```
                1          TITLE EDIT-359
                2  ********************************************************
                3  *
                4  *          EEEEEE    DDDDD     IIIII    TTTTT
                5  *          E         D    D      I        T
                6  *          E         D    D      I        T
                7  *          EEEE      D    D      I        T
                8  *          E         D    D      I        T
                9  *          E         D    D      I        T
               10  *          EEEEEE    DDDDD     IIIII     T
               11  *
               12  *
               13  *              33333     555555     99999
               14  *             3     3    5         9     9
               15  *                   3    5         9     9
               16  *               3333     5555       99999
               17  *                   3         5         9
               18  *             3     3    5    5        9
               19  *              33333     5555         9
               20  *
               21  ********************************************************
A040           22  CPUBAS EQU  >A040          CPU base
               23  ********************************************************
0010           24  M$MON  EQU  >0010          Module MONITOR branch table addres
6A70           25  M$PSCN EQU  >6A70          Module PSCAN branch table address
A000           26  M$EXEC EQU  >A000          Module EXEC branch table address
8000           27  M$FLMG EQU  >8000          Module FLMGR branch table address
               28  ********************************************************
               29  *      EQUATES FOR ROUTINES FROM OTHER SECTIONS
               30
0032           31  ATN$$  EQU  M$MON+>22           Arctangent routine
               32
6A70           33  PRESCN EQU  M$PSCN+>00          PRESCaN a program
6A74           34  LLIST  EQU  M$PSCN+>04          Line LIST routine
6A76           35  READLN EQU  M$PSCN+>06          READ a LiNe
6A7C           36  DISO   EQU  M$PSCN+>0C          Display line number
6A82           37  WARN$$ EQU  M$PSCN+>12          Warning routine
6A84           38  ERR$$  EQU  M$PSCN+>14          ERRor routine
6A86           39  READL1 EQU  M$PSCN+>16          READ Line 1
               40
A00C           41  EXEC1  EQU  M$EXEC+>0C
A018           42  INTRND EQU  M$EXEC+>18          Initialize random number
A026           43  LINK1  EQU  M$EXEC+>26          LINK to subprograms
               44
8012           45  CLSALL EQU  M$FLMG+>12          CLoSe ALL open files
8014           46  SAVE   EQU  M$FLMG+>14          SAVE
8016           47  OLD    EQU  M$FLMG+>16          OLD
8018           48  LIST   EQU  M$FLMG+>18          LIST a program
8024           49  CHKEND EQU  M$FLMG+>24          Check EOS
8026           50  OLD1   EQU  M$FLMG+>26          A subroutine for LOAD
8028           51  MERGE  EQU  M$FLMG+>28          Merge a program
802A           52  GRMLST EQU  M$FLMG+>2A          List program line from ERAM
802C           53  GRSUB2 EQU  M$FLMG+>2C          Read from ERAM(GREAD1) or VDP
802E           54  GRSUB3 EQU  M$FLMG+>2E          Read from ERAM(use GREAD1) or
               55  *                              VDP, reset possible bkpt too
```

```
                  ********************************************************
                  *          Equates for routines in MONITOR
0018              CHAR2$ EQU  >18          CHARACTER TABLE ADDRESS
                  ********************************************************
                  *                  Equates for XMLs
0070              COMPCT EQU  >70              PERFORM A GARBAGE COLLECTION
0078              VPOP   EQU  >78              Pop off value stack
0079              PGMCHR EQU  >79              GET PROGRAM CHARACTER
007E              SPEED  EQU  >7E              SPEED UP XML
0000              SYNCHK EQU  0                SYNCHK XML selector
0003              SEETWO EQU  3                SEETWO XML selector
007F              CRUNCH EQU  >7F              Crunch an input line
0081              CONTIN EQU  >81              Continue after a break
0083              SCROLL EQU  >83              SCROLL THE SCREEN
0085              GREAD  EQU  >85              READ DATA FROM ERAM
0086              GWRITE EQU  >86              WRITE DATA TO ERAM
0087              DELREP EQU  >87              REMOVE CONTENT FROM VDP/ERAM
0088              MVDN   EQU  >88              MOVE DATA IN VDP/ERAM
0089              MVUP   EQU  >89              MOVE DATA IN VDP/ERAM
008A              VGWITE EQU  >8A              MOVE DATA FROM VDP TO ERAM
008E              GDTECT EQU  >8E              ERAM DETECT&ROM PAGE 1 ENABLE
                  ********************************************************
                  *                  GPL Status Block
0072              STACK  EQU  >72              STACK FOR DATA
0073              SUBSTK EQU  >73              SUBROUTINE STACK
0074              KEYBD  EQU  >74              KEYBOARD SELECTION
0075              RKEY   EQU  >75              KEY CODE
0078              RANDOM EQU  >78              RANDOM NUMBER GENERATOR
0079              TIMER  EQU  >79              TIMING REGISTER
007A              MOTION EQU  >7A              NUMBER OF MOVING SPRITES
007B              VDPSTS EQU  >7B              VDP STATUS REGISTER
007C              ERCODE EQU  >7C              STATUS REGISTER
                  ********************************************************
                  *              Temporary workspaces in EDIT
0000              VARO   EQU  >00
0002              STPT   EQU  >02              TWO BYTES
0004              VARY   EQU  >04
0006              VARY2  EQU  VARY+2
0004              PABPTR EQU  >04
0006              CCPPTR EQU  >06
0007              RECLEN EQU  >07
0008              CCPADR EQU  >08
0008              VARC   EQU  >08
000A              RAMPTR EQU  >0A              Pointer for crunching
000C              BYTE   EQU  >0C
000E              CURINC EQU  >0E              Increment for auto-num
0014              CURLIN EQU  >14              Current line for auto-num
0016              VAR9   EQU  >16
0017              DSRFLG EQU  >17
0002              AAA1   EQU  >02
000C              BBB1   EQU  >0C
0008              CCC1   EQU  >08
                  ********************************************************
                  *              Permanent workspace variables
0018              STRSP  EQU  >18              String space beginning
```

```
001A    111  STREND EQU  >1A          String space end
001C    112  SREF   EQU  >1C          Temporary string pointer
001E    113  SMTSRT EQU  >1E          Start of current statement
0020    114  VARW   EQU  >20          Screen address
0022    115  ERRCOD EQU  >22          Return error code from ALC
0024    116  STVSPT EQU  >24          Value-stack base
002A    117  VARA   EQU  >2A          Ending display location
002C    118  PGMPTR EQU  >2C          Program text pointer
002E    119  EXTRAM EQU  >2E          Line number table pointer
0030    120  STLN   EQU  >30          Start of line number table
0032    121  ENLN   EQU  >32          End of line number table
0034    122  DATA   EQU  >34          DATA pointer for READ
0036    123  LNBUF  EQU  >36          Line table pointer for READ
0038    124  INTRIN EQU  >38          Addr of intrinsic poly consts
003A    125  SUBTAB EQU  >3A          Subprogram symbol table
003C    126  IOSTRT EQU  >3C          PAB list
003E    127  SYMTAB EQU  >3E          Symbol table pointer
0040    128  FREPTR EQU  >40          Free space pointer
0042    129  CHAT   EQU  >42          Current character/token
0043    130  BASE   EQU  >43          OPTION BASE value
0044    131  PRGFLG EQU  >44          Program/imperative flag
0045    132  FLAG   EQU  >45          General 8-bit flag
0046    133  BUFLEV EQU  >46          Crunch-buffer destruction lev
0048    134  LSUBP  EQU  >48          Last subprogram block on stac
004A    135  FAC    EQU  >4A          Floating-point ACcumulator
004B    136  FAC1   EQU  FAC+1
004C    137  FAC2   EQU  FAC+2
004E    138  FAC4   EQU  FAC+4
0050    139  FAC6   EQU  FAC+6
0051    140  FAC7   EQU  FAC+7
0052    141  FAC8   EQU  FAC+8
0054    142  FAC10  EQU  FAC+10
0056    143  FAC12  EQU  FAC+12
0057    144  FAC13  EQU  FAC+13
0058    145  FAC14  EQU  FAC+14
0059    146  FAC15  EQU  FAC+15
004C    147  AAA    EQU  FAC+2
004E    148  CCC    EQU  FAC+4
0050    149  BBB    EQU  FAC+6
004C    150  DDD    EQU  FAC+2
004E    151  FFF    EQU  FAC+4
0050    152  EEE    EQU  FAC+6
0054    153  DDD1   EQU  FAC+10
0056    154  FFF1   EQU  FAC+12
0058    155  EEE1   EQU  FAC+14
005C    156  ARG    EQU  >5C          Floating-point ARGument
005E    157  ARG2   EQU  ARG+2
005F    158  ARG3   EQU  ARG+3
0060    159  ARG4   EQU  ARG+4
0061    160  ARG5   EQU  ARG+5
0062    161  ARG6   EQU  ARG+6
006E    162  VSPTR  EQU  >6E          Value stack pointer
0076    163  EXP$   EQU  >76          Exponent in floating-point
        164
0084    165  RAMTOP EQU  >84          Highest address in ERAM
```

```
0086    166  RAMFRE EQU  >86                 Free pointer in the ERAM
0088    167  RSTK   EQU  >88                 Subroutine stack base
0089    168  RAMFLG EQU  >89                 Eram flag
        169  ********************************************************** **
        170  *                    VDP addresses
02E2    171  NLNADD EQU  >2E2                New LiNe ADDress
02FE    172  ENDSCR EQU  >2FE                END of SCReen address
0371    173  LODFLG EQU  >371                Auto-boot needed flag
0372    174  START  EQU  >372                Line to start execution at
0382    175  SPGMPT EQU  >382                Saved PGMPTR for continue
0384    176  SBUFLV EQU  >384                Saved BUFLEV for continue
0386    177  SEXTRM EQU  >386                Saved EXTRAM for continue
0388    178  SAVEVP EQU  >388                Saved VSPTR for continue
038A    179  ERRLN  EQU  >38A                On-error line pointer.
038C    180  BUFSRT EQU  >38C                Edit recall start addr(VARW)
038E    181  BUFEND EQU  >38E                Edit recall end addr(VARA)
0396    182  SLSUBP EQU  >396                Saved LSUBP for continue
0398    183  SFLAG  EQU  >398                Saved on-warning/break bits
0820    184  CRNBUF EQU  >820                CRUnch BUFfer address
08BE    185  CRNEND EQU  >8BE                CRuNch buffer END
08C0    186  RECBUF EQU  >8C0                Edit RECall BUFfer
0958    187  VRAMVS EQU  >958                Default base of value stack
0390    188  CSNTMP EQU  >0390               Use as temporary stored place
        189  ********************************************************** ***
        190  *                    IMMEDIATE VALUES
000B    191  UPARR  EQU  >0B
000A    192  DWNARR EQU  >0A
000D    193  CHRTN  EQU  >0D
0060    194  OFFSET EQU  >60
0002    195  LIST$  EQU  >02
0005    196  OLD$   EQU  >05
0006    197  RESEQ$ EQU  >06
0007    198  SAVE$  EQU  >07
0008    199  MERGE$ EQU  >08
        200  *                  Bits in FLAG
0000    201  NUMBIT EQU  0                   Autonum bit
```

```
                    *
                    *              B A S I C   T O K E N   T A B L E
                    *
                    *
                    *       EQU   >80          SPARE
0081                ELSE$   EQU   >81          "ELSE"
0082                SSEP$   EQU   >82          "::"
0083                TREM$   EQU   >83          "!"
0084                IF$     EQU   >84          "IF"
0085                GO$     EQU   >85          "GO"
0086                GOTO$   EQU   >86          "GOTO"
0087                GOSUB$  EQU   >87          "GOSUB"
0088                RETUR$  EQU   >88          "RETURN"
0089                DEF$    EQU   >89          "DEF"
008A                DIM$    EQU   >8A          "DIM"
008B                END$    EQU   >8B          "END"
008C                FOR$    EQU   >8C          "FOR"
008D                LET$    EQU   >8D          "LET"
008E                BREAK$  EQU   >8E          "BREAK"
008F                UNBRE$  EQU   >8F          "UNBREAK"
0090                TRACE$  EQU   >90          "TRACE"
0091                UNTRA$  EQU   >91          "UNTRACE"
0092                INPUT$  EQU   >92          "INPUT"
0093                DATA$   EQU   >93          "DATA"
0094                RESTO$  EQU   >94          "RESTORE"
0095                RANDO$  EQU   >95          "RANDOMIZE"
0096                NEXT$   EQU   >96          "NEXT"
0097                READ$   EQU   >97          "READ"
0098                STOP$   EQU   >98          "STOP"
0099                DELET$  EQU   >99          "DELETE"
009A                REM$    EQU   >9A          "REM"
009B                ON$     EQU   >9B          "ON"
009C                PRINT$  EQU   >9C          "PRINT"
009D                CALL$   EQU   >9D          "CALL"
009E                OPTIO$  EQU   >9E          "OPTION"
009F                OPEN$   EQU   >9F          "OPEN"
00A0                CLOSE$  EQU   >A0          "CLOSE"
00A1                SUB$    EQU   >A1          "SUB"
00A2                DISPL$  EQU   >A2          "DISPLAY"
00A3                IMAGE$  EQU   >A3          "IMAGE"
00A4                ACCEP$  EQU   >A4          "ACCEPT"
00A5                ERROR$  EQU   >A5          "ERROR"
00A6                WARN$   EQU   >A6          "WARNING"
00A7                SUBXT$  EQU   >A7          "SUBEXIT"
00A8                SUBND$  EQU   >A8          "SUBEND"
00A9                RUN$    EQU   >A9          "RUN"
                    *       EQU   >AA          SPARES
                    *       EQU   >AB
                    *       EQU   >AC
                    *       EQU   >AD
                    *       EQU   >AE
                    *       EQU   >AF
00B0                THEN$   EQU   >B0          "THEN"
00B1                TO$     EQU   >B1          "TO"
00B2                STEP$   EQU   >B2          "STEP"
```

```
00B3      258  COMMA$  EQU  >B3          ","
00B4      259  SEMIC$  EQU  >B4          ";"
00B5      260  COLON$  EQU  >B5          ":"
00B6      261  RPAR$   EQU  >B6          ")"
00B7      262  LPAR$   EQU  >B7          "("
00B8      263  CONC$   EQU  >B8          "&"
          264  *       EQU  >B9          SPARE
00BA      265  OR$     EQU  >BA          "OR"
00BB      266  AND$    EQU  >BB          "AND"
00BC      267  XOR$    EQU  >BC          "XOR"
00BD      268  NOT$    EQU  >BD          "NOT"
00BE      269  EQUAL$  EQU  >BE          "="
00BF      270  LESS$   EQU  >BF          "<"
00C0      271  GREAT$  EQU  >C0          ">"
00C1      272  PLUS$   EQU  >C1          "+"
00C2      273  MINUS$  EQU  >C2          "-"
00C3      274  MULT$   EQU  >C3          "*"
00C4      275  DIVI$   EQU  >C4          "/"
00C5      276  CIRCU$  EQU  >C5          "^"
          277  *       EQU  >C6          SPARE
00C7      278  STRIN$  EQU  >C7          QUOTED STRING
00C8      279  UNQST$  EQU  >C8          UNQUOTED STRING
00C8      280  NUM$    EQU  UNQST$       ALSO NUMERICAL STRING
00C9      281  LN$     EQU  >C9          LINE NUMBER CONSTANT
          282  *       EQU  >CA          SPARE
00CB      283  ABS$    EQU  >CB          "ABS"
00CC      284  ATN$    EQU  >CC          "ATN"
00CD      285  COS$    EQU  >CD          "COS"
00CE      286  EXP$$   EQU  >CE          "EXP"
00CF      287  INT$    EQU  >CF          "INT"
00D0      288  LOG$    EQU  >D0          "LOG"
00D1      289  SGN$$   EQU  >D1          "SGN"
00D2      290  SIN$    EQU  >D2          "SIN"
00D3      291  SQR$    EQU  >D3          "SQR"
00D4      292  TAN$    EQU  >D4          "TAN"
00D5      293  LEN$    EQU  >D5          "LEN"
00D6      294  CHR$$   EQU  >D6          "CHR$"
00D7      295  RND$    EQU  >D7          "RND"
00D8      296  SEG$$   EQU  >D8          "SEG$"
00D9      297  POS$    EQU  >D9          "POS"
00DA      298  VAL     EQU  >DA          "VAL"
00DB      299  STR$$   EQU  >DB          "STR$"
          300  *
          301  *
          302  *
00E8      303  NUMER$  EQU  >E8          "NUMERIC"
00E9      304  DIGIT$  EQU  >E9          "DIGIT"
00EA      305  UALPH$  EQU  >EA          "UALPHA"
00EB      306  SIZE$   EQU  >EB          "SIZE"
00EC      307  ALL$    EQU  >EC          "ALL"
00ED      308  USING$  EQU  >ED          "USING"
00EE      309  BEEP$   EQU  >EE          "BEEP"
00EF      310  ERASE$  EQU  >EF          "ERASE"
00F0      311  AT$     EQU  >F0          "AT"
00F1      312  BASE$   EQU  >F1          "BASE"
```

```
                313  *       EQU  >F2          "VARIABLE"
                314  *       EQU  >F3          "TEMPORARY"
                315  *       EQU  >F4          "RELATIVE"
                316  *       EQU  >F5          "INTERNAL"
00F6            317  SEQUE$  EQU  >F6          "SEQUENTIAL"
00F7            318  OUTPU$  EQU  >F7          "OUTPUT"
00F8            319  UPDAT$  EQU  >F8          "UPDATE"
00F9            320  APPEN$  EQU  >F9          "APPEND"
00FA            321  FIXED$  EQU  >FA          "FIXED"
00FB            322  PERMA$  EQU  >FB          "PERMANENT"
00FC            323  TAB$    EQU  >FC          "TAB"
00FD            324  NUMBE$  EQU  >FD          "#"
00FE            325  VALID$  EQU  >FE          VALIDATE
                326  *       EQU  >FF          ILLEGAL VALUE
                327  ********************************************************************
                328  *                 ASCII CODES FOR EACH CHARACTER
0030            329  ZERO    EQU  :0:          0
0039            330  NINE    EQU  :9:          9
003A            331  COLON   EQU  ::::         :
003B            332  SEMIC   EQU  :;:          ;
003C            333  LESS    EQU  :<:          <
003D            334  EQUAL   EQU  :=:          =
003E            335  GREAT   EQU  :>:          >
003F            336  QUEST   EQU  :?:          ?
0041            337  A       EQU  :A:          A
0045            338  E       EQU  :E:          E
005A            339  Z       EQU  :Z:          Z
005B            340  OPENB   EQU  :[:
005C            341  BACKS   EQU  :\:
005D            342  CLOSEB  EQU  :]:
005F            343  UNLN    EQU  :_:
0020            344  SPACE   EQU  : :
0021            345  EXCL    EQU  :!:          !
0022            346  QUOTE   EQU  :":          "
0024            347  DOLLAR  EQU  :$:          $
0028            348  LPAR    EQU  :(:          (
0029            349  RPAR    EQU  :):          )
002B            350  PLUS    EQU  :+:          +
002C            351  COMMA   EQU  :,:          ,
002D            352  MINUS   EQU  :-:          -
002E            353  DOT     EQU  :.:          .
007E            354  CURSOR  EQU  >1E+OFFSET       SPECIAL CURSOR CHARACTER
007F            355  EDGECH  EQU  >1F+OFFSET       SPECIAL SCREEN EDGE CHARACTER
```

```
357  ********************************************************************
358  *       FLAGS IN TEXT EDITTING:
359  *       @FLAG  BIT  RESET                    SET
360  *                0   AUTONUM OFF            *AUTONUM ON
361  *                4   UNTRACE IS ON          *TRACE IS ON
362  *                5   PRESCAN OR RUN MODE    *EDIT MODE
363  ********************************************************************
```

```
                    365  ********************************************************************
                    366  *                          GROM HEADER
                    367  ********************************************************************
                    368          GROM 3
                    369          ORG  0
6000 AA             370          DATA >AA                 GROM ID
6001 01             371          DATA 1                   VERSION
6002 01             372          DATA 1                   NUMBER OF USER PROGRAMS
6003 00             373          DATA 0                   RESERVED
6004 0000           374          DATA #0
6006 633B           375          DATA #USER               USER PROGRAM HEADER POINTER
6008 0000           376          DATA #0                  NO DSRS
600A A026           377          DATA #LINK1              LINK TO SUBPROGRAMS
600C 000000         378          DATA #0,#0               RESERVED
600F 00
```

```
                380  ************************************************************
                381  *               Branch table for routines in EDIT
                382  ************************************************************
6010 454F       383          BR   AUTON
6012 43CC       384          BR   TOPL15
6014 4978       385          BR   INITPG
6016 4A02       386          BR   SPRINT            Initialize sprites.
6018 4018       387          BR   $                 WAS ILL1
601A 43C9       388          BR   TOPL10
601C 49DA       389          BR   CHRTAB
601E 4481       390          BR   S$RUN
6020 4020       391          BR   $                 WAS GETLNB
6022 4991       392          BR   KILSYM
6024 4024       393          BR   $                 WAS CRUNCH
6026 48FA       394          BR   GETNB
6028 48FF       395          BR   GETNB2
602A 4905       396          BR   GETCHR
602C 491F       397          BR   GETLN
602E 455A       398          BR   AUTO1
6030 63C2       399          DATA #TOPL02
6032 4784       400          BR   EDITLN
6034 48C5       401          BR   GRSUB1            Read from ERAM(use GREAD)/VDP
6036 48DD       402          BR   GWSUB             Write a few bytes to ERAM/VDP
```

```
                    404  *              Error and system messages
                    405  *
                    406          BASE 0,0,>300,>300,0,0,>60
6038 A9AE80         408  MSGERR DATA :IN ERROR:
6040 07B2A5         409  MSGFST DATA 7,:READY *:
6048 0AA2B2         410  MSGBRK DATA 10,:BREAKPOINT:
6053 B4B2B9         411  MSGTA  DATA :TRY AGAIN:
605C 8A80B7         412  MSGWRN DATA :* WARNING:
6065 10AEB5         413  MSG10  DATA 16,:NUMERIC OVERFLOW:
6076 0CB3B9         414  MSG14  DATA 12,:SYNTAX ERROR:
6083 18A9AC         415  MSG16  DATA 24,:ILLEGAL AFTER SUBPROGRAM:
609C 10B5AE         416  MSG17  DATA 16,:UNMATCHED QUOTES:
60AD 0DAEA1         417  MSG19  DATA 13,:NAME TOO LONG:
60BB 16B3B4         418  MSG24  DATA 22,:STRING-NUMBER MISMATCH:
60D2 11AFB0         419  MSG25  DATA 17,:OPTION BASE ERROR:
60E4 14A9AD         420  MSG28  DATA 20,:IMPROPERLY USED NAME:
60F9 16B5AE         421  MSG34  DATA 22,:UNRECOGNIZED CHARACTER:
6110 0BA9AD         422  MSG36  DATA 11,:IMAGE ERROR:
611C 0BADA5         423  MSG39  DATA 11,:MEMORY FULL:
6128 0EB3B4         424  MSG40  DATA 14,:STACK OVERFLOW:
6137 10AEA5         425  MSG43  DATA 16,:NEXT WITHOUT FOR:
6148 10A6AF         426  MSG44  DATA 16,:FOR-NEXT NESTING:
6159 15ADB5         427  MSG47  DATA 21,:MUST BE IN SUBPROGRAM:
616F 19B2A5         428  MSG48  DATA 25,:RECURSIVE SUBPROGRAM CALL:
6189 0EADA9         429  MSG49  DATA 14,:MISSING SUBEND:
6198 14B2A5         430  MSG51  DATA 20,:RETURN WITHOUT GOSUB:
61AD 10B3B4         431  MSG54  DATA 16,:STRING TRUNCATED:
61BE 0DA2A1         432  MSG57  DATA 13,:BAD SUBSCRIPT:
61CC 0EACA9         433  MSG60  DATA 14,:LINE NOT FOUND:
61DB 0FA2A1         434  MSG61  DATA 15,:BAD LINE NUMBER:
                    435  *MSG62 *** SEE THE LINE AFTER MSG135
61EB 0EA3A1         436  MSG67  DATA 14,:CAN'T CONTINUE:
61FA 1AA3AF         437  MSG69  DATA 26,:COMMAND ILLEGAL IN PROGRAM:
6215 17AFAE         438  MSG70  DATA 23,:ONLY LEGAL IN A PROGRAM:
622D 0CA2A1         439  MSG74  DATA 12,:BAD ARGUMENT:
623A 12AEAF         440  MSG78  DATA 18,:NO PROGRAM PRESENT:
624D 09A2A1         441  MSG79  DATA 9,:BAD VALUE:
6257 17A9AE         442  MSG81  DATA 23,:INCORRECT ARGUMENT LIST:
626F 0BA9AE         443  MSG83  DATA 11,:INPUT ERROR:
627B 0AA4A1         444  MSG84  DATA 10,:DATA ERROR:
6286 14B0B2         445  MSG97  DATA 20,:PROTECTION VIOLATION:
629B 0AA6A9         446  MSG109 DATA 10,:FILE ERROR:
62A6 09A98F         447  MSG130 DATA 9,:I/O ERROR:
62B0 14B3B5         448  MSG135 DATA 20,:SUBPROGRAM NOT FOUND:
62C5 0DACA9         449  MSG62  DATA 13,:LINE TOO LONG:
                    450  *
                    451  *       OTHER MESSAGES
                    452  *
62D3 A2B9B4         453  MSGFRE DATA :BYTES FREE:
62DD A2B9B4         454  MSGSFR DATA :BYTES OF STACK FREE:
62F0 A2B9B4         455  MSGGFR DATA :BYTES OF PROGRAM:
6300 B3B0A1         456  MSGGF1 DATA :SPACE FREE:
630A B5A4A6         457  MSGCIS DATA :UDF REFS ITSELF:
6319 A3A1AC         458  MSGCF  DATA :CALLED FROM:
                    459
```

```
6324 16B3B0    460  MSG56  DATA 22,:SPEECH STRING TOO LONG:
               462         BASE   0,0,>300,>300,0,0,0
               463
               464  *
633B 000063    465  USER   DATA #0,#TOPLEV,17,:TI EXTENDED BASIC:
633E 721154
6341 492045
6344 585445
6347 4E4445
634A 442042
634D 415349
6350 43
               466
6351 094453    467  DSCLOD DATA 9,:DSK1.LOAD:,0
6354 4B312E
6357 4C4F41
635A 4400
               468  SPCCHR
635C 7E4242    469         DATA >7E,>42,>42,>42,>42,>42,>42,>7E  CURSOR CHAR.
635F 424242
6362 427E
6364 000000    470         DATA 0,0,0,0,0,0,0,0     SCREEN EDGE CHAR.
6367 000000
636A 0000
               471
636C E00020    472  VDPREG DATA >E0,>00,>20,>00,>06,>00
636F 000600
```

```
                    474  ********************************************************************
                    475  *
                    476  *                    START OF BASIC INTERPRETER
                    477  *
                    478  ********************************************************************
6372 86A370         479  TOPLEV CLR  RAM(>370)           Initialize temp area
6375 35004D         480         MOVE 77 FROM RAM(>370) TO RAM(>371)
6378 A371A3
637B 70
637C BFA38C         481         DST  NLNADD,RAM(BUFSRT) Initialize edit-buffer start
637F 02E2
6381 BFA38E         482         DST  NLNADD,RAM(BUFEND) Initialize edit-buffer end
6384 02E2
6386 310002         483         MOVE 2 FROM ROM(#ATN$$) TO @INTRIN  Get addr of ATN$
6389 380032
638C B2381F         484         AND >1F,@INTRIN         Throw away the BR op code
638F A33800         485         DADD >5B,@INTRIN        Address of polynomial consts
6392 5B
6393 BEA371         486         ST   >FF,RAM(LODFLG)    Indicate try auto-boot
6396 FF
6397 BE7388         487  S$NEW   ST   RSTK,@SUBSTK      Load base of subroutine stack
639A 0669CC         488         CALL CHRTA2             Load character table
639D 8645           489         CLR  @FLAG              Initialize flag byte
639F 8746           490         DCLR @BUFLEV            Initialize crunch buffer leve
63A1 068012         491         CALL CLSALL             Close all open files
63A4 8634           492         CLR  @DATA              Initialize READ/DATA pointer
63A6 BF6E09         493         DST  VRAMVS,@VSPTR      Initialize base of value stac
63A9 58
63AA BD246E         494         DST  @VSPTR,@STVSPT     Save in permanent base
63AD BDA388         495         DST  @VSPTR,RAM(SAVEVP)
63B0 6E
63B1 066978         496         CALL INITPG             Initialize program & s.t.
63B4 06A018         497         CALL INTRND             Initialize random number
63B7 8EA371         498         $IF  RAM(LODFLG) .NE. 0 THEN  If need auto-boot
63BA 63C2
63BC 86A371         499           CLR  RAM(LODFLG)      Won't ever need to do again
63BF 066474         500           CALL AUTOLD           Attempt an auto-boot          1
                    501         $END I
                    502  *      Label TOPL02 is used by auto-boot in detection of
                    503  *      errors.
                    504  ERRRDY
63C2 066A84         505  TOPL02 CALL ERR$$              Say READY
63C5 00             506         DATA 0                  returns to TOPL15
                    507
63C6 066978         508  TOPL05 CALL INITPG             Initialize program space
                    509
63C9 066991         510  TOPL10 CALL KILSYM             Kill the symbol table
                    511
63CC B245F7         512  TOPL15 RB   @FLAG,3            If error in UDF execution
                    513
63CF BE7388         514  TOPL20 ST   RSTK,@SUBSTK       Initialize subroutine stack
                    515  *
63D2 BF2002         516  TOPL25 DST  NLNADD,@VARW       Screen addr =lower left corne
63D5 E2
63D6 868089         517         CLR  @RAMFLG            Clear the RAMFLG
```

```
63D9 8644      518          CLR  @PRGFLG            Make sure not in program mode
               519
               520  *       Check for auto-num mode
               521
63DB DA4501    522          $IF .BIT(NUMBIT) @FLAG .EQ. 1 THEN If auto-num on
63DE 6407
63E0 A1140E    523            DADD @CURINC,@CURLIN Generate new line number
63E3 D21400    524            $IF @CURLIN .LT. 0 THEN     >32767?
63E6 63EE
63E8 B245FE    525              RB   @FLAG,NUMBIT  If out of range->exit auto-nu
63EB 056407    526              B    TOPL35        Merge in below
               527  *           Must be long branch!!
               528            $END IF
               529
63EE D53032    530  TOPL30     $IF @STLN .DNE. @ENLN THEN Line might exist
63F1 63FB
63F3 BD4A14    531              DST  @CURLIN,@FAC  Ready for program search
63F6 0F7E      532              XML  SPEED
63F8 03        533              DATA SEETWO        Search for existence of line
63F9 66B7      534              BS   EDT$00        COND set = line found
               535            $END IF
63FB 0F83      536            XML  SCROLL          Scroll to the next line
63FD BD5E14    537            DST  @CURLIN,@ARG2   New line #
6400 066A7C    538            CALL DISO            Display the line number
6403 9120      539            DINC @VARW           Followed by a space
6405 4409      540          $SELSE
6407 0F83      541  TOPL35    XML  SCROLL          Scroll the screen
               542          $END IF
6409 BEA2E1    543          ST   :>:+>60,RAM(NLNADD-1)  Display the prompt c'
640C 9E
640D 066A76    544          CALL READLN          Read in a line
6410 06674B    545          CALL SAVLIN          Save input lin for recall
               546
               547  *             CRUNCH the input line
               548
6413 8622      549          CLR  @ERRCOD         Assume no-error return
6415 BF0A08    550          DST  CRNBUF,@RAMPTR  Initialize crunch pointer
6418 20
6419 0F7F      551          XML  CRUNCH          CRUNCH the input line
641B 00        552          DATA 0               Normal crunch mode
641C 8A23      553  TOPL42  CASE @ERRCOD+1
641E 442E      554          BR   TOPL45          No error detected
6420 4A25      555          BR   ERRSYN          *SYNTAX ERROR
6422 4A39      556          BR   ERRBLN          *BAD LINE NUMBER
6424 4A3D      557          BR   ERRLTL          *LINE TOO LONG
6426 4A2F      558          BR   ERRNTL          *NAME TOO LONG
6428 4A29      559          BR   ERRNQS          *UNMATCHED QUOTES
642A 4A43      560          BR   ERRCIP          *COMMAND ILLEGAL IN PROGRAM
642C 4A4D      561          BR   ERRIVN          *UNRECOGNIZED CHARACTER
               562
642E 8F4A64    563  TOPL45  $IF @FAC .DNE. 0 THEN   Line # present                1
6431 48
6432 DA4501    564            $IF .BIT(NUMBIT) @FLAG .EQ. 0 THEN   Not AUTONUM
6435 4441
6437 D6750D    565              $IF @RKEY .NE. CHRTN THEN   Must be up or do
```

```
643A 6441
643C D64201    566              $IF @CHAT .EQ. 1 GOTO EDT$$0 Start EDIT mode
643F 66AD
               567                $END IF
               568              $END IF
6441 066784    569              CALL EDITLN          EDIT the line into the progra
6444 63D2      570              BS   TOPL25          If didn't change the line
6446 43C9      571              BR   TOPL10
               572  *           Jump always
               573            $END IF
6448 D64201    574            $IF @CHAT .EQ. 1 GOTO TOPL25 If blank line - ignore
644B 63D2
644D D6A820    575            $IF RAM(CRNBUF) .EQ. SIZE$ GOTO S$SIZE
6450 EB65A7
6453 C6A820    576            $IF RAM(CRNBUF) .H. MERGE$ GOTO S$RUN4 If imperative
6456 086500
6459 BF2C08    577            DST  CRNBUF+1,@PGMPTR  Anticipate usage of PGMCHR
645C 21
645D 0F79      578            XML  PGMCHR            Prepare CHAT for OLD and SAVE
               579  *
645F 8AA820    580            CASE RAM(CRNBUF)       Select the keyword
6462 4397      581            BR   S$NEW           'NEW'                  0
6464 4503      582            BR   S$CONT          'CONTINUE'             1
6466 459C      583            BR   S$LIST          'LIST'                 2
6468 4598      584            BR   S$BYE           'BYE'                  3
646A 4543      585            BR   S$NUM           'NUMBER'               4
646C 4595      586            BR   S$OLD           'OLD'                  5
646E 4603      587            BR   S$RES           'RESEQUENCE'           6
6470 458D      588            BR   S$SAVE          'SAVE'                 7
6472 45A4      589            BR   S$MERG          'MERGE'                8
```

```
                    591  *       AUTO-BOOT - attempt a 'RUN "DSK1.LOAD"'
6474 31000B         592  AUTOLD  MOVE 11 FROM ROM(#DSCLOD) TO RAM(CRNBUF)
6477 A82063
647A 51
647B BF2C08         593          DST CRNBUF,@PGMPTR    DSK1.LOAD is in crunch buffer
647E 20
647F 4486           594          BR S$RUNL             Go to the RUN "NAME" code
                    595
                    596
                    597  ****** RUN ***********
                    598
6481 D642C7         599  S$RUN   $IF @CHAT .EQ. STRIN$ THEN Ready for 'RUN "NAME"---'
6484 44A5
6486 BD582C         600  S$RUNL    DST @PGMPTR,@FAC14   Save pointer to name
6489 0F79           601            XML PGMCHR           Get the length of the string
648B BC5742         602            ST @CHAT,@FAC13      Put it in FAC13
648E 8656           603            CLR @FAC12           Make it a double byte
6490 A12C56         604            DADD @FAC12,@PGMPTR  Skip the string
6493 0F79           605            XML PGMCHR           To see there is ln.no. ahead
6495 068024         606            CALL CHKEND          Only RUN "NAME" ?
6498 4A25           607            BR   ERRSYN          No - junk on end so error
649A BE42C7         608            ST STRIN$,@CHAT      Prepare for LOAD routine
649D BD2C58         609            DST @FAC14,@PGMPTR   Restore the saved PGMPTR
64A0 068026         610            CALL OLD1            Load the program
64A3 44C0           611            BR S$RUNO            Go ahead from here
                    612          $END IF                No RUN NAME : just run the
                    613  *                               current program in memory
64A5 D642C9         614          $IF @CHAT .EQ. LN$ THEN Is there a line # after RUN?
64A8 44BB
64AA 0F79           615            XML PGMCHR           Get the line number
64AC BC4A42         616            ST @CHAT,@FAC        Put it in FAC for SEETWO
64AF 0F79           617            XML PGMCHR
64B1 BC4B42         618            ST @CHAT,@FAC1
64B4 0F79           619            XML PGMCHR           Should be EOS now
64B6 068024         620            CALL CHKEND          Is it?
64B9 64D0           621            BS S$RUN2            Yes - Go ahead from here
                    622          $END IF                Just 'RUN'
64BB 068024         623          CALL CHKEND            Should be EOS now
64BE 4A25           624          BR ERRSYN              No-SYNTAX ERROR
64C0 D53032         625  S$RUNO  $IF @STLN .DEQ. @ENLN GOTO ILLST  Refuse w/o program
64C3 64D5
64C5 BDA372         626          DST  @ENLN,RAM(START)  Default to beginning
64C8 32
64C9 A7A372         627          DSUB 3,RAM(START)      Offset into the table
64CC 0003
64CE 44E8           628          BR   S$RUN1            Merge in below
                    629  *       Jump always
64D0 D53032         630  S$RUN2  $IF @STLN .DEQ. @ENLN THEN  Refuse w/o program
64D3 44DF
64D5 0F83           631  ILLST     XML  SCROLL          Scroll the screen for message
64D7 8644           632            CLR @PRGFLG          Prevent line # printing
64D9 066A82         633  WRNNPP    CALL WARN$$
64DC 1D             634            DATA 29              * NO PROGRAM PRESENT
64DD 43CC           635            BR   TOPL15
                    636  *       Condition can never be set since line 0 is prohi  e
```

```
                    637           $END IF
64DF 0F7E           638           XML   SPEED
64E1 03             639           DATA SEETWO                 Find the line in the program
64E2 4A35           640           BR    ERRLNF                * LINE NOT FOUND
64E4 BDA372         641           DST   @EXTRAM,RAM(START)    Program run starts here
64E7 2E
64E8 8644           642  S$RUN1   CLR @PRGFLG                 No line #s if error in CLSALL
64EA 068012         643           CALL CLSALL                 Close any open files
64ED 9244           644           DEC   @PRGFLG               Put it back in execution
64EF BC8089         645           ST   @RAMTOP+1,@RAMFLG      Set/reset RAMFLG flag -- when
64F2 8085
                    646  *                                     in program mode & ERAM exist
64F4 87A386         647           DCLR RAM(SEXTRM)            Disallow CONTINUE after RUN
64F7 87A38A         648           DCLR RAM(ERRLN)             Reset ERR handling to default
64FA 066991         649           CALL KILSYM                 Set the stack empty
64FD BE7388         650           ST    RSTK,@SUBSTK          Re-initialize subr stack
6500 056A70         651  S$RUN4   B     PRESCN
```

```
                    653  ****** CONTINUE *******
                    654
                    655  S$CONT
6503 0668FA         656          CALL GETNB                 Check for END-OF-LINE
6506 4A23           657          BR   ERRSY1                Junk on end of command
6508 8FA386         658          $IF RAM(SEXTRM) .DNE. 0 THEN If can continue
650B 653F
650D 0F83           659            XML  SCROLL
650F BD2EA3         660            DST  RAM(SEXTRM),@EXTRAM Copy old line table ptr
6512 86
6513 BD2CA3         661            DST  RAM(SPGMPT),@PGMPTR Copy old text pointer
6516 82
6517 BD46A3         662            DST  RAM(SBUFLV),@BUFLEV Copy old buffer level
651A 84
651B BD48A3         663            DST  RAM(SLSUBP),@LSUBP Copy last subprog on stack
651E 96
651F B445A3         664            OR   RAM(SFLAG),@FLAG Restore on-warn/break bits
6522 98
6523 C56EA3         665            $WHILE @VSPTR .DH. RAM(SAVEVP) While extra on stac
6526 88452D
6529 0F78           666              XML VPOP               Pop them off
652B 4523           667            $SEND WHILE
652D BE44FF         668            ST   -1,@PRGFLG        Indicate program mode
6530 BC8089         669            ST @RAMTOP+1,@RAMFLG Set/reset RAMFLG flag -- when
6533 8085
                    670  *                                 in program mode & ERAM exists
6535 87A386         671            DCLR RAM(SEXTRM)       Prevent unauthorized CONTINUE
6538 BFA388         672            DST  VRAMVS,RAM(SAVEVP) Init for program completio
653B 0958
653D 0F81           673            XML  CONTIN            Resume normal execution
                    674          $END IF
653F 066A84         675  ERRCC   CALL ERR$$              Indicate error
6542 19             676          DATA 25                "* CAN'T CONTINUE"
```

```
                    678  ****** NUMBER ********
                    679
6543 06654F         680  S$NUM  CALL AUTON              Get start line # and incremer
6546 B64501         681         SB   @FLAG,NUMBIT       Set AUTONUM bit for future
6549 BF2002         682         DST  NLNADD,@VARW       Initialize screen address
654C E2
654D 43EE           683         BR   TOPL30             Jump back into it
                    684  *      Jump always
                    685
                    686  ********************************************************
                    687  *      AUTON - scans the NUM, LIST and RES commands for
                    688  *      line numbers.  Leaves 1st line number in CURLIN
                    689  *      and 2nd line number in CURINC.
                    690  *      AUTON is entry point for NUM to default to 100,10.
                    691  *      AUTO1 is entry point for LIST.
                    692  ********************************************************
654F BF1400         693  AUTON  DST  100,@CURLIN         Default start
6552 64
6553 BF0E00         694         DST  10,@CURINC         Default increment
6556 0A
6557 BE082C         695         ST   COMMA,@VARC        Comma is the separator
655A 9320           696  AUTO1  DDEC @VARW              Don't miss the first characte
655C 0668FA         697         CALL GETNB              Get 1st char after keyword
655F 658C           698         BS   AUTO2              If end of line
6561 06691F         699         CALL GETLN              Try to get a line number
6564 8E0C65         700         $IF  @BYTE .NE. 0 THEN  If digits gotten
6567 6B
6568 BD144A         701           DST  @FAC,@CURLIN     Set initial
                    702         $END IF
656B 0668FF         703         CALL GETNB2             Allow spaces before sep
656E C5202A         704         $IF  @VARW .DH. @VARA GOTO AUTO2  Check end of line
6571 658C
6573 D44208         705         $IF  @CHAT .NE. @VARC GOTO ERRSY1 If not correct sep
6576 4A23
6578 0668FA         706         CALL GETNB              Get char after separator
657B 658C           707         BS   AUTO2              If end of line
657D 06691F         708         CALL GETLN              Try to get 2nd number
6580 8E0C65         709         $IF  @BYTE .NE. 0 THEN  If digits gotten
6583 87
6584 BD0E4A         710           DST  @FAC,@CURINC     Save the increment
                    711         $END IF
6587 0668FF         712         CALL GETNB2             Check EOL
658A 4A23           713         BR   ERRSY1             NOT EOL: SYNTAX ERROR
658C 00             714  AUTO2  RTN
```

```
                    716  ****** SAVE ********
                    717
658D D53032         718  S$SAVE $IF  @STLN .DEQ. @ENLN GOTO ILLST   If no progra
6590 64D5
6592 058014         719          B     SAVE
                    720
                    721  ****** OLD ********
                    722
6595 058016         723  S$OLD  B     OLD
                    724
                    725  ****** BYE **********
                    726
6598 068012         727  S$BYE  CALL CLSALL              Properly close all files
659B 0B             728         EXIT                    Return to MONITOR
                    729
                    730  ****** LIST ********
                    731
659C D53032         732  S$LIST $IF @STLN .DEQ. @ENLN GOTO ILLST Refuse LIST w/o pro
659F 64D5
65A1 058018         733         B     LIST              LIST the program
                    734  ****** MERGE *******
                    735
65A4 058028         736  S$MERG B    MERGE
                    737
```

```
                    739  ****** SIZE ******
                    740
65A7 8EA821         741  S$SIZE $IF RAM(CRNBUF+1) .NE. 0 GOTO ERRSYN Must have EOL
65AA 4A25
65AC 0F70           742         XML   COMPCT              Garbage collect to free space
65AE BD5E1A         743         DST   @STREND,@ARG2       Get end of string space
65B1 A55E6E         744         DSUB  @VSPTR,@ARG2        Subtract stack pointer
65B4 A75E00         745         DSUB  63,@ARG2            Require 64-byte buffer
65B7 3F
65B8 0A65BD         746         $IF   .NOT. .GT. THEN     If less than 64 bytes left
65BB 875E           747           DCLR  @ARG2             Then indicate zero
                    748         $END IF
65BD 0F83           749         XML   SCROLL              Scroll the screen
65BF BF2002         750         DST   NLNADD+2,@VARW      Begin a new line
65C2 E4
65C3 066A7C         751         CALL DISO                 Display the number
65C6 8E8084         752         $IF   @RAMTOP .EQ. 0 THEN  If no ERAM present
65C9 45D5
65CB 31000A         753           MOVE 10 FROM ROM(#MSGFRE) TO RAM(1(VARW))
65CE E00120
65D1 62D3
65D3 45FF           754         $ELSE                     If ERAM present
65D5 310013         755           MOVE 19 FROM ROM(#MSGSFR) TO RAM(1(VARW))
65D8 E00120
65DB 62DD
65DD 0F83           756           XML   SCROLL            Scroll the screen
65DF BF2002         757           DST NLNADD+2,@VARW      Beginning of line
65E2 E4
65E3 BD5E80         758           DST   @RAMFRE,@ARG2     Calc space in ERAM
65E6 86
65E7 A75EA0         759           DSUB CPUBAS-1,@ARG2     Subtract base
65EA 3F
65EB 066A7C         760           CALL DISO               Display the number
65EE 310010         761           MOVE 16 FROM ROM(#MSGGFR) TO RAM(1(VARW))
65F1 E00120
65F4 62F0
65F6 0F83           762           XML   SCROLL
65F8 31000A         763           MOVE 10 FROM ROM(#MSGGF1) TO RAM(NLNADD+4)
65FB A2E663
65FE 00
                    764         $END IF
65FF 0F83           765         XML   SCROLL              Scroll the screen
6601 43CC           766         BR    TOPL15              Return to top-level
```

```
                    768  **** RESEQUENCE ****
                    769
6603 D53032         770  S$RES  $IF @STLN .DEQ. @ENLN GOTO ILLST  If no program
6606 64D5
6608 06654F         771         CALL AUTON                Get start line & increment
660B BD4A32         772         DST  @ENLN,@FAC           Compute # of increments req
660E A54A30         773         DSUB @STLN,@FAC           Actual number of lines - 1
6611 E74A00         774         DSRL @FAC,2               Also takes care of this ^^^
6614 02
6615 A94A0E         775         DMUL @CURINC,@FAC         Compute space taken by incr
6618 8F4A4A         776         $IF @FAC .DNE. 0 GOTO ERRBLN Bad line number
661B 39
661C A1144C         777         DADD @FAC2,@CURLIN        Compute highest address used
661F 0C6A39         778         $IF .CARRY. GOTO ERRBLN Watch out for overflow
6622 C6147F         779         $IF @CURLIN .H. >7F GOTO ERRBLN Overflow is > 32767
6625 6A39
6627 BC8089         780         ST @RAMTOP+1,@RAMFLG      Set/reset RAMFLG to use PGMCH
662A 8085
662C 8660           781         CLR  @ARG4                To be used for double add
662E BD2C32         782         DST  @ENLN,@PGMPTR        Start at end of pro
6631 BD0070         783         DST  @>70,@VARO           Assume VDP-top
6634 8E8089         784         $IF @RAMFLG .NE. 0 THEN   But if ERAM exists
6637 663D
6639 BD0080         785           DST @RAMTOP,@VARO       Top for ERAM
663C 84
                    786         $END IF
663D 952C           787         DINCT @PGMPTR             Skip EOL and count
                    788         $REPEAT
663F 0F79           789           XML  PGMCHR             VDP RAM or ERAM
6641 D642C7         790           $IF @CHAT .EQ. STRIN$ GOTO SEQ$2 Skip strings
6644 664B
6646 D642C8         791           $IF @CHAT .EQ. NUM$ THEN If numeric
6649 4655
664B 0F79           792  SEQ$2      XML PGMCHR            Get next token (count)
664D BC6142         793             ST   @CHAT,@ARG5      For double add
6650 A12C60         794             DADD @ARG4,@PGMPTR    Up to end of string
6653 4688           795           $SELSE
6655 D642C9         796             $IF @CHAT .EQ. LN$ THEN Check for line #
6658 4688
665A 06802C         797               CALL GRSUB2         Get the ln. # in the text
665D 2C             798               DATA PGMPTR         @PGMPTR:Source addr.on ER/VDP
665E BD5258         799               DST @EEE1,@FAC8     Save it temporary place
6661 BD5E14         800               DST @CURLIN,@ARG2   Set for searching
6664 BD5C30         801               DST @STLN,@ARG      Compare w/entries in table
                    802               $REPEAT
6667 06802E         803                 CALL GRSUB3       Read the ln. # from ERAM(use
                    804  *                    GREAD1) or VDP, reset possible bkpt to
666A 5C             805                 DATA ARG          @ARG:Source addr. on ERAM/VDP
666B D55258         806                 $IF @FAC8 .DEQ.   @EEE1 GOTO SEQ$3
666E 6680
6670 A55E0E         807                 DSUB @CURINC,@ARG2 Update new line #
6673 A35C00         808                 DADD 4,@ARG       And entry in line # table
6676 04
6677 C55C32         809                 $UNTIL @ARG .DH. @ENLN Stop if end of table
667A 4667
```

```
667C BF5E7F   810                 DST  >7FFF,@ARG2 Default = 32767
667F FF
6680 0668DD   811  SEQ$3          CALL GWSUB       Write a few bytes of data to
              812  *                                ERAM(use GWRITE) or VDP
              813  *              @PGMPTR : Destination addr. on ERAM/VDP
              814  *              @ARG2 : Data
              815  *              2 : Byte count
6683 2C5E02   816                 DATA PGMPTR,ARG2,2
6686 952C     817                 DINCT @PGMPTR    Pass two byte ln. # in text
              818               $END IF
              819             $END IF
6688 8780D6   820           DCLR @>D6               Reset VDP timeout
668B C92C00   821           $UNTIL @PGMPTR .DHE. @VARO   And on end of prograam
668E 463F
              822  *        Now update the line # table itself
6690 BD4A30   823           DST  @STLN,@FAC         Start at beginning of table
6693 BD5C14   824           DST  @CURLIN,@ARG       With start address off course
              825           $REPEAT                 At least one entry
6696 0668DD   826             CALL GWSUB            Write a few bytes of data to
              827  *                                ERAM(use GWRITE) or VDP
6699 4A5C02   828             DATA FAC,ARG,2
              829  *          @FAC : Destination addr. on ERAM/VDP
              830  *          @ARG : Data
              831  *          2 : Byte count
669C A55C0E   832             DSUB @CURINC,@ARG     Compute next line #
669F A34A00   833             DADD 4,@FAC           And next entry in line # tabl
66A2 04
66A3 C54A32   834           $UNTIL @FAC .DH. @ENLN Stop at end of line # table
66A6 4696
66A8 868089   835           CLR  @RAMFLG          Restore the ERAM flag
66AB 43D2     836           BR   TOPL25           Return to top-level
```

```
                    838  *
                    839  *       EDIT routine - display requested line and edit
                    840  *            any changes in the program segment
                    841  *
                    842  *       FAC contains the line number just read in
                    843
66AD D53032         844  EDT$$0 $IF @STLN .DEQ. @ENLN GOTO ILLST If no program
66B0 64D5
                    845         $REPEAT
66B2 0F7E           846           XML  SPEED
66B4 03             847           DATA SEETWO           Try to find the line(# in FAC
66B5 4A35           848           BR   ERRLNF           * LINE NOT FOUND
66B7 BE061D         849  EDT$00   ST   29,@CCPPTR      Force new record on first lin
                    850
                    851  *        The entry in the line number table is in EXTRAM
                    852
66BA BE1760         853           ST   OFFSET,@DSRFLG   Set screen output mode
66BD BE071C         854           ST   28,@RECLEN       Select standard record length
66C0 8704           855           DCLR @PABPTR          I/O to the screen
66C2 8E8084         856           $IF @RAMTOP .NE. 0 THEN  If ERAM
66C5 66CA
66C7 06802A         857             CALL GRMLST         Prepare to list from ERAM
                    858           $END IF
66CA 066A74         859           CALL LLIST            List the line
                    860
                    861  *       VARW contains the position of the first character
                    862  *       following the line number
                    863
66CD C40607         864           $IF @CCPPTR .H. @RECLEN THEN Exactly at end of n
66D0 46DC
66D2 0F83           865             XML  SCROLL         Scroll up one line
66D4 A72000         866             DSUB 32,@VARW       And correct both VARW
66D7 20
66D8 A70800         867             DSUB 28,@CCPADR                 and CCPADR
66DB 1C
                    868           $END IF
66DC BD5E20         869           DST  @VARW,@ARG2      Set cursor at start position
66DF B25FE0         870           AND  >E0,@ARG3        Back to beginning of line
66E2 A35E00         871           DADD 157,@ARG2        Compute theoretically highest
66E5 9D
66E6 BD2A08         872           DST  @CCPADR,@VARA    Use current high pos. as high
66E9 C95E2A         873           $IF @ARG2 .DL. @VARA THEN  If > 4 lines-correct
66EC 66F2
66EE BF5E03         874             DST  >31D,@ARG2     Allow for one more line
66F1 1D
                    875           $END IF
66F2 066A86         876           CALL READL1           Allow the user to make change
66F5 06674B         877           CALL SAVLIN           Save the line for recall
66F8 8E8084         878           $IF  @RAMTOP .NE. 0 THEN  If ERAM exists
66FB 6700
66FD BD2E58         879             DST  @FAC14,@EXTRAM GRMLST saves EXTRAM in FAC 1
                    880           $END IF
6700 DA4501         881           $IF .BIT(NUMBIT) @FLAG .EQ. 1 GOTO EDT$01 Autonum
6703 4723
6705 D6750B         882           $IF @RKEY .EQ. UPARR THEN Ended in UP arrow
```

```
6708 4715
670A A32E00   883             DADD 4,@EXTRAM      Point at next line to list
670D 04
670E C52E32   884             $IF @EXTRAM .DH. @ENLN GOTO EDT$01 Doesn't exist
6711 6723
6713 4728     885             BR   EDT$02         Exists-set up to edit it
              886           $END IF
6715 D6750A   887           $IF @RKEY .EQ. DWNARR THEN Want next program line
6718 472F
671A A72E00   888             DSUB 4,@EXTRAM      Point at next line to list
671D 04
671E C92E30   889             $IF @EXTRAM .DL. @STLN THEN Passed high program
6721 6728
6723 BE750D   890 EDT$01        ST   CHRTN,@RKEY Set no more editing
6726 472F     891             $SELSE
6728 06802E   892 EDT$02        CALL GRSUB3           Read from ERAM, use GREAD
              893 *                              or VDP, Reset possible breakpt too
672B 2E       894               DATA EXTRAM         @EXTRAM:Source addr. on ERAM
672C BD6258   895               DST @EEE1,@ARG6     Save for general use
              896             $END IF
              897           $END IF
672F 8E6047   898           $IF @ARG4 .EQ. 0 THEN If current line was changed
6732 41
6733 BF0A08   899             DST  CRNBUF,@RAMPTR Initialize crunch pointer
6736 20
6737 0F7F     900             XML  CRUNCH         Crunch the input line
6739 00       901             DATA 0              Normal crunch mode
673A 8F2244   902             $IF  @ERRCOD .DNE. 0 GOTO TOPL42 If error
673D 1C
673E 066784   903             CALL EDITLN         And edit into program buffer
              904           $END IF
6741 BD4A62   905           DST  @ARG6,@FAC       Line number for next line
6744 D6750D   906         $UNTIL @RKEY .EQ. CHRTN  Stop on carriage return
6747 46B2
6749 43CC     907         BR   TOPL15             Don't kill the symbol table
              908 *       Jump always
              909
              910 *       Save input line for edit recall
674B B221E0   911 SAVLIN AND  >E0,@VARW+1        Correct in case autonum
674E 9421     912         INCT @VARW+1            Skip edge characters
6750 BD4A2A   913         DST  @VARA,@FAC         Get pointer to end of line
6753 A54A20   914         DSUB @VARW,@FAC         Compute length of line
6756 6768     915         BS   SAVLN5             If zero-length line
6758 C74A00   916         $IF  @FAC .DH. 160 THEN  If line longer than buffer
675B A04762
675E BF4A00   917           DST  160,@FAC         Default to max buffer size
6761 A0
              918         $END IF
6762 344AA8   919         MOVE @FAC FROM RAM(@VARW) TO RAM(RECBUF)  Save line
6765 C0B020
6768 BDA38C   920 SAVLN5 DST  @VARW,RAM(BUFSRT) Save pointer to line start
676B 20
676C BDA38E   921         DST  @VARA,RAM(BUFEND) Save pointer to line end
676F 2A
6770 CBA38C   922         $WHILE RAM(BUFSRT) .DL. >262  If try more than 160
```

```
6773 026267
6776 83
6777 BFA38E   923          DST >2FE,RAM(BUFEND) Update pointer to line en
677A 02FE
677C A3A38C   924          DADD 32,RAM(BUFSRT)  Update pointer for 160 chars
677F 0020
6781 4770     925          $SEND WHILE
6783 00       926          RTN
```

```
                    928  ***************************************************************
                    929  *                  EDIT a line into a program
                    930  *
                    931  *       Must be called with the following set up:
                    932  *            FAC -  line number of line to edited into prog
                    933  *            CHAT - length of line
                    934  *            CRNBUF - crunched line
                    935  ***************************************************************
6784 DA4580         936  EDITLN $IF .BIT7 @FLAG .EQ. 1 GOTO ERRPV Protection violati
6787 4A49
6789 068012         937         CALL CLSALL                 Close any open files
678C 066991         938         CALL KILSYM                 Kill the symbol table
678F 8602           939         CLR  @STPT                  Restore STPT
6791 BC0342         940         ST   @CHAT,@STPT+1
                    941  ***************************************************************
                    942  *  @CHAT=1 ? YES : LINE NO. ONLY ...GO TO DELETE THE LINE
                    943  *             NO : INSERT A NEW LINE OR REPLACE EXISTING LN
                    944  ***************************************************************
6794 D64201         945         $IF @CHAT .NE. 1 GOTO INSREP Something besides line
6797 47E5
6799 DA4501         946         $IF .BIT(NUMBIT) @FLAG .EQ. 1 THEN AUTONUM mode on
679C 67A5
679E B245FE         947           RB   @FLAG,NUMBIT         Reset AUTONUM mode
                    948
67A1 D40000         949  RTNSET   CEQ  @0,@0               Set condition bit
67A4 01             950           RTNC                      And return
                    951         $END IF
                    952
67A5 D53032         953         $IF @STLN .DEQ. @ENLN GOTO RTNSET If no program
67A8 67A1
                    954
                    955  ***************************************************************
                    956  *EDIT$1 Delete the line # from line-#-buffer.
                    957  *       Delete the text from program text area.
                    958  ***************************************************************
67AA 0F7E           959  EDIT$1 XML  SPEED                 Try to find the given line #
67AC 03             960         DATA SEETWO
67AD 47A1           961         BR   RTNSET               Return if not found
67AF 0F87           962         XML  DELREP               Remove it's text from program
                    963
                    964  *      Delete the 4 bytes from the line # table
                    965
67B1 BD062E         966         DST  @EXTRAM,@VARY2       Pointer to line pointer
67B4 9106           967         DINC @VARY2               Advance to last byte of entry
67B6 972E           968         DDECT @EXTRAM             Point to first byte of entry
67B8 BD002E         969         DST  @EXTRAM,@VARO
67BB 9300           970         DDEC @VARO                Last byte of next line entry
                    971  *                                Move down 4 bytes from here
67BD A52E30         972         DSUB @STLN,@EXTRAM        # of bytes to move down
67C0 8F2E67         973         $IF @EXTRAM .DNE. 0 THEN
67C3 C9
67C4 BD5C2E         974           DST  @EXTRAM,@ARG       Put in arg for MVDN
67C7 0F88           975           XML  MVDN               Move one byte at a time
                    976         $END IF
67C9 A33000         977         DADD >04,@STLN            New start addr of line # tab
```

```
67CC 04
67CD 8E8084    978          $IF @RAMTOP .EQ. 0 THEN  IF ERAM not exist
67D0 47D9
67D2 C53070    979            $IF @STLN .DH. @>70 GOTO TOPLO5 Delete the only in
67D5 63C6
67D7 47E3      980          $SELSE                    With ERAM
67D9 8F3063    981            $IF @STLN .DEQ. 0 GOTO TOPLO5
67DC C6
67DD C53080    982            $IF @STLN .DH. @RAMTOP GOTO TOPLO5
67E0 8463C6
               983          $END IF
67E3 4991      984          BR  KILSYM                Kill symbol table and return
               985
               986  ***************************************************************
               987  *       INSERT A NEW LINE OR REPLACE AN EXISTING LINE
               988  ***************************************************************
               989  INSREP
               990  *  BUILD LN # AND LN PTR IN VARY,+1,+2,+3,+4
67E5 BD044A    991          DST  @FAC,@VARY           2 bytes of line #
67E8 BD0632    992          DST  @ENLN,@VARY2         Last addr of line-#-tabl
67EB BD2E32    993          DST  @ENLN,@EXTRAM        Prepare to search the ln # ta
               994
               995  ***************************************************************
               996  * 1ST LN IN MEMORY : EDIT$5--EDIT$6--EDIT$8--DONE
               997  ***************************************************************
67EE D53032    998          $IF @STLN .DEQ. @ENLN GOTO EDIT$5 1st text ?
67F1 683D
               999
              1000  ***************************************************************
              1001  * EDIT$3
              1002  *       COMPARE LN # IN FAC WITH LN # IN THE LN # TABLE
              1003  *       EQUATE : --DELTX--EDIT$8--DONE
              1004  *       HIGHER : HIGHEST LN ? YES : EDIT$6--EDIT$8--DONE
              1005  *                                NO  : BACK TO EDIT$3
              1006  *       LOWER : EDIT$4--EDIT$8--DONE                    *
              1007  ***************************************************************
67F3 912E     1008          DINC @EXTRAM              Get line
67F5 A72E00   1009  EDIT$3 DSUB 4,@EXTRAM            Go to next line in program
67F8 04
67F9 0668C5   1010          CALL GRSUB1               Read from ERAM(use GREAD)/VDP
67FC 2E       1011          DATA EXTRAM               @EXTRAM : Source addr. on ERA
              1012  *                                 or VDP
67FD B2507F   1013          AND >7F,@EEE              Reset possible breakpt
6800 D54A50   1014          $IF @FAC .DEQ. @EEE GOTO DELTX If #s match-delete ol
6803 68A1
6805 BF2A00   1015          DST  4,@VARA              For MEMFUL
6808 04
6809 094813   1016          $IF .H. THEN              New line # is greater
680C D52E30   1017            $IF @EXTRAM .DEQ. @STLN GOTO EDIT$6 Line to be
680F 6841
              1018  *                 inserted got the highest line no. in ln #
              1019  *                 table::add to the end of line-#table
6811 47F5     1020            BR   EDIT$3
              1021          $END IF
              1022
```

```
                1023  ********************************************************************
                1024  * EDIT$4
                1025  *     ALLOCATE SPACE IN LINE # TABLE BY MOVING
                1026  *     PART(ARG=4) OF THE LN # TABLE UP
                1027  ********************************************************************
6813 BF5C00     1028          DST   4,@ARG
6816 04
6817 A15C2E     1029  EDIT$4 DADD @EXTRAM,@ARG
681A A55C30     1030          DSUB @STLN,@ARG          # of bytes in between
681D BD1630     1031          DST   @STLN,@VAR9        Copy old start address of ln
6820 066958     1032          CALL MEMFUL              Check for memory full
6823 A13002     1033          DADD @STPT,@STLN
6826 8E8084     1034          $IF @RAMTOP .EQ. 0 THEN
6829 4833
682B 345CB0     1035            MOVE @ARG FROM RAM(@VAR9) TO RAM(@STLN) Move ln #t
682E 30B016
6831 4838       1036          $ELSE
6833 BD0030     1037            DST @STLN,@VAR0        Destination addr. for MVUP
6836 0F89       1038            XML  MVUP              Move the ln # table up
                1039          $END IF
6838 BD0632     1040          DST   @ENLN,@VARY2       Set up ln ptr in ln # entry
683B 484A       1041          BR    EDIT$8
                1042
                1043  ********************************************************************
                1044  *  EDIT$5
                1045  *  EDIT$6
                1046  *     * SET UP 1ST ENTRY IN LINE # TABLE BY GIVING @VARA=3
                1047  *     * WHEN INSERT THE HIGHEST LINE  :
                1048  *       CONCATENATE LINE # ENTRY TO LN # TABLE
                1049  ********************************************************************
683D BF2A00     1050  EDIT$5 DST   >03,@VARA          Subtract >03 from STLN(>@70)
6840 03
                1051  *                              to get new start addr. of tab
6841 066958     1052  EDIT$6 CALL MEMFUL             Check for memory full
6844 A13002     1053          DADD @STPT,@STLN         Concatenate line # entry to
6847 BD2E30     1054          DST   @STLN,@EXTRAM                          table
                1055
                1056  ********************************************************************
                1057  * EDIT$8
                1058  *      UPDATE ENTRY IN LINE # TABLE, PUT TEXT IN --DONE
                1059  ********************************************************************
                1060  EDIT$8
                1061  ******* Update the 4 bytes entry in ln # table ***********
684A 9106       1062          DINC @VARY2              Point to 1st token(not length
684C A50602     1063            DSUB @STPT,@VARY2      Set up the ln. ptr for VDP
684F 0668DD     1064          CALL GWSUB               Write a few bytes of data to
                1065  *                              ERAM(use GWRITE) or VDP
6852 2E0404     1066          DATA EXTRAM,VARY,4
                1067  *       @EXTRAM : Destination addr. on ERAM/VDP
                1068  *       @VARY : Data
                1069  *       4 : Byte count
                1070  ********************************************************************
                1071  *       Now insert the line's text between the line number
                1072  *       table and the rest of the program's text
                1073  ********************************************************************
```

```
                    1074  ******   GET THE LENGTH OF LN # TAB IN @ARG              *****
6855 BD5C32         1075          DST  @ENLN,@ARG          Highest addr for line # table
6858 A55C30         1076          DSUB @STLN,@ARG          Total length of line # ta
685B 915C           1077          DINC @ARG                Add one for extra offset
                    1078  ******** MOVE THE LN # TABLE ***************
685D BD1630         1079          DST  @STLN,@VAR9         Old start addr of line # tab
6860 9102           1080          DINC @STPT               Point to next free byte in VD
6862 A53002         1081          DSUB @STPT,@STLN         New entry to line # table
6865 A53202         1082          DSUB @STPT,@ENLN
6868 8E8084         1083          $IF @RAMTOP .EQ. 0 THEN  If ERAM not exist
686B 4875
686D 345CB0         1084            MOVE @ARG FROM RAM(@VAR9) TO RAM(@STLN) Move ln #t
6870 30B016
6873 487A           1085          $SELSE
6875 BD0030         1086            DST  @STLN,@VAR0       Set up destination addr. for
6878 0F89           1087            XML MVUP               Move ln # table          MVU
                    1088          $END IF
                    1089  ******** WRITE THE LENGTH BYTE **************
687A 9302           1090          DDEC @STPT               Update length of text
687C 9306           1091          DDEC @VARY2              Point to the length byte
687E 0668DD         1092          CALL GWSUB               Write a few bytes of data
                    1093  *                                to ERAM(use GWRITE) or VDP
6881 060301         1094          DATA VARY2,STPT+1,1
                    1095  *       @VARY2 : Destination addr. on ERAM or VDP
                    1096  *       @(STPT+1) : Data
                    1097  *       1 : Byte count
6884 9106           1098          DINC @VARY2
                    1099  ******** WRITE THE TEXT **********************
6886 8E8084         1100          $IF @RAMTOP .EQ. 0 THEN  If ERAM not exist
6889 4893
688B 3402B0         1101            MOVE @STPT FROM RAM(CRNBUF) TO RAM(@VARY2) Move te
688E 06A820
6891 489F           1102          $SELSE
6893 BF4C08         1103            DST CRNBUF,@AAA        Copy the text from crunch
6896 20                                        VAR9
                    1104  *                    VAR0        buffer(which is on VDP)to ERA
6897 BD5006         1105            DST @VARY2,@BBB
689A BD4E02         1106            DST @STPT,@CCC ARG     @CCC:Byte count
689D 0F8A           1107            XML VGWITE MVUP
                    1108          $END IF
689F 4991           1109          BR   KILSYM              Kill symbol table and return
                    1110  ****************************DONE****************************
```

```
                    1112  *       REPLACE AN EXISTING LINE
                    1113
                    1114  *********     Compute length of old entry  *************
68A1 952E           1115  DELTX   DINCT @EXTRAM             Point to the ln pointer
68A3 0668C5         1116          CALL  GRSUB1              Read from ERAM(use GREAD)/VDP
68A6 2E             1117          DATA  EXTRAM              @EXTRAM:Source addr.on ERAM/V
68A7 972E           1118          DDECT @EXTRAM             Restore back
68A9 9350           1119          DDEC  @EEE                Point to the length byte
68AB 0668C5         1120          CALL  GRSUB1              Read the length from ERAM/VDP
68AE 50             1121          DATA  EEE                 @EEE:Source addr. on ERAM/VDP
68AF BC2B50         1122          ST    @EEE,@VARA+1
68B2 862A           1123          CLR   @VARA               Make a double byte
68B4 832A           1124          DNEG  @VARA               And get length difference
68B6 066958         1125          CALL  MEMFUL              Check for memory full
68B9 A1302A         1126          DADD  @VARA,@STLN         Update STLN
68BC 0F87           1127          XML   DELREP              Remove old text (same line #)
68BE 972E           1128          DDECT @EXTRAM             Correct pointer
                    1129  ********  SET UP THE LN PTR IN LN # ENTRY  **********
68C0 BD0632         1130          DST   @ENLN,@VARY2        Prepare setting up the ln ptr
68C3 484A           1131          BR    EDIT$8              Go update entry in line # tab
                    1132  *                                 and put text in
                    1133  ************************************************************
                    1134  *       SUBROUTINE TO READ 2 BYTES OF DATA FROM VDP OR ERAM
                    1135  *       (USE GREAD)
                    1136  ************************************************************
                    1137  GRSUB1
68C5 884E           1138          FETCH @FFF                Fetch the source addr. on ERA
68C7 BD4C90         1139          DST   *FFF,@DDD           Put it in @DDD
68CA 4E
68CB 8E8084         1140          $IF   @RAMTOP .NE. 0 THEN  If ERAM exists
68CE 68D8
68D0 BF4E00         1141            DST 2,@FFF              @FFF:Byte count
68D3 02
68D4 0F85           1142            XML GREAD               Read data from ERAM
                    1143  *                                 @EEE:Destination addr. on CPU
68D6 48DC           1144          $ELSE                     ERAM not exists
68D8 BD50B0         1145            DST RAM(@DDD),@EEE      Read data from VDP
68DB 4C
                    1146          $END IF
68DC 00             1147          RTN
                    1148  ************************************************************
                    1149  *        SUBROUTINE TO WRITE A FEW BYTES OF DATA TO VDP OR
                    1150  *        ERAM (USE GWRITE)
                    1151  ************************************************************
                    1152  GWSUB
68DD 884C           1153          FETCH @AAA                Fetch the destination addr. o
68DF BD4C90         1154          DST   *AAA,@AAA           ERAM/VDP
68E2 4C
68E3 8851           1155          FETCH @BBB+1              Fetch the source addr. on CPU
                    1156  *                                 where data is stored
68E5 8650           1157          CLR   @BBB                Make a double byte
68E7 884F           1158          FETCH @CCC+1              Fetch the byte count
68E9 864E           1159          CLR   @CCC                Make a double byte
68EB 8E8084         1160          $IF   @RAMTOP .NE. 0 THEN If ERAM exists
68EE 68F3
```

```
68F0 0F86     1161            XML GWRITE          Write the data to ERAM
68F2 00       1162            RTN
              1163          $END IF
68F3 344EB0   1164          MOVE @CCC FROM *BBB+1 TO RAM(@AAA) Write to VDP
68F6 4C9051
68F9 00       1165          RTN
```

```
                    1193  ************************************************************
                    1194  *       GETLN - Gets an line number after a command and put
                    1195  *       it into the FAC.  If the character in CHAT when
                    1196  *       is called is not in the legal numeric range (0-9,
                    1197  *       then GETLN returns with no other action.
                    1198  *
                    1199  *       Called by: AUTON, RUN, EDITLN
                    1200  ************************************************************
691F 874A           1201  GETLN   DCLR @FAC                   Assume no number
6921 860C           1202          CLR  @BYTE                  Assume no digits
6923 A64230         1203  GETLN2  SUB  :0:,@CHAT              ASCII to normal range
6926 CA420A         1204          $IF  @CHAT .L. 10 THEN      If numeric digit
6929 6946
692B AB4A00         1205            DMUL  10,@FAC             Multiply by 10
692E 0A
692F 8F4A49         1206            $IF  @FAC .DNE. 0 GOTO GTLNER Error if overflow
6932 52
6933 BC4B42         1207            ST   @CHAT,@FAC1          Need to add in this digit
6936 A14A4C         1208            DADD @FAC2,@FAC           Add accum into last digit
6939 D34A00         1209            $IF  @FAC .DLT. 0 GOTO GTLNER  Error if overflow
693C 004952
693F 900C           1210            INC  @BYTE                Got another digit
6941 066905         1211            CALL GETCHR               Get the next character
6944 4923           1212            BR   GETLN2               If not EOS
                    1213          $END IF
6946 8E0C69         1214          $IF  @BYTE .NE. 0 THEN      If digits gotten
6949 4E
694A 8F4A69         1215            $IF  @FAC .DEQ. 0 GOTO GTLNER If hit natural zero
694D 52
                    1216          $END IF
694E A24230         1217          ADD  :0:,@CHAT            Put back into ASCII
6951 00             1218          RTN
                    1219
6952 0F83           1220  GTLNER  XML  SCROLL               Scroll the screen
6954 8644           1221          CLR  @PRGFLG              Don't print a line number
6956 4A39           1222          BR   ERRBLN               Issue the error message
                    1223  *                                 * BAD LINE NUMBER
```

```
6958 A12A02  1225  MEMFUL DADD @STPT,@VARA        Total # of bytes to be added
695B A5302A  1226         DSUB @VARA,@STLN       New STLN
695E 8E8084  1227         $IF @RAMTOP .NE. 0 THEN
6961 696A
6963 CB30A0  1228           $IF @STLN .DL. CPUBAS GOTO MEM$1  Not enough memor
6966 404970
6969 00      1229           RTN
             1230         $END IF
696A CB300A  1231         $IF @STLN .DL. VRAMVS+64+256  THEN    Memory full
696D 986977
6970 A1302A  1232  MEM$1    DADD @VARA,@STLN     Back to old start line # tab
6973 066A84  1233           CALL ERR$$
6976 0B      1234           DATA 11              "* MEMORY FULL"
             1235         $END IF
6977 00      1236         RTN
```

```
                1238  *               Initialize program space
                1239  INITPG
6978 868089     1240          CLR   @RAMFLG             Reset RAMFLG
697B 0F8E       1241          XML   GDTECT              Search for ERAM & select RUM
697D BD8086     1242          DST   @RAMTOP,@RAMFRE     Initialize free pointer
6980 8084
6982 BD3070     1243          DST   @>70,@STLN          Assume VDP - initialize STLN
6985 8E8084     1244          $IF @RAMTOP .NE. 0 THEN   If ERAM is present
6988 698E
698A BD3080     1245            DST @RAMTOP,@STLN       Initialize STLN for ERAM
698D 84
                1246          $END IF
698E BD3230     1247          DST   @STLN,@ENLN         Init ENLN based upon STLN
                1248  *               Kill the symbol table
6991 BD4030     1249  KILSYM  DST   @STLN,@FREPTR       Assume VDP and init free ptr
6994 D54070     1250          $IF @FREPTR .DNE. @>70 THEN
6997 699B
6999 9340       1251            DDEC @FREPTR            Back off 1 if program present
                1252          $END IF
699B 8E8084     1253          $IF @RAMTOP .NE. 0 THEN   If ERAM exists
699E 69B1
69A0 BD8086     1254            DST   @STLN,@RAMFRE     Update the @RAMFRE
69A3 30
69A4 D58086     1255            $IF @RAMFRE .DNE. @RAMTOP THEN
69A7 808469
69AA AE
69AB 938086     1256              DDEC @RAMFRE          Back off 1 if program present
                1257            $END IF
69AE BD4070     1258            DST   @>70,@FREPTR      Initialize VDP free point
                1259          $END IF
69B1 873E       1260          DCLR  @SYMTAB             Kill symbol table
69B3 873A       1261          DCLR  @SUBTAB             Kill subprogram table
69B5 BD1840     1262          DST   @FREPTR,@STRSP      Initialize string space
69B8 BD1A18     1263          DST   @STRSP,@STREND
69BB 8643       1264          CLR   @BASE               Reset OPTION BASE to 0
69BD 87A386     1265          DCLR  RAM(SEXTRM)         Disallow CONTINUE
69C0 BF2409     1266          DST   VRAMVS,@STVSPT      Initialize base of value stac
69C3 58
69C4 BD6E24     1267          DST   @STVSPT,@VSPTR      Initialize value stack pointe
69C7 BDA388     1268          DST   @VSPTR,RAM(SAVEVP)  Initialize pointer in VDP to
69CA 6E
69CB 00         1269          RTN
```

```
                    1271  ************************************************************
                    1272  *           Data for the color tables ( starts at >800 )
                    1273  *                        looks like:
                    1274  *       DATA >D0,>00,>00,>00,>00,>00,>00,>00
                    1275  *       DATA >00,>00,>00,>00,>00,>00,>00,>10
                    1276  *       DATA >10,>10,>10,>10,>10,>10,>10,>10
                    1277  *       DATA >10,>10,>10,>10,>10,>10,>10,>10
                    1278  ************************************************************
                    1279  CHRTA2
69CC 0780           1280          ALL  : :+OFFSET          Clear the screen
69CE BF80C0         1281          DST  >3567,@>C0          Initialize GPL random number
69D1 3567
69D3 310010         1282          MOVE 16 FROM ROM(#SPCCHR) TO RAM(>3F0) Cursor char
69D6 A3F063
69D9 5C
69DA BF4A04         1283  CHRTAB  DST  >400,@FAC           Addr of character tables
69DD 00
69DE 060018         1284          CALL CHAR2$              Load the character set
69E1 0407           1285          BACK 7                   Border color = CYAN
69E3 86A800         1286          CLR  RAM(>800)
69E6 35000E         1287          MOVE 14 FROM RAM(>800) TO RAM(>801)
69E9 A801A8
69EC 00
69ED BEA80F         1288          ST   >10,RAM(>80F)       Black fore/transparent back
69F0 10
69F1 350010         1289          MOVE 16 FROM RAM(>80F) TO RAM(>810)
69F4 A810A8
69F7 0F
69F8 066A02         1290          CALL SPRINT
                    1291  *       This part might be moved up later.
                    1292  *       Load special characters here. Don't load before
                    1293  *       hiding all sprites.
                    1294  *
69FB 390006         1295          MOVE 6 FROM ROM(#VDPREG) TO VDP(1)
69FE 01636C
6A01 00             1296          RTN
                    1297  *
                    1298  ***     Initialization of sprites.  Enable 28 sprites.   ***
                    1299  *
                    1300  SPRINT
6A02 86A780         1301          CLR  RAM(>780)          Clear motion of all sprites.
6A05 35006F         1302          MOVE >6F FROM RAM(>780) TO RAM(>781)
6A08 A781A7
6A0B 80
6A0C BE7A1C         1303          ST   28,@MOTION          All in motion.
6A0F BEA370         1304          ST   >D0,RAM(>370)       Sprites 29 to 32 unavailable.
6A12 D0
6A13 BFA300         1305          DST  >C000,RAM(>300)     Hide the first sprite.
6A16 C000
6A18 87A302         1306          DCLR RAM(>302)           Make first sprite transparent
6A1B 35006C         1307          MOVE 108 FROM RAM(>300) TO RAM(>304) Ripple for rest
6A1E A304A3
6A21 00
6A22 00             1308          RTN
                    1309  *
```

```
                1310  *       ERROR messages in this file
                1311  *
                1312  *
6A23 8644       1313  ERRSY1 CLR @PRGFLG            W/o a line number
6A25 066A84     1314  ERRSYN CALL ERR$$
6A28 03         1315         DATA 3                 SYNTAX ERROR
                1316  *
6A29 0F83       1317  ERRNQS XML  SCROLL             Scroll up the screen.
6A2B 066A84     1318         CALL ERR$$             EOL before end of string.
6A2E 05         1319         DATA 5                 * UNMATCHED QUOTES message.
                1320  *
6A2F 8644       1321  ERRNTL CLR @PRGFLG            Don't print a line #
6A31 066A84     1322         CALL ERR$$
6A34 06         1323         DATA 6                 * NAME TOO LONG
                1324  *
6A35 066A84     1325  ERRLNF CALL ERR$$             * LINE NOT FOUND
6A38 16         1326         DATA 22
                1327  *
6A39 066A84     1328  ERRBLN CALL ERR$$            * BAD LINE NUMBER
6A3C 17         1329         DATA 23
                1330  *
6A3D 8644       1331  ERRLTL CLR @PRGFLG            Don't print line number
6A3F 066A84     1332         CALL ERR$$             Issue the error
6A42 18         1333         DATA 24                * LINE TOO LONG
                1334  *
6A43 0F83       1335  ERRCIP XML  SCROLL            Scroll the screen
6A45 066A84     1336         CALL ERR$$             * COMMAND ILLEGAL IN PROG
6A48 1A         1337         DATA 26
                1338  *
6A49 066A84     1339  ERRPV  CALL ERR$$            * PROTECTION VIOLATION
6A4C 27         1340         DATA 39
                1341  *
6A4D 8644       1342  ERRIVN CLR  @PRGFLG           Don't print line number
6A4F 066A84     1343         CALL ERR$$             * UNRECOGNIZED CHARACTER
6A52 28         1344         DATA 40
                1345  *
                1346  *
                1347  *       Other ERROR messages in the program
                1348  *
                1349  * ERRRDY     * READY                    DATA 0
                1350  * ERRMEM     * MEMORY FULL              DATA 11
                1351  * ERRCC      * CAN'T CONTINUE           DATA 25
                1352  * WRNNPP     * NO PORGRAM PRESENT       DATA 29
                1353  *
                1354         END


ERRORS=     0

LENGTH= 2643 (>0A53)


429 SYMBOLS USED
```

```
SYMBOL VALUE  DEF    REFERENCE TABLE

COLON  003A   331
COLON$ 00B5   260
COMMA  002C   351      695
COMMA$ 00B3   258
COMPCT 0070    61      742
CONC$  00B8   263
CONTIN 0081    68      673
COS$   00CD   285
CPUBAS A040    22      759 1228
CRNBUF 0820   184      550  575  576  577  580  592  593  741  899 1101
                       1103
CRNEND 08BE   185
CRUNCH 007F    67      551  900
CSNTMP 0390   188
CURINC 000E   101      523  694  710  775  807  832
CURLIN 0014   102      523  524  531  537  693  701  777  779  800  824
CURSOR 007E   354
DATA   0034   122      492
DATA$  0093   226
DDD    004C   150     1139 1145
DDD1   0054   153
DEF$   0089   216
DELET$ 0099   232
DELREP 0087    72      962 1127
DELTX  68A1  1115     1014
DIGIT$ 00E9   304
DIM$   008A   217
DISO   6A7C    36      538  751  760
DISPL$ 00A2   241
DISPLA 0009     1
DIVI$  00C4   275
DOLLAR 0024   347
DOT    002E   353
DSCLOD 6351   467      592
DSRFLG 0017   104      853
DWNARR 000A   192      887
E      0045   338
EDGECH 007F   355     1184
EDIT$1 67AA   959
EDIT$3 67F5  1009     1020
EDIT$4 6817  1029
EDIT$5 683D  1050      998
EDIT$6 6841  1052     1017
EDIT$8 684A  1060     1041 1131
EDITLN 6784   936      400  569  903
EDT$$0 66AD   844      566
EDT$00 66B7   849      534
EDT$01 6723   890      881  884
EDT$02 6728   892      885
EEE    0050   152     1013 1014 1119 1121 1122 1145
EEE1   0058   155      799  806  895
ELSE$  0081   208
END$   008B   218
ENDSCR 02FE   172
ENLN   0032   121      530  625  626  630  718  732  770  772  782  809
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
			834 844 884 953 992 993 998 1040 1075 1082 1130 1247
EQUAL	003D	334	
EQUAL$	00BE	269	
ERASE$	00EF	310	
ERCODE	007C	87	
ERR$$	6A84	38	505 675 1233 1314 1318 1322 1325 1328 1332 1336 1339 1343
ERRBLN	6A39	1328	556 776 778 779 1222
ERRCC	653F	675	
ERRCIP	6A43	1335	560
ERRCOD	0022	115	549 553 902
ERRIVN	6A4D	1342	561
ERRLN	038A	179	648
ERRLNF	6A35	1325	640 848
ERRLTL	6A3D	1331	557
ERRNQS	6A29	1317	559
ERRNTL	6A2F	1321	558
ERROR$	00A5	244	
ERRPV	6A49	1339	936
ERRRDY	63C2	504	
ERRSY1	6A23	1313	657 705 713
ERRSYN	6A25	1314	555 607 624 741
EXCL	0021	345	
EXEC1	A00C	41	
EXP$	0076	163	
EXP$$	00CE	286	
EXTRAM	002E	119	641 660 879 883 884 888 889 894 966 968 969 972 973 974 993 1008 1009 1011 1017 1029 1054 1066 1115 1117 1118 1128
FAC	004A	135	136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 531 563 616 701 710 772 773 774 775 776 823 828 833 834 905 913 914 916 917 919 991 1014 1201 1205 1206 1208 1209 1215 1283
FAC1	004B	136	618 1207
FAC10	0054	142	
FAC12	0056	143	603 604
FAC13	0057	144	602
FAC14	0058	145	600 609 879
FAC15	0059	146	
FAC2	004C	137	777 1208
FAC4	004E	138	
FAC6	0050	139	
FAC7	0051	140	
FAC8	0052	141	799 806
FFF	004E	151	1138 1139 1141
FFF1	0056	154	
FIGURE	0002	1	
FIXED$	00FA	321	
FLAG	0045	132	489 512 522 525 564 664 681 881 936 946 947
FOR$	008C	219	
FREPTR	0040	128	1249 1250 1251 1258 1262
GDTECT	008E	76	1241

SYMBOL	VALUE	DEF	REFERENCE TABLE
GETCHR	6905	1181	396 1171 1190 1211
GETLN	691F	1201	397 699 708
GETLN2	6923	1203	1212
GETNB	68FA	1170	394 656 697 706
GETNB2	68FF	1173	395 703 712
GO$	0085	212	
GOSUB$	0087	214	
GOTO$	0086	213	
GREAD	0085	70	1142
GREAT	003E	335	
GREAT$	00C0	271	
GRMLST	802A	52	857
GRSUB1	68C5	1137	401 1010 1116 1120
GRSUB2	802C	53	797
GRSUB3	802E	54	803 892
GTLNER	6952	1220	1206 1209 1215
GWRITE	0086	71	1161
GWSUB	68DD	1152	402 811 826 1064 1092
IF$	0084	211	
ILLST	64D5	631	625 718 732 770 844
IMAGE$	00A3	242	
INITPG	6978	1239	385 496 508
INPUT$	0092	225	
INSREP	67E5	989	945
INT$	00CF	287	
INTRIN	0038	124	483 484 485
INTRND	A018	42	497
IOSTRT	003C	126	
KEYBD	0074	81	
KILSYM	6991	1249	392 510 649 938 984 1109
LEN$	00D5	293	
LESS	003C	333	
LESS$	00BF	270	
LET$	008D	220	
LINK1	A026	43	377
LIST	8018	48	733
LIST$	0002	195	
LLIST	6A74	34	859
LN$	00C9	281	614 796
LNBUF	0036	123	
LODFLG	0371	173	486 498 499
LOG$	00D0	288	
LPAR	0028	348	
LPAR$	00B7	262	
LSUBP	0048	134	663
M$EXEC	A000	26	41 42 43
M$FLMG	8000	27	45 46 47 48 49 50 51 52 53 54
M$MON	0010	24	31
M$PSCN	6A70	25	33 34 35 36 37 38 39
MEM$1	6970	1232	1228
MEMFUL	6958	1225	1032 1052 1125
MERGE	8028	51	736
MERGE$	0008	199	576
MINUS	002D	352	
MINUS$	00C2	273	

SYMBOL	VALUE	DSF	REFERENCE TABLE
MOTION	007A	85	1303
MSG10	6065	413	
MSG109	629B	446	
MSG130	62A6	447	
MSG135	62B0	448	
MSG14	6076	414	
MSG16	6083	415	
MSG17	609C	416	
MSG19	60AD	417	
MSG24	60BB	418	
MSG25	60D2	419	
MSG28	60E4	420	
MSG34	60F9	421	
MSG36	6110	422	
MSG39	611C	423	
MSG40	6128	424	
MSG43	6137	425	
MSG44	6148	426	
MSG47	6159	427	
MSG48	616F	428	
MSG49	6189	429	
MSG51	6198	430	
MSG54	61AD	431	
MSG56	6324	460	
MSG57	61BE	432	
MSG60	61CC	433	
MSG61	61DB	434	
MSG62	62C5	449	
MSG67	61EB	436	
MSG69	61FA	437	
MSG70	6215	438	
MSG74	622D	439	
MSG78	623A	440	
MSG79	624D	441	
MSG81	6257	442	
MSG83	626F	443	
MSG84	627B	444	
MSG97	6286	445	
MSGBRK	6048	410	
MSGCF	6319	458	
MSGCIS	630A	457	
MSGERR	6038	408	
MSGFRE	62D3	453	753
MSGFST	6040	409	
MSGGF1	6300	456	763
MSGGFR	62F0	455	761
MSGSFR	62DD	454	755
MSGTA	6053	411	
MSGWRN	605C	412	
MULT$	00C3	274	
MVDN	0088	73	975
MVUP	0089	74	1038 1087
NEXT$	0096	229	
NINE	0039	330	
NLNADD	02E2	171	481 482 516 543 682 750 757 763

SYMBOL	VALUE	DEF	REFERENCE TABLE
NOT$	00BD	268	
NUM$	00C8	280	791
NUMBE$	00FD	324	
NUMBIT	0000	201	522 525 564 681 881 946 947
NUMER$	00E8	303	
OFFSET	0060	194	354 355 853 1185 1280
OLD	8016	47	723
OLD$	0005	196	
OLD1	8026	50	610
ON$	009B	234	
OPEN$	009F	238	
OPENB	005B	340	
OPTIO$	009E	237	
OR$	00BA	265	
OUTPU$	00F7	318	
PABPTR	0004	94	855
PERMA$	00FB	322	
PGMCHR	0079	63	578 601 605 615 617 619 789 792
PGMPTR	002C	118	577 593 600 604 609 661 782 787 794 798 816 817 821
PLUS	002B	350	
PLUS$	00C1	272	
POP	0005	1	
POS$	00D9	297	
PRESCN	6A70	33	651
PRGFLG	0044	131	518 632 642 644 668 1221 1313 1321 1331 1342
PRINT$	009C	235	
QUEST	003F	336	
QUOTE	0022	346	
RAM	0001	1	479 480 480 481 482 486 495 498 499 543 575 576 580 592 626 627 641 647 648 658 660 661 662 663 664 665 671 672 741 753 755 761 763 919 919 920 921 922 923 924 1035 1035 1084 1084 1101 1101 1145 1164 1183 1265 1268 1282 1286 1287 1287 1288 1289 1289 1301 1302 1302 1304 1305 1306 1307 1307
RAMFLG	0089	168	517 645 669 780 784 835 1240
RAMFRE	0086	166	758 1242 1254 1255 1256
RAMPTR	000A	99	550 899
RAMTOP	0084	165	645 669 752 780 785 856 878 978 982 1034 1083 1100 1140 1160 1227 1242 1244 1245 1253 1255
RANDO$	0095	228	
RANDOM	0078	83	
READ$	0097	230	
READL1	6A86	39	876
READLN	6A76	35	544
RECBUF	0BC0	186	919
RECLEN	0007	96	854 864
REM$	009A	233	
RESEQ$	0006	197	
RESTO$	0094	227	
RETUR$	0088	215	
RKEY	0075	82	565 882 887 890 906
RND$	00D7	295	
ROM	000A	1	483 592 753 755 761 763 1282 1295

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

RPAR    0029   349
RPAR$   00B6   261
RSTK    0088   167      487  514  650
RTNSET  67A1   949      953  961 1172 1182
RUN$    00A9   248
S$BYE   6598   727      584
S$CONT  6503   655      582
S$LIST  659C   732      583
S$MERG  65A4   736      589
S$NEW   6397   487      581
S$NUM   6543   680      585
S$OLD   6595   723      586
S$RES   6603   770      587
S$RUN   6481   599      390
S$RUNO  64C0   625      611
S$RUN1  64E8   642      628
S$RUN2  64D0   630      621
S$RUN4  6500   651      576
S$RUNL  6486   600      594
S$SAVE  658D   718      588
S$SIZE  65A7   741      575
SAVE    8014    46      719
SAVE$   0007   198
SAVEVP  0388   178      495  665  672 1268
SAVLIN  674B   911      545  877
SAVLN5  6768   920      915
SBUFLV  0384   176      662
SCROLL  0083    69      536  541  631  659  749  756  762  765  865 1220
                       1317 1335
SEETWO  0003    66      533  639  847  960
SEG$$   00D8   296
SEMIC   003B   332
SEMIC$  00B4   259
SEQ$2   664B   792      790
SEQ$3   6680   811      806
SEQUE$  00F6   317
SEXTRM  0386   177      647  658  660  671 1265
SFLAG   0398   183      664
SGN$$   00D1   289
SIN$    00D2   290
SIZE$   00EB   306      575
SLSUBP  0396   182      663
SMTSRT  001E   113
SPACE   0020   344     1173
SPCCHR  635C   468     1282
SPEED   007E    64      532  638  846  959
SPGMPT  0382   175      661
SPRINT  6A02  1300      386 1290
SPRITE  0006     1
SQR$    00D3   291
SREF    001C   112
SSEP$   0082   209
STACK   0072    79
START   0372   174      626  627  641
STEP$   00B2   257
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
STLN	0030	120	530 625 630 718 732 770 773 801 823 844 889 953 972 977 979 981 982 998 1017 1030 1031 1033 1035 1037 1053 1054 1076 1079 1081 1084 1086 1126 1226 1228 1231 1232 1243 1245 1247 1249 1254
STOP$	0098	231	
STPT	0002	91	939 940 1033 1053 1063 1080 1081 1082 1090 1094 1101 1106 1225
STR$$	00DB	299	
STREND	001A	111	743 1263
STRIN$	00C7	278	599 608 790
STRSP	0018	110	1262 1263
STVSPT	0024	116	494 1266 1267
SUB$	00A1	240	
SUBND$	00A8	247	
SUBSTK	0073	80	487 514 650
SUBTAB	003A	125	1261
SUBXT$	00A7	246	
SYMTAB	003E	127	1260
SYNCHK	0000	65	
TAB$	00FC	323	
TABLE	000B	1	
TAN$	00D4	292	
THEN$	00B0	255	
TIMER	0079	84	
TO$	00B1	256	
TOP	0003	1	
TOPL02	63C2	505	399
TOPL05	63C6	508	979 981 982
TOPL10	63C9	510	388 571
TOPL15	63CC	512	384 635 766 907
TOPL20	63CF	514	
TOPL25	63D2	516	570 574 836
TOPL30	63EE	530	683
TOPL35	6407	541	526
TOPL42	641C	553	902
TOPL45	642E	563	554
TOPLEV	6372	479	465
TRACE$	0090	223	
TREM$	0083	210	
UALPH$	00EA	305	
UNBRE$	008F	222	
UNLN	005F	343	
UNQST$	00C8	279	280
UNTRA$	0091	224	
UPARR	0005	191	882
UPDAT$	00F8	319	
USER	633B	465	375
USING$	00ED	308	
VAL	00DA	298	
VALID$	00FE	325	
VAR0	0000	90	783 785 821 969 970 1037 1086
VAR9	0016	103	1031 1035 1079 1084
VARA	002A	117	704 872 873 913 921 1015 1050 1122 1123 1124 1126 1182 1225 1226 1232

SYMBOL	VALUE	DEF	REFERENCE TABLE
VARC	0008	98	695 705
VARW	0020	114	516 539 682 696 704 750 753 755 757 761 866 869 911 912 914 919 920 1182 1183 1186 1189
VARY	0004	92	93 991 1066
VARY2	0006	93	966 967 992 1040 1062 1063 1091 1094 1098 1101 1105 1130
VDP	000C	1	1295
VDPREG	636C	472	1295
VDPSTS	007B	86	
VEL	0007	1	
VGWITE	008A	75	1107
VPOP	0078	62	666
VRAMVS	0958	187	493 672 1231 1266
VSPTR	006E	162	493 494 495 665 744 1267 1268
WARN$	00A6	245	
WARN$$	6A82	37	633
WRNNPP	64D9	633	
XOR$	00BC	267	
XPT	000E	1	
YPT	000D	1	
Z	005A	339	
ZERO	0030	329	