	208	
LINE	A006	77	787
LINP10	885F	1629	1622
LINPUT	8841	1619	395
LIST	9068	2614	382
LIST$0	908D	2632	2647
LIST$1	90FB	2680	2685
LIST$3	914E	2714	2709
LIST$4	915F	2720	2710
LIST$5	9148	2711	2713
LITS05	A002	76	1988 1997
LLIST	6A74	70	2707
LN$	00C9	269	1075 3345
LNBUF	0036	142	795 796 803 806 807 810 2003 2006 2007
LOG$	00D0	276	
LPAR$	00B7	250	964 1761 2836 2905 3262 3292
LRTOP$	8D32	2211	2091 2111
LRTOPL	8EB1	2388	2457 2542 2600
LSUBP	0048	153	
M$EDIT	6010	25	60 61 62 63 64 65 66 67 68
M$EXEC	A000	24	76 77 78 79 80 81
M$PSCN	6A70	26	70 71 72 73 74
MEMCHK	0072	31	669 3439
MERG$1	905B	2595	
MERGE	9002	2556	390
MFLAG	0000	119	535 536
MINUS	002D	322	2618
MINUS$	00C2	261	
MNUM	0002	86	609 610 612 651 652 654 671 672 688 689 690 975 989 990 991 992 993 994 995 996 1165 1294 1494 1496 1497 2074 2075 2076 2077 2078 2144 2145 2147 2148 2149 2156 2157 2158 2159 2160 2161 2164 2224 2226 2227 2229 2230 2231 2235 2254 2255 2264 2268 2337 2338 2339 2341 3103 3104 3105 3105 3107 3108 3109 3110 3112 3125 3126 3128 3132 3435 3436 3438 3442 3447 3448 3479 3480 3503 3504 3579 3581 3582 3583 3584 3585 3590 3592 3593
MRGPAB	039E	338	2588 2591
MSGTA	6053	65	
MULT$	00C3	262	
MVDN	0088	48	2241 2247
NEWTOP	03BE	352	2087 2106 2197 2225 2239 2255 2261 2472
NEXT$	0096	217	
NLEN	000D	101	2063 2677 2870 2871 2873 2878 3099 3449 3452 3631
NUDTAB	0028	135	
NUM$	00C8	268	1984 2861 2955 2991 3338
NUMBE$	00FD	313	769 843 1334 1620 3028
NUMER$	00E8	292	1767 1770
NXTC$0	95C9	3339	3337

SYMBOL	VALUE	DEF	REFERENCE TABLE
NXTCHR	95BA	3334	1700
OFFSET	0060	331	334 1422 1448 1451 1456 1642 1658 1664 1714 1725 1856 1894 1923 2204 2544 2646 2647 2732 3242 3604 3628
OFS	0003	93	610 653 689 856 1042 1349 1353 1393 1402 1412 1418 1442 1460 1581 1636 1648 1653 3079 3219 3447 3510 3523
OL$$	8CDE	2167	
OLD	8BC4	2056	381
OLD$2	8D8E	2258	2271
OLD$3	8C63	2117	2071
OLD$5	8C32	2090	
OLD$7	8C5A	2108	2201
OLD$9	8D16	2194	2185
OLD1	8BCA	2059	389 2057
OLDC$0	8227	788	771 786
OLDC$1	823A	800	808
OLDCD	821F	784	769
OLDER	8D2D	2208	2078 2118 2136 2139 2149 2163 2164 2180 2191
OLDTOP	03BC	350	2085 2105 2157 2224 2254 2471
ON$	009B	222	
OPEN	8032	423	374
OPEN$	009F	226	
OPERR	8106	615	454 472 497 516 535 571 575 592 625 686
OPT$0	8072	471	
OPT$01	80BE	551	456
OPT$02	807C	482	457
OPT$03	80E5	583	458
OPT$1	8081	496	459
OPT$2	80A8	524	460
OPT$3	80AD	531	461 545 603
OPT$4	80B7	543	462
OPT$5	80C3	561	463
OPT$55	80DB	574	565
OPT$6	80EA	591	451
OPT$7	80F4	601	452
OPTFLG	0017	117	472 473 497 498 535 536 575 576 592 593 639 3456
OPTI$0	8047	441	508 577
OPTI$1	804C	446	503
OPTIO$	009E	225	
OPTION	8045	436	474 537 594
OSTRNG	970E	3568	936 941 1708
OTHE$0	8308	935	932
OTHE$1	8303	933	917
OUTEOF	9390	3076	710 746 774 1686 3218
OUTPU$	00F7	307	
OUTREC	96A5	3478	383 971 1006 1022 1035 1167 2722 3081 3369 3586
PAB	9170	2731	2676
PAB1	8FF9	2543	2492 2571
PAB3	8D24	2203	2125 2399
PABLEN	000E	326	2064 2867 3100 3436 3446
PABPTR	0004	87	483 507 525 544 552 572 584 602 610 632 633 640 645 650 651 653 659 666 668 668 672 689 711 717 726 728 729 730 743 743

SYMBOL	VALUE	DEF	REFERENCE TABLE
			745 748 752 775 827 852 853 855 856 858
			874 876 1021 1031 1032 1042 1048 1052 1062 1348
			1349 1353 1355 1362 1393 1398 1401 1402 1411 1412
			1418 1420 1428 1430 1442 1459 1460 1581 1583 1625
			1636 1639 1640 1648 1650 1652 1653 1685 1687 1762
			1819 1823 1826 1827 1828 1833 1855 1880 1935 1936
			1944 1946 1959 2062 2063 2065 2066 2067 2068 2069
			2125 2129 2136 2180 2191 2223 2225 2232 2233 2238
			2273 2378 2380 2381 2399 2400 2404 2423 2438 2451
			2492 2531 2537 2571 2572 2580 2588 2590 2591 2597
			2674 2676 2688 2691 2792 2794 2811 2816 2818 2865
			2866 2867 2867 2869 2870 2871 2873 2877 3056 3061
			3062 3063 3063 3078 3079 3096 3099 3101 3102 3104
			3107 3112 3114 3119 3135 3136 3137 3138 3139 3139
			3141 3147 3150 3151 3153 3155 3156 3181 3182 3183
			3184 3184 3216 3219 3224 3237 3244 3248 3252 3253
			3258 3285 3286 3290 3297 3302 3303 3310 3359 3362
			3376 3377 3443 3444 3445 3446 3446 3447 3449 3450
			3451 3484 3485 3502 3509 3510 3522 3523 3528 3603
			3619 3628 3630 3632 3639
PARFN	965A	3428	434 684
PARREC	94C2	3212	776 865 1688
PARSE	0074	33	513 906 965 1113 1198 1705 1778 2837 3029 3220
			3263 3274 3293 3432
PERMA$	00FB	311	
PFLAG	0002	121	472 473
PGMCHR	0079	37	437 447 499 562 713 872 963 1016 1076 1078
			1080 1195 1395 1440 1557 1704 1760 1766 1774 1976
			2026 2031 2061 2303 2560 2668 2673 2682 2868 2914
			3214 3217 3238 3254 3259 3291 3340 3349
PGMPTR	002C	137	435 1337 1439 1467 1518 1553 1554 1697 1703 1710
			2021 2036 2307 2309 2310 2311 2312 2317 2318 2319
			2320 2321 2322 2323 2667 2873 2876 2881 2894 3343
			3346
PLUS$	00C1	260	
POP	0005	1	
PR$$O	9701	3524	3515
PRCOL	8377	1005	3418
PREXIT	83A4	1039	1020 1021 1025
PRGFLG	0044	150	2023
PRIN$O	82B1	873	866
PRIN$1	82C4	883	827 835
PRINIT	96F5	3520	859 3080
PRINT	8266	841	372
PRINT$	009C	223	
PRN$10	82A5	867	845
PRSEM	837A	1015	3411
PRSM$1	8381	1019	1304
PRTAB	832D	960	891
PRTCOM	835C	988	3415
PRTNFN	00CE	190	1729
RAM	0001	1	483 507 525 544 552 572 584 602 610 632
			633 640 645 650 651 653 663 664 666 668
			672 689 711 717 726 730 743 745 748 775
			852 853 855 856 858 876 929 947 948 1042

SYMBOL VALUE DEF REFERENCE TABLE

			1062 1104 1104 1126 1146 1148 1149 1149 1154 1156
			1157 1178 1180 1181 1184 1185 1196 1200 1205 1206
			1207 1210 1217 1227 1229 1232 1266 1276 1283 1283
			1314 1337 1338 1348 1349 1353 1355 1362 1363 1372
			1393 1398 1401 1402 1411 1412 1415 1418 1420 1422
			1428 1430 1442 1448 1451 1456 1459 1460 1467 1468
			1497 1499 1505 1518 1519 1581 1583 1625 1636 1639
			1640 1642 1648 1650 1652 1653 1658 1662 1665 1685
			1687 1714 1725 1728 1752 1800 1804 1804 1808 1809
			1846 1849 1850 1856 1866 1872 1873 1882 1894 1910
			1918 1922 1922 1923 1937 1941 1942 1943 1949 1950
			1993 2063 2065 2066 2067 2068 2069 2074 2075 2076
			2078 2083 2084 2085 2085 2087 2105 2106 2125 2129
			2136 2139 2141 2143 2147 2149 2156 2157 2180 2191
			2197 2223 2224 2225 2239 2244 2245 2248 2254 2255
			2261 2268 2273 2302 2307 2309 2310 2311 2312 2317
			2318 2319 2320 2321 2322 2323 2325 2332 2367 2369
			2371 2373 2374 2375 2376 2378 2380 2381 2399 2400
			2404 2410 2412 2414 2416 2417 2418 2419 2422 2423
			2438 2451 2471 2472 2492 2506 2522 2522 2523 2525
			2527 2527 2529 2531 2536 2537 2571 2572 2577 2580
			2583 2587 2587 2588 2591 2594 2597 2646 2647 2676
			2681 2688 2691 2741 2742 2774 2797 2816 2819 2820
			2866 2867 2867 2870 2871 2873 2873 2948 2948 2964
			2975 3062 3063 3078 3079 3096 3099 3104 3105 3107
			3107 3108 3109 3110 3112 3114 3129 3129 3136 3137
RAMFLG	0089	188	2025 2027 2872
RAMFRE	0086	187	2109 2110
RAMPTR	000A	108	1409 1415 1657 1665 1667 1736
RAMTOP	0084	186	1103 2024 2091 2106 2118 2164 2197 2239 2245 2248
			2261 2328 2337 2382 2422 2435 2467 2471 2499 2692
			2704 2715 2770 2971 3149
RAND0$	0095	216	
READ	8B3A	1977	377
READ$	0097	218	
READL1	6A86	74	1885
READLN	6A76	71	1733
REC$	00DE	290	3215
RECENT	8827	1580	1419 1427 1441
RECLEN	0007	104	924 946 968 995 1410 1437 1438 1444 1445 1446
			1843 1850 1859 1950 2688 2689 3364 3374 3375 3376
			3377 3479 3503 3522 3581 3610
RELAT$	00F4	304	
RELO$1	8DB3	2272	2256 2259
RELOCA	8D36	2222	2088 2107 2200 2473
REM$	009A	221	
RESTO$	0094	213	
RESTOR	81FC	767	376
RETUR$	0088	203	
RFLAG	0004	123	575 576 639
RKEY	0075	183	2710
RND$	00D7	283	
RNM	000A	99	507 645 775 2065 2066 2067 2380 2381 2797 3224
ROM	000A	1	2125 2399 2492 2571 2676 2821 3652
RPAR$	00B6	249	966 1779 1794 2911 3275 3278

SYMBOL	VALUE	DEF	REFERENCE TABLE
RSTK	0088	189	3653 3654
RSTRIN	960B	3382	922 940
RTC	938C	3065	1061 3006 3041 3062 3215 3324 3336 3484
RTNG	0026	134	
SA$1	8E2B	2326	2305
SAPROT	03B9	349	2302 2325 2375 2418
SAVE	8D88	2298	380
SAVMG	8F76	2482	2314
SCR	000C	100	2820 3628
SCR$	883A	1588	1517
SCRNBS	02E0	332	1587 1590 3246 3614
SCRO	96CC	3496	3370
SCROLL	0083	46	1481 1589 1671 1961 3497
SEARCH	8B99	2020	1095
SEETWO	0003	43	1084 2654 2658
SEMIC$	00B4	247	1199 1302
SEQUE$	00F6	306	
SETVW	89A2	1773	1771
SFLAG	0003	122	497 498
SGN$$	00D1	277	
SIGN$	0075	184	
SIN$	00D2	278	
SIZCCP	03B4	344	1849 1949 2223 2273
SIZE$	00EB	295	3289
SIZE1	95DF	3358	1829 3311
SIZREC	03B6	345	1850 1950 2244 2245 2248
SIZXPT	03B8	348	1846 1937
SMB	007B	39	3723
SMTSRT	001E	130	
SPACE	0020	321	947 1658
SPEED	007E	41	431 714 832 879 1072 1083 1300 1689 1716 1792 1820 2653 2657 2921 3026 3239 3260 3265 3276
SPRITE	0006	1	
SQR$	00D3	279	
SRDA$0	8BBD	2035	2033
SRDA$1	8BB5	2032	2028
SRDATA	8B96	2018	386
SREF	001C	129	1104 1108 1144 1145 1146 1151 1799 1808 1908 1909 1910 1913 1922 1923 1925 1989 1990 1991 1993 2938 2939 2948 3386
SSEP$	0082	197	
STADDR	000A	107	661 663 664 664 666 1408 1464 1479 1553 1697 1698 1703 1710 1735 2062 2063 2064 2066 2068 2074 2075 2076 2078 2083 2084 2085 2086 2101 2170 2228 2236 2257 2259 2260 2263 2266 2268 2270 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2376 2378 2379 2381 2466 2871 2873 2875 2881 2894 3096 3097 3110 3117 3125 3129 3130 3134
STEP$	00B2	245	
STLN	0030	139	789 806 2006 2084 2097 2103 2109 2141 2145 2158 2227 2228 2230 2233 2257 2330 2338 2369 2373 2383 2412 2416 2431 2436 2462 2464 2468 2474 2535 2634 2696 3150
STOP$	0098	219	
STREND	001A	128	

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

STRIN$  00C7   266     2860 2990 3337
STRSP   0018   127
STRVAL  0065   329      918  934 1115 1564 1633 1707 1780 1927 1928 1983
                       3037 3434
STVSPT  0024   133     2672
SUB$    00A1   228
SUBEOF  921E  2835     2793 2813
SUBREC  91CA  2791      387
SUBSTK  0073   182     3407 3411 3415 3418 3654 3663 3664 3670
SUBTAB  003A   144
SYM     007A    38     2902 3722
SYMTAB  003E   146     3144 3145 3146 3147
SYNCHK  0000    42      432  715  833  880 1073 1301 1690 1717 1793 1821
                       2922 3027 3240 3261 3266 3277
TAB$    00FC   312      891
TABLE   000B     1
TAN$    00D4   280
TEMP5   0066   180     1104 1107 1355 1356 1363 1364 1372 1390 1640 1642
                       1643 1647 1648 1651 1652 1657 2948 2997
THEN$   00B0   243
TO$     00B1   244
TONE1   0034    58     1053 1730 1824
TOP     0003     1
TOPL02  6030    66     3652
TOPL10  601A    62     2725
TOPL15  6012    60      793 2058 2390 2727
TRACE$  0090   211
TREM$   0083   198     3324
TSTINT  83C5  1060      916  961 3412
TSTS$0  9631  3408     3406
TSTS$1  9659  3420     3409 3410 3413
TSTSEP  9627  3403      890  954
UALPH$  00EA   294     1768
UBSUB   A020    81     2336
UNBRE$  008F   210
UNQST$  00C8   267      268
UNTRA$  0091   213
UPDAT$  00F8   308
USING   83CF  1071      874  877  884
USING$  00ED   297      884 1074
USN$42  84E9  1190     1181
USN$55  8564  1244     1232 1235
USN$67  85AA  1278     1288
USN$68  85BE  1286     1282
USN$95  85D3  1296     1293
USNG$0  8452  1125     1129
USNG$1  845E  1130     1087 1096 1115 1122 1142
USNG$3  8478  1147     1176
USNG$4  8493  1155     1159 1269 1280
USNG$5  8533  1218     1205
USNG$9  85C3  1289     1161 1194
VALCD   A016    80     1929
VALID$  00FE   314     1759 3294 3306
VALIDL  03B2   343     1809
VALIDP  03B0   342     1808
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
VAR0	0000	85	1274 1276 1287 1527 1530 1662 1663 1664 1665 1984 2001 2002 2022 2037 2096 2097 2098 2099 2236 2340 2341 2342 2355 2498 2512 2518 2525 2527 2530 2531 2955 2958 2959 2975 2981 2990 2991 2996 3004 3012 3014
VAR4	000E	110	1204 1217 1227 1228 1229 1230 1232 1233 1237 1268 1271 1279 1313 1314 1494 1520 1555 1629 1732 1734 1867 1884 1886 2896
VAR5	0010	112	1438 1444 1445 1463 1484 2237 2250 2348 2355 2357 2465 2895 2918
VAR6	0011	113	1432 1436 1437 1446 1457 1463 1476 1483 1484 1528 1532 2903 2906 2908 2912
VARA	002A	136	1353 1354 1356 1362 1365 1391 1398 1401 1402 1420 1421 1424 1428 1429 1430 1431 1658 1659 1661 1763 1782 1797 1809 1813 1816 1862 1863 1873 1892 1893 1894 1895 1901 1909 1917 1942
VARC	0003	106	2618
VARIA$	00F3	303	453
VARW	0020	131	1422 1423 1448 1450 1451 1452 1455 1456 1459 1460 1581 1582 1583 1661 1662 1669 1723 1764 1773 1788 1800 1802 1806 1812 1813 1872 1892 1901 1908 1917 1918 1919 1922 1924 1941 2645 2646 2647
VARY2	0006	88	2238
VDP	000C	1	
VEL	0007	1	
VGWITE	008A	51	2104 2182 2193
VLID$0	89C4	1790	1775
VPOP	0078	36	1162 1173 1290 1297 1803 1897 3441
VPUSH	0077	35	1139 1153 1174 1360 1562 1634 1787 1870 1898 1980 2917 3437
VRAMVS	0958	328	2348 2671 2674 2677 2865
VSAV$	8E69	2364	
VSPTR	006E	181	1148 1154 1178 1181 1196 1200 1520 1555 1629 1732 1734 1867 1884 1886 2671 2672 2896
VWIDTH	001C	330	1828 2700 3280 3610
WARN$$	6A82	72	791 1513 1902 1932 3666
WR$$5	87BF	1515	1508
WRNINP	87BB	1512	1480 1484
WRNIO	97B3	3665	
WRNNPP	822C	790	
WRNSNM	8AF8	1931	
XPT	000E	1	1049 1050 1482 1672 1842 1843 1844 1846 1937 1962 2700 3243 3606 3607
YPT	000D	1	