```
                    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
0001               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  LITS05 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
```

```
                83  *       Temporary workspace variables in file-manager
                84
0000            85  VAR0    EQU  >00
0002            86  MNUM    EQU  >02           Usually a counter
0004            87  PABPTR  EQU  >04           Pointer to current PAB
0006            88  VARY2   EQU  >06           Use in MVDN only
0008            89  CCC1    EQU  >08
000C            90  BBB1    EQU  >0C
                91  *       P A B   o f f s e t s
0002            92  FIL      EQU  2            File number within BASIC (0-255)
0003            93  OFS      EQU  3            Offset within record
0004            94  COD      EQU  4            I/O opcode
0005            95  FLG      EQU  5            I/O mode flag byte
0006            96  BUF      EQU  6            Start of data buffer
0008            97  LEN      EQU  8            Record length
0009            98  CNT      EQU  9            Character count
000A            99  RNM      EQU  10           Record number
000C           100  SCR      EQU  12           Screen base offset
000D           101  NLEN     EQU  13           Length of file descriptor
               102  *
0006           103  CCPPTR  EQU  >06           Pointer to current column (base 1)
0007           104  RECLEN  EQU  >07           Length of current record (maximum)
0008           105  CCPADR  EQU  >08           Actual buffer address of column
0008           106  VARC    EQU  >08
000A           107  STADDR  EQU  >0A           Start address - usually for copying
000A           108  RAMPTR  EQU  >0A           Pointer for crunching INPUT stmt
000C           109  BYTE    EQU  >0C           String length for GETSTR
000E           110  VAR4    EQU  >0E
000E           111  CURINC  EQU  >0E
0010           112  VAR5    EQU  >10
0011           113  VAR6    EQU  >11
0014           114  CURLIN  EQU  >14           Starting line number for LIST
0017           115  FNUM    EQU  >17           Current file number for search
0017           116  DSRFLG  EQU  FNUM          Internal = 60, External = 0
0017           117  OPTFLG  EQU  FNUM          Option flag byte during OPEN
               118  *       SUBDIVIDED INTO :
0000           119  MFLAG    EQU  0             Bit 0 - Mode flag (I/O mode)
0001           120  DFLAG    EQU  1             Bit 1 - DISPLAY/INTERNAL
0002           121  PFLAG    EQU  2             Bit 2 - PERMANENT
0003           122  SFLAG    EQU  3             Bit 3 - SEQUENTIAL/RELATIVE
0004           123  RFLAG    EQU  4             Bit 4 - FIXED/VARIABLE length
               124  *
               125  ********************************************************
               126  *               Permanent workspace variables
0018           127  STRSP   EQU  >18           Start of string space pointer
001A           128  STREND  EQU  >1A           End of string space pointer
001C           129  SREF    EQU  >1C           Temporary string reference
001E           130  SMTSRT  EQU  >1E           Beginning of current statement
0020           131  VARW    EQU  >20           Usually for cursor position
0022           132  ERRCOD  EQU  >22           Return vector from 9900 code
0024           133  STVSPT  EQU  >24           Base of value stack
0026           134  RTNG    EQU  >26           Return address from 9900 code
0028           135  NUDTAB  EQU  >28           Start of NUD table
002A           136  VARA    EQU  >2A           End of input string
002C           137  PGMPTR  EQU  >2C           Program token pointer
```

```
002E    138  EXTRAM EQU  >2E      Line number table pointer
0030    139  STLN   EQU  >30      Line number table limit pointers
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$2
039E                338  MRGPAB EQU  >039E        MERGED TEMPORARY FOR PAB PTR
03AA                339  INPUTP EQU  >03AA        INPUT TEMPORARY FOR PTR TO PROMPT
03AC                340  ACCVRW EQU  >03AC        ACCEPT TEMPORARY FOR @VARW,@VARA T
03AE                341  ACCVRA EQU  >03AE        TRY AGAIN
03B0                342  VALIDP EQU  >03B0        PTR TO STANDARD STRING IN VALIDATE
03B2                343  VALIDL EQU  >03B2        LENGTH OF STANDARD STRING IN VALID
03B4                344  SIZCCP EQU  >03B4        SIZE TEMPORARY FOR CCPADR
03B6                345  SIZREC EQU  >03B6        SIZE TEMPORARY FOR RECLEN
                    346  *                        Also use as temporary in RELOCA
03B7                347  ACCTRY EQU  >03B7        ACCEPT "TRY AGAIN" FLAG
03B8                348  SIZXPT EQU  >03B8        Save XPT in SIZE when "try again"
03B9                349  SAPROT EQU  >03B9        PROTECTION flag in SAVE
03BC                350  OLDTOP EQU  >03BC        Old top of memory for RELOCA
03BC                351  CPTEMP EQU  >03BC        CCPPTR,RECLEN temp in INPUT
03BE                352  NEWTOP EQU  >03BE        New top of memory for RELOCA
                    353  *
                    354  *       I / O   C O D E   D E F I N I T I O N S
0000                355  C$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 45E6     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 4B34     377          BR    READ      READ ROUTINE
8010 52F3     378          BR    GETDAT    GET DATA FROM ERAM/VDP (NOT USED)
8012 41F4     379          BR    CLSALL    CLOSE ALL OPEN FILES (SUBROUTINE)
8014 4DAD     380          BR    SAVE      PROGRAM SAVE ROUTINE
8016 4BBF     381          BR    OLD       PROGRAM LOAD ROUTINE
8018 5058     382          BR    LIST      LIST routine
801A 5690     383          BR    OUTREC    Output record routine
801C 51CE     384          BR    EOF       End of file routine
801E 4964     385          BR    ACCEPT    "ACCEPT" routine
8020 4B91     386          BR    SRDATA    Search "DATA$" routine
8022 51BA     387          BR    SUBREC    RECORD routine
8024 5598     388          BR    CHKEND    Check EOS
8026 4BC5     389          BR    OLD1      A subroutine for LOAD
8028 4FF7     390          BR    MERGE     Merge a program
802A 5169     391          BR    GRMLST    List a line out of the ERAM
802C 5194     392          BR    GRSUB2    Read 2 bytes of data from ERAM/VDP
802E 51AC     393          BR    GRSUB3    Read 2 bytes of data from ERAM/VDP
              394  *                       with resetting possible breakpt
8030 4839     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 069340         424    CALL CHKFN              See if we specified any file
8035 77C2           425    BS   ERRFE              Definitely not...no # or #0 or
8037 06935C         426    CALL CHKCON             Check and search given filenumber
803A 77C2           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 069645         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 069598         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 069347         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 FIXED
```

```
                    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 069598         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.-siz1
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     $
```

```
                598     $     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 8602       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$$                  Then give an error
810C 03         619     DATA 3                      * SYNTAX ERROR
                620     $
                621     $     Continue with CHECK to complete the actual OPEN
                622     $
                623   CHECK
810D 069598     624     CALL CHKEND                 Check EOS
8110 4106       625     BR    OPERR                 Not EOS : SYNTAX ERROR
                626     $
                627     $     If the user hasn't specified VARIABLE or FIXED,
                628     $     the default specification depends on the file
                629     $     type.  Change current default (=VARIABLE) to
                630     $     FIXED for RELATIVE files.
                631     $
8112 DAE005     632     $IF .BIT0 RAM(FLG(PABPTR)) .EQ. 1 THEN RELATIVE RECORD    1
8115 040161
8118 27
8119 DAE005     633       $IF .BIT4 RAM(FLG(PABPTR)) .EQ. 1 THEN  VARIABLE mode   2
811C 041061
811F 25
                634   FIL$$
8120 0680FB     635         CALL CLRFRE             Undo the PAB allocation           2
8123 57C2       636         BR    ERRFE             FILE ERROR                        2
                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 069756     643     CALL CDSR                   Call the DSR...return with erro
8134 577F       644     BR    ERR$2B                  indication in COND.....
```

```
8136 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 77B2     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 069645         684    CALL  PARFN                 Handle as normal PAB OPEN
8181 069598         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 069749         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 069340         706    CALL CHKFN              Check for "no #" / "#0" cases
8199 77C2           707    BS   ERRFE             Not for "CLOSE" you don't
819B 06935C         708    CALL CHKCON            Check file number etc...
819E 57C2           709    BR   ERRFE             *** FILE NUMBER NOT IN SYSTEM
81A0 06937B         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 069598         719    CALL CHKEND            EOS?
81BB 4109           720    BR   ERRSYN            NO:SYNTAX ERROR
81BD 069756         721    CALL CDSR              Call DSR with whatever we have
81C0 41C7           722    BR   CLOS$1            Reset means error.....
81C2 069391         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 069391         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 57A6           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    $UNTIL RAM(@PABPTR) .DEQ. 0 Find last PAB in chain
81E4 41DD
81E6 06937B         746    CALL OUTEOF              Take care of pending records
                    747  CLSLBL
81E9 BEE004         748    ST   C$CLOS,RAM(COD(PABPTR)) Select CLOSE code
81EC 0401
81EE 069756         749    CALL CDSR                CLOSE to DSR routine
81F1 069391         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 069340         770    CALL CHKFN                 Check for #<filenumber>
8206 8F4A62         771    $IF @FAC .DEQ. 0 GOTO OLDC$0 Found equivalent of #0
8209 27
820A 06935C         772    CALL CHKCON                Check and decode file #
820D 57C2           773    BR   ERRFE                 Give error if file not there
820F 06937B         774    CALL OUTEOF                Output pending record
8212 87E00A         775    DCLR RAM(RNM(PABPTR))      Initialize to record 0
8215 04
8216 0694AD         776    CALL PARREC                Parse possible record clause
8219 069749         777    CALL IOCALL                Call DSR routine with
821C 04             778    DATA C$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 069598         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 0691AC         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 77BE
8248 A73600    807      DSUB 4,@LNBUF          Get next entry in line # table  1
824B 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 0694DA         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 06972A         842    CALL INITKB            initialize keyboard I/O
8269 D642FD         843    $IF @CHAT .EQ. NUMBE$ THEN Could still be anything      1
826C 42C4
826E 069340         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 06935C         846      CALL CHKCON          Check and convert expression     1
8278 57C2           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        2
8284 040277
8287 C2
                    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 0696E0   859      CALL PRINIT            Initialize some variables      1
              860  *
              861  *   Next character has to be either EOL,COMMA or COLON
              862  *
829B 069598   863      CALL CHKEND                                           1
829E 6395     864      BS   EOLEX             exit on end of statement       1
82A0 0694AD   865      CALL PARREC            Parse possible record clause   1
82A3 62B1     866      BS   PRIN$0            found "," but no REC clause    1
              867  PRN$10
82A5 069598   868      CALL CHKEND                                           1
82A8 6395     869      BS   EOLEX             Exit on end of statement for   1
              870  *                          "PRINT #0" or "PRINT file position"
82AA D642B3   871      $IF @CHAT .EQ. COMMA$ THEN                            2
82AD 42BE
82AF 0F79     872        XML  PGMCHR          get next in line               2
              873  PRIN$0
82B1 8E0463   874        $IF @PABPTR .EQ. 0 GOTO USING For "PRINT #0......"  2
82B4 CF
              875  *                 Internal type of file ?
82B5 DAE005   876        $IF .BIT3 RAM(FLG(PABPTR)) .EQ. 1 GOTO ERRFE        2
82B8 040857
82BB C2
82BC 43CF     877        BR  USING            execute USING clause           2
              878      $END IF
82BE 0F7E     879      XML  SPEED             Must be at a                   1
82C0 00       880      DATA SYNCHK                colon at this point        1
82C1 B5       881      DATA COLON$                  and error off on others  1
82C2 42C9     882    $SELSE                   make it a short branched ELSE  1
              883  PRIN$1
82C4 D642ED   884      $IF @CHAT .EQ. USING$ GOTO USING                      1
82C7 63CF
              885    $END IF                  end standard initialization
              886  *
              887  *   Test standard separators
              888  *
              889  CONPRT
82C9 069612   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 B5          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 0695F6      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 77C2
                 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$O
                 933  OTHE$1
8303 D64C65      934       $IF @FAC2 .EQ. STRVAL THEN Print the string result  1
8306 430D
                 935  OTHE$O
8308 0696F9      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 0695F6      940         CALL RSTRING       Reserve and copy string          1
8314 0696F9      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       $SEND IF
                 953  CHKSEP
```

```
8328 069612   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 57C2     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 0694A3   967       CALL CNVDEF            Check and convert to integer
833F BC4C07   968       ST   @RECLEN,@FAC2     Set modulo number
8342 069605   969       CALL COMMOD            Compute remainder
8345 C4064B   970       $IF @CCPPTR .H. @FAC1 THEN Position on next output rec 1
8348 434F
834A 069690   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 069690         1006        CALL OUTREC          Output the current record
                    1007      $SEND 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 069598         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 069690         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
8398 A1
8399 DA0404         1031      $IF .BIT2 @PABPTR .EQ. 0 THEN    "AT" clause unused
839C 43A1
839E B204E7         1032        AND  >E7,@PABPTR   remove flag 3 (SIZE used)
                    1033      $END IF
                    1034    $END IF
83A1 069690         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
83A7 B1
83A8 9206           1041      DEC  @CCPPTR           Back to actual offset
83AA BCE003         1042      ST   @CCPPTR,RAM(OFS(PABPTR)) Save for next time
83AD 0406
83AF 0F75           1043      XML  CONT              Continue with next stmt
                    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
83B4 43BB
83B6 BC7F06         1049      ST  @CCPPTR,XPT      Save current value of pointer
83B9 947F           1050      INCT XPT             CCPPTR: 1-28
                    1051    $END IF
83BB DA0402         1052    $IF .BIT1 @PABPTR .EQ. 1 THEN Used BEEP clause
```

```
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 77
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$1             has to match exactly           1
83EF 954C           1088      DINCT @FAC2             move up to the pointer field   1
83F1 BD5234         1089      DST   @DATA,@FAC8       save DATA pointer for READ usage 1
83F4 069194         1090      CALL GRSUB2             Read 2 bytes of data from ERAM/VDP1
83F7 4C             1091      DATA FAC2               @FAC2:Source address on ERAM/VDP 1
83F8 BD3458         1092      DST @EEE1,@DATA         @EEE1:Destination addr. on CPU 1
                    1093  *                           Put it in @DATA
83FB BE4CA3         1094      ST    IMAGE$,@FAC2      search for an IMAGE token       1
83FE 068B94         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 0692EF         1097      CALL GETGFL             get first part of format string 1
8406 06930E         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 0692B8         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 VD3
842C 0F8B           1109          XML  GVWITE        write data from ERAM to VDP     3
                    1110        $END IF
                    1111      $END IF
842E 4438           1112    $SELSE                                                   1
8430 0F74           1113      XML PARSE             parse up to the ending ":"       1
8432 B5             1114      DATA COLON$                                            1
8433 D64C65         1115      $IF @FAC2 .NE. STRVAL GOTO USNG$1 "* IMAGE ERROR"      1
```

```
8436 445E
                1116     $SEND IF
8438 D642B5     1117     $IF @CHAT .NE. COLON$ THEN  probably no variable list  1
843B 6448
843D 069598     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     $SELSE                     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 57AE       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     $SEND 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 3450B0     1148     MOVE @FAC6 FROM RAM(@FAC4) TO RAM(@CURLIN) copy format
847B 14B04E
847E BD4E14     1149     DST @CURLIN,@FAC4         complete preps for VPUSH
8481 BF4A00     1150     DST SREF,@FAC
8484 1C
8485 9150       1151     DINC @FAC6                include 0 in string length
8487 0F77       1152     XML VPUSH                 make the string temporary
                1153   USNG$4
8489 D6B014     1154     $IF RAM(@CURLIN) .NE. :#: THEN try to locate the next for1
848C 2364B9
848F 8EB014     1155       $IF RAM(@CURLIN) .NE. 0 THEN not end of string yet   2
8492 6498
8494 9114       1156         DINC @CURLIN          update pointer if not found  2
8496 4489       1157         BR  USNG$4            and continue searching       2
                1158       $END IF
8498 D642B3     1159       $IF @CHAT .NE. COMMA$ GOTO USNG$9 stop on last variabl1
849B 45B9
849D 0F78       1160       XML VPOP                restore original workstring dat 1
```

```
849F BC0C51  1161        ST  @FAC7,@BYTE      print the current format string   1
84A2 920C    1162        DEC  @BYTE           don't count the last "0"          1
84A4 BE0301  1163        ST  1,@MNUM+1        indicate direct output without ski1
84A7 069703  1164        CALL CHKR$O          copy string to output record      1
84AA 069690  1165        CALL OUTREC          also output current record        1
             1166   *
             1167   *    FAC still contains the right data, however it is
             1168   *    easier just to copy the original string again.
             1169   *
84AD BD144E  1170        DST @FAC4,@CURLIN    reconstruct CURLIN                1
84B0 0F78    1171        XML VPOP             copy original string info.        1
84B2 0F77    1172        XML VPUSH            without actually removing it      1
84B4 A51450  1173        DSUB @FAC6,@CURLIN   reconstruct start address        1
84B7 4478    1174        BR  USNG$3           continue for the next variable    1
             1175      $END IF
84B9 D514E0  1176      $IF @CURLIN .DNE. RAM(4(VSPTR)) THEN avoid "#" as count 1
84BC 046E64
84BF DF
84C0 9314    1177        DDEC @CURLIN         backup to the sign                1
84C2 D6B014  1178        $IF RAM(@CURLIN) .EQ. :.: THEN used ".#####"           2
84C5 2E44D1
84C8 D514E0  1179          $IF @CURLIN .DEQ. RAM(4(VSPTR)) GOTO USN$42           2
84CB 046E64
84CE DF
84CF 9314    1180          DDEC @CURLIN       avoid checking count bit          2
             1181        $END IF
84D1 D6B014  1182        $IF RAM(@CURLIN) .NE. :-: THEN check for minus or plus2
84D4 2D64DF
84D7 D6B014  1183          $IF RAM(@CURLIN) .NE. :+: THEN                        3
84DA 2B64DF
84DD 9114    1184            DINC  @CURLIN    it's neither, so we undo          3
             1185          $END IF
             1186        $END IF
             1187      $END IF
             1188   USN$42
             1189   *
             1190   *     Check for availabililty of variables
             1191   *
84DF D642B3  1192      $IF @CHAT .NE. COMMA$ GOTO USNG$9 exit if no more pt item
84E2 45B9
84E4 0F79    1193      XML  PGMCHR            get next expression
84E6 A514E0  1194      DSUB RAM(4(VSPTR)),@CURLIN  make CURLIN offset for garbage
84E9 046E
             1195   *                                collection
84EB 0F74    1196      XML  PARSE             parse up to ";" or ","
84ED B4      1197      DATA SEMIC$
84EE A114E0  1198      DADD RAM(4(VSPTR)),@CURLIN reconstruct new CLN after G.C.
84F1 046E
84F3 8752    1199      DCLR @FAC8             start with clean sheet for counts
84F5 8755    1200      DCLR @FAC11
84F7 8657    1201      CLR  @FAC13
84F9 BD0E14  1202      DST  @CURLIN,@VAR4     now start checking process
84FC D6B014  1203      $IF RAM(@CURLIN) .EQ. :.: GOTO USNG$5
84FF 2E6529
8502 D6B014  1204      $IF RAM(@CURLIN) .NE. :#: THEN has to be "+" or "-"       1
```

```
8505 23651D
8508 D6B014  1205      $IF RAM(@CURLIN) .EQ. :-: THEN                          2
850B 2D4511
850E B65502  1206        SB  @FAC11,1        set explict sign flag for CNS     2
             1207      $END IF
8511 D6B014  1208      $IF RAM(@CURLIN) .EQ. :+: THEN                          2
8514 2B451D
8517 B65502  1209        SB  @FAC11,1        set explict sign flag for CNS     2
851A B65504  1210        SB  @FAC11,2        set positive sign flag for CNS    2
             1211      $END IF
             1212    $END IF
851D 0685DB  1213    CALL ACCNM              accept first character plus "#"'s
8520 BC5653  1214    ST @FAC9,@FAC12         set up FAC12 for CNS
8523 D6B00E  1215    $IF RAM(@VAR4) .EQ. :.: THEN found decimal point          1
8526 2E4536
             1216  USNG$5
8529 8653    1217      CLR @FAC9             prepare for use as counter of no. 1
             1218  *                         of # sign after decimal point
852B 0685DB  1219      CALL ACCNM            accept some more "#"'s            1
852E BC5753  1220      ST @FAC9,@FAC13       set up FAC13 for CNS              1
8531 A05356  1221      ADD @FAC12,@FAC9      FAC9 now contains the total no. of1
             1222  *                         "#" sign ,d.p.and maybe a sign bit
8534 9253    1223      DEC @FAC9             exclude the d.p.                  1
             1224    $END IF
8536 D7B00E  1225    $IF RAM(@VAR4) .DEQ. :^^: THEN attemp to decode "^^^^^"   1
8539 5E5E45
853C 5A
853D 950E    1226      DINCT @VAR4           update address                    1
853F D7B00E  1227      $IF RAM(@VAR4) .DEQ. :^^: THEN                          2
8542 5E5E45
8545 58
8546 950E    1228        DINCT @VAR4         update address                    2
8548 B65508  1229        SB  @FAC11,3        set E-format bit for CNS          2
854B D6B00E  1230        $IF RAM(@VAR4) .NE. :^: GOTO USN$55                   2
854E 5E455A
8551 910E    1231        DINC @VAR4          update end address                2
8553 B65510  1232        SB  @FAC11,4        set extended E-format bit for CNS 2
8556 455A    1233        BR  USN$55                                            2
             1234      $END IF
8558 970E    1235      DDECT @VAR4           correct for previous errors       1
             1236    $END IF
             1237  *
             1238  *   At this point, CURLIN is pointing at the first
             1239  *   item of the format, VAR4 is pointing at the
             1240  *   character following the item
             1241  *
             1242  USN$55
855A CA4C64  1243    $IF @FAC2 .L. >64 THEN detected numerical argument          1
855D 658C
855F DA5502  1244      $IF .BIT1 @FAC11 .EQ. 1 THEN  exclude the sign count    2
8562 6566
8564 9253    1245        DEC  @FAC9          FAC9: no. of significant digits   2
             1246      $END IF
8566 DA5508  1247      $IF .BIT3 @FAC11 .EQ. 1 THEN   If E-format is used      2
8569 6572
```

```
856B CE530A  1248         $IF @FAC9 .GT. 10 GOTO ERRIM more than 10 significant
856E 77AE
             1249  *                          digits in image clause ****ERROR
8570 4577    1250       $SELSE                If E-format is not used
8572 CE530E  1251         $IF @FAC9 .GT. 14 GOTO ERRIM more than 14 significant
8575 77AE
             1252  *                          digits in image clause *****ERROR
             1253       $END IF
8577 B65501  1254       SB @FAC11,0           set fixed format output  it for CN
857A 0F73    1255       XML CNS               convert no. to fixed format string
             1256  *
             1257  *     FAC11 points to the beginning of the string after
             1258  *     supressing leading 0's, FAC12 contains the length of
             1259  *     the string
             1260  *
857C BC5755  1261       ST   @FAC11,@FAC13      FAC13 now point to beginning of
             1262  *                          the string
857F 8655    1263       CLR  @FAC11           clear high byte
8581 3455B0  1264       MOVE @FAC11 FROM *FAC13 TO RAM(@CURLIN) copy the
8584 149057
             1265  *                          result string from temp
8587 BD140E  1266       DST  @VAR4,@CURLIN    move pointer behind print field
858A 4489    1267       BR   USNG$4           continue after printing
             1268     $END IF
858C BD540E  1269     DST  @VAR4,@FAC10       compute total length
858F A55414  1270     DSUB @CURLIN,@FAC10
8592 C45155  1271     $IF @FAC7 .H. @FAC11 THEN string exceeds limits
8595 45A7
8597 BE002A  1272       ST  :*:,@VAR0         prepare a "*****.." string
             1273       $REPEAT
859A BCB014  1274         ST  @VAR0,RAM(@CURLIN) fill the remainder of field
859D 00
859E 9114    1275         DINC @CURLIN            up to the end
             1276   USN$67
85A0 D5140E  1277       $UNTIL @CURLIN .DEQ. @VAR4 which is stored in VAR4
85A3 459A
85A5 4489    1278       BR  USNG$4            continue with next item
             1279     $END IF
85A7 8F5065  1280     $IF @FAC6 .DEQ. 0 GOTO USN$68
85AA B4
85AB 3450B0  1281     MOVE @FAC6 FROM RAM(@FAC4) TO RAM(@CURLIN) copy result
85AE 14B04E
             1282  *                                                  string
85B1 A11450  1283     DADD @FAC6,@CURLIN      and update address in string
             1284   USN$68
85B4 BE0020  1285     ST  : :,@VAR0           fill remainder with spaces
85B7 45A0    1286     BR  USN$67
             1287   USNG$9
85B9 0F78    1288     XML  VPOP               temporary string back out
85BB BC0C15  1289     ST   @CURLIN+1,@BYTE    output up to the current pos.
85BE A40C4F  1290     SUB  @FAC5,@BYTE        create one byte result
85C1 65C9    1291     BS   USN$95             avoid empty strings
85C3 BE0301  1292     ST   1,@MNUM+1          prevent skip if field too small
85C6 069703  1293     CALL  CHKR$0            perform all normal I/O stuff
             1294   USN$95
```

```
85C9 0F78     1295     XML   VPOP              remove source format string
85CB 069598   1296     CALL CHKEND             check for end of line exit
85CE 6395     1297     BS   EOLEX              take end of line exit
85D0 0F7E     1298     XML  SPEED
85D2 00       1299     DATA SYNCHK             then it HAS to be a ";"
85D3 B4       1300     DATA  SEMIC$
85D4 069598   1301     CALL CHKEND             Now - must be EOS
85D7 6381     1302     BS   PRSM$1             Supressed end of record, make it
              1303  *                                        a pending record
85D9 4109     1304     BR   ERRSYN             SYNTAX ERROR
              1305  *
              1306  *      Collect string of "#"'s
              1307  *
              1308  ACCNM
              1309     $REPEAT
85DB 9053     1310       INC  @FAC9            update item count                1
85DD 910E     1311       DINC  @VAR4           and item address                 1
85DF D6B00E   1312     $UNTIL RAM(@VAR4) .NE.  :#: decode as many "#"'s as possibl
85E2 2365DB
85E5 00       1313     RTN                     return from duty
```

```
                    1315  *
                    1316  *
                    1317  ***************************************************************
                    1318  *                  I N P U T   R O U T I N E                  *
                    1319  *                                                             *
                    1320  *       First check for file or screen I/O.  If file I/O      *
                    1321  *       then check for pending output and print that.         *
                    1322  *       If screen I/O then check for input prompt.            *
                    1323  *                                                             *
                    1324  *       Next collect the INPUT variable list on the V-stack.  *
                    1325  *       Get enough input from either file or keyboard, and    *
                    1326  *       compare types with entries on V-stack.                *
                    1327  *                                                             *
                    1328  *       After verification and approval, assign the values.   *
                    1329  ***************************************************************
                    1330  INPUT
85E6 06972A         1331    CALL INITKB             Assume keyboard INPUT
85E9 D642FD         1332    $IF @CHAT .EQ. NUMBE$ THEN Might be #0 or #1-255          1
85EC 4751
85EE 069340         1333      CALL CHKFN            Check for default #0              1
85F1 8F4A46         1334      $IF @FAC .DEQ. 0 THEN If luno #0                        2
85F4 01
85F5 BDA3AA         1335        DST @PGMPTR,RAM(INPUTP)  Save PGMPTR for "try again" 2
85F8 2C
85F9 91A3AA         1336        DINC RAM(INPUTP)    Pass the ":" for the "prompt" code2
                    1337  *                         handler later, ( using #0 will not
                    1338  *                         take care the prompt in INPUT)
85FC 068926         1339        CALL INPU$2         #0 is equivalent to no #        2
85FF 475F           1340        BR   INP$2                                          2
                    1341      $END IF
8601 0688E1         1342      CALL INSU1            Get info about file             1
                    1343      $
                    1344      $  INTERNAL files get special treatment
                    1345      $
8604 DAE005         1346      $IF .BIT3 RAM(FLG(PABPTR)) .EQ. 1 THEN INTERNAL file  2
8607 040866
860A A4
860B 8EE003         1347        $IF RAM(OFS(PABPTR)) .EQ. 0 THEN Fresh start....    3
860E 044614
                    1348  INTR$0
8611 069750         1349          CALL IOCL$1       Get a new record through the DSR  3
                    1350        $END IF
8614 BC2BE0         1351        ST   RAM(OFS(PABPTR)),@VARA+1  Regain possible offset2
8617 0304
8619 862A           1352        CLR  @VARA          Make that a two byte constant   2
861B BD66E0         1353        DST  RAM(BUF(PABPTR)),@TEMP5  Get first address     2
861E 0604
8620 A1662A         1354        DADD @VARA,@TEMP5  Compute actual address within rec.2
                    1355  INTR$1
8623 0F7A           1356        XML  SYM            Get the symbol table entry      2
8625 0F7B           1357        XML  SMB              of the given variable         2
8627 0F77           1358        XML  VPUSH          And save it on the stack        2
8629 870C           1359        DCLR @BYTE          Assume no data available        2
862B C82BE0         1360        $IF @VARA+1 .L. RAM(CNT(PABPTR)) THEN Pick up data  3
862E 090466
```

```
8631 3A
8632 BC0DB0  1361          ST   RAM(@TEMP5),@BYTE+1 Length byte first      3
8635 66
8636 9166    1362          DINC @TEMP5       Update both actual address    3
8638 902B    1363          INC  @VARA+1        and offset                  3
             1364        $END IF
863A D64C65  1365        $IF @FAC2 .EQ. >65 THEN Has to be string variable  3
863D 4647
863F BD500C  1366          DST  @BYTE,@FAC6 Set length of string           3
8642 0692C9  1367          CALL CTMPST       Create temporary string       3
8645 4675    1368        $SELSE                                            3
8647 D60D08  1369          $IF @BYTE+1 .NE. 8 GOTO ERRFE "* FILE ERROR"    3
864A 57C2
864C 340C4A  1370          MOVE @BYTE FROM RAM(@TEMP5) TO @FAC Copy value  3
864F B066
8651 8F4A66  1371          $IF @FAC .DNE. 0 THEN Watch out for non-scaled stuf4
8654 73
8655 BE5C51  1372            ST   FAC7,@ARG      Test for legal numeric      4
             1373            $REPEAT
8658 C6905C  1374              $IF *ARG .H. 99 GOTO ERRFE "* FILE ERROR"     5
865B 6377C2
865E 925C    1375              DEC  @ARG     Next digit for test           5
8660 D65C4B  1376            $UNTIL @ARG .EQ. FAC1                         4
8663 4658
8665 BD5C4A  1377            DST  @FAC,@ARG Copy in ARG for some testing    4
8668 815C    1378            DABS @ARG       Be sure we're positive        4
             1379  *       If first byte after expon. byte =0 : incorrect
             1380  *       normalization has occured : FILE ERROR
             1381  *       Or >99 :illegal numeric:FILE ERROR
866A 925D    1382            DEC  @ARG1      0 would cause underflow here  4
866C C65D62  1383            $IF @ARG1 .H. 98 GOTO ERRFE                   4
866F 77C2
8671 4675    1384          $SELSE                                          4
8673 874C    1385            DCLR @FAC2      Be sure FAC2 = 0 (no strings) 4
             1386          $SEND IF
             1387        $SEND IF
8675 A1660C  1388        DADD @BYTE,@TEMP5  Update address and             2
8678 A02B0D  1389        ADD  @BYTE+1,@VARA+1    offset again              2
867B 0F7C    1390        XML  ASSGNV        Assign value to variable       2
867D 86E003  1391        CLR  RAM(OFS(PABPTR))   Undo allocated offsets    2
8680 04
8681 D642B3  1392        $IF @CHAT .EQ. COMMA$ THEN                        3
8684 46A2
8686 0F79    1393          XML  PGMCHR       Get next text character       3
8688 069598  1394          CALL CHKEND       Check for end of statement    3
868B 6696    1395          BS   INTR$2       O.K. EOS is fine              3
868D C82BE0  1396            $IF @VARA+1 .HE. RAM(CNT(PABPTR)) GOTO INTR$0 3
8690 090466
8693 11
8694 4623    1397            BR   INTR$1     Still something left          3
             1398  INTR$2
8696 C82BE0  1399          $IF @VARA+1 .L. RAM(CNT(PABPTR)) THEN             4
8699 090466
869C A2
869D BCE003  1400            ST   @VARA+1,RAM(OFS(PABPTR)) Save value of offse4
```

```
86A0 042B
                1401          $END IF
                1402        $END IF
86A2 0F75       1403        XML  CONT           And CONTINUE                    2
                1404      $END IF
86A4 06927A     1405      CALL GETVAR           Collect variable list on stack  1
86A7 BD140A     1406      DST  @STADDR,@CURLIN  Save it in temp                 1
86AA BF0A08     1407      DST  CRNBUF,@RAMPTR   Initialize crunch buffer pointer 1
86AD 20
86AE 8607       1408      CLR  @RECLEN          Initialize field counter        1
86B0 BEE004     1409      ST   C$READ,RAM(COD(PABPTR)) Select READ operation    1
86B3 0402
86B5 8EE003     1410      $IF RAM(OFS(PABPTR)) .NE. 0 GOTO INP$31               1
86B8 0446E0
86BB 46C3       1411      BR   INP$3            Adjust for used record usage    1
                1412      $REPEAT
86BD BEEFFF     1413        ST   COMMA$,RAM(-1(RAMPTR)) Fake legal separator    2
86C0 FF0AB3
                1414  INP$3
86C3 069750     1415        CALL IOCL$1         Get next input record           2
86C6 86E003     1416        CLR  RAM(OFS(PABPTR)) Reset offset within record    2
86C9 04
86CA 06881F     1417        CALL RECENT         Compute record entry            2
86CD BC2AE0     1418        ST   RAM(CNT(PABPTR)),@VARA Get record length       2
86D0 0904
86D2 8E2A66     1419        $WHILE @VARA .NE. 0                                 3
86D5 E0
86D6 A2B020     1420          ADD  OFFSET,RAM(@VARW)  Add video offset for normal3
86D9 60
86DA 9120       1421          DINC @VARW            screen-type crunch - proceed for3
86DC 922A       1422          DEC  @VARA            entire record.              3
86DE 46D2       1423        $SEND WHILE                                         2
                1424  INP$31
86E0 06881F     1425        CALL RECENT         Compute actual record entry     2
86E3 BC2BE0     1426        ST   RAM(CNT(PABPTR)),@VARA+1 Compute end of record 2
86E6 0904
86E8 862A       1427        CLR  @VARA          Make that a double byte         2
86EA A12AE0     1428        DADD RAM(BUF(PABPTR)),@VARA  Add buffer start addr. 2
86ED 0604
86EF 932A       1429        DDEC @VARA          Point to last position in record 2
86F1 8611       1430        CLR  @VAR6          Assume no values input          2
86F3 0F7F       1431        XML  CRUNCH         Scan data fields as in DATA stmt 2
86F5 01         1432        DATA 1              Indicate input stmt crunch      2
86F6 8F2257     1433        $IF @ERRCOD .DNE. 0 GOTO  ERRINP If some crunch error2
86F9 BA
86FA 9011       1434          INC  @VAR6        Get correct # of fields (one off) 2
86FC A00711     1435          ADD  @VAR6,@RECLEN Update # of fields up to now   2
86FF C80710     1436      $UNTIL @RECLEN .HE. @VAR5 O.K...THAT'S ENOUGH !!!!!!!!!1
8702 46BD
8704 972C       1437      DDECT @PGMPTR         Backup program pointer          1
8706 0F79       1438      XML  PGMCHR           Re-inspect last token before EOL 1
8708 06881F     1439      CALL RECENT           Precompute record entry         1
870B 86E003     1440      CLR  RAM(OFS(PABPTR))  Assume no pending record       1
870E 04
870F D642B3     1441      $IF @CHAT .EQ. COMMA$ THEN Make record pending        2
```

```
8712 4749
8714 D40710  1442        $IF @RECLEN .NE. @VAR5 THEN Enough left for pendir 3
8717 6749
8719 A40710  1443          SUB  @VAR5,@RECLEN Compute remaining # of fields  3
871C A41107  1444          SUB  @RECLEN,@VAR6 # of fields used in last rec   3
             1445  INP$32
871F D6B020  1446          $WHILE RAM(@VARW) .EQ. :":+OFFSET                 4
8722 824731
             1447            $REPEAT         Skip quoted strings
8725 9120    1448              DINC @VARW                                      5
8727 D6B020  1449            $UNTIL RAM(@VARW) .EQ. :":+OFFSET               4
872A 824725
872D 9120    1450            DINC @VARW                                      4
872F 471F    1451          $SEND WHILE                                       3
             1452          $REPEAT           Search for Nth data item
8731 9120    1453            DINC @VARW      Update pointer                  4
8733 D6EFFF  1454          $UNTIL RAM(-1(VARW)) .EQ. :,:+OFFSET              3
8736 FF208C
8739 4731
873B 9211    1455          DEC  @VAR6        Commas denote end of field      3
873D 471F    1456          BR   INP$32       Continue until done             3
873F A520E0  1457          DSUB RAM(BUF(PABPTR)),@VARW  Compute current offset3
8742 0604
8744 BCE003  1458          ST   @VARW+1,RAM(OFS(PABPTR)) Store for next round 3
8747 0421
             1459        $END IF
             1460      $END IF
8749 BC1110  1461      ST   @VAR5,@VAR6      Copy # of variables for check   1
874C BD0A14  1462      DST @CURLIN,@STADDR   Restore from temp               1
874F 477D    1463    $SELSE                  Now we're in for screen I/O     1
8751 06972A  1464      CALL INITKB           Initialize some variables for KB 1
8754 BDA3AA  1465      DST @PGMPTR,RAM(INPUTP) Save for "try again" case      1
8757 2C
8758 BDA3BC  1466      DST @CCPPTR,RAM(CPTEMP) Save CCPPTR,RECLEN for "try agn1
875B 06
             1467  INP$33
             1468  *                       Entry point for "try again" case
875C 0688FF  1469      CALL INSUB1          Put out prompt                   1
             1470  INP$2
875F 06927A  1471      CALL GETVAR          Get variable list on V-stack     1
             1472  INPU$3
8762 068934  1473      CALL INSUB2          Read from the screen             1
8765 8611    1474      CLR  @VAR6           Assume no values input           1
8767 0F7F    1475      XML  CRUNCH          Crunch the input line            1
8769 01      1476      DATA 1               Indicate input stmt scan         1
876A BD0A14  1477      DST  @CURLIN,@STADDR Restore from temp                1
876D 8F2247  1478      $IF @ERRCOD .DNE. 0 GOTO WRNINP If got some crunch erro1
8770 B2
8771 0F83    1479      XML  SCROLL          Scroll up after crunching        1
8773 BE7F03  1480      ST   3,XPT           Reset XPT too - pending records  1
8776 9011    1481      INC  @VAR6           # fields = # commas + 1          1
8778 D41011  1482      $IF @VAR5 .NE. @VAR6 GOTO WRNINP # of variables wrong 1
877B 47B2
             1483    $SEND IF
             1484    $
```

```
                    1485     $     Once we're here, all information should be available
                    1486     $     After type verification for input and variables, we
                    1487     $     push all value entries on the V-stack.
                    1488     $     VAR6 = VAR5 = number of variables
                    1489     $
877D BD1434         1490       DST  @DATA,@CURLIN      Save current DATA pointer
8780 BF3408         1491       DST  CRNBUF,@DATA       Get crunch entry
8783 20
8784 BD020E         1492       DST  @VAR4,@MNUM        Get entry in V-stack before PUSH
                    1493     INPU$4
8787 A30200         1494       DADD 8,@MNUM            Point to first symbol table entry
878A 08
878B BD06B0         1495       DST  RAM(@MNUM),@CCPPTR Get intermediate result
878E 02
878F 0692F7         1496       CALL GETRAM             Get value descriptor from RAM
8792 DAB006         1497       $IF .BIT7 RAM(@CCPPTR) .EQ. 0 THEN Numerical value      1
8795 8047C6
8798 0692D7         1498         CALL CHKNUM           Check entered value against numeri1
879B 47AB           1499         BR   INPU$5           Found error                       1
879D 8E1767         1500         $IF @DSRFLG .EQ. 0 GOTO INPU$6 Do not check overflow in1
87A0 CB
                    1501     *   file I/O, supply machine infinity with appropriate sign
                    1502     *   and contine
87A1 8EA3BA         1503         $IF RAM(CSNTP1) .EQ. 0 GOTO INPU$6 Watch out for over- 1
87A4 67CB
                    1504     *                                          flow in screen I/O
87A6 BD3414         1505         DST  @CURLIN,@DATA    Restore DATA pointer              1
87A9 47B6           1506         BR   WR$$5            Ask for input reenter             1
                    1507     INPU$5
87AB 8E1777         1508         $IF @DSRFLG .EQ. 0 GOTO ERRINP  FILE I/O IS FATAL       1
87AE BA
87AF BD3414         1509         DST  @CURLIN,@DATA    Restore DATA pointer on error     1
                    1510     WRNINP
87B2 066A82         1511         CALL WARN$$           Go here for simple warnings too   1
87B5 20             1512         DATA 32               INPUT ERROR - TRY AGAIN           1
                    1513     WR$$5
                    1514
87B6 068832         1515         CALL SCR$              Scroll the screen and reset CCPAD1
87B9 BD2CA3         1516         DST RAM(INPUTP),@PGMPTR Restore ptr to "promp" if any  1
87BC AA
87BD BD06A3         1517         DST RAM(CPTEMP),@CCPPTR Restore CCPPTR,RECLEN for try a1
87C0 BC
87C1 BD6E0E         1518         DST @VAR4,@VSPTR      Restore original stack ptr.       1
87C4 475C           1519         BR   INP$33                                             1
                    1520       $END IF
87C6 06930E         1521       CALL CHKSTR             Check string input
87C9 67AB           1522       BS   INPU$5             ERROR...CHECK I/O TYPE
                    1523     INPU$6
87CB 0692F7         1524       CALL GETRAM             Get separation character (RAM)
87CE D601B3         1525       $IF @VAR0+1 .NE. COMMA$ THEN                            1
87D1 67DD
87D3 9211           1526         DEC  @VAR6            Has to be end of data             1
87D5 47AB           1527         BR   INPU$5           If not...ERROR                    1
87D7 8E0147         1528         $IF @VAR0+1 .NE. 0 GOTO INPU$5                          1
87DA AB
```

```
87DB 47E1    1529    $SELSE                                                  1
87DD 9211    1530      DEC  @VAR6          Count number of value entries    1
87DF 4787    1531      BR   INPU$4         Continue                         1
             1532    $SEND IF
             1533    $
             1534    $    Assign cycle - assign values to variables
             1535    $    Because it rescans the program line, this code
             1536    $    can not be used for imperative statements, since
             1537    $    the crunch buffer get's destroyed on input.
             1538    $         The rescan is necessary because subscripts
             1539    $    should be evaluated AFTER all previous values
             1540    $    have been assigned, i.e.
             1541    $
             1542    $         INPUT I,A(I)   with values 2,3
             1543    $
             1544    $    Should assign value 3 to A(2) !!!!!!
             1545    $
             1546    $         No error-checking is done here, since types
             1547    $    are already validated.  We might get subscripts
             1548    $    out of range though !!!!
             1549    $
87E1 BF3408  1550    DST  CRNBUF,@DATA       Prepare for input rescan
87E4 20
87E5 BD2C0A  1551    DST  @STADDR,@PGMPTR   Restore token pointer for rescan
87E8 932C    1552    DDEC @PGMPTR           Backup one token
87EA BD6E0E  1553    DST  @VAR4,@VSPTR      Restore original stack pointer
             1554  INP$65
87ED 0F79    1555      XML  PGMCHR          Get next program characters
87EF 069598  1556      CALL CHKEND          Might have , before EOS
87F2 681A    1557      BS   INPU$7
87F4 0F7A    1558      XML  SYM             Rescan variable name
87F6 0F7B    1559      XML  SMB             Get correct entry for arrays
87F8 0F77    1560      XML  VPUSH           Save on stack for ASSGNV
87FA 0692F7  1561      CALL GETRAM          Get first token of input value
87FD D64C65  1562      $IF @FAC2 .NE. STRVAL THEN Numerical case               1
8800 6807
8802 0692D7  1563        CALL CHKNUM        Check for numerical value        1
8805 6810    1564        BS   INP$67        COND should be set (valid numeric)1
             1565      $END IF
8807 06930E  1566      CALL CHKSTR          Get the correct string value
880A BD0C50  1567      DST  @FAC6,@BYTE     Length for temporary string
880D 0692C9  1568      CALL CTMPST          Create temporary string
             1569  INP$67
8810 0F7C    1570      XML  ASSGNV          Assign value to variable
8812 0692F7  1571      CALL GETRAM          Skip separator (already checked)
8815 069598  1572      CALL CHKEND          Check for end of statement
8818 47ED    1573      BR   INP$65          Found it
             1574  INPU$7
881A BD3414  1575    DST  @CURLIN,@DATA     Restore DATA pointer
881D 0F75    1576    XML  CONT              Continue in PARSE
             1577
             1578  RECENT
881F BC21E0  1579    ST   RAM(OFS(PABPTR)),@VARW+1 Get record offset
8822 0304
8824 8620    1580    CLR  @VARW             Double byte value reqr'd.....
```

```
8826 A120E0   1581     DADD RAM(BUF(PABPTR)),@VARW Got it.................
8829 0604
882B 00       1582     RTN                        AND NOW, THE END IS NEAR.......
              1583
              1584   CHKRM
882C C70802   1585     $IF @CCPADR .DH. SCRNBS+29 THEN not enough room for "?"   1
882F FD4838
              1586   SCR$
8832 0F83     1587        XML  SCROLL             Scroll one line for "?"
8834 BF0802   1588        DST  SCRNBS+2,@CCPADR  and update CCPADR accordingly   1
8837 E2
              1589     $END IF
8838 00       1590     RTN
```

```
                1592  ************************************************************
                1593  *                  L I N P U T    R O U T I N E              *
                1594  *                                                            *
                1595  *       If file-I/O then                                     *
                1596  *            Get file number and check it                    *
                1597  *            Internal file not allowed                       *
                1598  *       End if                                               *
                1599  *       Get variable info                                    *
                1600  *       Must be string variable                              *
                1601  *       If file-I/O then                                     *
                1602  *          If no-partial-record or REC clause included       *
                1603  *            Read new record                                 *
                1604  *          End if                                            *
                1605  *          Set up copy pointers                              *
                1606  *       Else                                                 *
                1607  *          Call readln to read from keyboard                 *
                1608  *          Copy to crunch buffer adjusting for screen offset*
                1609  *          Set up copy pointers                              *
                1610  *       End if                                               *
                1611  *       Get string of proper length                          *
                1612  *       Move data into string                                *
                1613  *       Assign string                                        *
                1614  *       Done.                                                *
                1615  ************************************************************
                1616
8839 06972A     1617  LINPUT CALL INITKB              Assume input from keyboard
883C D642FD     1618         $IF @CHAT .EQ. NUMBE$ THEN  If '#' - then device    1
883F 4854
8841 069340     1619           CALL CHKFN             Check for default = 0       1
8844 8F4A68     1620           $IF @FAC .DEQ. 0 GOTO LINP10  #0 is same as keyboa1
8847 57
8848 0688E1     1621           CALL INSU1             Parse the device #          1
                1622  *        If internal file-then error
884B DAE005     1623           $IF .BIT3 RAM(FLG(PABPTR)) .EQ. 1 GOTO ERRFE      1
884E 040857
8851 C2
8852 4857       1624         $SELSE                                              1
8854 0688FF     1625           CALL INSUB1            Handle possible prompt      1
                1626         $END IF
8857 BD0E6E     1627  LINP10 DST @VSPTR,@VAR4        Save original V-ptr
                1628  *                             incase BREAK in READLN
885A 0F7A       1629         XML  SYM                Get info about the symbol
885C 0F7B       1630         XML  SMB                Get value pointer and type
885E D64C65     1631         $IF  @FAC2 .NE. STRVAL GOTO ERRMUV Must be string
8861 57CA
8863 0F77       1632         XML VPUSH               Push entry on the stack
8865 8E1748     1633         $IF @DSRFLG .EQ. 0 THEN If device I/O               1
8868 A8
8869 8EE003     1634           $IF RAM(OFS(PABPTR)) .EQ. 0 THEN  If new record   2
886C 044874
886F 069750     1635             CALL IOCL$1          Read the record             2
8872 488C       1636           $SELSE                 Else partial left by INPUT  2
8874 BC0CE0     1637             ST   RAM(CNT(PABPTR)),@BYTE   Get length of rec 2
8877 0904
8879 BD66E0     1638             DST  RAM(BUF(PABPTR)),@TEMP5  Get addr of buf   2
```

```
887C 0604
887E 8E0C68  1639            $WHILE @BYTE .NE. 0   While chars in buffer     3
8881 8C
8882 A6B066  1640              SUB  OFFSET,RAM(@TEMP5) Remove INPUT's offset 3
8885 60
8886 9166    1641              DINC @TEMP5       Increment pointer           3
8888 920C    1642              DEC  @BYTE        Decrement count             3
888A 487E    1643            $SEND WHILE         Drop out directly when done 2
             1644          $END IF
888C 8666    1645          CLR  @TEMP5           Need a word value           1
888E BC67E0  1646          ST   RAM(OFS(PABPTR)),@TEMP5+1 Restore offset     1
8891 0304
8893 860C    1647          CLR  @BYTE            Need a word value           1
8895 BC0DE0  1648          ST   RAM(CNT(PABPTR)),@BYTE+1  Get the length     1
8898 0904
889A A50C66  1649          DSUB @TEMP5,@BYTE     Calculate length            1
889D A166E0  1650          DADD RAM(BUF(PABPTR)),@TEMP5 Current buffer addres1
88A0 0604
88A2 86E003  1651          CLR  RAM(OFS(PABPTR)) Read next record next time  1
88A5 04
88A6 48DA    1652        $SELSE                  Else if keyboard input      1
88A8 068934  1653          CALL INSUB2           Clear line and call READLN  1
88AB 870C    1654          DCLR @BYTE            Initialize byte counter     1
88AD BD660A  1655          DST  @RAMPTR,@TEMP5   Initialize "crunch" pointer 1
88B0 D6B02A  1656          $IF  RAM(@VARA) .EQ. SPACE+OFFSET THEN If space   2
88B3 8048B8
88B6 932A    1657            DDEC @VARA          Don't include space on end  2
             1658          $END IF
88B8 CD202A  1659          $WHILE @VARW .DLE. @VARA THEN   While not at end  2
88BB 68D5
88BD BC00B0  1660            ST   RAM(@VARW),@VARO Get the character         2
88C0 20
88C1 D6007F  1661            $IF  @VARO .NE. EDGECH THEN  If not at edge char3
88C4 68D1
88C6 A60060  1662              S    OFFSET,@VARO   Subtract screen offset    3
88C9 BCB00A  1663              ST   @VARO,RAM(@RAMPTR)  And put into crnbuf  3
88CC 00
88CD 910C    1664              DINC @BYTE         Count it                    3
88CF 910A    1665              DINC @RAMPTR       And update "crunch" pointer 3
             1666            $END IF
88D1 9120    1667            DINC @VARW          Update input pointer        2
88D3 48B8    1668          $SEND WHILE                                       1
88D5 0F83    1669          XML  SCROLL           Scroll the screen           1
88D7 BE7F03  1670          ST   3,XPT            Initialize X-pointer        1
             1671        $END IF
88DA 0692C9  1672        CALL CTMPST             Create temporary string
88DD 0F7C    1673        XML  ASSGNV             Assign the value to it
88DF 0F75    1674        XML  CONT               And continue execution
```

```
                1676  *       Get file number and info about the file
                1677  INSU1
88E1 06935C     1678          CALL CHKCON             Check & convert & search..
88E4 57C2       1679          BR   ERRFE              Give error if required
                1680  *
                1681  *       INPUT allowed for input and update modes
                1682  *
88E6 DAE005     1683          $IF .BIT1 RAM(FLG(PABPTR)) .NE. 0 GOTO ERRFE
88E9 040257
88EC C2
88ED 06937B     1684          CALL OUTEOF             Output pending PRINT stuff
88F0 BEE004     1685          ST   C$READ,RAM(COD(PABPTR))  Ensure read operation
88F3 0402
88F5 0694AD     1686          CALL PARREC             Parse REC clause
88F8 0F7E       1687          XML  SPEED              Must be at a
88FA 00         1688          DATA SYNCHK                colon else
88FB B5         1689          DATA COLON$                   its an error
88FC 8617       1690          CLR  @DSRFLG            Clear keyboard input flag
88FE 00         1691          RTN
```

```
                    1693  *       Parse and put out input prompt
                    1694
88FF BD0A2C         1695  INSUB1  DST   @PGMPTR,@STADDR    Save pointer for prompt check
8902 930A           1696          DDEC  @STADDR            Backup to previous token
                    1697          $REPEAT                  Go into a tight loop
8904 0695A5         1698            CALL NXTCHR            Get next program character   1
8907 6920           1699            BS   INP$37            Detected end of stmt         1
8909 D642B5         1700          $UNTIL @CHAT .EQ. COLON$  Stop if we find a colon
890C 4904
890E BD2C0A         1701          DST   @STADDR,@PGMPTR    Backup for actual prompt scan
8911 0F79           1702          XML   PGMCHR             Jump into 1st char of prompt
8913 0F74           1703          XML   PARSE              And try to decode string expr.
8915 B5             1704          DATA  COLON$
8916 D64C65         1705          $IF @FAC2 .NE. STRVAL GOTO ERRSNM Num. prompt illegal
8919 57AA
891B 0696F9         1706          CALL  OSTRNG             Output the given prompt
891E 492F           1707          BR    INP$39             Exit without prompt backup
8920 BD2C0A         1708  INP$37  DST   @STADDR,@PGMPTR    Backup to beginning of line
8923 BE42B5         1709          ST    COLON$,@CHAT       Fake prompt with ":"
                    1710
8926 06882C         1711  INPU$2  CALL  CHKRM              Check for room for ?
8929 BEB008         1712          ST    :?:+OFFSET,RAM(@CCPADR) Display ?
892C 9F
892D 9508           1713          DINCT @CCPADR            Count it too
892F 0F7E           1714  INP$39  XML   SPEED              Must be at a
8931 00             1715          DATA  SYNCHK              colon else
8932 B5             1716          DATA  COLON$                 its an error
8933 00             1717          RTN
```

```
                    1719  *       Issue 'BEEP' and call readln to read from screen
8934 06882C         1720  INSUB2 CALL CHKRM               Check for room for answer
8937 BD2008         1721         DST  @CCPADR,@VARW       Copy current cursor positi
                    1722         $REPEAT
893A BEB008         1723           ST   : :+OFFSET,RAM(@CCPADR) Clear the remainder  1
893D 80
893E 9108           1724           DINC @CCPADR                 of the current line    1
8940 CB0802         1725         $UNTIL @CCPADR .DHE. >2FE Stop if we're there
8943 FE493A
8946 BFA2FE         1726         DST  >7F7F,RAM(>2FE)     Replace edgechars
8949 7F7F
894B 8E80CE         1727         $IF @PRTNFN .EQ. 0 THEN If previous tone finished   1
894E 4953
8950 060034         1728           CALL TONE1             ------ "BEEP" ------        1
                    1729         $END IF
8953 C16E0E         1730         DEX  @VAR4,@VSPTR        Don't destroy V-stack on BREAK
8956 066A76         1731         CALL READLN              Input a line from the keyboard
8959 C16E0E         1732         DEX  @VAR4,@VSPTR        Restore V-stack pointer
895C BD140A         1733         DST  @STADDR,@CURLIN     Save in a temp
895F BF0A08         1734         DST  CRNBUF,@RAMPTR      Init crunch buffer pointer
8962 20
8963 00             1735         RTN
```

```
                    1737  ************************************************************
                    1738  *           A C C E P T   S T A T E M E N T                *
                    1739  *                                                          *
                    1740  *      Accept input anywhere on the screen. The            *
                    1741  *      total number of input variables is limited to one.  *
                    1742  *          On an ACCEPT AT( , ),' the maximum number of    *
                    1743  *      that can be accepted is up to the right margin !!!  *
                    1744  *          If SIZE() is used, the maximum number is        *
                    1745  *      limited to the given SIZE, or to the number of      *
                    1746  *      characters remaining on the line, whichever is      *
                    1747  *      the lesser.                                         *
                    1748  ************************************************************
                    1749  ACCEPT
8964 86A3B7         1750    CLR  RAM(ACCTRY)          Clear "try again" flag
8967 0694DA         1751    CALL DISACC               Use common code for DISPLAY/ACCEPT
896A 6109           1752    BS   ERRSYN                 COND set means end of statement
896C BE63FF         1753    ST   >FF, @ARG7           Assume we don't have VALIDATE
                    1754  *
                    1755  *           V A L I D A T E   O P T I O N   H A N D L I N G
                    1756  *
896F D642FE         1757    $IF @CHAT .EQ. VALID$ THEN  Detected VALIDATE option     1
8972 49F6
8974 0F79           1758      XML  PGMCHR             Next character should start option1
8976 D642B7         1759      $IF @CHAT .NE. LPAR$ GOTO ERRSYN  "* SYNTAX ERROR "    1
8979 4109
897B B60440         1760      SB   @PABPTR,6          Indicate usage of VALIDATE clause 1
897E BF2A00         1761      DST  1,@VARA            Use VARA as length of option strin1
8981 01
8982 8720           1762      DCLR @VARW              VARW= options used, VARW+1 = # of 1
                    1763      $REPEAT                    stack entries for strings
8984 0F79           1764        XML PGMCHR            Skip separator token             2
8986 CA42E8         1765        $IF @CHAT .HE. NUMER$ THEN Could be valid option     3
8989 49A3
898B CA42EB         1766          $IF @CHAT .L. UALPH$+1 THEN It is...               4
898E 69A3
8990 BE5C01         1767            ST   1,@ARG         Select bit 0 as number option 4
8993 A642E8         1768            SUB NUMER$,@CHAT    Creat correct offset          4
8996 699B           1769            BS    SETVW         Skip the shift stat.          4
8998 E05C42         1770            SLL   @ARG,@CHAT Then select whatever option we s4
899B B4205C         1771  SETVW     OR    @ARG,@VARW     Remember options in VARW     4
899E 0F79           1772            XML   PGMCHR         Get next token               4
89A0 0589BD         1773            B VLID$0           Must use a long branch here    4
                    1774          $END IF
                    1775        $END IF
89A3 0F74           1776        XML  PARSE             Try to decode a string expression 2
89A5 B6             1777        DATA RPAR$                                           2
89A6 D64C65         1778        $IF @FAC2 .NE. STRVAL GOTO ERRSNM String-number mism2
89A9 57AA
89AB 8E5169         1779        $IF @FAC7 .NE. 0 THEN   Only count non-null strings 3
89AE BD
89AF A02B51         1780          ADD  @FAC7,@VARA+1    Now watch out for overflow   3
89B2 0C49B9         1781          $IF .CARRY. THEN      String truncated             4
89B5 066A84         1782  ERRST     CALL ERR$$          * STRING TRUNCATED ERROR     4
89B8 13             1783            DATA 19                                          4
                    1784          $END IF
```

```
89B9 0F77     1785            XML  VPUSH              Push the result for future re3
89BB 9021     1786            INC  @VARW+1         Count number of entries on st 3
              1787          $END IF
              1788  VLID$O
89BD D642B3   1789      $UNTIL @CHAT .NE. COMMA$  Evaluate all fields            1
89C0 6984
89C2 0F7E     1790       XML  SPEED                                              1
89C4 00       1791       DATA SYNCHK              Check for ")" on end           1
89C5 B6       1792       DATA RPAR$               If not ..."* SYNTAX ERROR"     1
89C6 0694DD   1793       CALL DISP$1              Try to evaluate further optio1
89C9 6109     1794       BS   ERRSYN              Premature end of statement     1
89CB BD0C2A   1795       DST  @VARA,@BYTE         Allocate string for character1
89CE 0F71     1796       XML   GETSTR                                           1
89D0 BD5C1C   1797       DST   @SREF,@ARG         Get start of allocated string1
89D3 BCB05C   1798       ST   @VARW,RAM(@ARG)     Copy selected standard option1
89D6 20
89D7 915C     1799       DINC  @ARG               Leave room for standard optio1
89D9 8E2169   1800       $WHILE @VARW+1 .NE. 0    Copy all available informatio2
89DC EC
89DD 0F78     1801         XML  VPOP              Regain stack-entry             2
89DF 3450B0   1802         MOVE @FAC6 FROM RAM(@FAC4) TO RAM(@ARG) Copy string  2
89E2 5CB04E
89E5 A15C50   1803         DADD @FAC6,@ARG        Update destination address    2
89E8 9221     1804         DEC  @VARW+1           Count # of stack entries       2
89EA 49D9     1805       $SEND WHILE              and again drop out on 0 !!!!   1
89EC BDA3B0   1806       DST  @SREF,RAM(VALIDP)   Copy start address of string  1
89EF 1C
89F0 BDA3B2   1807       DST  @VARA,RAM(VALIDL)     and total string length     1
89F3 2A
89F4 8663     1808         CLR  @ARG7           Indicate VALIDATE usage for READLN1
              1809      $END IF
89F6 BD2008   1810      DST  @CCPADR,@VARW      Save start address of the field
89F9 BD2A20   1811      DST  @VARW,@VARA        Set default highest address used
89FC BD5E08   1812      DST  @CCPADR,@ARG2      Select absolute highest usable loc.
89FF A35E01   1813      DADD  290,@ARG2         290=2+32*9 maximum of 254 character
8A02 22
8A03 C62BFC   1814      $IF @VARA+1 .H. >FC THEN Start at the end of line        1
8A06 4A0C
8A08 A35E00   1815        DADD  4, @ARG2                                         1
8A0B 04
              1816      $END IF
8A0C 8E046A   1817      $IF @PABPTR .NE. 0 THEN  We used some options like AT,SIZ1
8A0F 5F
8A10 0F7E     1818        XML  SPEED                                           1
8A12 00       1819        DATA SYNCHK            Should always end on ":"      1
8A13 B5       1820        DATA COLON$                                          1
8A14 DA0402   1821        $IF .BIT1 @PABPTR .EQ. 1 THEN  Used BEEP clause      2
8A17 6A1C
8A19 060034   1822          CALL TONE1           Wake up the user              2
              1823        $END IF                Continue with option evaluation
8A1C DA0404   1824        $IF .BIT2 @PABPTR .EQ. 1 THEN  Used AT option...SIZE!! 2
8A1F 6A2E
8A21 DA0408   1825          $IF .BIT3 @PABPTR .EQ. 0 THEN Use default SIZE optio3
8A24 4A2C
8A26 BE051C   1826            ST  VWIDTH,@PABPTR+1   Limit current record leng 3
```

```
8A29 0695CA  1827          CALL SIZE1              Compute some intermediary dat3
             1828        $END IF
8A2C 4A40    1829        BR  ACCP$1                                             2
             1830      $END IF
8A2E DA0408  1831      $IF .BIT3 @PABPTR .EQ. 1 THEN SIZE option used somewher2
8A31 6A5F
             1832  *
             1833  *       We're sure now that SIZE has been used WITHOUT
             1834  *       the AT option...this means that we should
             1835  *       set XPT to point behind the SIZE field.
             1836  *       This can be done by adding the record length
             1837  *       to the current screen base address and
             1838  *       the line's screen base address
             1839  *
8A33 BC7F09  1840       ST  @CCPADR+1,@XPT  Start of with current address      2
8A36 A07F07  1841       ADD  @RECLEN,@XPT   Add in the current record length   2
8A39 A67FDF  1842       SUB  >DF,@XPT       And subtract the lower base addres2
             1843  *                        Also adjust for edge char.
8A3C BCA3B8  1844       ST @XPT,RAM(SIZXPT) Save it for "try again"case becaus2
8A3F 7F
             1845  *                        in WARNING, XPT gets changed
             1846  ACCP$1
8A40 BDA3B4  1847         DST @CCPADR,RAM(SIZCCP) Save for "try again" case     2
8A43 08
8A44 BCA3B6  1848         ST @RECLEN,RAM(SIZREC)  Save for "try again" case     2
8A47 07
             1849  ********************************************************
             1850  *ENTRY POINT FOR "TRY AGAIN" CASE WHEN SIZE OR ACCEPT USED
             1851  ********************************************************
             1852  ACCP$9
8A48 DA0480  1853         $IF .BIT7 @PABPTR .EQ. 0 THEN  Blank current field   3
8A4B 4A51
8A4D BEB008  1854           ST  : :+OFFSET,RAM(@CCPADR)                        3
8A50 80
             1855         $END IF
8A51 9108    1856         DINC @CCPADR        Update screen address           2
8A53 9207    1857         DEC  @RECLEN        Reduce count ...always at least on2
8A55 4A48    1858         BR   ACCP$9         Loop until at end of field      2
8A57 9308    1859         DDEC @CCPADR        Fix end of field for maximum size 2
8A59 BD2A08  1860         DST  @CCPADR,@VARA  Set highest field address used   2
8A5C BD5E2A  1861         DST  @VARA,@ARG2    Also highest location available  2
             1862       $END IF               OK all set to go
             1863     $END IF
8A5F D6A3B7  1864     $IF RAM(ACCTRY) .EQ. 1 GOTO ACCP$7  Skip if in "try again"
8A62 016A6E
8A65 BD0E6E  1865     DST @VSPTR,@VAR4        Save first entry in V-stack
8A68 0F7A    1866     XML  SYM                Collect the symbol designator
8A6A 0F7B    1867     XML  SMB                Take care of arrays too
8A6C 0F77    1868     XML  VPUSH              Save symbol table entry
             1869  ACCP$7
8A6E BDA3AC  1870     DST @VARW,RAM(ACCVRW)   Save for trying again case
8A71 20
8A72 BDA3AE  1871     DST @VARA,RAM(ACCVRA)   Save for trying again case
8A75 2A
             1872  ********************************************************
```

```
                    1873  *ENTRY POINT FOR "TRY AGAIN" WHEN NEITHER SIZE OR ACCEPT IS
                    1874  ************************************************************
                    1875  ACCP$5
                    1876  *  In case a CALL CLEAR or ERASE ALL or CALL HCHAR has just
                    1877  *  been processed, EDGE CHARS. are gone at the bottom line
8A76 DA040C         1878    CLOG >0C,@PABPTR        If AT/SIZE used, maximum field is 1
8A79 4A80           1879    BR   A$1                line, so no need to worry about it
8A7B BFA2FE         1880    DST >7F7F,RAM(>2FE)     Put the EDGE CHARS. back
8A7E 7F7F
                    1881  A$1
8A80 C10E6E         1882    DEX  @VSPTR,@VAR4       Don't destroy V-stack on BREAK
8A83 066A86         1883    CALL READL1             Ask for some input that can be used
8A86 C10E6E         1884    DEX  @VSPTR,@VAR4       Restore V-stack ptr
                    1885  *
                    1886  *      At this point, VARA contains the highest location
                    1887  *      used, and VARW contains the string's start address
                    1888  *
                    1889  ACCP$2
8A89 D52A20         1890    $IF @VARA .DNE. @VARW THEN Only non-empty string          1
8A8C 6A98
8A8E 932A           1891      DDEC @VARA            Go to the next position         1
8A90 D6B02A         1892      $IF RAM(@VARA) .EQ. : :+OFFSET GOTO ACCP$2            1
8A93 806A89
8A96 912A           1893      DINC @VARA            Back to the last space          1
                    1894    $END IF
8A98 0F78           1895    XML VPOP                Check the symbol designator is a
8A9A 0F77           1896    XML VPUSH               string or numeric variable
8A9C D64C65         1897    $IF @FAC2 .NE. >65 THEN If numeric : empty string is r  1
8A9F 6AAC
                    1898  *                          allowed to be entered
8AA1 D5202A         1899      $IF @VARW .DEQ. @VARA THEN  If an empty string was ente2
8AA4 4AAC
8AA6 066A82         1900        CALL WARN$$           ***INPUT ERROR***             2
8AA9 20             1901        DATA 32                                             2
8AAA 4AF6           1902        BR  ACCP$8            Try again                     2
                    1903      $END IF
                    1904    $END IF
8AAC 870C           1905    DCLR  @BYTE             Compute length of input string
8AAE BD1C20         1906    DST  @VARW,@SREF        Use SREF as temporary variable
8AB1 D51C2A         1907    $WHILE @SREF .DNE. @VARA                                1
8AB4 6AC2
8AB6 D6B01C         1908      $IF RAM(@SREF) .NE. EDGECH THEN Exclude edge character 2
8AB9 7F6ABE
8ABC 910C           1909        DINC @BYTE                                        2
                    1910      $END IF
8ABE 911C           1911      DINC @SREF            Decrement the counter           1
8AC0 4AB1           1912    $END WHILE
8AC2 0692BC         1913    CALL  CTSTRO            Create a temporary string
                    1914  ACCP$3
8AC5 D5202A         1915    $WHILE @VARW .DNE. @VARA                                1
8AC8 6AE5
8ACA D6B020         1916      $IF RAM(@VARW) .EQ. EDGECH THEN Skip the edge character2
8ACD 7F4AD6
8AD0 A32000         1917        DADD 4, @VARW                                     2
8AD3 04
```

```
8AD4 4AC5    1918        BR  ACCP$3                                          2
             1919      $END IF
8AD6 BCB01C  1920      ST RAM(@VARW),RAM(@SREF)  Copy the string             1
8AD9 B020
8ADB A6B01C  1921      SUB  OFFSET,RAM(@SREF)    Subtract the screen offset  1
8ADE 60
8ADF 9120    1922      DINC @VARW                Update pointers             1
8AE1 911C    1923      DINC @SREF                                            1
8AE3 4AC5    1924    $SEND WHILE                 Result can't be 0
8AE5 D64C65  1925    $IF @FAC2 .NE. STRVAL THEN  Numerical variable          1
8AE8 6B24
8AEA BE4C65  1926      ST  STRVAL,@FAC2          Create temp string          1
8AED 06A016  1927      CALL VALCD                Use VAL code for translation 1
8AF0 4B24    1928      BR   ACCP$6               No error - o.k. go on       1
             1929  WRNSNM
8AF2 066A82  1930      CALL WARN$$               Error                       1
8AF5 07      1931      DATA 7                    STRING NO. MISMATCH         1
             1932  ACCP$8
8AF6 DA0408  1933      $IF .BIT3 @PABPTR .EQ. 1 THEN  If SIZE is used        2
8AF9 6B04
8AFB DA0404  1934        $IF .BIT2 @PABPTR .EQ. 0 THEN  Also AT is not used  3
8AFE 4B04
8B00 BC7FA3  1935          ST  RAM(SIZXPT),@XPT  Restore XPT: in WARNING XPT  3
8B03 B8
             1936  *                             got changed
             1937        $END IF
             1938      $END IF
8B04 BD20A3  1939      DST RAM(ACCVRW),@VARW     Restore @VARA,@VARW         1
8B07 AC
8B08 BD2AA3  1940      DST RAM(ACCVRA),@VARA                                 1
8B0B AE
8B0C BEA3B7  1941      ST 1,RAM(ACCTRY)          Set the "try again" flag    1
8B0F 01
8B10 DA0408  1942      $IF .BIT3 @PABPTR .EQ. 0 THEN  If SIZE is not used    2
8B13 4B1A
             1943  *   IF ACCEPT ALSO NOT USED, GOTO "TRY AGAIN" FROM HERE
8B15 DA0404  1944        $IF .BIT2 @PABPTR .EQ. 0 GOTO ACCP$5                2
8B18 6A76
             1945      $END IF
             1946  *   IF "EITHER SIZE OR ACCEPT IS USED" THEN
8B1A BD08A3  1947      DST RAM(SIZCCP),@CCPADR Restore CCPADR                1
8B1D B4
8B1E BC07A3  1948      ST RAM(SIZREC),@RECLEN  Restore RECLEN                1
8B21 B6
8B22 4A48    1949      BR  ACCP$9              Go blanking the field and "try 1
             1950  *                             again"
             1951    $END IF
             1952  ACCP$6
8B24 0F7C    1953    XML  ASSGNV               Should be o.k. now
             1954  *
             1955  *       Scroll the screen one line if no AT or SIZE
             1956  *
8B26 DA040C  1957    CLOG >C, @PABPTR          Test usage of AT and/or SIZE
8B29 4B30    1958    BR  ACCP$4                At least one of the two used
8B2B 0F83    1959    XML  SCROLL               Scroll the screen up
```

```
8B2D BE7F03   1960     ST 3,XPT                And reset XPT
              1961   ACCP$4
8B30 0F75     1962     XML  CONT
```

```
                    1964  ************************************************************
                    1965  *           R E A D   S T A T E M E N T                    *
                    1966  *                                                          *
                    1967  *      Assign DATA values to variables in READ-list        *
                    1968  *      one at a time.  Possibly search for new DATA        *
                    1969  *      statements in the program if the current DATA       *
                    1970  *      statement has been used.  Be careful with null      *
                    1971  *      entries.....!                                       *
                    1972  ************************************************************
                    1973    $REPEAT
8B32 0F79           1974      XML  PGMCHR           Get character following ","      1
                    1975  READ
8B34 0F7A           1976      XML  SYM              Get pointers and correct entries 1
8B36 0F7B           1977      XML  SMB                also allow for array-variables 1
8B38 0F77           1978      XML  VPUSH            Push on Vstack for assignment    1
8B3A 8E3477         1979      $IF @DATA .EQ. 0 GOTO ERRDAT  DATA ERROR               1
8B3D BE
8B3E 0692EF         1980      CALL GETGFL           Get next data item (RAM/GROM)    1
8B41 D64C65         1981      $IF @FAC2 .NE. STRVAL THEN                             2
8B44 6B66
8B46 D601C8         1982        $IF @VAR0+1 .NE. NUM$ GOTO ERRSNM Not a numeric      2
8B49 57AA
                    1983  *                         string-number mismatch error
8B4B 06931A         1984        CALL CHKS$0         Build up string info             2
8B4E 9150           1985        DINC @FAC6          Force legal delimiter on end     2
8B50 06A002         1986        CALL LITS05         Copy numeric into string space   2
8B53 BD561C         1987        DST  @SREF,@FAC12   Copy string start address        2
8B56 A11C50         1988        DADD @FAC6,@SREF    Compute end address of string    2
8B59 931C           1989        DDEC @SREF          Back up over delimiter           2
8B5B 06A012         1990        CALL CONVER         Convert string to number         2
8B5E D5A390         1991        $IF RAM(CSNTMP) .DNE. @SREF GOTO ERRDAT WRONG !!!!!!!2
8B61 1C57BE
8B64 4B6E           1992      $SELSE                                                 2
8B66 06930E         1993        CALL CHKSTR         Check string input               2
8B69 77BE           1994        BS   ERRDAT        Give error on error              2
8B6B 06A002         1995        CALL LITS05         Allocate string in string space  2
                    1996      $SEND IF
8B6E 0F7C           1997      XML  ASSGNV           Assign variable                  1
8B70 0692EF         1998      CALL GETGFL           Get next datum from DATA stmt    1
8B73 D601B3         1999      $IF @VAR0+1 .NE. COMMA$ THEN   Has to be an end of DATA2
8B76 6B8A
8B78 8E0157         2000        $IF @VAR0+1 .NE. 0 GOTO ERRDAT Check  for end of data2
8B7B BE
8B7C 9736           2001        DDECT @LNBUF        Pointer to line # of DATA stmt   2
8B7E 8634           2002        CLR   @DATA         Assume the worst - no more DATAs 2
                    2003        $  WATCH OUT FOR "DATA" STMT AT END OF PROGRAM
8B80 D53630         2004        $IF @LNBUF .DNE. @STLN THEN                        3
8B83 6B8A
8B85 9336           2005          DDEC @LNBUF       Next line's 1st token address    3
8B87 06A008         2006          CALL DATAST       Get next DATA statement          3
                    2007        $END IF
                    2008      $END IF
8B8A D642B3         2009    $UNTIL @CHAT .NE. COMMA$ Worry about junk in CONT
8B8D 6B32
8B8F 0F75           2010    XML  CONT
```

```
                2011  *
                2012  *       SRDATA-Search for DATA statements (DATA stat.
                2013  *              must be the only stat. on one line)
                2014  *       SEARCH-also uesed for searching IMAGE stat.
                2015  *
                2016  SRDATA
8B91 BE4C93     2017    ST  DATA$,@FAC2         search for a DATA token
                2018  SEARCH
8B94 C12C34     2019    DEX  @DATA,@PGMPTR      exchange with normal PC
8B97 C00142     2020    EX   @CHAT,@VAR0+1      preserve current PGM character
8B9A 8E444B     2021    $IF @PRGFLG .EQ. 0     If imperative statement          1
8B9D AE
8B9E 8E8084     2022      $IF @RAMTOP .NE. 0   With ERAM: text itself in ERAM   2
8BA1 6BAE
8BA3 BE8089     2023        ST  >FF,@RAMFLG    Fake RAMFLG in this case         2
8BA6 FF
8BA7 0F79       2024        XML PGMCHR         Get first char. on the line      2
8BA9 868089     2025        CLR  @RAMFLG       Restore it back                  2
8BAC 4BB0       2026        BR   SRDA$1        Skip that PGMCHR                 2
                2027      $END IF
                2028    $END IF
8BAE 0F79       2029    XML  PGMCHR            get first character on the line
                2030  SRDA$1
8BB0 D4424C     2031    $IF @CHAT .EQ. @FAC2 GOTO SRDA$0 search for specific toker
8BB3 6BB8
8BB5 D40000     2032    CEQ @0,@0              set COND if no DATA found
                2033  SRDA$0
8BB8 C12C34     2034    DEX @DATA,@PGMPTR      exchanges won't  affect the CO
8BBB C00142     2035    EX  @CHAT,@VAR0+1      situation o.k.
8BBE 01         2036    RTNC                   return to caller with COND
```

```
                    2038  ********************************************************************
                    2039  *                  O L D   S T A T E M E N T                       *
                    2040  *     A normal load:                                               *
                    2041  *       Get a program from an external device to VDP and           *
                    2042  *       reinitialize the program pointers.  Also                   *
                    2043  *       update the line pointer table, since the                   *
                    2044  *       memory size of the machine on which the program            *
                    2045  *       was created doesn't have to be the same as on the          *
                    2046  *       current system !!!!!! Then check if ERAM existed,          *
                    2047  *       move it to ERAM if does exist (in relocated form)          *
                    2048  *     Load a sequential file :                                     *
                    2049  *       When program is bigger than 13.5K and ERAM exists,         *
                    2050  *       maximum-length record reads are performed to               *
                    2051  *       read the file and each record is copied into the           *
                    2052  *       ERAM as it is read.                                        *
                    2053  ********************************************************************
                    2054  OLD
8BBF 068BC5         2055    CALL OLD1                   Make OLD1 a subroutine for LOAD
8BC2 056012         2056    B  TOPL15                   Go back to top level
                    2057  OLD1
8BC5 069225         2058    CALL GPNAME                 Get program name & reinitialize ^s
8BC8 0F79           2059    XML  PGMCHR                 Check for EOL
8BCA 8645           2060    CLR @FLAG                   Clear the PROTECTION bit too
8BCC BD0A04         2061    DST  @PABPTR,@STADDR        Compute memory start address
8BCF A10AE0         2062    DADD RAM(NLEN-1(PABPTR)),@STADDR Add PAB-name length
8BD2 0C04
8BD4 A30A00         2063    DADD PABLEN-4,@STADDR                 and PAB length
8BD7 0A
8BD8 BDE00A         2064    DST  @>70,RAM(RNM(PABPTR))  Compute # of available bytes
8BDB 0470
8BDD A5E00A         2065    DSUB @STADDR,RAM(RNM(PABPTR))
8BE0 040A
8BE2 91E00A         2066    DINC RAM(RNM(PABPTR))  Include current address
8BE5 04
8BE6 BDE006         2067    DST  @STADDR,RAM(BUF(PABPTR)) for copy start
8BE9 040A
8BEB BEE004         2068    ST   C$LOAD,RAM(COD(PABPTR)) Select LOAD I/O code
8BEE 0405
8BF0 069756         2069    CALL CDSR                   Call device service routine
8BF3 4C5C           2070    BR   OLD$3                  Not a program file, may be a
                    2071  *                             sequential file
                    2072    $    STADDR still points to the info bytes
8BF5 BD02E0         2073    DST  RAM(2(STADDR)),@MNUM   First test checksum
8BF8 020A
8BFA B902E0         2074    DXOR RAM(4(STADDR)),@MNUM         which is a simple XOR
8BFD 040A
8BFF D5B00A         2075    $IF  RAM(@STADDR) .DNE. @MNUM THEN Try PROTECTION option 1
8C02 026C10
8C05 8302           2076      DNEG @MNUM                                              1
8C07 D5B00A         2077      $IF RAM(@STADDR) .DNE. @MNUM GOTO OLDER No-ERROR        1
8C0A 024D22
8C0D B64580         2078      SB @FLAG,7          Yes-Set LIST/EDIT PROTECTION flag   1
                    2079    $END IF
8C10 BD32E0         2080    DST  RAM(2(STADDR)),@ENLN   Copy new ENLN,
8C13 020A
```

```
8C15 BD30E0  2081    DST  RAM(4(STADDR)),@STLN            STLN and
8C18 040A
8C1A BDA3BC  2082    DST  RAM(6(STADDR)),RAM(OLDTOP)      top of memory in
8C1D E0060A
8C20 A30A00  2083    DADD 8,@STADDR          Point to program data
8C23 08
8C24 BDA3BE  2084    DST  @>70,RAM(NEWTOP)   Set up the new top
8C27 70
8C28 068D2B  2085    CALL RELOCA             Relocate according to @>70
             2086  *******
             2087  OLD$5
8C2B 8E8084  2088    $IF @RAMTOP .EQ. 0 GOTO LRTOP$ ERAM present ?
8C2E 6D27
             2089  *                               No-Go back to toplevel
             2090  *                               Yes-move from VDP to ERAM (in relo-
             2091  *                               cated form)
             2092  ******************Move to the ERAM from CPUBAS first********
8C30 BD0070  2093    DST @>70,@VARO
8C33 A50030  2094    DSUB @STLN,@VARO
8C36 9100    2095    DINC @VARO              # of bytes to move
8C38 BD4E00  2096    DST  @VARO,@CCC         @CCC : Byte count for VGWITE
8C3B BF50A0  2097    DST CPUBAS,@BBB         @BBB : Destination addr. on ERAM
8C3E 40
8C3F BD0A50  2098    DST @BBB,@STADDR        For later use as the base of curent
             2099  *                         program image in RELOCA
8C42 BD4C30  2100    DST @STLN,@AAA          @AAA : Source addr. on VDP
8C45 0F8A    2101    XML VGWITE              Move from VDP to ERAM
8C47 BDA3BC  2102    DST @>70,RAM(OLDTOP)    Set up old memory top
8C4A 70
8C4B BDA3BE  2103    DST @RAMTOP,RAM(NEWTOP) Set up new memory top
8C4E 8084
8C50 068D2B  2104    CALL RELOCA             Relocate the program image
             2105  OLD$7
8C53 BD8086  2106    DST @STLN,@RAMFRE       Reset the RAMFRE on ERAM
8C56 30
8C57 938086  2107    DDEC @RAMFRE
8C5A 4D27    2108    BR  LRTOP$              Go back to toplevel
             2109  ************************************************************
             2110  *    At this point : if ERAM not exist - ERROR off          *
             2111  *      else open sequential file to load program to ERAM     *
             2112  *      through VDP RAM                                      *
             2113  ************************************************************
             2114  OLD$3
8C5C 8E8084  2115    $IF @RAMTOP .EQ. 0 GOTO OLDER
8C5F 6D22
             2116  *   Set up PAB for OPEN
             2117  *   File type : Sequential file,
             2118  *   Mode of operation : Input
             2119  *   Data type : internal
             2120  *   Record type : Variable length records
             2121  *   Logical record length : 254 maximum
8C61 310009  2122    MOVE 9 FROM ROM(PAB3) TO RAM(4(PABPTR)) Build the PAB
8C64 E00404
8C67 8D19
8C69 BD4A70  2123    DST @>70,@FAC           Compute the data buffer address
```

```
8C6C A74A00  2124     DSUB 253,@FAC
8C6F FD
8C70 BD4C4A  2125     DST @FAC,@AAA            Save it for later use in VGWITE
8C73 BDE006  2126     DST @FAC,RAM(BUF(PABPTR))
8C76 044A
8C78 069756  2127     CALL CDSR                 Call the device service routine
8C7B 577F    2128     BR ERR$2B            Return with ERROR indication in COND..
             2129  *    Start to read in file
8C7D 069749  2130     CALL IOCALL              Read in the first record
8C80 02      2131     DATA C$READ
             2132  *    Check the control information
8C81 D6E009  2133     $IF RAM(CNT(PABPTR)) .NE. 10 GOTO OLDER 10 bytes contr. inf
8C84 040A4D
8C87 22
             2134  *    >ABCD is the flag set at SAVE time indicating a program
             2135  *    file
8C88 D7B04A  2136     $IF RAM(@FAC) .DNE. >ABCD GOTO OLDER
8C8B ABCD4D
8C8E 22
8C8F 954A    2137     DINCT @FAC
8C91 BD30B0  2138     DST RAM(@FAC),@STLN      Copy the new STLN
8C94 4A
8C95 954A    2139     DINCT @FAC
8C97 BD32B0  2140     DST RAM(@FAC),@ENLN      ENLN too
8C9A 4A
8C9B BD0232  2141     DST @ENLN,@MNUM          Test checksum
8C9E B90230  2142     DXOR @STLN,@MNUM
8CA1 954A    2143     DINCT @FAC
8CA3 D5B04A  2144     $IF RAM(@FAC) .DNE. @MNUM THEN Try PROTECTION option      1
8CA6 026CB4
8CA9 8302    2145       DNEG @MNUM                                             1
8CAB D5B04A  2146       $IF RAM(@FAC) .DNE. @MNUM GOTO OLDER No-ERROR          1
8CAE 024D22
8CB1 B64580  2147       SB @FLAG,7             Yes-set LIST/EDIT PROTECTION flag 1
             2148     $END IF
8CB4 954A    2149     DINCT @FAC
             2150  *  Check is there enough memory in ERAM
8CB6 BD02B0  2151     DST RAM(@FAC),@MNUM      Get the old top of memory out
8CB9 4A
8CBA BDA3BC  2152     DST @MNUM,RAM(OLDTOP)    For later use in RELOCA
8CBD 02
8CBE A50230  2153     DSUB @STLN,@MNUM
8CC1 9102    2154     DINC @MNUM               Total # of bytes in program
8CC3 BD0802  2155     DST @MNUM,@CCC1          For later use as the byte count
8CC6 A302A0  2156     DADD CPUBAS,@MNUM        Add the total # of bytes to CPUBAS
8CC9 40
             2157  *      Check if enough memory in ERAM
8CCA 0A6D22  2158     $IF .GT. GOTO OLDER      Greater than >ffff case
8CCD C50280  2159     $IF @MNUM .DH. @RAMTOP GOTO OLDER Greater than >dfff case
8CD0 846D22
             2160  *        Move to ERAM starting from CPUBAS first,
             2161  *        then relocate according the new top of memory inERAM
             2162  OL$$
8CD3 BF50A0  2163     DST CPUBAS,@BBB          @BBB : Destination addr. in ERAM
8CD6 40
```

```
                    2164  *                                  for VGWITE
8CD7 BD0A50         2165    DST @BBB,@STADDR          For later use as base of the
                    2166  *                           current program image in RELOC
                    2167  * DST @>70,@AAA             @AAA has been set up before
                    2168  * DSUB 253,@AAA             For copy start on VDP RAM
                    2169  * @CCC1 : Total # of bytes to move to ERAM, set up above
8CDA 069749         2170    CALL IOCALL               Read in the second record
8CDD 02             2171    DATA C$READ
                    2172    $REPEAT                   Read in the file and each record
                    2173  *  Should be a full(maximum length 254) record at this time,
                    2174  *  because program supposed to be bigger than 13.5K
8CDE D6E009         2175      $IF RAM(CNT(PABPTR)) .NE. 254 GOTO OLDER                1
8CE1 04FE4D
8CE4 22
8CE5 BF4E00         2176      DST 254,@CCC            @CCC:# of bytes to move         1
8CE8 FE
8CE9 0F8A           2177      XML VGWITE              Move data from VDP RAM to ERAM  1
8CEB A35000         2178      DADD 254,@BBB           Update the destination addr. on ERA1
8CEE FE
8CEF A70800         2179      DSUB 254,@CCC1          # of bytes left to move        1
8CF2 FE
8CF3 6D0B           2180      BS   OLD$9              No more bytes to move           1
8CF5 069749         2181      CALL IOCALL             Read in the file and each record i1
8CF8 02             2182      DATA C$READ             copied into the ERAM as it is read
8CF9 CB0800         2183    $UNTIL @CCC1 .DL. 254     Leave the last record alone
8CFC FE6CDE
                    2184  *  The record length should be the same as the # of bytes
                    2185  *  left to move at this time
8CFF D4E009         2186    $IF RAM(CNT(PABPTR)) .NE. @CCC1+1 GOTO OLDER
8D02 04094D
8D05 22
8D06 BD4E08         2187    DST @CCC1,@CCC            Set up byte count for the last read
8D09 0F8A           2188    XML VGWITE                Move data from VDP RAM to ERAM
                    2189  OLD$9
8D0B 069749         2190    CALL IOCALL               close the file
8D0E 01             2191    DATA C$CLOS
8D0F BDA3BE         2192    DST @RAMTOP,RAM(NEWTOP) New top of memory
8D12 8084
                    2193  * RAM(OLDTOP) : old top of memory, set up above
                    2194  * @STADDR : base of current program image in ERAM,set above
8D14 068D2B         2195    CALL RELOCA               Relocate the program
8D17 4C53           2196    BR OLD$7                  Go to set the RAMFRE and back to
                    2197  *                           toplevel
                    2198  PAB3
8D19 001C00         2199    DATA 0,>1C,#0,254,0,#0,OFFSET
8D1C 00FE00
8D1F 000060
                    2200    $
                    2201    $     OLD error exit code...don't kill machine....
                    2202    $
                    2203  OLDER
8D22 066014         2204    CALL INITPG               Initialize program space
8D25 5782           2205    BR   ERR$2                And take error exit
                    2206  LRTOP$
8D27 066022         2207    CALL KILSYM               Release string space/symbol tab
```

```
8D2A 00       2208    RTN
              2209  ********************************************************
              2210  *  RELOCATE THE PROGRAM IMAGE ACCORDING TO THE NEW TOP OF *
              2211  *          STLN : old stln                          MEMORY  *
              2212  *          ENLN : old enln                                  *
              2213  *          RAM(OLDTOP) : old top of memory                  *
              2214  *          RAM(NEWTOP) : new top of memory                  *
              2215  *          @STADDR : current base for the old image         *
              2216  ********************************************************
              2217  RELOCA
8D2B BDA3B4   2218    DST @PABPTR,RAM(SIZCCP) Save in temp.
8D2E 04
8D2F BD02A3   2219    DST RAM(OLDTOP),@MNUM  Get the old top of memory
8D32 BC
8D33 BD04A3   2220    DST RAM(NEWTOP),@PABPTR Get the new top of memory
8D36 BE
8D37 A53202   2221    DSUB @MNUM,@ENLN       Compute ENLN relative to top (-)
8D3A A53002   2222    DSUB @MNUM,@STLN          "    STLN     "    "   "  (-)
8D3D A50A30   2223    DSUB @STLN,@STADDR     Highest memory address used
8D40 8702     2224    DCLR @MNUM             Total # of bytes to be moved
8D42 A50230   2225    DSUB @STLN,@MNUM       STLN = - (# of bytes - 1)
8D45 9102     2226    DINC @MNUM             Take care of that one
8D47 A13204   2227    DADD @PABPTR,@ENLN     Compute new address of ENLN
8D4A A13004   2228    DADD @PABPTR,@STLN             and STLN
              2229  *     @PABPTR : destination addr., @STADDR : source addr.
8D4D BD5C02   2230    DST @MNUM,@ARG         @ARG : byte count
8D50 BD000A   2231    DST @STADDR,@VARO      @VARO : source addr. for MVDN
8D53 BD1006   2232    DST @CCPPTR,@VAR5      Save in tem. (CCPPTR,VARY2 EQU >06)
8D56 BD0604   2233    DST @PABPTR,@VARY2     @VARY2 : desitination addr. for MVDN
8D59 D5A3BE   2234    $IF RAM(NEWTOP) .DEQ. @RAMTOP THEN Relocate the program  1
8D5C 80844D
8D5F 64
              2235  *                        in ERAM
8D60 0F88     2236      XML MVDN             Move from lower memory to higher  1
              2237  *                        memory one byte at a time
8D62 4D73     2238    $SELSE                                                   1
8D64 87A3B6   2239      DCLR RAM(SIZREC)     Clear a temporary var             1
8D67 C1A3B6   2240      DEX @RAMTOP,RAM(SIZREC)   Save the RAMTOP, also fake as1
8D6A 8084
              2241  *                        if ERAM not exist for MVDN in this
8D6C 0F88     2242      XML  MVDN            Move in VDP                       1
8D6E C1A3B6   2243      DEX @RAMTOP,RAM(SIZREC) Restore RAMTOP                 1
8D71 8084
              2244    $END IF
8D73 BD0610   2245    DST @VAR5,@CCPPTR      Restore back
              2246    $
              2247    $    Update line # links according to new size
              2248    $
8D76 BD02A3   2249    DST RAM(OLDTOP),@MNUM  Old memory top
8D79 BC
8D7A A502A3   2250    DSUB RAM(NEWTOP),@MNUM Stop if sizes are same
8D7D BE
8D7E 6DA8     2251    BS  RELO$1
8D80 BD0A30   2252    DST  @STLN,@STADDR     Start relocations at STLN
              2253  OLD$2
```

```
8D83 C9320A  2254     $IF @ENLN .DL. @STADDR GOTO RELO$1 and continue up to ENLN
8D86 4DA8
8D88 950A    2255     DINCT @STADDR            Skip the line #
8D8A D4A3BE  2256     $IF RAM(NEWTOP) .EQ. @RAMTOP THEN  If in ERAM                1
8D8D 80844D
8D90 A0
8D91 069194  2257       CALL GRSUB2            Read the link out                  1
8D94 0A      2258       DATA STADDR                                               1
8D95 A55802  2259       DSUB @MNUM,@EEE1       Update it                          1
8D98 066036  2260       CALL GWSUB             Write it back                      1
8D9B 0A5802  2261       DATA STADDR,EEE1,2                                        1
8D9E 4DA4    2262     $SELSE                                                      1
8DA0 A5B00A  2263       DSUB @MNUM,RAM(@STADDR)     Update the link                1
8DA3 02
             2264     $END IF
8DA4 950A    2265     DINCT @STADDR            Skip the link...next line #
8DA6 4D83    2266     BR   OLD$2               And continue until done
             2267   RELO$1
8DA8 BD04A3  2268     DST RAM(SIZCCP),@PABPTR Restore from temp.
8DAB B4
8DAC 00      2269     RTN
```

```
                2271  ********************************************************
                2272  *              S A V E   S T A T E M E N T             *
                2273  *                                                      *
                2274  *  SAVE "NAME",MERGE :Save in crunched form of a program  *
                2275  *      into a file one line at a time with the ln #.   *
                2276  *      File opened with sequential accessed, variable-len  *
                2277  *      records (161 max), display type & outpt mode,   *
                2278  *      move one ln # and one ln text to the crunch      *
                2279  *      buffer then write to the file one ln at a time  *
                2280  *   A normal SAVE : When ERAM not exist or the size of  *
                2281  *      the program and ln # table in ERAM can fit in VDP  *
                2282  *      (can be moved into VDP from ERAM once), then    *
                2283  *      the save statement saves a program image to an  *
                2284  *      external device, including all the information  *
                2285  *      the system needs for rebuilding the program image on*
                2286  *      a machine with a different memory size..also include*
                2287  *      is a checksum for rudimetary error checking and for *
                2288  *      PROTECTION VIOLATION                            *
                2289  *   A sequential SAVE : Maximum-length records are     *
                2290  *      performed to write the file and each record is  *
                2291  *      copied into the VDP from ERAM before it is written  *
                2292  ********************************************************
                2293  SAVE
8DAD DA4580     2294    $IF .BIT7 @FLAG .EQ. 1 GOTO ERRPV *PROTECTION VIOLATION
8DB0 57C6
8DB2 069225     2295    CALL GPNAME             This will also close all files
                2296  *    Check SAVE "NAME",MERGE or SAVE "NAME",PROTECTED first
8DB5 86A3B9     2297    CLR RAM(SAPROT)         Clear "PROTECTED" flag
8DB8 0F79       2298    XML PGMCHR
8DBA 8E42       2299    CZ  @CHAT               EOL ?
8DBC 6E20       2300    BS SA$1                 Yes- no need to check any option
8DBE D642B3     2301    $IF @CHAT .NE. COMMA$ GOTO ERRSYN Has to be a comma here
8DC1 4109
8DC3 D7B02C     2302    $IF RAM(@PGMPTR) .DEQ. >C805 THEN  Unquoted string with  1
8DC6 C8054D
8DC9 E9
                2303  *                       length 5 : has to be MERGE at this time
8DCA D7E002     2304      $IF RAM(2(PGMPTR)) .DNE. :ME: GOTO ERRSYN  If not:SYNTA1
8DCD 2C4D45
8DD0 4109
8DD2 D7E004     2305      $IF RAM(4(PGMPTR)) .DNE. :RG: GOTO ERRSYN        ERROR 1
8DD5 2C5247
8DD8 4109
8DDA D6E006     2306      $IF RAM(6(PGMPTR)) .NE. :E: GOTO ERRSYN If not:SYNTAX  1
8DDD 2C4541
8DE0 09
8DE1 8EE007     2307      CZ RAM(7(PGMPTR))   Check for EOL                ERROR 1
8DE4 2C
8DE5 4109       2308      BR ERRSYN           Not EOL: SYNTAX ERROR             1
8DE7 4F6B       2309      BR SAVMG            Go to handle this option          1
                2310    $END IF
                2311  * Has to be PROTECTED option here, crunched as unquoted str.
8DE9 D7B02C     2312    $IF RAM(@PGMPTR) .DNE. >C809 GOTO ERRSYN   SYNTAX ERROR
8BEC C80941
8DEF 09
```

```
8DF0 D7E002  2313     $IF RAM(2(PGMPTR)) .DNE. :PR: GOTO ERRSYN
8DF3 2C5052
8DF6 4109
8DF8 D7E004  2314     $IF RAM(4(PGMPTR)) .DNE. :OT: GOTO ERRSYN
8DFB 2C4F54
8DFE 4109
8E00 D7E006  2315     $IF RAM(6(PGMPTR)) .DNE. :EC: GOTO ERRSYN
8E03 2C4543
8E06 4109
8E08 D7E008  2316     $IF RAM(8(PGMPTR)) .DNE. :TE: GOTO ERRSYN
8E0B 2C5445
8E0E 4109
8E10 D6E00A  2317     $IF RAM(10(PGMPTR)) .NE. :D: GOTO ERRSYN
8E13 2C4441
8E16 09
8E17 8EE00B  2318     CZ  RAM(11(PGMPTR)) Check EOL
8E1A 2C
8E1B 4109    2319     BR  ERRSYN              No-SYNTAX ERROR
8E1D 90A3B9  2320     INC RAM(SAPROT)         Set PROTECTION flag
             2321   SA$1
             2322   ************************************************************
8E20 8E8084  2323     $IF @RAMTOP .EQ. 0 THEN     If ERAM NOT present then     1
8E23 4E37
             2324   ************CLEAR THE BREAKPT IN VDP ALONE TO SPEED UP ***
8E25 BD5230  2325       DST @STLN,@FAC8     End of ln # buffer                 1
             2326       $REPEAT
8E28 B2B052  2327         RB  RAM(@FAC8),7  Clear the breakpt                  2
8E2B 7F
8E2C A35200  2328         DADD 4,@FAC8      Move to the next one               2
8E2F 04
8E30 C55232  2329       $UNTIL @FAC8 .DH. @ENLN  Until done                  1
8E33 4E28
8E35 4E5E    2330     $ELSE                                                  1
8E37 06A020  2331       CALL UBSUB          Clear the breakpt in ERAM        1
8E3A BD0280  2332       DST  @RAMTOP,@MNUM  Top of memory in ERAM            1
8E3D 84
8E3E A50230  2333       DSUB @STLN,@MNUM                                     1
8E41 9102    2334       DINC @MNUM          # of bytes total in ERAM         1
8E43 BD0070  2335       DST @>70,@VARO      Top of memory in VDP             1
8E46 A50002  2336       DSUB @MNUM,@VARO                                     1
8E49 9100    2337       DINC @VARO                                           1
             2338   *                        Check is there enough memory in VDP
             2339   *                        to move the program text and ln #
             2340   *                        table from ERAM to VDP
8E4B 0A4EAC  2341       $IF .NOT. .GT. GOTO GSAVE Not enough memory in VDP for 1
             2342   *                                                    sure
8E4E BF100A  2343       DST  VRAMVS+64+256,@VAR5 64 bytes are for savety buffer1
8E51 98
             2344   * DSR routine gives file error when loading a program which
             2345   * has VDP maximum size and was saved from VDP to be a progra
             2346   * file on disk when ERAM not exist.
             2347   * In order to fix this problem, restrict the program memory
             2348   * to be 256 bytes less than the real space in VDP when ERAM
             2349   * not exist
8E52 C90010  2350       $IF @VARO .DL. @VAR5 GOTO GSAVE  Not enough memory in  1
```

```
8E55 4EAC
                2351  *                               VDP, do a sequential file save
8E57 A71000     2352      DSUB 10,@VAR5               10 bytes for control inform.    1
8E5A 0A
                2353  *                               Destination addr. on VDP for GVMOV
8E5B 068F3F     2354      CALL GVMOV                  Enough memory in VDP, move it over1
                2355  *                               and do the normal save later
                2356    $END IF
                2357  ************** Without ERAM, or after GVMOV :**************
                2358  ************** do the normal save      **********************
                2359  VSAV$
8E5E BD0A40     2360    DST  @FREPTR,@STADDR    Store additional control info
8E61 930A       2361    DDEC @STADDR            Back up some more for 2 byte save
8E63 BDB00A     2362    DST  @>70,RAM(@STADDR)  First current top of memory
8E66 70
8E67 970A       2363    DDECT @STADDR
8E69 BDB00A     2364    DST  @STLN,RAM(@STADDR) Then STLN
8E6C 30
8E6D 970A       2365    DDECT @STADDR
8E6F BDB00A     2366    DST  @ENLN,RAM(@STADDR) Then ENLN
8E72 32
8E73 970A       2367    DDECT @STADDR           Then
8E75 BDB00A     2368    DST  @STLN,RAM(@STADDR)
8E78 30
8E79 B9B00A     2369    DXOR @ENLN,RAM(@STADDR)     STLN XORed with ENLN
8E7C 32
8E7D D6A3B9     2370    $IF RAM(SAPROT) .EQ. 1 THEN Check is there PROTECTED opti1
8E80 014E86
8E83 83B00A     2371      DNEG RAM(@STADDR)     Negate the CHECKSUM to indicate   1
                2372    $END IF                 LIST/EDIT protection
8E86 BDE006     2373    DST  @STADDR,RAM(BUF(PABPTR)) Save start address in PAB
8E89 040A
8E8B 930A       2374    DDEC @STADDR
8E8D BDE00A     2375    DST  @>70,RAM(RNM(PABPTR))  Compute # of bytes used
8E90 0470
8E92 A5E00A     2376    DSUB @STADDR,RAM(RNM(PABPTR))  and store that in PAB too
8E95 040A
8E97 8E8084     2377    $IF @RAMTOP .NE. 0 THEN   If ERAM exists then           1
8E9A 6EA2
8E9C BD300C     2378      DST @BBB1,@STLN       Restore the original STLN,ENLN    1
8E9F BD3208     2379      DST @CCC1,@ENLN       which points to ERAM              1
                2380    $END IF
8EA2 069749     2381    CALL IOCALL             Call Device Service Routine for
8EA5 06         2382    DATA C$SAVE             SAVE operation
                2383  LRTOPL
8EA6 066022     2384    CALL KILSYM             Release string space/symbol table
8EA9 056012     2385    B  TOPL15               Go back to toplevel
                2386  ************************************************************
                2387  GSAVE
                2388  *  Open the sequential file, set the PAB
                2389  *  File type : sequential file
                2390  *  Mode of operation : output
                2391  *  Data type : internal
                2392  *  Record type : variable length records
                2393  *  Logical record length : 254 maximum
```

```
8EAC 310009  2394    MOVE 9 FROM ROM(PAB3) TO RAM(4(PABPTR)) Build the PAB
8EAF E00404
8EB2 8D19
8EB4 96E005  2395    DECT RAM(FLG(PABPTR))  Put in the correct i/o mode, :>1A
8EB7 04
             2396  *  Compute the data buffer address
8EB8 BD4A70  2397    DST @>70,@FAC
8EBB A74A00  2398    DSUB 253,@FAC
8EBE FD
8EBF BDE006  2399    DST @FAC,RAM(BUF(PABPTR))
8EC2 044A
8EC4 BD584A  2400    DST @FAC,@EEE1          Save it for later use in GVWITE
8EC7 069756  2401    CALL CDSR               Call device service routine to oper
8ECA 577F    2402    BR ERR$2B               Return with ERROR indication inCONI
             2403  *                         Put 8 bytes control info. at the
             2404  *                         beginning of the data buffer
8ECC BFB04A  2405    DST >ABCD,RAM(@FAC)     >ABCD indentifies a program file
8ECF ABCD
8ED1 954A    2406    DINCT @FAC              when doing LOAD later
8ED3 BDB04A  2407    DST @STLN,RAM(@FAC)     Save STLN in control info.
8ED6 30
8ED7 954A    2408    DINCT @FAC
8ED9 BDB04A  2409    DST @ENLN,RAM(@FAC)     ENLN too
8EDC 32
8EDD 954A    2410    DINCT @FAC
8EDF BDB04A  2411    DST @STLN,RAM(@FAC)
8EE2 30
8EE3 B9B04A  2412    DXOR @ENLN,RAM(@FAC)    Save the checksum
8EE6 32
8EE7 D6A3B9  2413    $IF RAM(SAPROT) .EQ. 1 THEN Check is there PROTECTED opt 1
8EEA 014EF0
8EED 83B04A  2414      DNEG RAM(@FAC)        Negate the CHECKSUM to indicate   1
             2415    $END IF                 the LIST/EDIT protection
8EF0 954A    2416    DINCT @FAC
8EF2 BDB04A  2417    DST @RAMTOP,RAM(@FAC)   Save the top of memory info.
8EF5 8084
8EF7 BEE009  2418    ST 10,RAM(CNT(PABPTR)) Set the character count in PAB
8EFA 040A
8EFC 069749  2419    CALL IOCALL             Call device service routine
8EFF 03      2420    DATA C$WRIT             With I/O opcode : write,
             2421  *                         to save the control info. for
             2422  *                         the first record
             2423  *      Now start to use maximum-length record to write
             2424  *      the file and each record is copied into the VDP
             2425  *      from ERAM before  it is written
8F00 BD5430  2426    DST @STLN,@DDD1         Staring address on ERAM
             2427  * DST @>70,@EEE1          @EEE1 has been set up before
             2428  * DSUB 253,@EEE1          Staring address of the data buffer
             2429  *                              on VDP RAM
8F03 BD0880  2430    DST @RAMTOP,@CCC1
8F06 84
8F07 A50830  2431    DSUB @STLN,@CCC1        Total # of bytes in ERAM to move
8F0A 9108    2432    DINC @CCC1
8F0C BEE009  2433    ST 254,RAM(CNT(PABPTR)) Set the character count of PAI
8F0F 04FE
```

```
                2434    $REPEAT
8F11 BF5600     2435      DST 254,@FFF1         @FFF1:Byte count                1
8F14 FE
8F15 0F8B       2436      XML GVWITE            Move data from ERAM to VDP      1
8F17 069749     2437      CALL IOCALL           Call device service routine     1
8F1A 03         2438      DATA C$WRIT                                           1
8F1B A35400     2439      DADD 254,@DDD1        Update the source addr.on ERAM  1
8F1E FE
8F1F A70800     2440      DSUB 254,@CCC1        # of bytes left to move          1
8F22 FE
8F23 6F39       2441      BS   GSAV1           No more bytes to save           1
8F25 CB0800     2442    $UNTIL @CCC1 .DL. 254     Leave the last record alone
8F28 FE6F11
                2443  *       Move the last @CCC1 bytes from ERAM to VDP RAM
8F2B BD5608     2444    DST @CCC1,@FFF1           @FFF1:Byte count
8F2E 0F8B       2445    XML GVWITE                Write data from ERAM to VDP RAM
8F30 BCE009     2446    ST @CCC1+1,RAM(CNT(PABPTR)) Update the character count in
8F33 0409
8F35 069749     2447    CALL IOCALL             Call device service routine.   PAB
8F38 03         2448    DATA  C$WRIT
                2449  GSAV1
8F39 069749     2450    CALL IOCALL             Close the file
8F3C 01         2451    DATA C$CLOS
8F3D 4EA6       2452    BR   LRTOPL             Continue
                2453  ********************************************************
                2454  ********************************************************
                2455  * Move the program text & ln # table to VDP, and relocate
                2456  GVMOV
8F3F BD0C30     2457    DST @STLN,@BBB1         Save STLN,ENLN for later use
8F42 BD0832     2458    DST @ENLN,@CCC1
8F45 BD5430     2459    DST @STLN,@DDD1         Source addr. on ERAM
8F48 BD5810     2460    DST @VAR5,@EEE1         Destination addr. on VDP
8F4B BD0A58     2461    DST @EEE1,@STADDR       Use later for RELOCA
8F4E BD5680     2462    DST @RAMTOP,@FFF1
8F51 84
8F52 A55630     2463    DSUB @STLN,@FFF1        # of bytes to move
8F55 9156       2464    DINC @FFF1              @FFF1 : Byte count for GVWITE
8F57 0F8B       2465    XML  GVWITE             Move from ERAM to VDP
8F59 BDA3BC     2466    DST @RAMTOP,RAM(OLDTOP) Set up @RAMTOP for old top of memo
8F5C 8084
8F5E BDA3BE     2467    DST @>70,RAM(NEWTOP)    Set up @>70 for new top of memory
8F61 70
8F62 068D2B     2468    CALL RELOCA             Relocate the program
8F65 BD4030     2469    DST @STLN,@FREPTR       Set up @FREPTR
8F68 9340       2470    DDEC @FREPTR
8F6A 00         2471    RTN
                2472  ********************************************************
                2473  *       Save the crunched form of a program into a file     *
                2474  *       Move the ln # and text to the crunch buffer, then   *
                2475  *       write to the file one ln at a time                  *
                2476  ********************************************************
                2477  SAVMG
                2478  * Open the file with
                2479  *    I/O opcode : OPEN
                2480  *    File type : SEQUENTIAL file
```

```
                2481  *   Mode of operation : OUTPUT
                2482  *   Data type : DISPLAY type data
                2483  *   Recordtype : VARIABLE LENGTH records
                2484  *   Data buffer address : crunch buffer address
                2485  *   Logical record length : 163 (length of crunch buffer +
                2486  *                           2 bytes for ln #) maximum
8F6B 310009     2487      MOVE 9 FROM ROM(PAB1) TO RAM(4(PABPTR)) Build the PAB
8F6E E00404
8F71 8FEE
8F73 069750     2488      CALL IOCL$1               Call the DSR routine to open file
8F76 BD5032     2489      DST  @ENLN,@FAC6          Start from the first ln #
8F79 A75000     2490      DSUB >03,@FAC6            @FAC6 now points to the 1st ln #
8F7C 03
                2491      $REPEAT                   Write to the file from crunch buffr
                2492  *                             one ln at a time
8F7D 8600       2493        CLR @VARO               Make it a two byte later          1
8F7F 8E8084     2494        $IF @RAMTOP .NE. 0 THEN   If ERAM exists then             2
8F82 6FAB
8F84 BD5450     2495          DST @FAC6,@DDD1       Write the 4 bytes (ln # and ln ptr2
                2496  *                             from ERAM to crunch buffer
                2497  *                             @DDD1:Source addr. on ERAM
8F87 BF5808     2498          DST CRNBUF,@EEE1      @EEE1:Destination addr. on VDP    2
8F8A 20
8F8B BF5600     2499          DST 4,@FFF1           @FFF1:Byte count                  2
8F8E 04
8F8F 0F8B       2500          XML GVWITE            Write data from ERAM to VDP       2
8F91 BD54A8     2501          DST RAM(CRNBUF+2),@DDD1 Ln ptr now points to len byte2
8F94 22
8F95 9354       2502          DDEC @DDD1            Get the length of this ln         2
                2503  *                             @DDD1:Source addr. on ERAM
8F97 9156       2504          DINC  @FFF1           @FFF1:Byte count, coming back from2
                2505  *                             GVWITE above, =0.
8F99 0F8C       2506          XML GREAD1            Read the length byte from ERAM    2
8F9B BC0158     2507          ST @EEE1,@VARO+1      @EEE1:Destination addr. on CPU    2
8F9E BF5808     2508          DST CRNBUF+2,@EEE1    Write the text from ERAM to 3rd   2
8FA1 22
                2509  *                             byte of crunch buffer
                2510  *                             @EEE1:Destination addr. on VDP
                2511  *                             @DDD1:Source addr. on ERAM
8FA2 9154       2512          DINC @DDD1            Back to point to the text         2
8FA4 BD5600     2513          DST @VARO,@FFF1       @FFF1:Byte count                  2
8FA7 0F8B       2514          XML GVWITE            Write data from ERAM to VDP       2
8FA9 4FC2       2515        $ELSE                   ERAM not exist : ln # table and   2
                2516  *                             text in VDP
8FAB BDA820     2517          DST RAM(@FAC6),RAM(CRNBUF) PUT THE LN # IN              2
8FAE B050
8FB0 BD4CE0     2518          DST RAM(2(FAC6)),@FAC2 Get the ln ptr out               2
8FB3 0250
8FB5 934C       2519          DDEC @FAC2            Ln ptr now points to the len byte 2
8FB7 BC01B0     2520          ST RAM(@FAC2),@VARO+1  Get the length out               2
8FBA 4C
                2521  *             Move the text into the crunch buffer
8FBB 3400A8     2522          MOVE @VARO FROM RAM(1(FAC2)) TO RAM(CRNBUF+2)           2
8FBE 22E001
8FC1 4C
```

```
                2523        $END IF
8FC2 B2A820     2524        RB RAM(CRNBUF),7        Reset possible breakpt          1
8FC5 7F
8FC6 9500       2525        DINCT @VARO             Total len = text len + ln # len(2)1
8FC8 BCE009     2526        ST @VARO+1,RAM(CNT(PABPTR)) Store in the character coun1
8FCB 0401
8FCD 069749     2527        CALL IOCALL             Call the device service routine to1
8FD0 03         2528        DATA C$WRIT             write                           1
8FD1 A75000     2529        DSUB 4,@FAC6            Go to the next ln #             1
8FD4 04
8FD5 C95030     2530      $UNTIL @FAC6 .DL. @STLN   Finish moving all
8FD8 6F7D
8FDA BFA820     2531      DST >FFFF,RAM(CRNBUF)     Set up a EOF for the last record
8FDD FFFF
8FDF BEE009     2532      ST 2,RAM(CNT(PABPTR)) Only write this 2 bytes
8FE2 0402
8FE4 069749     2533      CALL IOCALL             Call the device service routine to
8FE7 03         2534      DATA C$WRIT             write
8FE8 069749     2535      CALL IOCALL             Call the device service routine to
8FEB 01         2536      DATA C$CLOS             close the file
8FEC 4EA6       2537      BR LRTOPL               Go back to top level
                2538    PAB1
8FEE 001208     2539      DATA 0,>12,#CRNBUF,163,0,#0,OFFSET
8FF1 20A300
8FF4 000060
```

```
                    2541  ********************************************************
                    2542  *            M E R G E    R O U T I N E                  *
                    2543  *                                                        *
                    2544  *       MERGE  load a file which is in crunched program  *
                    2545  *       form into the CRNBUF one record (one ln) at a time *
                    2546  *       then take the ln # out in FAC, text length into  *
                    2547  *       @CHAT, and edit it into the program.             *
                    2548  *       Identify EOF by the last record which is set up  *
                    2549  *       at SAVE time                                     *
                    2550  ********************************************************
                    2551  MERGE
8FF7 069225         2552    CALL GPNAME             Close all file, set up PAB
8FFA 8645           2553    CLR @FLAG               Does not check PROTECTION VIOLATION
                    2554  *                         in MERGE
8FFC 0F79           2555    XML PGMCHR              Check EOL
8FFE 8E42           2556    CZ  @CHAT
9000 4109           2557    BR  ERRSYN                Not EOL : SYNTAX ERROR
                    2558  * Open the file with
                    2559  *   I/O opcode : OPEN
                    2560  *   File type : SEQUENTIAL file
                    2561  *   Mode of operation : INPUT
                    2562  *   Data type : DISPLAY type data
                    2563  *   Recordtype : VARIABLE LENGTH records
                    2564  *   Data buffer addr. : crnbuf addr.
                    2565  *   Logical record length : 163 maximum
9002 310009         2566    MOVE 9 FROM ROM(PAB1) TO RAM(4(PABPTR)) Set up PAB
9005 E00404
9008 8FEE
900A 94E005         2567    INCT RAM(FLG(PABPTR))   Put in correct i/o mode :>14
900D 04
900E 069750         2568    CALL IOCL$1             Call the device service routine to
                    2569  *                         open the file
9011 069749         2570    CALL IOCALL             Call the device service routine to
9014 02             2571    DATA C$READ             read
9015 D7A820         2572    $IF RAM(CRNBUF) .DEQ. >FFFF GOTO ERR$2B If 1st rec is EOF
9018 FFFF77
901B 7F
                    2573    $REPEAT                 Read in one ln and edit it to progm
901C 8780D6         2574      DCLR @>D6             Reset VDP timeout                1
901F BC42E0         2575      ST RAM(CNT(PABPTR)),@CHAT Len of this record           1
9022 0904
9024 9642           2576      DECT @CHAT            Text len = total len - 2 (ln # len
                    2577  *                         Put it in @CHAT for EDITLN
9026 BD4AA8         2578      DST RAM(CRNBUF),@FAC  Put the ln # in @FAC for EDITLN  1
9029 20
902A 8656           2579      CLR @FAC12            Make it a double byte            1
902C BC5742         2580      ST @CHAT,@FAC13                                        1
                    2581  *       Move the text up 2 bytes
902F 3456A8         2582      MOVE @FAC12 FROM RAM(CRNBUF+2) TO RAM(CRNBUF)        1
9032 20A822
9035 BDA39E         2583      DST @PABPTR,RAM(MRGPAB) SAVE PAB PTR                 1
9038 04
9039 066032         2584      CALL EDITLN           EDIT IT TO THE PROGRAM           1
903C BD04A3         2585      DST RAM(MRGPAB),@PABPTR RESTORE PAB PTR
903F 9E
```

```
9040 069749  2586      CALL IOCALL            CALL THE DEVICE SERVICE ROUTINE TO1
9043 02      2587      DATA  C$READ           read another record or another lin1
9044 D7A820  2588    $UNTIL RAM(CRNBUF) .DEQ. >FFFF  End on EOF
9047 FFFF50
904A 1C
             2589  MERG$1
             2590  *     Double check EOF record
904B D6E009  2591    $IF RAM(CNT(PABPTR)) .NE. 2 GOTO ERR$2B I/O ERROR
904E 040257
9051 7F
9052 069749  2592    CALL IOCALL            Call the device service routine to
9055 01      2593    DATA C$CLOS            close the file
9056 4EA6    2594    BR LRTOPL              Go back to top level
```

ERR$4

```
DST  RAM(MGRPAB), @PABPTR      4
$IF  @PABPTR .NE. Ø THEN       5
     CALL IOCALL               3
     DATA C$CLOS               1
$END IF                       ---
                              13
                               4
                              ---
                              17
                               2
                              ---
                              19
```

```
                    2596  ****************************************************************
                    2597  *              L I S T   R O U T I N E                         *
                    2598  *                                                              *
                    2599  *      LIST lists a readable copy of the current program       *
                    2600  *      image to the specified device. In case no device        *
                    2601  *      is specified, the listing is copied to the screen.      *
                    2602  *                                                              *
                    2603  *           This routine uses the fact that ERR$$ returns      *
                    2604  *      to the caller if the call has been issued in EDIT       *
                    2605  *      mode.  Therefore, ERR$2 will return to TOPL10,          *
                    2606  *      which will reinitiate the variable stuff                *
                    2607  ****************************************************************
                    2608  LIST
9058 DA4580         2609    $IF .BIT7 @FLAG .EQ. 1 GOTO ERRPV  PROTECTION VIOLATION EF
905B 57C6
905D 8714           2610    DCLR @CURLIN            Create some kind of control for
905F 870E           2611    DCLR @CURINC              defaults
9061 BE082D         2612    ST   MINUS,@VARC        Select "-" as separator
9064 06602E         2613    CALL AUTO1              Pick up any available arguments
                    2614    $
                    2615    $     If either CURLIN or CURINC is non-zero, use it.
                    2616    $     For zero values replace the default (ENLN-3, STLN)
                    2617    $
9067 8F1450         2618    $IF @CURLIN .DEQ. 0 THEN                                    1
906A 84
906B BD5432         2619        DST @ENLN,@DDD1     Get the first ln.'s ln. #       1
906E A75400         2620        DSUB 3,@DDD1        DDD1 : Source addr on ERAM/VDP  1
9071 03
9072 0691AC         2621        CALL GRSUB3         Read the ln. # from ERAM/VDP    1
9075 54             2622        DATA DDD1           @DDD1:Source addr. on ERAM/VDP  1
                    2623  *                         Reset possible breakpt too
9076 BD1458         2624        DST @EEE1,@CURLIN   Use standard default            1
9079 8F0E50         2625      $IF @CURINC .DEQ. 0 THEN                              2
907C 84
                    2626  LIST$0
907D 0691AC         2627        CALL GRSUB3         Read last ln. # from ERAM/VDP   2
9080 30             2628        DATA STLN           @STLN:Source addr. on ERAM/VDP  2
                    2629  *                         Reset possible breakpt too
9081 BD0E58         2630        DST @EEE1,@CURINC   @EEE1:Destination addr. on ERAM/VD2
                    2631  *                         Also default for end line
                    2632      $END IF
                    2633    $END IF
                    2634    $
                    2635    $     Now first evaluate what we've got in CURLIN
                    2636    $
9084 8F0E50         2637    $IF @CURINC .DEQ. 0 THEN  Check for combination xxx-       1
9087 96
                    2638      $REPEAT
9088 9320           2639        DDEC @VARW          Backup to the separation mark   2
908A D6B020         2640      $UNTIL RAM(@VARW) .NE. : :+OFFSET                     1
908D 807088
9090 D6B020         2641      $IF RAM(@VARW) .EQ. :-:+OFFSET GOTO LIST$0 Select last 1
9093 8D707D
                    2642    $END IF
9096 C90E14         2643    $IF @CURINC .DL. @CURLIN THEN If something like LIST i-- 11
```

```
9099 709E
909B BD0E14  2644       DST  @CURLIN,@CURINC Replace by LIST 15-15          1
             2645     $END IF
909E BD4A14  2646     DST  @CURLIN,@FAC     Prepare for line # search
90A1 0F7E    2647     XML  SPEED            Search the line number table
90A3 03      2648     DATA SEETWO
90A4 BD142E  2649     DST  @EXTRAM,@CURLIN  Get first real line # in CURLIN
90A7 BD4A0E  2650     DST  @CURINC,@FAC
90AA 0F7E    2651     XML  SPEED
90AC 03      2652     DATA SEETWO           Evaluate second line #
90AD 0691AC  2653     CALL GRSUB3           Read 2 bytes of data from ERAM/VDP
90B0 2E      2654     DATA EXTRAM           @EXTRAM:Source addr. on ERAM/VDP
             2655  *                        Reset possible breakpt too
90B1 C5580E  2656     $IF  @EEE1 .DH. @CURINC THEN                           1
90B4 50BA
90B6 A32E00  2657       DADD 4,@EXTRAM        Else take next lower line    1
90B9 04
             2658     $END IF
90BA BD0E2E  2659     DST  @EXTRAM,@CURINC  Which could be equal to CURLIN
90BD BD2E14  2660     DST  @CURLIN,@EXTRAM  For use below by LLIST
90C0 932C    2661     DDEC @PGMPTR          Backup to last CHAT
90C2 0F79    2662     XML  PGMCHR           Retrieve last CHAT
90C4 8E4271  2663     $IF @CHAT .NE. 0 THEN Devicename available             1
90C7 22
90C8 0681F4  2664       CALL CLSALL           Close all files that are open  1
90CB BF6E09  2665       DST  VRAMVS,@VSPTR    Re-initialize the V-stack      1
90CE 58
90CF BD246E  2666       DST  @VSPTR,@STVSPT   And it's base                  1
90D2 0F79    2667       XML  PGMCHR         Get name length in CHAT          1
90D4 BF0409  2668       DST  VRAMVS+16,@PABPTR Get entrypoint in PAB         1
90D7 68
90D8 8617    2669       CLR  @DSRFLG          Indicate device I/O            1
90DA 310009  2670       MOVE 9 FROM ROM(PAB) TO RAM(4(PABPTR))               1
90DD E00404
90E0 9160
90E2 BF0809  2671       DST  VRAMVS+16+NLEN,@CCPADR Select start address for co1
90E5 75
90E6 BC4C42  2672       ST   @CHAT,@FAC2      Number of characters to copy   1
90E9 904C    2673       INC  @FAC2            Plus length byte               1
             2674  LIST$1
90EB BCB008  2675       ST   @CHAT,RAM(@CCPADR) Copy the bytes one by one     1
90EE 42
90EF 0F79    2676       XML  PGMCHR           Get next character             1
90F1 9108    2677       DINC @CCPADR          CCPADR ends up with highest addres1
90F3 924C    2678       DEC  @FAC2            Count total # of characters    1
90F5 50EB    2679       BR   LIST$1                                          1
90F7 069750  2680       CALL IOCL$1           Perform OPEN on DSR            1
90FA 864A    2681       CLR  @FAC             Create double byte PAB length  1
90FC BC07E0  2682       ST   RAM(LEN(PABPTR)),@RECLEN Get record length      1
90FF 0804
9101 BC4B07  2683       ST   @RECLEN,@FAC1    Compute record length          1
9104 A14A08  2684       DADD @CCPADR,@FAC     Get highest address used       1
9107 BDE006  2685       DST  @CCPADR,RAM(BUF(PABPTR)) Store it               1
910A 0408
910C 8E8084  2686       $IF @RAMTOP .NE. 0 THEN  If ERAM exists then         2
```

```
910F 7118
9111 C54A70  2687        $IF @FAC . DH. @>70 GOTO ERRIO  Compare with top of
9114 77A6
             2688  *                      VDP: if higher then 'not enough room'
9116 511D    2689      $SELSE
9118 C54A30  2690        $IF @FAC . DH. @STLN GOTO ERRIO   Not enough room
911B 77A6
             2691      $END IF
911D BE0601  2692      ST   1,@CCPPTR          Clear first line in output
9120 5128    2693    $SELSE
9122 BE7F1F  2694      ST   VWIDTH+3,XPT       For common code usage
9125 06972A  2695      CALL INITKB             Reset current record length
             2696    $SEND IF
             2697    $REPEAT                   Display at least one line
9128 8E8084  2698      $IF @RAMTOP .NE. 0 THEN  If ERAM exists then
912B 7130
912D 069169  2699      CALL GRMLST             Fake it: move each ln to the
             2700      $END IF                 CRUNCH buffer from ERAM
9130 066A74  2701      CALL LLIST              List the current line
9133 03      2702      SCAN                    Test for a break key
9134 513E    2703      BR   LIST$3             No key
9136 D67502  2704      $IF @RKEY .EQ. BREAK GOTO LIST$4
9139 714F
             2705  LIST$5
913B 03      2706      SCAN
913C 513B    2707      BR LIST$5
             2708  LIST$3
913E 8E8084  2709      $IF @RAMTOP .NE. 0 THEN If ERAM exists
9141 7146
9143 BD2E58  2710        DST @FAC14,@EXTRAM Restore the @EXTRAM
             2711      $END IF
9146 A72E00  2712      DSUB 4,@EXTRAM          Pointer to next line
9149 04
914A C50E2E  2713    $UNTIL @CURINC . DH. @EXTRAM Displayed all lines in range
914D 5128
             2714  LIST$4
914F 8E1751  2715    $IF @DSRFLG .EQ. 0 THEN Device I/O -> output last rec.
9152 5D
9153 069690  2716      CALL OUTREC             Output the last record
9156 069749  2717      CALL IOCALL             Close the device properly
9159 01      2718      DATA C$CLOS
915A 05601A  2719      B    TOPL10
             2720    $END IF
915D 056012  2721    B    TOPL15               Restart the variables too
             2722  *
             2723  *       PAB image used in LIST function
             2724  *
             2725  PAB
9160 001200  2726    DATA 0,>12,#0,0,0,#0,OFFSET
9163 000000
9166 000060
             2727  *  Move each line in ERAM to CRNBUF area, put ln. no. in
             2728  *  (CRNBUF), Put CRNBUF+4 in (CRNBUF+2) which is the ln. ntr
             2729  *  field, put the text itself from ERAM to (CRNBUF+4),B   e
             2730  *  calling LLIST, trick it by moving CRNBUF to @EXTRAM
```

```
                    2731  GRMLST
9169 0691AC         2732        CALL GRSUB3          Get ln. # from ERAM(use GREAD1)
916C 2E             2733        DATA EXTRAM          @EXTRAN : Source addr. on ERAM
                    2734  *                          Reset possible break pt too
916D BDA820         2735        DST @EEE1,RAM(CRNBUF)  Put it in CRNBUF
9170 58
9171 BFA822         2736        DST CRNBUF+4,RAM(CRNBUF+2) Put CRNBUF+4 into
9174 0824
                    2737  *                          the ln ptr field
9176 9554           2738        DINCT @DDD1          Get the ptr to the text from GR
9178 06919A         2739        CALL GRSUB4          Read the ln. pointer in (use
                    2740  *                          GREAD1)
917B 9358           2741        DDEC @EEE1           Get the ptr to the lengh byte
917D 069194         2742        CALL GRSUB2          Read the length form ERAM, use
9180 58             2743        DATA EEE1            GREAD1,@EEE1:Source addr. on ERA
9181 BC5758         2744        ST @EEE1,@FFF1+1     Use the length as byte count
                    2745  *                          to move the text from ERAM to
                    2746  *                          VDP CRNBUF+4 area
9184 BF5808         2747        DST CRNBUF+4,@EEE1   EEE1 : Destination addr. on VDP
9187 24
9188 9154           2748        DINC @DDD1           DDD1 : Source addr. on ERAM
                    2749  *                          Now point to the text
918A 0F8B           2750        XML GVWITE           Move data form ERAM to VDP
918C BD582E         2751        DST @EXTRAM,@FAC14   Save for later use
918F BF2E08         2752        DST CRNBUF,@EXTRAM   Fake it
9192 20
9193 00             2753        RTN
```

```
                    2755  *
                    2756  *       SUBROUTINE TO READ 2 BYTES OF DATA FROM ERAM OR VDP
                    2757  *       WITH THE OPTION TO RESET THE POSSIBLE BREAKPT
                    2758  *
                    2759  GRSUB2
9194 8856           2760          FETCH @FFF1               Fetch the source addr. on ERAM
9196 BD5490         2761          DST *FFF1,@DDD1           or VDP
9199 56
                    2762  *                                 @DDD1:Source addr. on ERAM/VDP
                    2763  GRSUB4
919A 8E8084         2764          $IF @RAMTOP .NE. 0 THEN If ERAM exists                1
919D 71A7
919F BF5600         2765            DST 2,@FFF1             @FFF1:Byte count            1
91A2 02
91A3 0F8C           2766            XML GREAD1              Read data from ERAM to CPU  1
91A5 51AB           2767          $SELSE                    ERAM not exists             1
91A7 BD58B0         2768            DST RAM(@DDD1),@EEE1    Read data from VDP to CPU   1
91AA 54
                    2769          $END IF
91AB 00             2770          RTN
                    2771  GRSUB3
91AC 8856           2772          FETCH @FFF1               Fetch the source addr. on ERAM
91AE BD5490         2773          DST *FFF1,@DDD1           or VDP
91B1 56
                    2774  *                                 @DDD1:Source addr. on ERAM/VDP
91B2 06919A         2775          CALL GRSUB4               Do the actual read
91B5 B3587F         2776          DAND >7FFF,@EEE1          Reset possible breakpt
91B8 FF
91B9 00             2777          RTN
```

```
                    2779  *
                    2780  *              R E C   R O U T I N E
                    2781  *
                    2782  *          REC(X) returns the current record to which file
                    2783  *          X is positioned.
                    2784  *
                    2785  SUBREC
91BA BD5C04         2786    DST  @PABPTR,@ARG       Save the current PAB^ & set new one
91BD 06920E         2787    CALL SUBEOF             Try to find the correct PAB
91C0 C15C04         2788    DEX @PABPTR,@ARG        @ARG: new PAB ^
                    2789  *                         @PABPTR : restore current PAB ^
91C3 5202           2790    BR   EOF$2              Didn't find the corresponding PAB
91C5 BD4AE0         2791    DST  RAM(RNM(ARG)),@FAC  Obtain integer record number
91C8 0A5C
91CA 0F80           2792    XML    CIF              Convert integer to floating
91CC 0F75           2793    XML   CONT                and continue
                    2794  ********************************************************
                    2795  *            E O F   R O U T I N E                     *
                    2796  *                                                      *
                    2797  *       EOF(X) returns status codes on file X.  The meaning *
                    2798  *       of the resultcodes is :                        *
                    2799  *                                                      *
                    2800  *            -1            Physical End Of File        *
                    2801  *            0             Not at End Of File yet      *
                    2802  *            +1            Logical End Of File         *
                    2803  ********************************************************
                    2804  EOF
91CE BD5C04         2805    DST @PABPTR,@ARG        Save the current PAB ^ and set the
                    2806  *                         new one in SUBEOF
91D1 06920E         2807    CALL   SUBEOF           Try to find the PAB somewhere
91D4 57C2           2808    BR     ERRFE            Can't find
91D6 BE5E09         2809    ST   C$STAT,@ARG2       Select status code without
91D9 C0E004         2810    EX   @ARG2,RAM(COD(PABPTR))  destroying original code
91DC 045E
91DE 069750         2811    CALL IOCL$1             Get the information from the DSR
91E1 C1045C         2812    DEX  @ARG,@PABPTR       Restore original PAB ^ and original
91E4 BCE004         2813    ST   @ARG2,RAM(COD(ARG))   I/O code
91E7 5C5E
91E9 BC5EE0         2814    ST   RAM(SCR(ARG)),@ARG2   And pick up STATUS
91EC 0C5C
91EE 310008         2815    MOVE 8 FROM ROM(FLOAT1) TO @FAC Get floating 1
91F1 4A9206
91F4 DA5E03         2816    CLOG 3,@ARG2            Test EOF bits
91F7 7202           2817    BS   EOF$2              No EOF indication
91F9 DA5E02         2818    $IF .BIT1 @ARG2 .EQ. 1 THEN Physical EOF                1
91FC 7200
91FE 834A           2819      DNEG @FAC             Make result -1                  1
                    2820    $END IF
9200 0F75           2821    XML  CONT
                    2822  EOF$2
9202 874A           2823    DCLR @FAC               Create result 0
9204 0F75           2824    XML  CONT
                    2825
                    2826  FLOAT1
9206 400100         2827    DATA >40,1,0,0,0,0,0,0  Floating point +1
```

```
9209 000000
920C 0000
                2828
                2829  SUBEOF
920E D642B7     2830    $IF @CHAT .NE. LPAR$ GOTO ERRSYN   * SYNTAX ERROR
9211 4109
9213 0F74       2831      XML  PARSE              Parse up to the matching ")"
9215 FF         2832      DATA >FF
9216 069347     2833      CALL  CHKCNV            Convert and search for PAB
9219 77B6       2834      BS    ERRBV             Avoid 0's and negatives Bad value!
921B BC6217     2835      ST   @DSRFLG,@ARG6      @DSRFLG got changed in CHKCON
921E 06935C     2836      CALL CHKCON             Check and search given filenumber
9221 BC1762     2837      ST   @ARG6,@DSRFLG     @DSRFLG got changed in CHKCON
9224 01         2838      RTNC                    Condition set:file no. exists
                2839
```

```
                    2841  ***************************************************************
                    2842  * L O A D / S A V E / M E R G E  U T I L I T Y  R O U T
                    2843  *
                    2844  *       GPNAME gets program name for OLD and SAVE
                    2845  *       Can also be used for future implementation of
                    2846  *       REPLACE statement.
                    2847  *            Also gives valuable contribution to updating
                    2848  *       of program pointers (VSPTR, STVSPT, FLAG, etc.)
                    2849  *       and creation of LOAD/SAVE PAB.
                    2850  ***************************************************************
                    2851  GPNAME
9225 B24580         2852    AND >80, @FLAG          Avoid returns from ERR$$ routin
                    2853  *                         Keep the protection bit
9228 D642C7         2854    $IF @CHAT .NE. STRIN$ THEN                                  1
922B 7232
922D D642C8         2855      $IF @CHAT .NE. NUM$ GOTO ERRSYN  "* SYNTAX ERROR"         1
9230 4109
                    2856    $END IF
9232 0681F4         2857    CALL CLSALL             First close all open files
9235 066022         2858    CALL KILSYM             Kill the symbol table
9238 BF0409         2859    DST  VRAMVS+8,@PABPTR   Create PAB as low as possible
923B 60
923C 86B004         2860    CLR  RAM(@PABPTR)       Clear PAB with ripple-move
923F 350009         2861    MOVE PABLEN-5 FROM RAM(@PABPTR) TO RAM(1(PABPTR))
9242 E00104
9245 B004
9247 0F79           2862    XML  PGMCHR             Get length of file-spec.
9249 A70400         2863    DSUB 4,@PABPTR          Make it a regular PAB ^
924C 04
924D BCE00D         2864    ST   @CHAT,RAM(NLEN(PABPTR)) Copy name length to PAB
9250 0442
9252 BD0AE0         2865    DST RAM(NLEN-1(PABPTR)),@STADDR Avoid problems(bugs!)
9255 0C04
9257 8E8089         2866    $IF @RAMFLG .EQ. 0 THEN If ERAM not exist or imperative si
925A 5265
925C 340AE0         2867      MOVE @STADDR FROM RAM(@PGMPTR) TO RAM(NLEN+1(PABPTR))     1
925F 0E04B0
9262 2C
9263 5274           2868    $ELSE                                                       1
9265 BD560A         2869      DST @STADDR,@FFF1          @FFF1:Byte count           1
9268 BD542C         2870      DST @PGMPTR,@DDD1          Source addr. on ERAM       1
926B BD5804         2871      DST @PABPTR,@EEE1                                     1
926E A35800         2872      DADD NLEN+1,@EEE1          Destination addr. on VDP   1
9271 0E
9272 0F8B           2873      XML GVWITE                 Write from ERAM to VDP     1
                    2874    $END IF
9274 A12C0A         2875    DADD @STADDR,@PGMPTR    Skip the string
                    2876    $
                    2877    $    OLD and SAVE can only be imperative
                    2878    $
9277 8634           2879    CLR   @DATA             Clear DATA line
9279 00             2880    RTN                     That's all folks
```

```
                    2882  ********************************************************************
                    2883  *                                                                  *
                    2884  *  R E A D / I N P U T   U T I L I T Y   R O U T I N E S           *
                    2885  *                                                                  *
                    2886  ********************************************************************
                    2887  GETVAR
927A BD0A2C         2888    DST   @PGMPTR,@STADDR    Save token pointer to first chr
927D 8610           2889    CLR   @VAR5              Clear # of parsed variables
927F BD0E6E         2890    DST   @VSPTR,@VAR4       Save first entry in V-stack
                    2891    $
                    2892    $     Start parse cycle for INPUT statement
                    2893    $
                    2894  GETV$0
9282 0F7A           2895    XML   SYM                Get correct symbol table entry
9284 8611           2896    CLR   @VAR6              Start with zero paren nesting
                    2897  GETV$1
9286 D642B7         2898    $IF @CHAT .EQ. LPAR$ THEN Increment counter for "("      1
9289 528D
928B 9011           2899      INC   @VAR6                                            1
                    2900    $END IF
928D 8E1172         2901    $IF @VAR6 .NE. 0 THEN    Watch out for final balance    1
9290 A1
9291 069598         2902      CALL  CHKEND           Check for unbalanced parenthesis 1
9294 6109           2903      BS    ERRSYN             Somebody forget something!!!!! 1
9296 D642B6         2904      $IF @CHAT .EQ. RPAR$ THEN    Decrement for ")"        2
9299 529D
929B 9211           2905        DEC   @VAR6                                          2
                    2906      $END IF
929D 0F79           2907      XML   PGMCHR           Get character following last "," 1
929F 5286           2908      BR    GETV$1                                           1
                    2909    $END IF
92A1 0F77           2910    XML   VPUSH              Push entry to V-stack
92A3 9010           2911    INC   @VAR5              Count all pushed variables
92A5 069598         2912    CALL  CHKEND           Next should either be EOS or ","
92A8 72B7           2913    BS    GETV$2           Found it EOS!!!!
92AA 0F7E           2914    XML   SPEED            Must be at a
92AC 00             2915    DATA  SYNCHK               comma else
92AD B3             2916    DATA  COMMA$                 its an error
92AE 069598         2917    CALL  CHKEND         Check for end of statement
92B1 5282           2918    BR    GETV$0         Haven't found it yet
92B3 8E1741         2919    $IF @DSRFLG .NE. 0 GOTO ERRSYN  Error for keyboard I/O
92B6 09
                    2920  GETV$2
92B7 00             2921    RTN
                    2922    $
                    2923    $     Create a temporary string in memory.  BYTE
                    2924    $     contains the length
                    2925    $
                    2926  CTSTR
92B8 BF4C65         2927    DST   >6500,@FAC2        Indicate string in FAC
92BB 00
                    2928  CTSTR0
92BC BD500C         2929    DST   @BYTE,@FAC6        Copy string length in FAC6
92BF 0F71           2930    XML   GETSTR             Reserve the string
92C1 BD4E1C         2931    DST   @SREF,@FAC4        Copy start address of string
```

```
92C4 BF4A00  2932     DST  SREF,@FAC          And indicate temp. string
92C7 1C
92C8 00      2933     RTN
             2934     $
             2935     $    Create a temporary string from TEMP5.  Length
             2936     $    is given in BYTE.
             2937     $
             2938   CTMPST
92C9 0692B8  2939     CALL CTSTR              Create the temporary string
92CC 8E5172  2940     $IF @FAC7 .NE. 0 THEN                                  1
92CF D6
92D0 340CB0  2941       MOVE @BYTE FROM RAM(@TEMP5) TO RAM(@SREF)            1
92D3 1CB066
             2942     $END IF                 non-empty
92D6 00      2943     RTN
             2944   *
             2945   *      CHKNUM - Check for numeric argument
             2946   *
             2947   CHKNUM
92D7 D601C8  2948     $IF @VAR0+1 .EQ. NUM$ THEN                             1
92DA 52EE
92DC 0692F7  2949       CALL GETRAM           Get string length              1
92DF BD5634  2950       DST  @DATA,@FAC12     Store entry for conversion      1
92E2 8600    2951       CLR  @VAR0            Prepare for double action       1
92E4 A13400  2952       DADD @VAR0,@DATA      Get end of data field           1
92E7 06A012  2953       CALL CONVER           Convert data to FAC #           1
             2954       $
             2955       $  Conversion should also end at end of field
             2956       $
92EA D5A390  2957       DCEQ @DATA,RAM(CSNTMP)    Set COND according to equalit1
92ED 34
             2958     $END IF
92EE 01      2959     RTNC                    Back to the caller
             2960   *
             2961   *      Get data from ERAM or VDP RAM
             2962   *
             2963   GETGFL
92EF BC4D80  2964     ST   @RAMTOP,@FAC3      Select target memory
92F2 84
             2965   GETDAT
92F3 8E4D52  2966     $IF @FAC3 .EQ. 0 THEN Get everything from RAM          1
92F6 FF
             2967   GETRAM
92F7 BC01B0  2968       ST   RAM(@DATA),@VAR0+1   Get data in VAR0+1         1
92FA 34
92FB 864D    2969       CLR  @FAC3            Be sure FAC3 = 0 !!!!          1
92FD 530B    2970     $SELSE                                                 1
92FF BF5600  2971       DST 1,@FFF1           FFF1 : byte count              1
9302 01
9303 BD5434  2972       DST @DATA,@DDD1       DDD1 : source addr on ERAM     1
9306 0F8C    2973       XML GREAD1            Read data from ERAM            1
9308 BC0158  2974       ST  @EEE1,@VAR0+1     EEE1 : Destination addr on CPU 1
             2975     $SEND IF
930B 9134    2976     DINC @DATA              Go to next datum for next time
930D 00      2977     RTN
```

```
                2978  *
                2979  *
                2980  *
                2981  CHKSTR
930E 8750       2982    DCLR @FAC6              Assume we'll have an empty string
9310 D601C7     2983    $IF @VAR0+1 .NE. STRIN$ THEN
9313 731A
9315 D601C8     2984      $IF @VAR0+1 .NE. NUM$ GOTO EMPSTR  See.......
9318 5329
                2985    $END IF
                2986  CHKS$0
931A 0692F3     2987    CALL GETDAT             Next datum is length byte
931D 8650       2988    CLR  @FAC6              Be sure high byte = 0 !!!!!!!
931F BC5101     2989    ST   @VAR0+1,@FAC7      Prepare FAC for string assignment
9322 BD6634     2990    DST  @DATA,@TEMP5       Save string address for assignment
9325 A13450     2991    DADD @FAC6,@DATA        Update DATA ^ for end of field
9328 00         2992    RTN
                2993    $
                2994    $     Empty strings are handled below
                2995    $
                2996  EMPSTR
9329 D601B3     2997    $IF @VAR0+1 .NE. COMMA$ THEN
932C 7333
932E 069336     2998      CALL DATEND           Check for end of data statement
9331 5377       2999      BR   RTC              Return with COND if not EOS
                3000    $END IF
9333 9334       3001    DDEC @DATA              Backup data pointer for empties
9335 00         3002    RTN
                3003  *
                3004  DATEND
9336 C04201     3005    EX   @VAR0+1,@CHAT       Exchange with CHAT for testing
9339 069598     3006    CALL  CHKEND            Check for EOS (=EOL or "::")
933C C04201     3007    EX    @VAR0+1,@CHAT     Restore original situation
933F 01         3008    RTNC
```

```
                    3010  ********************************************************
                    3011  *           O P E N / C L O S E / R E S T O R E        *
                    3012  *             U T I L I T Y   R O U T I N E S          *
                    3013  *                                                      *
                    3014  *       CHKFN - Check for token = "#" and collect and check *
                    3015  *       filenumber.  Also convert filenumber to (two byte)  *
                    3016  *       integer and check for range 0<X<256.           *
                    3017  ********************************************************
                    3018  CHKFN
9340 0F7E           3019    XML  SPEED              Must be at a
9342 00             3020    DATA SYNCHK               '#' else
9343 FD             3021    DATA NUMBE$                  its an error
9344 0F74           3022    XML  PARSE              Parse argument up to ":"
9346 B5             3023    DATA COLON$
                    3024    $
                    3025    $     Code to check for negative or zero result in
                    3026    $     floating point accu.  If not...convert to
                    3027    $     integer and return two byte integer in FAC
                    3028    $
                    3029  CHKCNV
9347 D64C65         3030    $IF @FAC2 .EQ. STRVAL GOTO ERRSNM String/number mismatch
934A 77AA
934C 8654           3031    CLR  @FAC10             Clear error-code byte
934E 0F12           3032    XML  CFI                Convert to two byte integer
9350 8E5457         3033    $IF @FAC10 .NE. 0 GOTO ERRBV      BAD VALUE ERROR
9353 B6
9354 DA4A80         3034    $IF .BIT7 @FAC .NE. 0 GOTO RTC Negative result
9357 5377
9359 8F4A           3035    DCZ  @FAC               And return with COND set/reset
935B 01             3036    RTNC
                    3037    $
                    3038  CHKCON
935C BC174B         3039    ST   @FAC1,@FNUM        Move result into FNUM
                    3040    $
                    3041    $     Check for high byte not zero ( >255 )
                    3042
935F 8E4A57         3043    $IF @FAC .NE. 0 GOTO ERRBV   Bad value error.
9362 B6
                    3044
                    3045    $     Search routine - Search for a given file number
                    3046    $     in the chain of allocated PABs.
                    3047    $     IOSTRT contains the start of the PAB-chain
                    3048    $
9363 BD043C         3049    DST  @IOSTRT,@PABPTR    Get first link in the chain
                    3050  CHKF$1
                    3051    $
                    3052    $     Check for last PAB in the chain and exit if found
                    3053    $
9366 8F0473         3054    $IF @PABPTR .DNE. 0 THEN  Check if file # is correct     1
9369 7A
936A D4E002         3055      $IF RAM(FIL(PABPTR)) .EQ. @FNUM GOTO RTC               1
936D 041773
9370 77
9371 BD04B0         3056      DST  RAM(@PABPTR),@PABPTR    Try the next PAB          1
9374 04
```

```
9375 5366     3057       BR    CHKF$1
              3058  RTC
9377 D40000   3059       CEQ   @0,@0            Force COND to "SET"
              3060     $END IF
937A 01       3061     RTNC                     Exit with no COND change
              3062
              3063  ***************************************************************
              3064  *       OUTEOF outputs the last record if this
              3065  *       record is non-empty, and if the PAB is
              3066  *       open for non-input mode (UPDATE, APPEND
              3067  *       or OUTPUT).
              3068  ***************************************************************
              3069  OUTEOF
937B 8617     3070     CLR   @DSRFLG
937D D6E004   3071     $IF RAM(COD(PABPTR)) .EQ. C$WRITE THEN Non-input mode
9380 040353
9383 90
9384 8EE003   3072       $IF RAM(OFS(PABPTR)) .NE. 0 THEN Non-empty record
9387 047390
938A 0696E0   3073         CALL PRINIT          Initiate for output
938D 069690   3074         CALL OUTREC          Output and remove pending cond.
              3075       $END IF
              3076     $END IF
9390 00       3077     RTN                      Return to whoever called
```

```
                3079  ********************************************************
                3080  *                                                      *
                3081  *      DELPAB routine - delete a given PAB from the chain  *
                3082  *      under the assumption that the PAB exists        *
                3083  *                                                      *
                3084  ********************************************************
                3085  DELPAB
                3086    $
                3087    $     First compute start and end address for block move
                3088    $
9391 BD0AE0     3089    DST   RAM(BUF(PABPTR)),@STADDR Get lowest used address
9394 0604
9396 930A       3090    DDEC @STADDR            Make that an address following PAB
9398 8608       3091    CLR  @CCPADR            Get highest address in CCPADR(2)
939A BC09E0     3092    ST   RAM(NLEN(PABPTR)),@CCPADR+1  complete the two bytes
939D 0D04
939F A2090D     3093    ADD  PABLEN-1,@CCPADR+1   Add PAB length - 1
93A2 A10804     3094    DADD @PABPTR,@CCPADR   Compute actual address within RAM
93A5 D53C04     3095    $IF @IOSTRT .DNE. @PABPTR THEN Watch out for first PAB   1
93A8 73D1
93AA BD023C     3096      DST  @IOSTRT,@MNUM   Figure out where link to PAB is   1
93AD D5B002     3097      $WHILE RAM(@MNUM) .DNE. @PABPTR Continue while not foun2
93B0 0473B9
93B3 BD02B0     3098        DST  RAM(@MNUM),@MNUM   Defer to next link in chain  2
93B6 02
93B7 53AD       3099      $SEND WHILE           Short end for code-savings        1
93B9 BDB002     3100      DST  RAM(@PABPTR),RAM(@MNUM) Copy link over deleted PAB1
93BC B004
93BE 8FB002     3101      $IF RAM(@MNUM) .DNE. 0 THEN Adjust link only if not zer2
93C1 73CB
93C3 A1B002     3102        DADD @CCPADR,RAM(@MNUM) Add deleted # of bytes for   2
93C6 08
93C7 A5B002     3103        DSUB @STADDR,RAM(@MNUM)    link correction           2
93CA 0A
                3104      $END IF
93CB BD04B0     3105      DST  RAM(@MNUM),@PABPTR   Get new PABPTR                1
93CE 02
93CF 53E2       3106    $SELSE                                                    1
93D1 BD3CB0     3107      DST  RAM(@PABPTR),@IOSTRT Update first link             1
93D4 04
93D5 8F3C73     3108      $IF @IOSTRT .DNE. 0 THEN  Only adjust if not last link 2
93D8 DF
93D9 A13C08     3109        DADD @CCPADR,@IOSTRT    Add deleted # of bytes        2
93DC A53C0A     3110        DSUB @STADDR,@IOSTRT                                  2
                3111      $END IF
93DF BD043C     3112      DST  @IOSTRT,@PABPTR      Get new PABPTR                1
                3113    $SEND IF
                3114    $
                3115    $     Move the bytes below the deleted block up in
                3116    $     memory. This includes both variables and PABs
                3117    $
93E2 BD020A     3118    DST  @STADDR,@MNUM      Get # of bytes to move
93E5 A50240     3119    DSUB @FREPTR,@MNUM
93E8 BD0608     3120    DST  @CCPADR,@CCPPTR    Save destination address
93EB 8F0273     3121    $WHILE @MNUM .DNE. 0                                      1
```

```
93EE FC
93EF BCB008  3122      ST   RAM(@STADDR),RAM(@CCPADR) Move byte-by-byte      1
93F2 B00A
93F4 930A    3123      DDEC @STADDR          Update source                  1
93F6 9308    3124      DDEC @CCPADR                 and destination pointers 1
93F8 9302    3125      DDEC @MNUM            Also update counter value      1
93FA 53EB    3126    $SEND WHILE             End WHILE loop (Also ends on 0)
93FC A5080A  3127    DSUB @STADDR,@CCPADR    Compute # of bytes of old PAB
93FF 8F0474  3128    $IF @PABPTR .DNE. 0 THEN    Avoid trouble with last PAB  1
9402 1C
9403 8FB004  3129      $WHILE RAM(@PABPTR) .DNE. 0 Ad infinitum (or fundum)   2
9406 7417
9408 A1B004  3130        DADD @CCPADR,RAM(@PABPTR)  Adjust link to next PAB   2
940B 08
940C A1E006  3131        DADD @CCPADR,RAM(BUF(PABPTR)) Update the buffer link 2
940F 0408
9411 BD04B0  3132        DST  RAM(@PABPTR),@PABPTR Get next link in chain     2
9414 04
9415 5403    3133      $SEND WHILE                                            1
9417 A1E006  3134      DADD @CCPADR,RAM(BUF(PABPTR))  Update buffer link      1
941A 0408
             3135    $END IF
             3136    $    Adjust symbol table links
941C 8F3E74  3137    $IF @SYMTAB .DNE. 0 THEN                                 1
941F 9F
9420 D13E06  3138      $IF @SYMTAB .DLT. @CCPPTR THEN Only updata lower links 2
9423 749F
9425 A13E08  3139        DADD @CCPADR,@SYMTAB Get symbol table pointer back   2
9428 BD043E  3140        DST  @SYMTAB,@PABPTR Get pointer for update          2
             3141  DELP$1
942B 8E8084  3142        $IF @RAMTOP .NE. 0 GOTO DELP$2                       2
942E 5437
9430 D1E004  3143        $IF RAM(4(PABPTR)) .DLT. @STLN THEN If imperative   3
9433 043074
9436 3C
9437 A1E004  3144  DELP$2  DADD @CCPADR,RAM(4(PABPTR))  Adjust name pointer   3
943A 0408
             3145        $END IF
943C D2B004  3146        $IF RAM(@PABPTR) .LT. 0 THEN If string-fix bkptrs   3
943F 007486
9442 BE4A07  3147          ST   >07,@FAC          Mask to get # of dims       3
9445 B04AB0  3148          AND  RAM(@PABPTR),@FAC Get # of dims               3
9448 04
9449 BD4C04  3149          DST  @PABPTR,@FAC2     Ptr to 1st dim max          3
944C A34C00  3150          DADD 6,@FAC2             or string pointer         3
944F 06
9450 BF5000  3151          DST  1,@FAC6           Number of pointers to change 3
9453 01
9454 864E    3152          CLR  @FAC4             For 2-byte use of option base3
9456 8E4A74  3153          $WHILE @FAC .NE. 0     While more dimensions       4
9459 6E
945A BE4F01  3154              ST   1,@FAC5       Assume option base 0        4
945D A44F43  3155              SUB  @BASE,@FAC5   But correct if base 1       4
9460 A14EB0  3156              DADD RAM(@FAC2),@FAC4   Get dim maximum         4
9463 4C
```

```
9464 A94E50  3157              DMUL @FAC6,@FAC4   Mpy it in                  4
9467 924A    3158              DEC  @FAC          Next dim                  4
9469 954C    3159              DINCT @FAC2                                  4
946B 059456  3160            $END WHILE                                     3
             3161  *
             3162  *       FAC2 now points at the 1st string pointer
             3163  *       FAC6 contains the # of ptrs that need to be changed
             3164  *
946E 8F5074  3165          $WHILE @FAC6 .DNE. 0  While pointers to change    4
9471 86
9472 BD4AB0  3166             DST   RAM(@FAC2),@FAC   Get ptr to string      4
9475 4C
9476 8F4A74  3167             $IF @FAC .DNE. 0 THEN   If string in non-null  5
9479 80
947A BDEFFF  3168               DST  @FAC2,RAM(-3(FAC))  Fix backpointer     5
947D FD4A4C
             3169             $END IF
9480 954C    3170             DINCT @FAC2          Point to next pointer     4
9482 9350    3171             DDEC  @FAC6          One less ptr to change    4
9484 546E    3172           $SEND WHILE                                       3
             3173         $END IF
9486 8FE002  3174         $IF RAM(2(PABPTR)) .DNE. 0 THEN                     3
9489 04749F
948C D1E002  3175           $IF RAM(2(PABPTR)) .DLT. @CCPPTR THEN             4
948F 040674
9492 9F
9493 A1E002  3176             DADD @CCPADR,RAM(2(PABPTR)) Adjust next value lin4
9496 0408
9498 BD04E0  3177             DST  RAM(2(PABPTR)),@PABPTR    Next entry     4
949B 0204
949D 542B    3178             BR    DELP$1                                  4
             3179           $END IF
             3180         $END IF
             3181       $END IF
             3182     $END IF
949F A14008  3183     DADD @CCPADR,@FREPTR   Update free word pointer
94A2 00      3184     RTN
```

```
                    3186  ************************************************************
                    3187  *       CNVDEF - Convert to 2-byte integer and default to  *
                    3188  *                  1 on negative or 0.......               *
                    3189  ************************************************************
                    3190  CNVDEF
94A3 069347         3191    CALL CHKCNV             Check and convert
94A6 54AC           3192    BR   CNVD$O
94A8 BF4A00         3193    DST  1,@FAC             Default to 1 for minus and 0
94AB 01
                    3194  CNVD$O
94AC 00             3195    RTN                     And return without COND set
```

```
                    3197  ************************************************************
                    3198  *       PARREC parses a possible REC clause in INPUT,      *
                    3199  *       PRINT or RESTORE.  In case a comma is detected     *
                    3200  *       without a REC clause following it, the COND        *
                    3201  *       is set upon return.  In case a REC clause is       *
                    3202  *       specified for a file opened for SEQUENTIAL access, *
                    3203  *       a * FILE ERROR is given.                           *
                    3204  ************************************************************
                    3205  PARREC
94AD D642B3         3206    $IF @CHAT .EQ. COMMA$ THEN Only check if we have a ","   1
94B0 54D9
94B2 0F79           3207       XML  PGMCHR           Check next token for REC        1
94B4 D642DE         3208       $IF @CHAT .NE. REC$ GOTO RTC May be a USING clause    1
94B7 5377
94B9 DAE005         3209       $IF .BITO RAM(FLG(PABPTR)) .NE. 1 GOTO ERRFE          1
94BC 040177
94BF C2
94C0 0F79           3210       XML  PGMCHR           Get first character of expression 1
94C2 06937B         3211       CALL OUTEOF           Output possible pending output  1
94C5 86E003         3212       CLR  RAM(OFS(PABPTR))  Clear record offset            1
94C8 04
94C9 0F74           3213       XML  PARSE            Translate the expression in REC 1
94CB B5             3214       DATA COLON$                                           1
94CC 069347         3215       CALL CHKCNV           Check numeric and convert to    1
94CF D24A00         3216       $IF @FAC .LT. 0 GOTO ERRBV  2 byte integer. Bad va    1
94D2 57B6
94D4 BDE00A         3217       DST  @FAC,RAM(RNM(PABPTR)) Store actual record number 1
94D7 044A
                    3218    $END IF
94D9 00             3219    RTN
```

```
                    3221  ********************************************************
                    3222  *                                                      *
                    3223  *      D I S P L A Y / A C C E P T   U T I L I T I E S *
                    3224  *                                                      *
                    3225  ********************************************************
                    3226  DISACC
94DA 06972A         3227    CALL INITKB              PABPTR is used as flag (no DSR I/O)
                    3228  DISP$1
94DD D642EF         3229    $IF @CHAT .EQ. ERASE$ THEN check for ERASE ALL          1
94E0 5503
94E2 DA0401         3230      $IF .BITO @PABPTR .EQ. 1 GOTO ERRSYN  already used once1
94E5 4109
94E7 0F79           3231      XML  PGMCHR            check next token for ALL       1
94E9 0F7E           3232      XML  SPEED                                            1
94EB 00             3233      DATA SYNCHK            has to be ALL                  1
94EC EC             3234      DATA ALL$                                             1
94ED 0780           3235      ALL  BKGD+OFFSET        clear screen to background color 1
94EF BE7F03         3236      ST   3,XPT            Reset pending output pointer   1
94F2 DA0404         3237      $IF .BIT2 @PABPTR .EQ. 0 THEN Didn't use AT yet       2
94F5 54FE
94F7 BE0601         3238        ST   1,@CCPPTR       Reset column pointer           2
94FA BF0802         3239        DST  SCRNBS+2,@CCPADR      and screen base address  2
94FD E2
                    3240      $END IF
94FE B60401         3241      SB   @PABPTR,0         Set "ERASE USED" flag          1
9501 54DD           3242      BR   DISP$1            Try next token                 1
                    3243    $END IF
9503 D642EE         3244    $IF @CHAT .EQ. BEEP$ THEN delay action for BEEP         1
9506 5514
9508 DA0402         3245      $IF .BIT1 @PABPTR .EQ. 1 GOTO ERRSYN  use it only once 1
950B 4109
950D B60402         3246      SB  @PABPTR,1          No syntax error detected here  1
9510 0F79           3247      XML PGMCHR             Evaluate next token            1
9512 54DD           3248      BR  DISP$1             Get set for second pass        1
                    3249    $END IF
9514 D642F0         3250    $IF @CHAT .EQ. AT$ THEN Generate "AT" clause            1
9517 555C
9519 DA0404         3251      $IF .BIT2 @PABPTR .EQ. 1 GOTO ERRSYN  Second usage not 1
951C 4109
951E 0F79           3252      XML  PGMCHR                          allowed...     1
9520 0F7E           3253      XML  SPEED                                          1
9522 00             3254      DATA SYNCHK            Skip left parenthesis          1
9523 B7             3255      DATA LPAR$                                          1
9524 0F74           3256      XML  PARSE             Now parse any expression       1
9526 B3             3257      DATA COMMA$                                         1
9527 0F7E           3258      XML  SPEED                                          1
9529 00             3259      DATA SYNCHK            Check for "," and skip it      1
952A B3             3260      DATA COMMA$                                         1
952B 0694A3         3261      CALL CNVDEF            Convert to 2 byte numeric      1
952E BE4C18         3262      ST   24,@FAC2          Convert modulo 24 (# screenlines) 1
9531 069605         3263      CALL COMMOD            Compute remainder              1
9534 924B           3264      DEC  @FAC1             Convert back to 0 (range was 1-24)1
9536 AA4B20         3265      MUL  32,@FAC1          Convert to line base address   1
9539 BD084B         3266      DST  @FAC1,@CCPADR     And replace CCPADR             1
953C 0F74           3267      XML  PARSE             Parse column expression        1
```

```
953E B6       3268      DATA RPAR$                                          1
953F 0F7E     3269      XML  SPEED                                         1
9541 00       3270      DATA SYNCHK          Check for ")"at end           1
9542 B6       3271      DATA RPAR$                                         1
9543 0694A3   3272      CALL CNVDEF          Again convert to two byte int. 1
9546 BE4C1C   3273      ST   VWIDTH,@FAC2    Convert modulo video width    1
9549 069605   3274      CALL COMMOD          Compute remainder             1
954C BC064B   3275      ST    @FAC1,@CCPPTR  Select current column         1
954F A1084A   3276      DADD @FAC,@CCPADR    Compute full address          1
9552 9108     3277      DINC @CCPADR         Adjust for column 0 (offset-1) 1
9554 B60404   3278      SB  @PABPTR,2        Set "AT-CLAUSE" used flag     1
9557 B60420   3279      SB  @PABPTR,5        Set "NON-STANDARD SCREEN ADDRESS" 1
955A 54DD     3280      BR  DISP$1           Continue for next item        1
              3281    $END IF
955C D642EB   3282    $IF @CHAT .EQ. SIZE$ THEN "SIZE" clause               1
955F 558B
9561 DA0408   3283      $IF .BIT3 @PABPTR .EQ. 1 GOTO ERRSYN   Only use once 1
9564 4109
9566 0F79     3284      XML  PGMCHR          Get chr following the SIZE    1
9568 D642B7   3285      $IF @CHAT .NE. LPAR$ GOTO ERRSYN  has to open ("(") 1
956B 4109
956D 0F74     3286      XML  PARSE           And close again (")")         1
956F FE       3287      DATA VALID$                                        1
9570 D24A00   3288      $IF @FAC .LT. 0 THEN Change to positive argument   2
9573 757A
9575 834A     3289        DNEG @FAC          For ACCEPT stmt with - size   2
9577 B60480   3290        SB   @PABPTR,7     indicate in highest bit       2
              3291      $END IF
957A 069347   3292      CALL CHKCNV                                        1
957D 77B6     3293      BS    ERRBV          "* BAD VALUE"                 1
957F 8E4A57   3294      $IF @FAC .NE. 0 GOTO  ERRBV  also for args > 255   1
9582 B6
9583 BC054B   3295      ST   @FAC1,@PABPTR+1     copy to PABPTR (always unused)1
9586 B60408   3296      SB   @PABPTR,3       prevent further use           1
9589 54DD     3297      BR   DISP$1          and go on                     1
              3298    $END IF
958B D642FE   3299    $IF @CHAT .NE. VALID$ THEN exclude VALIDATE option    1
958E 75A4
              3300  *
              3301  *       Start evaluating ERASE clause here
              3302  *
9590 DA0408   3303      $IF .BIT3 @PABPTR .EQ. 1 THEN                      2
9593 7598
9595 0695CA   3304        CALL SIZE1         evaluate field defined in SIZE 2
              3305      $END IF
              3306  *
              3307  *    if it's no DISPLAY keyword ( AT, SIZE, BEEP or USING)
              3308  *    it has to be a print separator or colon ":".
              3309  *    if anything is sprcified it has to be a colon or
              3310  *    end of line...for end-of-line output current record
              3311  *
              3312  *
              3313  *    check for end of statement
              3314  *
              3315  CHKEND
```

```
9598 DA4280  3316       $IF .BIT7 @CHAT .EQ. 1 THEN                          2
9598 75A2
959D CA4284  3317         $IF @CHAT  .L. TREM$+1 GOTO RTC                    2
95A0 5377
             3318       $END IF
95A2 8E42    3319       CZ  @CHAT              set COND according to CHAT    1
             3320     $END IF
95A4 01      3321     RTNC
             3322
             3323 ***************************************************************
             3324 *       NXTCHR - Get next program character - skip            *
             3325 *                all strings, numerics and line refs...       *
             3326 ***************************************************************
             3327 NXTCHR
95A5 069598  3328     CALL CHKEND            Check for end of stmts
95A8 7377    3329     BS   RTC               Avoid end of stmt
95AA D642C7  3330     $IF @CHAT .EQ. STRIN$ GOTO NXTC$O Skip all strings
95AD 75B4
95AF D642C8  3331     $IF @CHAT .EQ. NUM$ THEN   and numerics/unquoted strings 1
95B2 55C0
             3332 NXTC$O
95B4 0F79    3333       XML  PGMCHR          Get string length                1
95B6 BC4B42  3334       ST   @CHAT,@FAC1     Make that a double please....    1
95B9 864A    3335       CLR  @FAC            Hic.....Oops, sorry              1
95BB A12C4A  3336       DADD @FAC,@PGMPTR    Back to the serious stuff        1
95BE 55C7    3337     $SELSE                                                   1
95C0 D642C9  3338       $IF @CHAT .EQ. LN$ THEN Line # = skip 2 tokens         2
95C3 55C7
95C5 952C    3339         DINCT @PGMPTR      <----- That's the skip           2
             3340       $END IF
             3341     $SEND IF
95C7 0F79    3342     XML  PGMCHR            Get the next token
95C9 00      3343     RTN
```

```
                    3345  ********************************************************
                    3346  *     P R I N T / D I S P L A Y   U T I L I T I E S    *
                    3347  *                                                      *
                    3348  *     Use the parameters specified in SIZE for  further *
                    3349  *     evaluation of the limited field length           *
                    3350  ********************************************************
                    3351  SIZE1
95CA DA0404         3352    $IF .BIT2 @PABPTR .EQ. 0 THEN no "AT" clause used
95CD 55E7
95CF D60601         3353      $IF @CCPPTR .NE. 1 THEN might have to print current lin
95D2 75E7
95D4 BC4A06         3354        ST   @CCPPTR,@FAC  compute final position after size
95D7 A04A05         3355        ADD  @PABPTR+1,@FAC in FAC and compare with record
95DA 924A           3356        DEC  @FAC
95DC C44A07         3357        $IF @FAC .H. @RECLEN THEN size clause too long
95DF 55E7
                    3358  *
                    3359  *     We can't get here for AT( , ) output, since
                    3360  *     right margin is limited there
                    3361  *
95E1 069690         3362          CALL OUTREC       advance to next line
95E4 0696B7         3363          CALL SCRO         sccroll the screeen
                    3364        $END IF
                    3365      $END IF
                    3366    $END IF
95E7 A40706         3367    SUB  @CCPPTR,@RECLEN    limit field size to available
95EA 9007           3368    INC  @RECLEN             space...including current positior
95EC C40705         3369    $IF @RECLEN .NOT. .H. @PABPTR+1 GOTO INIT$1
95EF 56B3
95F1 BC0705         3370    ST   @PABPTR+1,@RECLEN only accept if available
95F4 56B3           3371    BR   INIT$1             reinitialize CCPPTR
                    3372  *
                    3373  *     Copy (converted) numerical datum in string
                    3374  *
                    3375  RSTRING
95F6 BC0D56         3376    ST   @FAC12,@BYTE+1     Get actual string length
95F9 860C           3377    CLR  @BYTE              Create double byte value
95FB 0692B8         3378    CALL CTSTR              Create a temporary string
95FE 340CB0         3379    MOVE @BYTE FROM *FAC11 TO RAM(@SREF) Copy value string
9601 1C9055
9604 00             3380    RTN
                    3381    $
                    3382    $     COMMOD - Compute FAC module FAC2
                    3383    $
                    3384  COMMOD
9605 AC4A4C         3385    DIV  @FAC2,@FAC         Compute remainder
9608 8E4B56         3386    $IF @FAC1 .EQ. 0 THEN Avoid zero remainders
960B 0F
960C BC4B4C         3387      ST   @FAC2,@FAC1      Assume maximum remainder
                    3388    $END IF
960F 864A           3389    CLR  @FAC               Clear upper byte
9611 00             3390    RTN
                    3391    $
                    3392    $     TSTSEP tests for separator in print and
                    3393    $     branches to the correct evaluation routine.
```

```
                    3394    $     if no separator is found, simple return.
                    3395    $
                    3396  TSTSEP
                    3397    $     Test case end of line
9612 069598         3398      CALL CHKEND
9615 561C           3399      BR   TSTS$0
9617 BF9073         3400      DST  #EOLEX,*SUBSTK  Replace returnaddress with EOLEX
961A 8395
                    3401  TSTS$0
961C CA42B3         3402    $IF @CHAT .L. COMMA$ GOTO TSTS$1
961F 5644
9621 C642B5         3403    $IF @CHAT .H. COLON$ GOTO TSTS$1
9624 7644
9626 BF9073         3404    DST  #PRSEM,*SUBSTK     Expect it to be a ";"
9629 837A
962B 0683C5         3405    CALL TSTINT             Test for INTERNAL files
962E 5644           3406    BR   TSTS$1             Treat all separators as ";"
9630 D642B3         3407    $IF @CHAT .EQ. COMMA$ THEN                                1
9633 563A
9635 BF9073         3408      DST  #PRTCOM,*SUBSTK                                    1
9638 835C
                    3409    $END IF
963A D642B5         3410    $IF @CHAT .EQ. COLON$ THEN                                1
963D 5644
963F BF9073         3411      DST  #PRCOL,*SUBSTK                                     1
9642 8377
                    3412    $END IF
                    3413  TSTS$1
9644 00             3414    RTN
                    3415  *
                    3416  *       PARFN - Parse string expression and create PAB
                    3417  *       automatically continue in CSTRIN for copy
                    3418  *       string to PAB
                    3419  *            Exit on non-string values
                    3420  *
                    3421  PARFN
                    3422    $
                    3423    $     First evaluate string expression
                    3424    $
9645 0F74           3425    XML  PARSE              Parse up to next comma
9647 B3             3426    DATA COMMA$
9648 D64C65         3427    $IF @FAC2 .NE. STRVAL GOTO ERRSNM Check for "STRING"
964B 57AA
964D BD0250         3428    DST  @FAC6,@MNUM        Copy length byte in MNUM
9650 A2030E         3429    ADD  PABLEN,@MNUM+1     Account for PAB length + control
9653 0F77           3430    XML  VPUSH              Save start of string somewhere
9655 BD4A02         3431    DST  @MNUM,@FAC         Setup for MEMCHK - check for memory
9658 0F72           3432    XML  MEMCHK                                  overflow
965A 77B2           3433    BS   ERRMEM            * MEMORY FULL
965C 0F78           3434    XML  VPOP               Restore all FAC information again
965E A54002         3435    DSUB @MNUM,@FREPTR      Update free word pointer
9661 BD0440         3436    DST  @FREPTR,@PABPTR    Assign PAB entry address
9664 9104           3437    DINC @PABPTR            Correct for byte within PAB
9666 86B004         3438    CLR  RAM(@PABPTR)       Clear PAB plus control info
9669 35000D         3439    MOVE PABLEN-1 FROM RAM(@PABPTR) TO RAM(1(PABPTR)) Ripp
```

```
966C E00104
966F B004
9671 BCE003  3440     ST   @MNUM+1,RAM(OFS(PABPTR)) Save length of PAB
9674 0403
9676 BC0251  3441     ST   @FAC7,@MNUM          Compute # of bytes in name
9679 BCE00D  3442     ST   @FAC7,RAM(NLEN(PABPTR)) Store name length
967C 0451
967E BCE002  3443     ST   @FNUM,RAM(FIL(PABPTR)) Copy file number in PAB
9681 0417
9683 BD0804  3444     DST  @PABPTR,@CCPADR   Get start address for string dest.
9686 A30800  3445     DADD NLEN+1,@CCPADR    Add offset to actual start address
9689 0E
             3446     $
             3447     $    TRICKY - OPTFLG also resets offset added in CSTRIN
             3448     $
968A 8617    3449     CLR  @OPTFLG           Clear all option flags
968C 0F84    3450     XML  IO                CSTRIN I/O utility
968E 02      3451     DATA CSTRIN
968F 00      3452     RTN
             3453
             3454  *>>>>>REMOVED 2/7/80 BECAUSE OF XML. SRH
             3455  *CSTRIN
             3456  * $WHILE @MNUM .NE. 0 THEN Watch out for empty names
             3457  *    ST   @DSRFLG,RAM(@CCPADR) Store offset
             3458  *    ADD  RAM(@FAC4),RAM(@CCPADR) Add in character
             3459  *    DINC @FAC4           Get next address for input
             3460  *    DINC @CCPADR           and for output
             3461  *    DEC  @MNUM           Decrement counter
             3462  * $SEND WHILE
             3463  * RTN
```

```
                    3465  ********************************************************
                    3466  *                      O U T R E C                     *
                    3467  *      OUTREC and INITRC are used to output a record to *
                    3468  *      either screen or external I/O devices, and to    *
                    3469  *      initiate pointers for further I/O                *
                    3470  ********************************************************
                    3471  OUTREC
9690 BC0307         3472    ST  @RECLEN,@MNUM+1          Compute number of character
9693 9003           3473    INC @MNUM+1                     positions we should fill
9695 8E1776         3474    $IF @DSRFLG .NE. 0 THEN      Screen I/O                  1
9698 BE
9699 0F84           3475      XML   IO                 Fill the remainder of the record  1
969B 01             3476      DATA  FILSPC               with appropriate fillers        1
969C DA0408         3477      $IF .BIT3 @PABPTR .EQ. 1 GOTO RTC block output on size 1
969F 5377
96A1 DA0404         3478      $IF .BIT2 @PABPTR .EQ. 1 THEN "AT CLAUSE USED"          2
96A4 76B7
                    3479  *
                    3480  *         Next test for xing the end of screen
                    3481  *
96A6 A30800         3482        DADD 4, @CCPADR                                       2
96A9 04
96AA CA0803         3483        $IF @CCPADR .L. 3 GOTO INIT$1                         2
96AD 56B3
96AF BF0800         3484        DST 2,@CCPADR        restart at upper left hand corner2
96B2 02
                    3485  INIT$1
96B3 BE0601         3486        ST 1,@CCPPTR         reset current column pointer     2
96B6 00             3487        RTN                                                   2
                    3488      $END IF
                    3489  SCRO
96B7 0F83           3490      XML  SCROLL              scroll the screen one line        1
96B9 BE0601         3491      ST   1,@CCPPTR           reinitialize CCPPTR               1
96BC 573F           3492      BR  INTKBO               and reinitialize                  1
                    3493    $END IF                    provide external I/O stuff
                    3494    $  This is also entry for last record output
96BE DAE005         3495    $IF .BIT4 RAM(FLG(PABPTR)) .EQ. 0 THEN FIXED records     1
96C1 041056
96C4 CD
96C5 BC0307         3496      ST   @RECLEN,@MNUM+1 Ready for space filling            1
96C8 9003           3497      INC  @MNUM+1         Move to first position outside rec1
96CA 0F84           3498      XML  IO              And do it up to end of record      1
96CC 01             3499      DATA FILSPC                                             1
                    3500    $END IF
96CD 9206           3501    DEC  @CCPPTR           Update last character position
96CF BCE009         3502    ST   @CCPPTR,RAM(CNT(PABPTR)) Store # of characters
96D2 0406
96D4 86E003         3503    CLR  RAM(OFS(PABPTR))  Undo pending record offsets
96D7 04
96D8 069749         3504    CALL IOCALL            Call DSR
96DB 03             3505    DATA C$WRIT                  for WRITE mode
                    3506    $
96DC 8609           3507    CLR  @CCPADR+1         Get address at bufferstart
96DE 56EC           3508    BR   PR$$O
                    3509  *
```

```
                3510  *       PRINIT initializes the variables CCPADR, CCPPTR
                3511  *       RECLEN and DSRFLG, for a given PABPTR.
                3512  *
                3513  PRINIT
96E0 8617       3514    CLR   @DSRFLG              Indicate external I/O in DSRFLG
96E2 BC07E0     3515    ST    RAM(LEN(PABPTR)),@RECLEN  Pick up record length
96E5 0804
96E7 BC09E0     3516    ST    RAM(OFS(PABPTR)),@CCPADR+1 Get offset in record
96EA 0304
                3517  PR$$O
96EC BC0609     3518    ST    @CCPADR+1,@CCPPTR Compute columnar position
96EF 9006       3519    INC   @CCPPTR             And convert from offset
96F1 8608       3520    CLR   @CCPADR             Clear upper byte
96F3 A108E0     3521    DADD  RAM(BUF(PABPTR)),@CCPADR Compute actual address
96F6 0604
96F8 00         3522    RTN
```

```
3524  *>>>>>>Removed 2/7/80 because XML to do it. SRH
3525  *
3526  ************************************************************
3527  *      FILSPC - Fill a record with spaces, starting from   *
3528  *      the current position at CCPADR to the position      *
3529  *      indicated in @MNUM+1 (1 byte)                       *
3530  ************************************************************
3531  *FILSPC
3532  *  $IF @MNUM+1 .EQ. 0 THEN     In case record length = 255
3533  *                              (255+1, overflow to be 0)
3534  *    $IF @CCPPTR .NE. 0 THEN
3535  *      B FILS$2                Skip the following "compare "
3536  *    $SELSE
3537  *      B FILS$3                Just RTN
3538  *    $END IF
3539  *  $END IF
3540  *  $IF @MNUM+1 .H. @CCPPTR THEN Make sure difference >=1
3541  *FILS$2
3542  *    SUB  @CCPPTR,@MNUM+1 Compute the # of bytes to move
3543  *    ADD  @MNUM+1,@CCPPTR Actually move pointer ahead
3544  *    ST   @DSRFLG,@MNUM   Assume zero filling
3545  *    CALL TSTINT          Which would be correct for
3546  *    BR   FILS$1            INTERNAL type files
3547  *    ADD  SPACE,@MNUM     Make that space filling
3548  *FILS$1
3549  *    ST   @MNUM,RAM(@CCPADR) Fill with fillers
3550  *    DINC @CCPADR         Next address to be treated
3551  *    DEC  @MNUM+1         Count that space too
3552  *    BR   FILS$1          One space is enough !!
3553  * $END IF
3554  *FILS$3
3555  * RTN                     Return after completing move
```

```
                    3557  ********************************************************
                    3558  *       OSTRNG - Copy the value of the string expression *
                    3559  *       to the screen                                    *
                    3560  ********************************************************
                    3561  OSTRNG
96F9 BC0C51         3562    ST    @FAC7,@BYTE         Pick up the string length
96FC 8E0C77         3563    $WHILE @BYTE .NE. 0       Output as many records as required
96FF 29
                    3564      $
                    3565      $  CHKREC check available space in current record.
                    3566      $  If the string to be output is too long, it
                    3567      $  is chunked up into digestable pieces.  If the
                    3568      $  current record is partly filled up, it is
                    3569      $  output before any chunking is done.
                    3570      $
                    3571  CHKREC
9700 BC0306         3572      ST    @CCPPTR,@MNUM+1 Use MNUM for current offset ind.  1
                    3573  CHKR$0
9703 BC0207         3574      ST    @RECLEN,@MNUM   Compute remaining area           1
9706 A40206         3575      SUB   @CCPPTR,@MNUM     between column and end      1
9709 9002           3576      INC   @MNUM           Also count current column       1
970B C8020C         3577      $IF @MNUM .L. @BYTE THEN Won't fit in current record  2
970E 771B
9710 D60301         3578        $IF @MNUM+1 .EQ. 1 GOTO CHKR$1 Unused record      2
9713 771E
9715 069690         3579        CALL OUTREC       Output whatever we have         2
9718 5700           3580        BR    CHKREC      And try again                   2
971A 00             3581        RTN                                               2
                    3582      $END IF
971B BC020C         3583      ST    @BYTE,@MNUM     Use actual count if fit         1
                    3584  CHKR$1
971E A40C02         3585      SUB   @MNUM,@BYTE     Update remaining chars count    1
9721 A00602         3586      ADD   @MNUM,@CCPPTR   Also new column pointer         1
9724 0F84           3587      XML   IO              Copy string to output           1
9726 02             3588      DATA CSTRIN                                           1
9727 56FC           3589    $END WHILE              Continue as long as needed
9729 00             3590    RTN
                    3591  ********************************************************
                    3592  *       INITKB - Initialize the variables needed       *
                    3593  *       for keyboard output                            *
                    3594  ********************************************************
                    3595  INITKB
972A 8604           3596    CLR   @PABPTR           Don't use any DISPLAY options
972C BE1760         3597    ST    OFFSET,@DSRFLG    Load for correction of screen chrs
972F BE0601         3598    ST    1,@CCPPTR         Assume un-initialized XPT
9732 C67F02         3599    $IF XPT .H. 2 THEN    *** Patch for un-initialized XPT  1
9735 573C
9737 BC067F         3600      ST    XPT,@CCPPTR     Initialize CCPPTR                 1
973A 9606           3601      DECT @CCPPTR          Correct for incorrect XPT offset  1
                    3602    $END IF
973C BE071C         3603    ST    VWIDTH,@RECLEN    Get video screen width
                    3604  INTKB0
973F BC0906         3605    ST    @CCPPTR,@CCPADR+1 Initialize screen address
9742 8608           3606    CLR   @CCPADR           Clear upper byte CCPADR
9744 A30802         3607    DADD SCRNBS+1,@CCPADR   Add start-address plus compensate
```

```
9747 E1
9748 00          3608    RTN
                 3609  *
                 3610  IOCALL
9749 8856        3611    FETCH @FAC12           I/O code to FAC12 (BUG!!!!)
974B BCE004      3612    ST    @FAC12,RAM(COD(PABPTR)) Pick up the I/O code
974E 0456
                 3613  IOCL$1
9750 069756      3614    CALL CDSR              Call the DSR routine
9753 5782        3615    BR   ERR$2             Give I/O error on error
9755 00          3616    RTN                    Or else return
                 3617  *
                 3618  *       DSR CALL ROUTINE - NORMAL ENTRY
                 3619  *
                 3620  CDSR
9756 BEE00C      3621    ST   OFFSET,RAM(SCR(PABPTR)) Always set screen offset
9759 0460
975B 35001E      3622    MOVE 30 FROM @FAC TO RAM(>3C0) Save FAC area
975E A3C04A
9761 BD5604      3623    DST  @PABPTR,@FAC12    Get PAB pointer in FAC
9764 A35600      3624    DADD NLEN,@FAC12       Compute name length entry
9767 0D
9768 B2E005      3625    AND >1F,RAM(FLG(PABPTR)) Clear error bits for ON ERROR
976B 041F
                 3626  *                              time, I/O process can still be continued
976D 060010      3627    CALL CALDSR            Call actual DSR link routine
9770 08          3628    DATA 8
9771 35001E      3629    MOVE 30 FROM RAM(>3C0) TO @FAC
9774 4AA3C0
                 3630  *            MOVE does not affect status
9777 777E        3631    BS   CDSR$0            ERROR = ERROR = ERROR = .....
9779 DAE005      3632    CLOG >E0,RAM(FLG(PABPTR)) Set COND if no error
977C 04E0
                 3633  CDSR$0
977E 01          3634    RTNC
```

```
                3636  *
                3637  *                E R R O R   M E S S A G E S
                3638  *
                3639  ERR$2B
977F 0680FB     3640    CALL CLRFRE             Undo allocation of PAB
                3641  ERR$2
                3642  * First check is it error coming from AUTOLD
                3643  * If it is  then do not print the error message
                3644  * and go back to TOPL02
9782 310002     3645    MOVE 2 FROM ROM(#TOPL02) TO RAM(AUTTMP)
9785 A39460
9788 30
9789 D5808A     3646    $IF @RSTK+2 .DEQ. RAM(AUTTMP) THEN                      1
978C A39457
978F 94
9790 BE738A     3647      ST RSTK+2,@SUBSTK                                     1
9793 00         3648      RTN                                                   1
                3649    $END IF
                3650  ************************************************************
                3651  *       Next code is to avoid recursion of errors in       *
                3652  *       CLSALL routine.  If this entry is taken from       *
                3653  *       CLSALL, the stack will contain CLSLBL as a         *
                3654  *       returnaddress in the third level.                  *
                3655  ************************************************************
9794 A67304     3656    SUB  4,@SUBSTK          Back down two levels
9797 D79073     3657    $IF *SUBSTK .DEQ. CLSLBL THEN                           1
979A 81E957
979D A3
                3658  WRNIO
979E 066A82     3659      CALL WARN$$           Give warning to the user        1
97A1 23         3660      DATA 35               * I/O ERROR  but warning        1
97A2 00         3661      RTN                   And return to close routine     1
                3662    $END IF
97A3 A27304     3663    ADD  4,@SUBSTK          Back up two levels for OLD/SAVE
                3664  ERRIO
97A6 066A84     3665    CALL ERR$$
97A9 24         3666    DATA 36                 * I/O ERROR
                3667  *
                3668  *
                3669  *       ERROR messages called in this file
                3670  *
97AA 066A84     3671  ERRSNM CALL ERR$$             * STRING-NUMBER MISMATCH
97AD 07         3672         DATA 7
                3673  *
97AE 066A84     3674  ERRIM  CALL ERR$$             * IMAGE ERROR
97B1 0A         3675         DATA 10
                3676  *
97B2 066A84     3677  ERRMEM CALL ERR$$             * MEMORY FULL
97B5 0B         3678         DATA 11
                3679  *
97B6 066A84     3680  ERRBV  CALL ERR$$             * BAD VALUE
97B9 1E         3681         DATA 30
                3682  *
97BA 066A84     3683  ERRINP  CALL ERR$$
97BD 20         3684          DATA 32             * INPUT ERROR
```

```
                3685  *
97BE 066A84     3686  ERRDAT  CALL ERR$$
97C1 21         3687          DATA 33                 * DATA ERROR"
                3688  *
97C2 066A84     3689  ERRFE   CALL ERR$$              * FILE ERROR
97C5 22         3690          DATA 34
                3691  *
97C6 066A84     3692  ERRPV   CALL ERR$$              * PROTECTION VIOLATION
97C9 27         3693          DATA 39
97CA 066A84     3694  ERRMUV  CALL ERR$$              * IMPROPERLY USED NAME
97CD 09         3695          DATA 9
                3696  *
                3697  **
                3698  *       Other errors called in this file
                3699  **
                3700  * ERRSYN     * SYNTAX ERROR    DATA 3
                3701  * ERRST      * STRING TRUNCATED ERROR     DATA 19
                3702  * WRNNPP     * NO PROGRAM PRESENT         DATA 29
                3703  * WRNINP     * INPUT ERROR       WARNING  DATA 32
                3704  * ERRIO      * I/O ERROR                  DATA 36
                3705  * WRNIO      * I/O ERROR         WARNING  DATA 36
                3706  * WRNSNM     * STRING NO. MISMATCH WARNING DATA 7
                3707          END


ERRORS=     0

LENGTH= 6094 (>17CE)


516 SYMBOLS USED
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
A$1	8A80	1881	1879
AAA	004C	169	2100 2125
ABS$	00CB	271	
ACCEP$	00A4	231	
ACCEPT	8964	1749	385
ACCNM	85DB	1308	1213 1219
ACCP$1	8A40	1846	1829
ACCP$2	8A89	1889	1892
ACCP$3	8AC5	1914	1918
ACCP$4	8B30	1961	1958
ACCP$5	8A76	1875	1944
ACCP$6	8B24	1952	1928
ACCP$7	8A6E	1869	1864
ACCP$8	8AF6	1932	1902
ACCP$9	8A48	1852	1858 1949
ACCTRY	03B7	347	1750 1864 1941
ACCVRA	03AE	341	1871 1940
ACCVRW	03AC	340	1870 1939
ALL$	00EC	296	3234
APPEN$	00F9	309	
ARG	005C	175	176 177 178 179 726 730 920 921 924 925 926 927 1123 1126 1127 1372 1374 1375 1376 1377 1378 1767 1770 1771 1797 1798 1799 1802 1803 2230 2786 2788 2791 2805 2812 2813 2814
ARG1	005D	176	1382 1383
ARG2	005E	177	1124 1128 1812 1813 1815 1861 2809 2810 2813 2814 2816 2818
ARG6	0062	178	2835 2837
ARG7	0063	179	1753 1808
ASSGNV	007C	40	1390 1570 1673 1953 1997
AT$	00F0	300	3250
ATN$	00CC	272	
AUTO1	602E	64	2613
AUTTMP	0394	337	3645 3646
BASE	0043	149	3155
BASE$	00F1	301	
BBB	0050	171	2097 2098 2163 2165 2178
BBB1	000C	90	2378 2457
BEEP$	00EE	298	3244
BKGD	0020	333	3235
BREAK	0002	320	2704
BREAK$	008E	209	
BUF	0006	96	668 672 1353 1428 1457 1581 1638 1650 2067 2126 2373 2399 2685 3089 3131 3134 3521
BUFLEV	0046	152	
BYTE	000C	109	1099 1140 1141 1161 1162 1289 1290 1359 1361 1366 1369 1370 1388 1389 1567 1637 1639 1642 1647 1648 1649 1654 1664 1795 1905 1909 2929 2941 3376 3377 3379 3562 3563 3577 3583 3585
C$CLOS	0001	356	711 748 2191 2451 2536 2593 2718
C$DELE	0007	362	692 717
C$LOAD	0005	360	2068
C$OPEN	0000	355	
C$READ	0002	357	855 1409 1685 2131 2171 2182 2571 2587
C$REST	0004	359	778

SYMBOL	VALUE	DEF	REFERENCE TABLE
C$SAVE	0006	361	2382
C$SCR	0008	363	
C$STAT	0009	364	2809
C$WRIT	0003	358	858 2420 2438 2448 2528 2534 3071 3505
CALDSR	0010	56	3627
CALL$	009D	224	
CB	0008	1	
CCC	004E	170	2096 2176 2187
CCC1	0008	89	2155 2179 2183 2186 2187 2379 2430 2431 2432 2440 2442 2444 2446 2458
CCPADR	0008	105	929 930 947 948 949 1023 1585 1588 1712 1713 1721 1723 1724 1725 1810 1812 1840 1847 1854 1856 1859 1860 1947 2671 2675 2677 2684 2685 3091 3092 3093 3094 3102 3109 3120 3122 3124 3127 3130 3131 3134 3139 3144 3176 3183 3239 3266 3276 3277 3444 3445 3482 3483 3484 3507 3516 3518 3520 3521 3605 3606 3607
CCPPTR	0006	103	925 931 946 950 970 974 989 1023 1024 1041 1042 1049 1466 1495 1497 1517 2232 2245 2692 3120 3138 3175 3238 3275 3353 3354 3367 3486 3491 3501 3502 3518 3519 3572 3575 3586 3598 3600 3601 3605
CDSR	9756	3620	643 721 749 2069 2127 2401 3614
CDSR$0	977E	3633	3631
CFI	0012	57	3032
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 1159 1192 1332 1392 1441 1618 1700 1709 1757 1759 1765 1766 1768 1770 1789 2009 2020 2031 2035 2299 2301 2556 2575 2576 2580 2663 2672 2675 2830 2854 2855 2864 2898 2904 3005 3007 3206 3208 3229 3244 3250 3282 3285 3299 3316 3317 3319 3330 3331 3334 3338 3402 3403 3407 3410
CHECK	810D	623	442 505
CHKCNV	9347	3029	515 2833 3191 3215 3292
CHKCON	935C	3038	426 708 772 846 1678 2836
CHKEND	9598	3315	388 504 564 624 685 719 785 863 868 1017 1118 1296 1301 1394 1556 1572 2902 2912 2917 3006 3328 3398
CHKF$1	9366	3050	3057
CHKFN	9340	3018	424 706 770 844 1333 1619
CHKNUM	92D7	2947	1498 1563
CHKPAR	809F	512	506 566
CHKR$0	9703	3573	1164 1293
CHKR$1	971E	3584	3578
CHKREC	9700	3571	3580
CHKRM	882C	1584	1711 1720
CHKS$0	931A	2986	1984
CHKSEP	8328	953	972 974 978
CHKSTR	930E	2981	1098 1521 1566 1993
CHR$$	00D6	282	
CIF	0080	45	2792
CIRCU$	00C5	264	
CLOS$1	81C7	725	722
CLOSE	8196	705	375

SYMBOL	VALUE	DEF	REFERENCE TABLE
CLOSE$	00A0	227	
CLRFRE	80FB	608	616 635 3640
CLSA$O	81E1	744	753
CLSALL	81F4	751	379 2664 2857
CLSLBL	81E9	747	3657
CNS	0073	32	939 1255
CNT	0009	98	1360 1396 1399 1418 1426 1637 1648 2133 2175 2186 2418 2433 2446 2526 2532 2575 2591 3502
CNVD$O	94AC	3194	3192
CNVDEF	94A3	3190	967 3261 3272
COD	0004	94	711 717 748 855 858 1409 1685 2068 2810 2813 3071 3612
COLON$	00B5	248	433 712 834 881 907 1114 1117 1689 1700 1704 1709 1716 1820 3023 3214 3403 3410
COMMA$	00B3	246	442 503 514 563 871 1137 1159 1192 1392 1413 1441 1525 1789 1999 2009 2301 2916 2997 3206 3257 3260 3402 3407 3426
COMMOD	9605	3384	969 3263 3274
CONPRT	82C9	889	1018
CONT	0075	34	673 693 724 779 812 1043 1055 1403 1576 1674 1962 2010 2793 2821 2824
CONVER	A012	79	1990 2953
COS$	00CD	273	
CPTEMP	03BC	351	1466 1517
CPUBAS	A040	22	2097 2156 2163
CRNBUF	0820	327	1407 1491 1550 1734 2498 2501 2508 2517 2522 2524 2531 2539 2572 2578 2582 2582 2588 2735 2736 2736 2747 2752
CRUNCH	007F	44	1431 1475
CSNTMP	0390	335	1991 2957
CSNTP1	03BA	336	1503
CSTRIN	0002	50	3451 3588
CTMPST	92C9	2938	1367 1568 1672
CTSTR	92B8	2926	1101 2939 3378
CTSTRO	92BC	2928	1913
CURINC	000E	111	2611 2625 2630 2637 2643 2644 2650 2656 2659 2713
CURLIN	0014	114	1144 1148 1149 1154 1155 1156 1170 1173 1176 1177 1178 1179 1180 1182 1183 1184 1194 1198 1202 1203 1204 1205 1208 1264 1266 1270 1274 1275 1277 1281 1283 1289 1406 1462 1477 1490 1505 1509 1575 1733 2610 2618 2624 2643 2644 2646 2649 2660
DATA	0034	141	1089 1092 1100 1490 1491 1505 1509 1550 1575 1979 2002 2019 2034 2879 2950 2952 2957 2968 2972 2976 2990 2991 3001
DATA$	0093	214	2017
DATAST	A008	78	811 2006
DATEND	9336	3004	2998
DDD1	0054	172	1107 2426 2439 2459 2495 2501 2502 2512 2619 2620 2622 2738 2748 2761 2768 2773 2870 2972
DEF$	0089	204	
DELET	817C	682	371
DELET$	0099	220	716
DELP$1	942B	3141	3178
DELP$2	9437	3144	3142
DELPAB	9391	3085	723 727 750

SYMBOL	VALUE	DEF	REFERENCE TABLE
DFLAG	0001	120	592 593
DIGIT$	00E9	293	
DIM$	008A	205	
DISACC	94DA	3226	822 1751
DISP$1	94DD	3228	1793 3242 3248 3280 3297
DISPL$	00A2	229	451
DISPL1	8257	821	370
DISPLA	0009	1	
DIVI$	00C4	263	
DSRFLG	0017	116	948 1020 1030 1040 1061 1500 1508 1633 1690 2669 2715 2835 2837 2919 3070 3474 3514 3597
EDGECH	007F	334	1661 1908 1916
EDITLN	6032	67	2584
EEE1	0058	174	805 1092 1108 2259 2261 2400 2460 2461 2498 2507 2508 2624 2630 2656 2735 2741 2743 2744 2747 2768 2776 2871 2872 2974
ELSE$	0081	196	
EMPSTR	9329	2996	2984
END$	008B	206	
ENLN	0032	140	789 795 2080 2140 2141 2221 2227 2254 2329 2366 2369 2379 2409 2412 2458 2489 2619
EOF	91CE	2804	384
EOF$2	9202	2822	2790 2817
EOLEX	8395	1029	823 864 869 1120 1297 3400
EQUAL$	00BE	257	
ERASE$	00EF	299	3229
ERR$$	6A84	73	618 1782 3665 3671 3674 3677 3680 3683 3686 3689 3692 3694
ERR$2	9782	3641	2205 3615
ERR$2B	977F	3639	644 2128 2402 2572 2591
ERRBV	97B6	3680	2834 3033 3043 3216 3293 3294
ERRCOD	0022	132	1433 1478
ERRDAT	97BE	3686	806 1979 1991 1994 2000
ERRFE	97C2	3689	425 427 636 707 709 773 847 853 876 927 962 1369 1374 1383 1623 1679 1683 2808 3209
ERRIM	97AE	3674	1131 1248 1251
ERRINP	97BA	3683	1433 1508
ERRIO	97A6	3664	731 2687 2690
ERRMEM	97B2	3677	670 3433
ERRMUV	97CA	3694	1631
ERRPV	97C6	3692	2294 2609
ERRSNM	97AA	3671	1705 1778 1982 3030 3427
ERRST	89B5	1782	
ERRSYN	8109	617	720 955 964 1119 1304 1752 1759 1794 2301 2304 2305 2306 2308 2312 2313 2314 2315 2316 2317 2319 2557 2830 2855 2903 2919 3230 3245 3251 3283 3285
EXP$	0076	185	
EXP$$	00CE	274	
EXTRAM	002E	138	1081 1085 2649 2654 2657 2659 2660 2710 2712 2713 2733 2751 2752
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 1150 1334 1370 1371 1377 1620 2123 2124 2125 2126 2136 2137 2138 2139 2140 2143 2144 2146 2149 2151 2397

SYMBOL	VALUE	DEF	REFERENCE TABLE
			2398 2399 2400 2405 2406 2407 2408 2409 2410 2411
			2412 2414 2416 2417 2578 2646 2650 2681 2684 2687
			2690 2791 2815 2819 2823 2932 3034 3035 3043 3147
			3148 3153 3158 3166 3167 3168 3193 3216 3217 3276
			3288 3289 3294 3335 3336 3354 3355 3356 3357 3385
			3389 3431 3622 3629
FAC1	004B	155	572 970 974 975 1079 1376 2683 3039 3264 3265
			3266 3275 3295 3334 3386 3387
FAC10	0054	164	1269 1270 3031 3033
FAC11	0055	165	921 938 1200 1206 1209 1210 1229 1232 1244 1247
			1254 1261 1263 1264 1271 3379
FAC12	0056	166	919 1214 1221 1987 2579 2582 2950 3376 3611 3612
			3623 3624
FAC13	0057	167	1201 1220 1261 1264 2580
FAC14	0058	168	2710 2751
FAC2	004C	156	918 934 968 1081 1085 1088 1091 1094 1115 1243
			1365 1385 1562 1631 1705 1778 1897 1925 1926 1981
			2017 2031 2518 2519 2520 2522 2672 2673 2678 2927
			3030 3149 3150 3156 3159 3166 3168 3170 3262 3273
			3385 3387 3427
FAC3	004D	157	2964 2966 2969
FAC4	004E	158	1123 1148 1149 1170 1281 1802 2931 3152 3156 3157
FAC5	004F	159	1290 3154 3155
FAC6	0050	160	1099 1102 1104 1106 1140 1145 1148 1151 1173 1280
			1281 1283 1366 1567 1802 1803 1985 1988 2489 2490
			2495 2517 2518 2529 2530 2929 2982 2988 2991 3151
			3157 3165 3171 3428
FAC7	0051	161	927 929 1120 1122 1124 1161 1271 1372 1779 1780
			2940 2989 3441 3442 3562
FAC8	0052	162	1089 1100 1199 2325 2327 2328 2329
FAC9	0053	163	1214 1217 1220 1221 1223 1245 1248 1251 1310
FFF1	0056	173	1106 2435 2444 2462 2463 2464 2499 2504 2513 2744
			2760 2761 2765 2772 2773 2869 2971
FIGURE	0002	1	
FIL	0002	92	3055 3443
FIL$$	8120	634	650
FILSPC	0001	49	977 998 3476 3499
FIXED$	00FA	310	
FLAG	0045	151	2060 2078 2147 2294 2553 2609 2852
FLG	0005	95	483 525 544 552 584 602 632 633 640 852
			853 876 1062 1346 1623 1683 2395 2567 3209 3495
			3625 3632
FLOAT1	9206	2826	2815
FNUM	0017	115	116 117 683 3039 3055 3443
FOR$	008C	207	
FREPTR	0040	147	612 671 690 728 2360 2469 2470 3119 3183 3435
			3436
GETDAT	92F3	2965	378 2987
GETGFL	92EF	2963	1097 1980 1998
GETRAM	92F7	2967	1496 1524 1561 1571 2949
GETSTR	0071	30	1143 1796 2930
GETV$0	9282	2894	2918
GETV$1	9286	2897	2908
GETV$2	92B7	2920	2913
GETVAR	927A	2887	1405 1471

SYMBOL	VALUE	DEF	REFERENCE TABLE
GO$	0085	200	
GOSUB$	0087	202	
GOTO$	0086	201	
GPNAME	9225	2851	2058 2295 2552
GREAD1	008C	53	2506 2766 2973
GREAT$	00C0	259	
GRMLST	9169	2731	391 2699
GRSUB2	9194	2759	392 1090 2257 2742
GRSUB3	91AC	2771	393 801 2621 2627 2653 2732
GRSUB4	919A	2763	2739 2775
GSAV1	8F39	2449	2441
GSAVE	8EAC	2387	2341 2350
GVMOV	8F3F	2456	2354
GVWITE	008B	52	1109 2436 2445 2465 2500 2514 2750 2873
GWRITE	0086	54	
GWSUB	6036	68	2260
IF$	0084	199	
IMAGE$	00A3	230	1094
INIT$1	96B3	3485	3369 3371 3483
INITKB	972A	3595	842 1331 1464 1617 2695 3227
INITPG	6014	61	2204
INP$2	875F	1470	1340
INP$3	86C3	1414	1411
INP$31	86E0	1424	1410
INP$32	871F	1445	1456
INP$33	875C	1467	1519
INP$37	8920	1708	1699
INP$39	892F	1714	1707
INP$65	87ED	1554	1573
INP$67	8810	1569	1564
INPU$2	8926	1711	1339
INPU$3	8762	1472	
INPU$4	8787	1493	1531
INPU$5	87AB	1507	1499 1522 1527 1528
INPU$6	87CB	1523	1500 1503
INPU$7	881A	1574	1557
INPUT	85E6	1330	373
INPUT$	0092	213	452
INPUTP	03AA	339	1335 1336 1465 1516
INSU1	88E1	1677	1342 1621
INSUB1	88FF	1695	1469 1625
INSUB2	8934	1720	1473 1653
INT$	00CF	275	
INTER$	00F5	305	
INTKBO	973F	3604	3492
INTR$0	8611	1348	1396
INTR$1	8623	1355	1397
INTR$2	8696	1398	1395
IO	0084	47	976 997 3450 3475 3498 3587
IOCALL	9749	3610	691 777 2130 2170 2181 2190 2381 2419 2437 2447 2450 2527 2533 2535 2570 2586 2592 2717 3504
IOCL$1	9750	3613	1349 1415 1635 2488 2568 2680 2811
IOSTRT	003C	145	658 659 661 752 753 3049 3095 3096 3107 3108 3109 3110 3112
KILSYM	6022	63	2207 2384 2858

SYMBOL	VALUE	DEF	REFERENCE TABLE
LEN	0008	97	572 650 651 2682 3515
LEN$	00D5	281	
LESS$	00BF	258	
LET$	008D	208	
LINE	A006	77	787
LINP10	8857	1627	1620
LINPUT	8839	1617	395
LIST	9058	2608	382
LIST$0	907D	2626	2641
LIST$1	90EB	2674	2679
LIST$3	913E	2708	2703
LIST$4	914F	2714	2704
LIST$5	913B	2705	2707
LITS05	A002	76	1986 1995
LLIST	6A74	70	2701
LN$	00C9	269	1075 3338
LNBUF	0036	142	795 796 803 806 807 810 2001 2004 2005
LOG$	00D0	276	
LPAR$	00B7	250	964 1759 2830 2898 3255 3285
LRTOP$	8D27	2206	2088 2108
LRTOPL	8EA6	2383	2452 2537 2594
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 3432
MERG$1	904B	2589	
MERGE	8FF7	2551	390
MFLAG	0000	119	535 536
MINUS	002D	322	2612
MINUS$	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
			1163 1292 1492 1494 1495 2073 2074 2075 2076 2077
			2141 2142 2144 2145 2146 2151 2152 2153 2154 2155
			2156 2159 2219 2221 2222 2224 2225 2226 2230 2249
			2250 2259 2263 2332 2333 2334 2336 3096 3097 3098
			3098 3100 3101 3102 3103 3105 3118 3119 3121 3125
			3428 3429 3431 3435 3440 3441 3472 3473 3496 3497
			3572 3574 3575 3576 3577 3578 3583 3585 3586
MRGPAB	039E	338	2583 2585
MSGTA	6053	65	
MULT$	00C3	262	
MVDN	0088	48	2236 2242
NEWTOP	03BE	352	2084 2103 2192 2220 2234 2250 2256 2467
NEXT$	0096	217	
NLEN	000D	101	2062 2671 2864 2865 2867 2872 3092 3442 3445 3624
NUDTAB	0028	135	
NUM$	00C8	268	1982 2855 2948 2984 3331
NUMBE$	00FD	313	769 843 1332 1618 3021
NUMER$	00E8	292	1765 1768
NXTC$0	95B4	3332	3330
NXTCHR	95A5	3327	1698
OFFSET	0060	331	334 1420 1446 1449 1454 1640 1656 1662 1712 1723
			1854 1892 1921 2199 2539 2640 2641 2726 3235 3597

SYMBOL	VALUE	DEF	REFERENCE TABLE
			3621
OFS	0003	93	610 653 689 856 1042 1347 1351 1391 1400 1410 1416 1440 1458 1579 1634 1646 1651 3072 3212 3440 3503 3516
OL$$	8CD3	2162	
OLD	8BBF	2054	381
OLD$2	8D83	2253	2266
OLD$3	8C5C	2114	2070
OLD$5	8C2B	2087	
OLD$7	8C53	2105	2196
OLD$9	8D0B	2189	2180
OLD1	8BC5	2057	389 2055
OLDC$0	8227	788	771 786
OLDC$1	823A	800	808
OLDCD	821F	784	769
OLDER	8D22	2203	2077 2115 2133 2136 2146 2158 2159 2175 2186
OLDTOP	03BC	350	2082 2102 2152 2219 2249 2466
ON$	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 3449
OPTI$0	8047	441	508 577
OPTI$1	804C	446	503
OPTIO$	009E	225	
OPTION	8045	436	474 537 594
OSTRNG	96F9	3561	936 941 1706
OTHE$0	8308	935	932
OTHE$1	8303	933	917
OUTEOF	937B	3069	710 746 774 1684 3211
OUTPU$	00F7	307	
OUTREC	9690	3471	383 971 1006 1022 1035 1165 2716 3074 3362 3579
PAB	9160	2725	2670
PAB1	8FEE	2538	2487 2566
PAB3	8D19	2198	2122 2394
PABLEN	000E	326	2063 2861 3093 3429 3439
PABPTR	0004	87	483 507 525 544 552 572 584 602 610 632 633 640 645 650 651 653 659 666 668 668 672 689 711 717 726 728 729 730 743 743 745 748 752 775 827 852 853 855 856 858 874 876 1021 1031 1032 1042 1048 1052 1062 1346 1347 1351 1353 1360 1391 1396 1399 1400 1409 1410

SYMBOL	VALUE	DEF	REFERENCE TABLE
			1416 1418 1426 1428 1440 1457 1458 1579 1581 1623
			1634 1637 1638 1646 1648 1650 1651 1683 1685 1760
			1817 1821 1824 1825 1826 1831 1853 1878 1933 1934
			1942 1944 1957 2061 2062 2064 2065 2066 2067 2068
			2122 2126 2133 2175 2186 2218 2220 2227 2228 2233
			2268 2373 2375 2376 2394 2395 2399 2418 2433 2446
			2487 2526 2532 2566 2567 2575 2583 2585 2591 2668
			2670 2682 2685 2786 2788 2805 2810 2812 2859 2860
			2861 2861 2863 2864 2865 2867 2871 3049 3054 3055
			3056 3056 3071 3072 3089 3092 3094 3095 3097 3100
			3105 3107 3112 3128 3129 3130 3131 3132 3132 3134
			3140 3143 3144 3146 3148 3149 3174 3175 3176 3177
			3177 3209 3212 3217 3230 3237 3241 3245 3246 3251
			3278 3279 3283 3290 3295 3296 3303 3352 3355 3369
			3370 3436 3437 3438 3439 3439 3440 3442 3443 3444
			3477 3478 3495 3502 3503 3515 3516 3521 3596 3612
			3621 3623 3625 3632
PARFN	9645	3421	434 684
PARREC	94AD	3205	776 865 1686
PARSE	0074	33	513 906 965 1113 1196 1703 1776 2831 3022 3213
			3256 3267 3286 3425
PERMA$	00FB	311	
PFLAG	0002	121	472 473
PGMCHR	0079	37	437 447 499 562 713 872 963 1016 1076 1078
			1080 1193 1393 1438 1555 1702 1758 1764 1772 1974
			2024 2029 2059 2298 2555 2662 2667 2676 2862 2907
			3207 3210 3231 3247 3252 3284 3333 3342
PGMPTR	002C	137	435 1335 1437 1465 1516 1551 1552 1695 1701 1708
			2019 2034 2302 2304 2305 2306 2307 2312 2313 2314
			2315 2316 2317 2318 2661 2867 2870 2875 2888 3336
			3339
PLUS$	00C1	260	
POP	0005	1	
PR$$O	96EC	3517	3508
PRCOL	8377	1005	3411
PREXIT	83A4	1039	1020 1021 1025
PRGFLG	0044	150	2021
PRIN$O	82B1	873	866
PRIN$1	82C4	883	827 835
PRINIT	96E0	3513	859 3073
PRINT	8266	841	372
PRINT$	009C	223	
PRN$10	82A5	867	845
PRSEM	837A	1015	3404
PRSM$1	8381	1019	1302
PRTAB	832D	960	891
PRTCOM	835C	988	3408
PRTNFN	00CE	190	1727
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
			1062 1104 1104 1126 1146 1148 1148 1154 1155 1176
			1178 1179 1182 1183 1194 1198 1203 1204 1205 1208
			1215 1225 1227 1230 1264 1274 1281 1281 1312 1335

SYMBOL	VALUE	DEF	REFERENCE TABLE
			1336 1346 1347 1351 1353 1360 1361 1370 1391 1396
			1399 1400 1409 1410 1413 1416 1418 1420 1426 1428
			1440 1446 1449 1454 1457 1458 1465 1466 1495 1497
			1503 1516 1517 1579 1581 1623 1634 1637 1638 1640
			1646 1648 1650 1651 1656 1660 1663 1683 1685 1712
			1723 1726 1750 1798 1802 1802 1806 1807 1844 1847
			1848 1854 1864 1870 1871 1880 1892 1908 1916 1920
			1920 1921 1935 1939 1940 1941 1947 1948 1991 2062
			2064 2065 2066 2067 2068 2073 2074 2075 2077 2080
			2081 2082 2082 2084 2102 2103 2122 2126 2133 2136
			2138 2140 2144 2146 2151 2152 2175 2186 2192 2218
			2219 2220 2234 2239 2240 2243 2249 2250 2256 2263
			2268 2297 2302 2304 2305 2306 2307 2312 2313 2314
			2315 2316 2317 2318 2320 2327 2362 2364 2366 2368
			2369 2370 2371 2373 2375 2376 2394 2395 2399 2405
			2407 2409 2411 2412 2413 2414 2417 2418 2433 2446
			2466 2467 2487 2501 2517 2517 2518 2520 2522 2522
			2524 2526 2531 2532 2566 2567 2572 2575 2578 2582
			2582 2583 2585 2588 2591 2640 2641 2670 2675 2682
			2685 2735 2736 2768 2791 2810 2813 2814 2860 2861
			2861 2864 2865 2867 2867 2941 2941 2957 2968 3055
			3056 3071 3072 3089 3092 3097 3098 3100 3100 3101
			3102 3103 3105 3107 3122 3122 3129 3130 3131 3132
RAMFLG	0089	188	2023 2025 2866
RAMFRE	0086	187	2106 2107
RAMPTR	000A	108	1407 1413 1655 1663 1665 1734
RAMTOP	0084	186	1103 2022 2088 2103 2115 2159 2192 2234 2240 2243
			2256 2323 2332 2377 2417 2430 2462 2466 2494 2686
			2698 2709 2764 2964 3142
RANDO$	0095	216	
READ	8B34	1975	377
READ$	0097	218	
READL1	6A86	74	1883
READLN	6A76	71	1731
REC$	00DE	290	3208
RECENT	881F	1578	1417 1425 1439
RECLEN	0007	104	924 946 968 995 1408 1435 1436 1442 1443 1444
			1841 1848 1857 1948 2682 2683 3357 3367 3368 3369
			3370 3472 3496 3515 3574 3603
RELAT$	00F4	304	
RELO$1	8DA8	2267	2251 2254
RELOCA	8D2B	2217	2085 2104 2195 2468
REM$	009A	221	
RESTO$	0094	215	
RESTOR	81FC	767	376
RETUR$	0088	203	
RFLAG	0004	123	575 576 639
RKEY	0075	183	2704
RND$	00D7	283	
RNM	000A	99	507 645 775 2064 2065 2066 2375 2376 2791 3217
ROM	000A	1	2122 2394 2487 2566 2670 2815 3645
RPAR$	00B6	249	966 1777 1792 2904 3268 3271
RSTK	0088	189	3646 3647
RSTRIN	95F6	3375	922 940
RTC	9377	3058	1061 2999 3034 3055 3208 3317 3329 3477

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

RTNG    0026   134
SA$1    8E20  2321      2300
SAPROT  03B9   349      2297 2320 2370 2413
SAVE    8DAD  2293       380
SAVMG   8F6B  2477      2309
SCR     000C   100      2814 3621
SCR$    8832  1586      1515
SCRNBS  02E0   332      1585 1588 3239 3607
SCRO    96B7  3489      3363
SCROLL  0083    46      1479 1587 1669 1959 3490
SEARCH  8B94  2018      1095
SEETWO  0003    43      1084 2648 2652
SEMIC$  00B4   247      1197 1300
SEQUE$  00F6   306
SETVW   899B  1771      1769
SFLAG   0003   122       497  498
SGN$$   00D1   277
SIGN$   0075   184
SIN$    00D2   278
SIZCCP  03B4   344      1847 1947 2218 2268
SIZE$   00EB   295      3282
SIZE1   95CA  3351      1827 3304
SIZREC  03B6   345      1848 1948 2239 2240 2243
SIZXPT  03B8   348      1844 1935
SMB     007B    39      1357 1559 1630 1867 1977
SMTSRT  001E   130
SPACE   0020   321       947 1656
SPEED   007E    41       431  714  832  879 1072 1083 1298 1687 1714 1790
                        1818 2647 2651 2914 3019 3232 3253 3258 3269
SPRITE  0006     1
SQR$    00D3   279
SRDA$0  8BB8  2033      2031
SRDA$1  8BB0  2030      2026
SRDATA  8B91  2016       386
SREF    001C   129      1104 1108 1144 1145 1146 1150 1797 1806 1906 1907
                        1908 1911 1920 1921 1923 1987 1988 1989 1991 2931
                        2932 2941 3379
SSEP$   0082   197
STADDR  000A   107       661  663  664  664  666 1406 1462 1477 1551 1695
                        1696 1701 1708 1733 2061 2062 2063 2065 2067 2073
                        2074 2075 2077 2080 2081 2082 2083 2098 2165 2223
                        2231 2252 2254 2255 2258 2261 2263 2265 2360 2361
                        2362 2363 2364 2365 2366 2367 2368 2369 2371 2373
                        2374 2376 2461 2865 2867 2869 2875 2888 3089 3090
                        3103 3110 3118 3122 3123 3127
STEP$   00B2   245
STLN    0030   139       789  806 2004 2081 2094 2100 2106 2138 2142 2153
                        2222 2223 2225 2228 2252 2325 2333 2364 2368 2378
                        2407 2411 2426 2431 2457 2459 2463 2469 2530 2628
                        2690 3143
STOP$   0098   219
STREND  001A   128
STRIN$  00C7   266      2854 2983 3330
STRSP   0018   127
STRVAL  0065   329       918  934 1115 1562 1631 1705 1778 1925 1926 1981
```

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

                       3030 3427
STVSPT  0024    133    2666
SUB$    00A1    228
SUBEOF  920E   2829    2787 2807
SUBREC  91BA   2785     387
SUBSTK  0073    182    3400 3404 3408 3411 3647 3656 3657 3663
SUBTAB  003A    144
SYM     007A     38    1356 1558 1629 1866 1976 2895
SYMTAB  003E    146    3137 3138 3139 3140
SYNCHK  0000     42     432  715  833  880 1073 1299 1688 1715 1791 1819
                       2915 3020 3233 3254 3259 3270
TAB$    00FC    312     891
TABLE   000B      1
TAN$    00D4    280
TEMP5   0066    180    1104 1107 1353 1354 1361 1362 1370 1388 1638 1640
                       1641 1645 1646 1649 1650 1655 2941 2990
THEN$   00B0    243
TO$     00B1    244
TONE1   0034     58    1053 1728 1822
TOP     0003      1
TOPL02  6030     66    3645
TOPL10  601A     62    2719
TOPL15  6012     60     793 2056 2385 2721
TRACE$  0090    211
TREM$   0083    198    3317
TSTINT  83C5   1060     916  961 3405
TSTS$0  961C   3401    3399
TSTS$1  9644   3413    3402 3403 3406
TSTSEP  9612   3396     890  954
UALPH$  00EA    294    1766
UBSUB   A020     81    2331
UNBRE$  008F    210
UNQST$  00C8    267     268
UNTRA$  0091    212
UPDAT$  00F8    308
USING   83CF   1071     874  877  884
USING$  00ED    297     884 1074
USN$42  84DF   1188    1179
USN$55  855A   1242    1230 1233
USN$67  85A0   1276    1286
USN$68  85B4   1284    1280
USN$95  85C9   1294    1291
USNG$0  8452   1125    1129
USNG$1  845E   1130    1087 1096 1115 1122 1142
USNG$3  8478   1147    1174
USNG$4  8489   1153    1157 1267 1278
USNG$5  8529   1216    1203
USNG$9  85B9   1287    1159 1192
VALCD   A016     80    1927
VALID$  00FE    314    1757 3287 3299
VALIDL  03B2    343    1807
VALIDP  03B0    342    1806
VAR0    0000     85    1272 1274 1285 1525 1528 1660 1661 1662 1663 1982
                       1999 2000 2020 2035 2093 2094 2095 2096 2231 2335
                       2336 2337 2350 2493 2507 2513 2520 2522 2525 2526
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
			2948 2951 2952 2968 2974 2983 2984 2989 2997 3005 3007
VAR4	000E	110	1202 1215 1225 1226 1227 1228 1230 1231 1235 1266 1269 1277 1311 1312 1492 1518 1553 1627 1730 1732 1865 1882 1884 2890
VAR5	0010	112	1436 1442 1443 1461 1482 2232 2245 2343 2350 2352 2460 2889 2911
VAR6	0011	113	1430 1434 1435 1444 1455 1461 1474 1481 1482 1526 1530 2896 2899 2901 2905
VARA	002A	136	1351 1352 1354 1360 1363 1389 1396 1399 1400 1418 1419 1422 1426 1427 1428 1429 1656 1657 1659 1761 1780 1795 1807 1811 1814 1860 1861 1871 1890 1891 1892 1893 1899 1907 1915 1940
VARC	0008	106	2612
VARIA$	00F3	303	453
VARW	0020	131	1420 1421 1446 1448 1449 1450 1453 1454 1457 1458 1579 1580 1581 1659 1660 1667 1721 1762 1771 1786 1798 1800 1804 1810 1811 1870 1890 1899 1906 1915 1916 1917 1920 1922 1939 2639 2640 2641
VARY2	0006	88	2233
VDP	000C	1	
VEL	0007	1	
VGWITE	008A	51	2101 2177 2188
VLID$0	89BD	1788	1773
VPOP	0078	36	1160 1171 1288 1295 1801 1895 3434
VPUSH	0077	35	1139 1152 1172 1358 1560 1632 1785 1868 1896 1978 2910 3430
VRAMVS	0958	328	2343 2665 2668 2671 2859
VSAV$	8E5E	2359	
VSPTR	006E	181	1176 1179 1194 1198 1518 1553 1627 1730 1732 1865 1882 1884 2665 2666 2890
VWIDTH	001C	330	1826 2694 3273 3603
WARN$$	6A82	72	791 1511 1900 1930 3659
WR$$5	87B6	1513	1506
WRNINP	87B2	1510	1478 1482
WRNIO	979E	3658	
WRNNPP	822C	790	
WRNSNM	8AF2	1929	
XPT	000E	1	1049 1050 1480 1670 1840 1841 1842 1844 1935 1960 2694 3236 3599 3600
YPT	000D	1	