```
                    917 ********************************************************************
                    918 *       EXTENDED STRING PACKAGE                                    *
                    919 *       THE ROUTINES ARE:                                          *
                    920 *         LITSO5 - Move a string literal from the                  *
                    921 *                  program to the string space                     *
                    922 *         INTARG - Checks that an argument is a numeric            *
                    923 *                  and converts it from floating point             *
                    924 *                  to an integer                                   *
                    925 *         PUSSTR - Checks that an argument is a string             *
                    926 *                  and pushes it on the stack                      *
                    927 *         CONCAT - Concatenates 2 strings together                 *
                    928 *         SEG$   - Segments a string                               *
                    929 *         LEN    - Puts the length of a string in the FAC          *
                    930 *         CHR$   - Converts an integer into its ASCII              *
                    931 *                  character                                       *
                    932 *         STR$   - Converts a number into its string               *
                    933 *                  equivalent                                      *
                    934 *         VAL    - Converts a string into its numeric              *
                    935 *                  equivalent                                      *
                    936 *         POS    - Gives the position of one string within         *
                    937 *                  another                                         *
                    938 *         RPT$   - Generates a single string with multiple         *
                    939 *                  copies of the original string                   *
                    940 *                                                                  *
                    941 *       AN ENTRY IN THE FAC LOOKS LIKE:                            *
                    942 * +-----------+-----+----+--------------+----------------+        *
                    943 * |addr of ptr| >65 | xx |addr of string|length of string|        *
                    944 * +-----------+-----+----+--------------+----------------+        *
                    945 * ^           ^     ^    ^              ^                ^        *
                    946 * FAC       FAC2  FAC3  FAC4          FAC6             FAC8        *
                    947 *                                                                  *
                    948 ********************************************************************
                    949 *       Support routine for functions to build FAC entry
A3FB 8650           950 LITSO5 CLR   @FAC6              Need as a double-byte value
A3FD BD0C50         951        DST   @FAC6,@BYTE        LENGTH FOR GETSTR
A400 BC5280         952        ST    @RAMTOP,@FAC8      Copy ERAM flag for later
A403 E4
A404 0F71           953 LITSO7 XML   GETSTR              ALLOCATE STRING SPACE
A406 BF4A00         954 LITSO8 DST   SREF,@FAC           SAVE ADDR OF STRING PTR
A409 1C
A40A BD4E1C         955        DST   @SREF,@FAC4         SAVE ADDR OF STRING
A40D BF4C65         956        DST   >6500,@FAC2         INDICATES A STRING CONSTANT.
A410 0C
                    957 ******COPY STRING INTO STRING SPACE
A411 8F0C64         958 LITSO9 $IF   @BYTE .DNE. 0 THEN If non-null string
A414 2B
A415 8E5244         959          $IF @FAC8 .EQ. 0 THEN If source string in VDP
A418 20
A419 340CB0         960            MOVE @BYTE FROM RAM(@TEMP5) TO RAM(@SREF)
A41C 1CB066
A41F 00             961            RTN
                    962          $END IF                  Else source string in ERAM
A420 BD560C         963          DST   @BYTE,@FFF1    FFF1 : BYTE COUNT
A423 BD581C         964          DST   @SREF,@EEE1    EEE1 : DESTINATION ADDRESS ON
                    965 *                                      VDP
```

```
A426 BD5466    966            DST  @TEMP5,@DDD1  DDD1 : Source address in ERAM  1
A429 0F8D      967            XML  GVWITE        Move data from ERAM to VDP    1
               968          $END IF
A42B 00        969          RTN
               970  *
A42C 8652      971  LITS06  CLR  @FAC8            SET FLAG TO VDP
A42E 4404      972          BR   LITS07           JUMP INTO CODE
               973
               974
               975
               976
               977
               978
               979  ********************************************************
               980  *      PUSSTR - Insures that the entry in the FAC      *
               981  *           is a string and pushes it onto the stack.  *
               982  ********************************************************
A430 D64C65    983  PUSSTR $IF @FAC2 .NE. >65 GOTO ERRSNM
A433 4D57
A435 0F77      984          XML  VPUSH            PUSH THE ARGUMENT
A437 00        985          RTN
```

```
                    987  ****************************************************************
                    988  *                                                              *
                    989  *       CONCAT - CONCATENATES TWO STRINGS TOGETHER              *
                    990  *                                                              *
                    991  *       INPUT: FLOATING POINT ACCUMULATOR ENTRIES              *
                    992  *       OUTPUT: CONCATENATED STRING AND (POSSIBLE)             *
                    993  *               ZEROED BACK-POINTERS FOR THE OLD STRINGS       *
                    994  *       USES: TEMP1,TEMP4 AND TEMP5 AS TEMPORARIES             *
                    995  ****************************************************************
                    996  CONCAT
A438 8623           997          CLR  @ERRCOD+1          CLEAR THE ERROR CODE
A43A 06A430         998          CALL PUSSTR             Push the string & get next tok
A43D 0F74           999          XML  PARSE              GET THE R.H. ARGUMENT
A43F B8            1000          DATA CONC$
A440 D64C65        1001          $IF  @FAC2 .NE. >65 GOTO ERRSNM  If not string - erro
A443 4D57
A445 BD0C50        1002          DST  @FAC6,@BYTE        GET R.H. LENGTH
A448 A10CE0        1003          DADD RAM(6(VSPTR)),@BYTE  ADD IN L.H. LENGTH
A44B 066E
A44D C70C00        1004          $IF  @BYTE .DH. 255 THEN                              1
A450 FF445B
A453 BF0C00        1005            DST  255,@BYTE        TRUNCATE IF TOO LONG          1
A456 FF
A457 066A82        1006  WRNST1    CALL WARN$$           display warning               1
A45A 13            1007            DATA 19               * STRING TRUNCATED message.   1
                   1008          $END IF
A45B BD6B0C        1009          DST  @BYTE,@TEMP6       KEEP LENGTH FOR LATER
A45E 0F77          1010          XML  VPUSH
A460 0F71          1011          XML  GETSTR             ALLOCATE THE RESULT STRING
A462 0F78          1012          XML  VPOP               RETRIEVE R.H.
A464 350008        1013          MOVE 8 FROM @FAC TO @ARG
A467 5C4A
A469 0F78          1014          XML  VPOP               RETREIEVE L.H.
A46B BD664E        1015          DST  @FAC4,@TEMP5       SET PTR TO L.H. ARG(FOR FRESTR
A46E BD0C50        1016          DST  @FAC6,@BYTE        LENGTH OF L.H. ARG.
A471 8652          1017          CLR  @FAC8              FORCE VDP MODE
A473 06A406        1018          CALL LITS08             SET UP FAC&COPY L.H. ARG IN
A476 8F6284        1019          $IF  @ARG6 .DEQ. 0 GOTO CONC06 IF R.H.=0 DON'T COPY
A479 8E
A47A BD641C        1020          DST  @SREF,@TEMP4       GET PTR TO NEW STRING
A47D A16450        1021          DADD @FAC6,@TEMP4       PTR TO WHERE 2nd STRING BEGINS
A480 A56B50        1022          DSUB @FAC6,@TEMP6       LENGTH OF 2nd STRING(POSSIBLY
                   1023  *                               TRUNCATED)
A483 648E          1024          BS   CONC06             If 2nd all truncated out
A485 3468B0        1025          MOVE @TEMP6 FROM RAM(@ARG4) TO RAM(@TEMP4) COPY IN 2
A488 64B060
A48B A15068        1026          DADD @TEMP6,@FAC6       ADD IN LENGTH OF 2nd ARG
                   1027  *       NOTE: FAC6 ALREADY CONTAINED LENGTH OF 1st ARG FROM
                   1028  *             THE PARSE THAT WAS DONE ON IT
                   1029  CONC06
A48E 0F75          1030          XML  CONT               Done.
```

```
                1032  ***************************************************************
                1033  *                                                             *
                1034  *       SEG$(A$,X,Y) - Extracts the designated string from    *
                1035  *            A$.  X specifies the character position within  *
                1036  *            A$ at which the extraction begins.  Y specifies*
                1037  *                                                             *
                1038  *            X or Y is negative an error occurs.  If X=0 an   *
                1039  *            error occurs.  If Y=0 or X > length of A$ then   *
                1040  *            the null string is returned.  If the remaining   *
                1041  *            length in A$ starting at the position specified  *
                1042  *            by X is less than the length specified by Y,     *
                1043  *            then the remainder of A$ starting at position X  *
                1044  *            is returned.                                     *
                1045  *       INPUT - Control is turned over to SEG$ from PARSE.    *
                1046  *            The only requirement is that a SEG$ was          *
                1047  *              encountered.                                   *
                1048  *       OUTPUT - The Floating Point Accumulator is set up     *
                1049  *            with the header for the segmented string.        *
                1050  *       USES - TEMP1(Others in calls to GETSTR and LITSUB)    *
                1051  *                                                             *
                1052  ***************************************************************
A490 06AC2B     1053  SEG$01  CALL  LPAR                 Insure '(', parse and check ,
A493 06A430     1054          CALL  PUSSTR               Push string and get next token
A496 0F7E       1055          XML   SPEED                Get the position
A498 01         1056          DATA  PARCOM                  within the source string
A499 06A975     1057          CALL  INTARG               CHECK & CONVERT ARG TO INTEGER
A49C 8F4A6D     1058          $IF   @FAC .DEQ. 0 GOTO ERRBV  CAN'T HAVE VALUE OF 0
A49F 87
A4A0 0F77       1059          XML   VPUSH                PUSH THE ARG
A4A2 0F74       1060          XML   PARSE                GET EXTRACTION LENGTH
A4A4 B6         1061          DATA  RPAR$
A4A5 0F7E       1062          XML   SPEED                Must have
A4A7 00         1063          DATA  SYNCHK                  ended on
A4A8 B6         1064          DATA  RPAR$                      a right parenthesis
A4A9 06A975     1065          CALL  INTARG               CHECK & CONVERT ARG TO INTEGER
A4AC BD5C4A     1066          DST   @FAC,@ARG            MOVE EXTRACTION LENGTH
A4AF 0F7B       1067          XML   VPOP                 GET POSITION BACK
A4B1 BD5E4A     1068          DST   @FAC,@ARG2           MOVE POSITION
A4B4 0F7B       1069          XML   VPOP                 RETRIEVE SOURCE STRING
A4B6 BD565E     1070          DST   @ARG2,@TEMP1         GET POSITION W/IN STRING
A4B9 C55650     1071          $IF   @TEMP1 .DH. @FAC6 GOTO SEG$08 If pos>len =>null
A4BC 64EF
A4BE A1565C     1072          DADD  @ARG,@TEMP1          COMPUTE END OF SUBSTRING
A4C1 A55650     1073          DSUB  @FAC6,@TEMP1         COMPUTE LENGTH BEYOND END
A4C4 9356       1074          DDEC  @TEMP1                 STRING
A4C6 D35600     1075          DCGE  0,@TEMP1
A4C9 00
A4CA 44D4       1076          BR    SEG$06               FINE IF SUBSTRING IS SHORTER
A4CC BD5C50     1077          DST   @FAC6,@ARG           ELSE, TRUNCATE LENGTH OF SUBST
A4CF A55C5E     1078          DSUB  @ARG2,@ARG           SUBT POS FROM SOURCE LENGTH
A4D2 915C       1079          DINC  @ARG                 INC TO INCLUDE LAST CHAR
A4D4 BD0C5C     1080  SEG$06  DST   @ARG,@BYTE           # OF BYTES NEEDED FOR SUBSTRI
A4D7 0F77       1081          XML   VPUSH                Save source string entry
A4D9 0F71       1082          XML   GETSTR               ALLOCATE RESULT STRING
A4DB 0F7B       1083          XML   VPOP                 Restore source string entry
```

```
A4DD BD664E  1084          DST   @FAC4,@TEMP5      PTR TO SOURCE FOR FRESTR & LIT
A4E0 A1665E  1085          DADD  @ARG2,@TEMP5      PTR TO START OF SUBSTRING
A4E3 9366    1086          DDEC  @TEMP5            Decrement since zero-based
A4E5 BD500C  1087          DST   @BYTE,@FAC6       SET LENGTH OF STRING.
A4E8 8652    1088          CLR   @FAC8             FORCE VDP MODE
A4EA 05A406  1089          CALL  LITS08            COPY IN & SET UP FAC
A4ED 0F75    1090          XML   CONT
A4EF B75C    1091  SEG$08  DCLR  @ARG              EXTRACT A NULL STRING
A4F1 44D4    1092          BR    SEG$06            >>JUMP ALWAYS<<
```

```
                1094  ********************************************************************
                1095  *                                                                  *
                1096  *       LEN(A$) - Calculate the length of a string and             *
                1097  *            leave the result in the FAC.                          *
                1098  *       CONTROL - Turned over to NLEN from the parser              *
                1099  *       USES - No temporaries.                                     *
                1100  *                                                                  *
                1101  ********************************************************************
                1102  *
A4F3 06A565     1103  LEN01   CALL  PARFF              Insure left paren and parse
A4F6 4D57       1104          BR    ERRSNM               If not string value
A4F8 BD4A50     1105          DST   @FAC6,@FAC         Length
A4FB 0F80       1106  LEN02   XML   CIF                Covert integer to floating no.
A4FD 0F75       1107          XML   CONT
```

```
                1109  ********************************************************************
                1110  *                                                                  *
                1111  *       CHR$(X) - Takes integer value X and converts               *
                1112  *            the number into the ASCII representation for          *
                1113  *            that number.                                          *
                1114  *       CONTROL - Turned over to NCHR by the parser.               *
                1115  *       OUPUT - FAC is set up with the string entry               *
                1116  *       USES - Uses temporaries when invoking LITS06(LITSTR)      *
                1117  *                                                                  *
                1118  ********************************************************************
A4FF 06A565     1119  CHR$01  CALL  PARFF                 Insure left paren and parse
A502 06A975     1120          CALL  INTARG                CONVERT INTO INTEGER
A505 BF0C00     1121          DST   1,@BYTE               CREATE A LENGTH 1 STRING
A508 01
A509 BCA378     1122          ST    @FAC1,RAM(ONECHR)     MOVE THE VALUE TO VDP(FOR LIT
A50C 4B
A50D BF6603     1123          DST   ONECHR,@TEMP5         ADDRESS OF CHARACTER.
A510 78
A511 06A42C     1124          CALL  LITS06                CREATE STRING AND SET UP FAC
A514 BF5000     1125          DST   1,@FAC6               LENGTH OF STRING
A517 01
A518 0F75       1126          XML   CONT
                1127
                1128  ********************************************************************
                1129  *                                                                  *
                1130  *       ASC(A$) - Takes the numeric value of the first             *
                1131  *            character in A$.                                      *
                1132  *                                                                  *
                1133  ********************************************************TE**
A51A 06A565     1134  ASC01   CALL  PARFF                 Insure left paren and parse
A51D 4D57       1135          BR    ERRSNM                   If not string
A51F 8E516D     1136          $IF   @FAC6+1 .EQ. 0 GOTO ERRBA  Bad argument.
A522 83
A523 BC4BB0     1137          ST    RAM(@FAC4),@FAC1      GET THE FIRST CHARACTER
A526 4E
A527 864A       1138          CLR   @FAC
A529 44FB       1139          BR    LEN02                 USE COMMON CODE
                1140  *       Jump always
```

```
                1142  ************************************************************
                1143  *                                                          *
                1144  *      STR$(X) - Takes as its input an integer X and       *
                1145  *           converts it to its string representation.      *
                1146  *           e.g.    3.1415 -> "3.1415"                     *
                1147  *      CONTROL - Turned over to STR$ by the parser          *
                1148  *      USES - The usual temporaries used by string function*
                1149  *           when it calls LITS06.  Uses the Roll-out       *
                1150  *           area for a temporary storage area when         *
                1151  *           allocating the result string.                  *
                1152  *      OUTPUT - FAC is set up in the usual manner for a    *
                1153  *           string.                                        *
                1154  ************************************************************
A52B 06A565     1155  STR$01 CALL PARFF                Insure left paren and parse
A52E 6D57       1156         BS   ERRSNM                  If not numeric-error
A530 8655       1157         CLR  @FAC11                Select BASIC floating type
A532 0F73       1158         XML  CNS                   Convert the number to string
A534 D69055     1159         $IF *FAC11 .EQ. SPACE THEN  If leading space          1
A537 20453E
A53A 9055       1160              INC @FAC11            Suppress it out             1
A53C 9256       1161              DEC @FAC12            Shorten the length          1
                1162         $END IF
A53E 860C       1163         CLR  @BYTE                 Prepare for 2-byte value
A540 BC0D56     1164         ST   @FAC12,@BYTE+1        Get length of string
A543 340CA3     1165         MOVE @BYTE FROM *FAC11 TO RAM(VROA$) Put the
A546 C09055
                1166  *                                 string in VDP
A549 BF6603     1167         DST  VROA$,@TEMP5          Copy-from addr(For LITSTR).
A54C C0
A54D 06A42C     1168         CALL LITS06                Allocate and set up FAC
A550 BD500C     1169         DST  @BYTE,@FAC6           Put in the length
A553 0F75       1170         XML  CONT
                1171
                1172  ************************************************************
                1173  *                                                          *
                1174  *      VAL(A$) - Takes as its input a string, A$, and      *
                1175  *           converts the string into a number if the string*
                1176  *           is a valid representation of a number.         *
                1177  *           e.g. - "3.1415" -> 3.1415                      *
                1178  *           VAL suppresses leading and trailing blanks from*
                1179  *           the source string.                             *
                1180  *           VAL is effectively the inverse of STR$.        *
                1181  *      CONTROL - from the parser                           *
                1182  *      OUTPUT - FAC contains the floation point number     *
                1183  *                                                          *
                1184  ************************************************************
A555 06A565     1185  VAL01  CALL PARFF                Insure left paren and parse
A558 4D57       1186         BR   ERRSNM                  If not string - error
A55A 8E516D     1187         $IF @FAC6+1 .EQ. 0 GOTO ERRBA  Can't have null string
A55D 83
A55E 06A571     1188         CALL  VALCD                    So bad argument error
A561 6D83       1189         BS  ERRBA
A563 0F75       1190         XML  CONT
                1191
                1192  *      Short routine to parse a single argument enclosed in
```

```
                    1193  *       parentheses for a function or a subprogram and set
                    1194  *       condition based upon whether the value parsed was
                    1195  *       a string or a numeric
A565 D642B7         1196  PARFF   $IF  @CHAT .NE. LPAR$ GOTO ERRSYN
A568 4D53
A56A 0F74           1197          XML  PARSE
A56C FF             1198          DATA >FF
A56D D64C65         1199          CEQ  >65,@FAC2
A570 01             1200          RTNC
```

```
                    1202  VALCD
A571 BD664E         1203          DST  @FAC4,@TEMP5       Ptr to string
A574 A16650         1204          DADD @FAC6,@TEMP5       Ptr to trailing length byte
A577 BD0C50         1205          DST  @FAC6,@BYTE        for suppressing trailing bl...
A57A 910C           1206          DINC @BYTE              prepare for undue subtraction
                    1207          $REPEAT
A57C 9366           1208            DDEC @TEMP5           keep track of end of string  1
A57E 930C           1209            DDEC @BYTE            decrease length of string    1
A580 6589           1210            BS   RTNSET           end up with empty string.    1
A582 D6B066         1211          $UNTIL RAM(@TEMP5) .NE. : :  while trailing blanks
A585 20657C
A588 910C           1212          DINC @BYTE              Allow for terminator
A58A 0F77           1213          XML  VPUSH              Save the ptr to the string
A58C 0F71           1214          XML  GETSTR             Get a new string
A58E 0F78           1215          XML  VPOP               Retrieve the ptr to the string
A590 BD664E         1216          DST  @FAC4,@TEMP5       Get the ptr to the string
A593 8652           1217          CLR  @FAC8              Force VDP mode
A595 06A411         1218          CALL LITS09             Copy the string and set up FAC
A598 A10C1C         1219          DADD @SREF,@BYTE        Point to the trailing length
A59B 930C           1220          DDEC @BYTE              Point at the last character
A59D BEB00C         1221          ST   SPACE,RAM(@BYTE)   Put in the terminator
A5A0 20
A5A1 BD561C         1222          DST  @SREF,@FAC12       Addr for the conversion
A5A4 D6B056         1223          $WHILE  RAM(@FAC12) .EQ. SPACE  While leading spaces1
A5A7 2045AE
A5AA 9156           1224            DINC  @FAC12          Skip leading blank           1
A5AC 45A4           1225          $END WHILE
A5AE 864C           1226          CLR  @FAC2              Get rid of string (in case=0)
A5B0 8654           1227          CLR  @FAC10             Assume no error
A5B2 0F10           1228          XML  CSNUM              Convert it
A5B4 D5560C         1229          DCEQ @BYTE,@FAC12       Convert all of it?
A5B7 68F1           1230          BS   WRNNO              Yes-check overflow & return
A5B9 D40000         1231  RTNSET  CEQ  @>00,@>00          No- return with condition set
A5BC 01             1232          RTNC
```

```
                    1234  ********************************************************************
                    1235  *                                                                  *
                    1236  *       POS(A$,B$,X) - Attempts to match the string, B$, in        *
                    1237  *            in A$ beginning at character # X in A$.  If X         *
                    1238  *            is > len(A$), a match is not found or A$ is the       *
                    1239  *            null string then the returned value is 0.            *
                    1240  *            If B$ is the null string then the returned valu      *
                    1241  *            is 1.  Otherwise, the returned value is the          *
                    1242  *            column # of the 1st character matched in A$.          *
                    1243  *       CONTROL - From the parser.  Returns through common        *
                    1244  *            code in LEN.                                         *
                    1245  *       USES - No temps - Utilizes FAC and ARG.                   *
                    1246  *                                                                  *
                    1247  ********************************************************************
A5BD 06AC28         1248  POS01   CALL LPAR                Insure '(', parse, insure ','
A5C0 06A430         1249          CALL PUSSTR              STACK THE STRING AND GET TOKEN
A5C3 0F7E           1250          XML  SPEED               Parse the match string and
A5C5 01             1251          DATA PARCOM                 insure end on comma
A5C6 06A430         1252          CALL PUSSTR              STACK THE STRING AND GET TOKEN
A5C9 0F74           1253          XML  PARSE               GET POSITION
A5CB B6             1254          DATA RPAR$
A5CC 0F7E           1255          XML  SPEED               Must have
A5CE 00             1256          DATA SYNCHK                 ended on a
A5CF B6             1257          DATA RPAR$                     right parenthesis
A5D0 06A975         1258          CALL INTARG              CHECK AND CONVERT IT
A5D3 8F4A6D         1259          $IF @FAC .DEQ. 0 GOTO ERRBV   VALUE OUT OF RANGE
A5D6 87
A5D7 BD0C4A         1260          DST  @FAC,@BYTE          KEEP THE OFFSET
A5DA 930C           1261          DDEC @BYTE               CORRECT FOR POSITION 0
A5DC 0F78           1262          XML VPOP                 GET MATCH STRING BACK
A5DE 350008         1263          MOVE 8 FROM @FAC TO @ARG    PUT MATCH IN ARG
A5E1 5C4A
A5E3 0F78           1264          XML  VPOP                GET SOURCE BACK
A5E5 8E5166         1265          $IF  @FAC6+1 .EQ. 0 GOTO POS12 If source null
A5E8 24
A5E9 C4510D         1266          CH   @BYTE+1,@FAC6+1     OFFSET > LENGTH?
A5EC 4624           1267          BR   POS12               YES - NO MATCH POSSIBLE
A5EE 8E6366         1268          $IF  @ARG6+1 .EQ. 0 GOTO POS06 If null string
A5F1 15
A5F2 A14E0C         1269          DADD @BYTE,@FAC4         ADJUST PTR FOR OFFSET
A5F5 A4510D         1270          SUB  @BYTE+1,@FAC6+1     ADJUST LENGTH
A5F8 C85163         1271  POS02   CHE  @ARG6+1,@FAC6+1    ENOUGH SPACE LEFT FOR A MATCH
A5FB 4624           1272          BR   POS12               NO - NO MATCH POSSIBLE
A5FD BD4A4E         1273          DST  @FAC4,@FAC          GET FIRST ARG
A600 BD5C60         1274          DST  @ARG4,@ARG          GET SECOND ARG
A603 BC6463         1275          ST   @ARG6+1,@ARG8       AND LENGTH OF SECOND
A606 D4B05C         1276  POS04   CEQ  RAM(@FAC),RAM(@ARG)  COMPARE THE CHARACTERS
A609 B04A
A60B 461C           1277          BR   POS10               DIDN'T MATCH
A60D 914A           1278          DINC @FAC                NEXT IN SOURCE
A60F 915C           1279          DINC @ARG                NEXT IN MATCH
A611 9264           1280          DEC  @ARG8               REACHED END OF MATCH?
A613 4606           1281          BR   POS04               NOT YET-SO LOOP
A615 900D           1282  POS06   INC  @BYTE+1            MATCHED! CORRECT FOR 1 INDEXI
A617 BD4A0C         1283  POS08   DST  @BYTE,@FAC         CHARACTER POSITION OF MATCH
```

```
A61A 44FB    1284          BR   LEN02              CONVERT TO FLOATING
             1285  *       Note: Utilizes the LEN code to do the conversion
             1286  *                   and finish up
A61C 900D    1287  POS10   INC  @BYTE+1            STEP INDEX OF MATCH CHARACTE
A61E 9251    1288          DEC  @FAC6+1            MOVE 1 POSITION DOWN 1ST
A620 914E    1289          DINC @FAC4                ARGUMENT
A622 45F8    1290          BR   POS02              TRY TO MATCH AGAIN
             1291  *       Jump always
A624 860D    1292  POS12   CLR  @BYTE+1            NO MATCH POSSIBLE
A626 4617    1293          BR   POS08
```

```
                    1295  ************************************************************
                    1296  *                                                          *
                    1297  *      RPT$(A$,X) - Creates a string consisting of X       *
                    1298  *           copies of A$.  If X is negative or non-numeric,*
                    1299  *           an exception occurs.  If A$ is not a string,   *
                    1300  *           an exception occurs.                           *
                    1301  *                                                          *
                    1302  ************************************************************
A628 06AC28         1303  RPT$01 CALL LPAR                 Insure '(', parse, insure ','
A62B 06A430         1304         CALL PUSSTR               Insure a string and push it
A62E 0F74           1305         XML  PARSE                Parse second arg
A630 B6             1306         DATA RPAR$
A631 0F7E           1307         XML  SPEED                Must have
A633 00             1308         DATA SYNCHK                  ended on a
A634 B6             1309         DATA RPAR$                      right parenthesis
A635 06A975         1310         CALL INTARG               Check numeric and convert
A638 A94AE0         1311         DMUL RAM(6(VSPTR)),@FAC Compute result length
A63B 066E
A63D BF4B66         1312         $IF @FAC1 .DNE. 0 THEN                                1
A640 49
A641 066AB2         1313  WRNST2   CALL WARN$$             Give truncation message      1
A644 13             1314           DATA 19                 * STRING TRUNCATED message.. 1
A645 BF4C00         1315           DST  255,@FAC2          Make it a maximum string     1
A648 FF
                    1316         $END IF
A649 BD0C4C         1317         DST  @FAC2,@BYTE          Copy requested string length
A64C 0F71           1318         XML  GETSTR               Get the new string
A64E 0F78           1319         XML  VPOP                 Retrieve the original string
                    1320  *      At this point BYTE should still contain the length
A650 BD5C50         1321         DST  @FAC6,@ARG           Copy original length in ARG
A653 8F0C46         1322         $IF @BYTE .DEQ. 0 THEN Zero copies requested          1
A656 59
A657 875C           1323           DCLR @ARG               So we copy zero !!!!!!!      1
                    1324         $END IF
A659 C10C5C         1325         DEX  @ARG,@BYTE           Original length to BYTE
A65C BD664E         1326         DST  @FAC4,@TEMP5         And also original start addre
A65F 8652           1327         CLR  @FAC8                clear flag for LITSOB
A661 06A406         1328         CALL LITSOB               Create FAC and copy one copy
                    1329  *      ARG contains total length now
A664 BD505C         1330         DST  @ARG,@FAC6           Store new length
                    1331  RPT$02
A667 A55C0C         1332         DSUB @BYTE,@ARG           Subtract one copy
A66A 8F5C66         1333         $IF @ARG .DEQ. 0 GOTO XMLCON <<<<The way out
A66D 84
A66E A11C0C         1334         DADD @BYTE,@SREF          Compute new start address
A671 C50C5C         1335         $IF @BYTE .DH. @ARG THEN                              1
A674 4679
A676 BD0C5C         1336           DST  @ARG,@BYTE         Truncate string              1
                    1337         $END IF
A679 340CB0         1338         MOVE @BYTE FROM RAM(@TEMP5) TO RAM(@SREF)
A67C 1CB066
A67F 4667           1339         BR   RPT$02               Loop until done
```

```
                    1341  ************************************************************
                    1342  *                    TRACE STATEMENT                       *
                    1343  ************************************************************
A6B1 B64510         1344  NTRACE SB    @FLAG,4            Set the trace bit
A6B4 0F75           1345  XMLCON XML   CONT               Continue on
                    1346
                    1347
                    1348  ************************************************************
                    1349  *                   UNTRACE STATEMENT                      *
                    1350  ************************************************************
A6B6 B245EF         1351  NUNTRC RB    @FLAG,4            Reset the trace bit
A6B9 0F75           1352         XML   CONT               Continue on
                    1353
                    1354
                    1355  ************************************************************
                    1356  *              BREAK AND UNBREAK STATEMENTS                *
                    1357  ************************************************************
A6BB BE5CFF         1358  NBREAK ST    >FF,@ARG           BREAK flag
A6BE 066A78         1359         CALL  CHKEND             Check for end of statement.
A6C1 46A4           1360         BR    LINEGP             If not goto LINEGP.
A6C3 032C           1361         DDEC  @PGMPTR            Back up so CON will rescan end
A6C5 BE4441         1362         $IF   @PRGFLG .NE. 0 GOTO EXEC6C  rative w/o line #
A6C8 27
A6C9 066A84         1363  ERROLP CALL  ERR$$              Only legal in a program.
A6CC 1B             1364         DATA  27
                    1365  *
A6CD B65C           1366  NUNBRK CLR   @ARG               UNBREAK flag for common
A6CF 066A78         1367         CALL  CHKEND             Check for end of statement.
A6D2 66F7           1368         BS    UNBK01             If end then goto UNBK01
                    1369
A6D4 06A8D3         1370  LINEGP CALL  LINE               Get line #
A6D7 BD5E32         1371         DST   @ENLN,@ARG2
A6DA A75E00         1372         DSUB  >03,@ARG2          1st line #
A6DD 03
A6DE C95E30         1373  LNGP1  $IF @ARG2 .DL. @STLN GOTO WRNLNF If line not found
A6E1 46F1
A6E3 05802E         1374         CALL  GRSUB3             Read ln # of data from ERAM
A6E6 5E             1375         DATA  ARG2              (use GREAD1) or VDP
                    1376  *     @ARG2:Source addr in ERAM/VDP, reset possible bkpt too
A6E7 D55B4A         1377         $IF @EEE1 .DEQ. @FAC GOTO LNGP2  If line found
A6EA 66C2
A6EC A75E00         1378         DSUB  4,@ARG2            Next line in VDP or ERAM
A6EF 04
A6F0 46AE           1379         BR    LNGP1
                    1380  *      Jump always
                    1381  LNGP2
A6F2 8E8084         1382         $IF @RAMTOP .NE. 0 THEN    If ERAM exist                1
A6F5 66DA
A6F7 B2587F         1383           RB @EEE1,7             Assume UNBREAK flag          1
A6FA 8E5C66         1384           $IF @ARG .NE. 0 THEN If BREAK flag                  2
A6FD D1
A6FE B65B80         1385             SB  @EEE1,7          Set the breakpoint           2
                    1386           $END IF
A701 066036         1387           CALL GWSUB             Write a few bytes of data to 1
                    1388  *                                  ERAM(use GWRITE)
```

```
A6D4 5E5B01  1389            DATA ARG2,EEE1,1                                  1
             1390  *         @ARG2 : Destination addr on ERAM
             1391  *         @EEE1 : Data
             1392  *         1 : Byte count
A6D7 054A6E6 1393          $ELSE                                               1
A6DA B2B05E  1394             RB RAM(@ARG2),7     Assume UNBREAK flag first  1
A6DD 7F
A6DE 8E5C66  1395            $IF @ARG .NE. 0 THEN If BREAK flag                2
A6E1 E6
A6E2 B6B05E  1396               SB   RAM(@ARG2),7    Set the breakpoint       2
A6E5 80
             1397            $END IF
             1398          $END IF
A6E6 066A78  1399  LNGP2B  CALL CHKEND             Check for end of statement.
A6E9 66FA    1400          BS   LNGP4              If end then continue.
A6EB 0F7E    1401          XML  SPEED              Must be
A6ED 00      1402          DATA SYNCHK                at a
A6EE B3      1403          DATA COMMA$                comma now
A6EF 46A4    1404          BR   LINEGP
             1405  *       Jump always
             1406  WRNLNF
A6F1 066A82  1407          CALL WARN$$             Note: warning not error.
A6F4 26      1408          DATA 38                 'LINE NOT FOUND'
A6F5 46E6    1409          BR   LNGP2B             And continue on.
             1410  *
             1411  *       Jump always
             1412
A6F7 06A6FC  1413  UNBK01  CALL UBSUB              Clear all bkpt in ln # table
A6FA 0F75    1414  LNGP4   XML CONT                Continue
             1415  *        CLEAR ALL BREAKPOINTS
A6FC BD5230  1416  UBSUB   DST  @STLN,@FAC8        END OF LINE # BUFFER
             1417          $REPEAT
A6FF 06A70C  1418            CALL UBSUB1           Reset one ln # at a time   1
A702 A35200  1419            DADD 4,@FAC8          Go to the nexy ln          1
A705 04
A706 C55232  1420          $UNTIL @FAC8 .DH. @ENLN  End of the table
A709 46FF
A70B 00      1421          RTN
             1422  UBSUB1
A70C 06802E  1423          CALL GRSUB3             Read the line # from ERAM/VDP
             1424  *                               Reset possible bkpt too
A70F 52      1425          DATA FAC8               @FAC8:Source addr on ERAM/VDP
A710 066036  1426          CALL GWSUB              Write a few bytes of data to
             1427  *                                  ERAM(use GWRITE) or VDP
A713 525B01  1428          DATA FAC8,EEE1,1
             1429  *       @FAC8 : Destination addr in ERAM/VDP
             1430  *       @EEE1 : Data
             1431  *       1 : Byte count
A716 00      1432          RTN
```

```
                1434  ********************************************************************
                1435  *                    USER-DEFINED FUNCTIONS                        *
                1436  *       Subroutine to store away the information of the            *
                1437  *       tokens in a function reference, go into the 'DEF'         *
                1438  *       statement, calculate the value of the expression          *
                1439  *       and then resume execution of the user's program           *
                1440  *       after the reference.                                      *
                1441  *                                                                 *
                1442  *       An entry in the FAC and on the stack for a function       *
                1443  *       reference looks like:                                     *
                1444  *                                                                 *
                1445  *    +--------+---+--------------------+--------+--------+       *
                1446  *    | PGMPTR |>68|string/numeric flag| SYMTAB | FREPTR |       *
                1447  *    +--------+---+--------------------+--------+--------+       *
                1448  *    ^        ^   ^                    ^        ^                *
                1449  *   FAC      FAC2 FAC3                 FAC4     FAC6             *
                1450  *                                                                 *
                1451  *       The 'PGMPTR' is where execution resumes after             *
                1452  *       evaluating the function.   String(80)/numeric(00)         *
                1453  *       flag is function type.  SYMTAB is the old symbol          *
                1454  *       table pointer and FREPTR is the old free space            *
                1455  *       pointer.  These are restored after the function is        *
                1456  *       evaluated.                                                *
                1457  ********************************************************************
A717 8E4447     1458  UDF     $IF  @PRGFLG .EQ. 0 THEN If imperative                    1
A71A 20
A71B 8E8085     1459            $IF  @RAMTOP+1 .NE. 0 GOTO ERROLP And ERAM-error      1
A71E 4699
                1460            $END IF
A720 8651       1461            CLR  @FAC7              Assume no args
A722 8722       1462            DCLR @ERRCOD            Clear the error code for cont
A724 865E       1463            CLR  @ARG2              Safety for VPUSH
A726 864C       1464            CLR  @FAC2              Safety for VPUSH
A728 D642B7     1465            $IF  @CHAT .EQ. LPAR$   THEN                          1
A72B 473B
A72D 0F77       1466              XML  VPUSH            Save ptr to func definition   1
A72F 0F74       1467              XML  PARSE            PARSE to get arg value        1
A731 FF         1468              DATA >FF                                            1
A732 350008     1469              MOVE 8 BYTES FROM @FAC TO @ARG  Save PARSE result   1
A735 5C4A
A737 0F78       1470              XML   VPOP            Get S.T. ptr to func def      1
A739 9051       1471              INC   @FAC7           Indicate that we have an arg  1
                1472            $END IF
A73B BC6651     1473            ST   @FAC7,@TEMP5       Move the parm count
A73E BD644A     1474            DST  @FAC,@TEMP4        S.T. ptr to definition
A741 0F77       1475            XML  VPUSH              Allow room for UDF entry
A743 350008     1476            MOVE 8 FROM @ARG TO @FAC  Retrieve parse result
A746 4A5C
A748 0F77       1477            XML  VPUSH              Save parse result
A74A BC4CB0     1478            ST   RAM(@TEMP4),@FAC2  Get S.T. declarations
A74D 64
A74E BC4D4C     1479            ST   @FAC2,@FAC3        Do this to save string bit
                1480  *
                1481  *       NOTE: THIS IS TO ALLOW THE CHECKING AFTER THE
                1482  *       FUNCTION HAS BEEN EVALUATED TO MAKE SURE THE
```

```
                1483  *       FUNCTION TYPE(STRING/NUM) MATCHES THE RESULT IT
                1484  *       PRODUCES.
                1485  *
A751 B24C07     1486          AND   >07,@FAC2           Mask all but # of parms
A754 D44C66     1487          $IF   @FAC2 .NE. @TEMP5 GOTO ERRIAL
A757 4D8B
                1488  *               Incorrect argument list error above.
A759 BD4A2C     1489          DST   @PGMPTR,@FAC        Will resume execution here
A75C BE4C70     1490          ST    >70,@FAC2           Entering par into symbol table
                1491  *                                 while in UDF statement executi
A75F B24D80     1492          AND   >80,@FAC3          Mask all but string bit
A762 A76E00     1493          DSUB  16,@VSPTR           Get below parse result
A765 10
A766 BD4E3E     1494          DST   @SYMTAB,@FAC4       Save current symbol table ptr
A769 BD5040     1495          DST   @FREPTR,@FAC6       Save current free space ptr
A76C 0F77       1496          XML   VPUSH               Save the return info
A76E A36E00     1497          DADD  8,@VSPTR            Get back to parse result
A771 08
                1498  *
                1499  *******SHIFT EXECUTION TO FUNCTION DEFINITION
                1500  *
A772 BD2CE0     1501          DST   RAM(6(TEMP4)),@PGMPTR  Set text ptr to def
A775 0664
A777 0F79       1502          XML   PGMCHR              Get 1st char in the def
A779 C673A4     1503          $IF   @SUBSTK .H. >A4 GOTO ERRSO Stack overflow.
A77C 6D63
A77E 350018     1504          MOVE 24 FROM @00 TO RAM(VROA$)  Roll out temporaries
A781 A3C000
A784 B64508     1505          SB    @FLAG,3             Set function flag for ENTER
A787 BE1680     1506          ST    >80,@XFLAG          Make calls look like 'ENTERX'
A78A D642BE     1507          $IF   @CHAT .EQ. EQUAL$  THEN
A78D 479C
                1508  *
                1509  *       Note: This is to keep the global/local variables
                1510  *       correct in the event that a function uses another
                1511  *       function in its evaluation.
                1512  *
A78F 8659       1513            CLR  @FAC15             Create a dummy entry in table
A791 066A80     1514            CALL ENT09              for no-parm function
A794 972C       1515            DDECT @PGMPTR           Back up to equal sign
A796 86E002     1516            CLR  RAM(2(VSPTR))      This is to keep ASSGNV(called
A799 6E
                1517  *             below) not to screw up in case FAC2 happens to
                1518  *             have a value .gt. >65
A79A 479F       1519          $SELSE
A79C 066A7E     1520            CALL ENTER              Enter the parameter
                1521          $END IF
A79F 0F79       1522          XML   PGMCHR              Get the '=' (Checked in PSCAN
A7A1 B245F7     1523          RB    @FLAG,3             Reset to normal ENTERs
A7A4 350018     1524          MOVE 24 FROM RAM(VROA$) TO @00  Roll in temporaries
A7A7 00A3C0
A7AA BEEFFF     1525          ST    >68,RAM(-6(VSPTR))    Correct stack entry ID
A7AD FA6E68
A7B0 BDE002     1526          DST   RAM(SYMBOL),RAM(2(SYMTAB))    Fudge link to
A7B3 3EA376
```

```
                    1527  *                                   get global values
A7B6 BD4A3E         1528          DST   @SYMTAB,@FAC          Set up for SMB
A7B9 0F7D           1529          XML   SMB                   Get value space
A7BB 350008         1530          MOVE 8 FROM @FAC TO @FAC8    Destination
A7BE 524A
A7C0 0F78           1531          XML   VPOP                  Get arg back
A7C2 350008         1532          MOVE 8 FROM @FAC TO @ARG  Argument value
A7C5 5C4A
A7C7 350008         1533          MOVE 8 FROM @FAC8 TO @FAC  Destination
A7CA 4A52
A7CC 0F77           1534          XML   VPUSH                 Push the destination
A7CE 350008         1535          MOVE 8 FROM @ARG TO @FAC  Argument value
A7D1 4A5C
A7D3 D64C65         1536          $IF @FAC2 .EQ. >65 THEN  If a string                1
A7D6 47E2
A7D8 D74A00         1537            $IF @FAC .NOT. .DEQ. SREF  THEN  If not temp      2
A7DB 1C67E2
A7DE BD4EB0         1538              DST  RAM(@FAC),@FAC4  Get new loc. of string    2
A7E1 4A
                    1539            $END IF             String probably moved when
                    1540          $END IF               Parameter was allocated in S.T.
A7E2 0F79           1541          XML   PGMCHR          Skip the '='
A7E4 0F7C           1542          XML   ASSGNV          Assign the value to the parameter
                    1543  *
A7E6 0F74           1544          XML   PARSE           PARSE to end of function definition
A7E8 83             1545          DATA TREM$
                    1546  *
                    1547  *******CHECK FOR TYPE MATCH (STRING/STRING OR NUM/NUM)
                    1548  *******BETWEEN THE RESULT AND THE FUNCTION TYPE
                    1549  *
A7E9 D64C65         1550          $IF  @FAC2 .EQ. >65 THEN  If result string          1
A7EC 47F6
A7EE 8EE003         1551            $IF RAM(3(VSPTR)) .EQ. 0 GOTO ERRSNM    If functio
A7F1 6E6D57
A7F4 47FC           1552          $SELSE                    if result not string      1
A7F6 8EE003         1553            $IF RAM(3(VSPTR)) .NE. 0 GOTO ERRSNM If function i
A7F9 6E4D57
                    1554          $END IF
                    1555  *
                    1556  *******NOW RESTORE SYMBOL TABLE AND
                    1557  *******RESUME EXECUTION AT THE ORIGINAL LINE
                    1558  *
A7FC 06A80A         1559          CALL DELINK                 Delink the parameter entry
A7FF BD2CE0         1560          DST  RAM(8(VSPTR)),@PGMPTR Manual pop to get ptr back
A802 086E
A804 932C           1561          DDEC @PGMPTR                Back up text pointer
A806 0F79           1562          XML  PGMCHR                 Get next token
A808 0F75           1563          XML  CONT                   And continue
                    1564  *
                    1565  *
                    1566  DELINK
A80A BD663E         1567          DST  @SYMTAB,@TEMP5    Save addr of S.T. entry just
                    1568  *             in case entry is a string(Must free the string)
A80D 350004         1569          MOVE 4 FROM RAM(4(VSPTR)) TO @SYMTAB  Restore old
A810 3EE004
```

```
A813 6E
                1570  *       symbol table ptr and free space pointer
                1571  *       This handles the freeing of the string value
                1572  *       which was assigned to the parameter.
A814 D2B066     1573          $IF RAM(@TEMP5) .LT. 0 THEN    If string parm          1
A817 00684C
A81A BD66E0     1574            DST  RAM(6(TEMP5)),@TEMP5    Where the string is     1
A81D 0666
A81F 8F6668     1575            $IF @TEMP5 .DNE. 0 THEN      If non-null string      2
A822 33
A823 BD56EF     1576              DST  RAM(-3(TEMP5)),@TEMP1 Get backpointer         2
A826 FFFD66
A829 C9563E     1577              $IF @TEMP1 .DL. @SYMTAB THEN  If not used          3
A82C 6833
A82E 87EFFF     1578                DCLR RAM(-3(TEMP5))      Free up the string    3
A831 FD66
                1579              $END IF
                1580            $END IF
                1581  *       This handles the special case of F$(X$)=X$
                1582  *       The result, which was permanent, must be made a temp.
A833 D64C65     1583            $IF @FAC2 .EQ. >65 THEN If string result          2
A836 484A
A838 C94A3E     1584              $IF @FAC .DL. @SYMTAB THEN  If came from argumen3
A83B 684A
A83D 8F4E68     1585               $IF @FAC4 .DNE. 0 THEN If non-null             4
A840 46
A841 87EFFF     1586                DCLR RAM(-3(FAC4)) Clear the backpointer      4
A844 FD4E
                1587               $END IF
A846 BF4A00     1588               DST  SREF,@FAC       Make it a temp           3
A849 1C
                1589              $END IF
                1590            $END IF
A84A 4856       1591          $SELSE                       If numeric parm         1
A84C BE8084     1592            $IF @RAMTOP .NE. 0 THEN  If ERAM exist           2
A84F 6856
A851 A38086     1593              DADD 8,@RAMFRE     Remove 8 bytes of value     2
A854 0008
                1594  *   allocated in the expansion ram for a numeric parm.
                1595            $END IF
                1596          $END IF
A856 A76E00     1597          DSUB 8,@VSPTR            Trash the stack entry
A859 08
A85A 00         1598          RTN                      And return
```

```
                    1600  ATTNUT
A85B 0F74           1601          XML   PARSE
A85D B5             1602          DATA  RPAR$
A85E 06A3ED         1603          CALL  CKSTNM        CHECK FOR NUMERIC OR STRING
A861 0F7E           1604          XML   SPEED         Insure argument is in
A863 02             1605          DATA  RANGE             range of 0-30
A864 00001E         1606          DATA  0,#30
A867 E64B01         1607          SRL   @FAC1,1       0,1:0000      ATTENUATION/2
                    1608  *                           2,3:0001
                    1609  *                           4,5:0010
                    1610  *                           6,7:0011      ETC.
A86A B64BF0         1611          OR    >F0,@FAC1     REGISTER BITS
A86D 00             1612          RTN
```

```
                    1614  ************************************************************
                    1615  *       SUBROUTINE TO SET POINTER TO EACH DATUM
                    1616  ************************************************************
A86E 9336           1617  DATAST DDEC @LNBUF                Point to 1st byte of line ptr
A870 068020         1618         CALL GRSUB2                Read 2 bytes from VDP or ERAM
A873 36             1619         DATA LNBUF                    (use GREAD1), @LNBUF:Source
                    1620  *                                    addr in ERAM or VDP
A874 BD3458         1621         DST @EEE1,@DATA            Put it in @DATA
A877 068020         1622         CALL  SRDATA               Look for 'DATA' on the line
A87A 4889           1623         BR    DATST1               O.K. FOUND ANOTHER 'DATA' STMT
A87C 9736           1624         DDECT @LNBUF               NO
A87E D53630         1625         $IF @LNBUF .DNE. @STLN THEN                              1
A881 6887
A883 9336           1626           DDEC @LNBUF              POINT TO 1ST TOKEN ADD.       1
A885 486E           1627           BR   DATAST                                            1
                    1628         $END IF
A887 8634           1629         CLR @DATA                  INDICATE NO DATA
                    1630  DATST1
A889 00             1631         RTN
                    1632
                    1633
                    1634
                    1635  ************************************************************
                    1636  *       Subroutine to get line number and goto routine      *
                    1637  *       to display it on the screen                         *
                    1638  ************************************************************
                    1639  ASC
A88A 8E8089         1640        $IF @RAMFLG .EQ. 0 THEN                                  1
A88D 4897
A88F BD5EEF         1641           DST RAM(-2(EXTRAM)),@ARG2 Get ln # in                 1
A892 FFFE2E
A895 48A5           1642        $SELSE                                                   1
A897 BF5600         1643           DST 2,@FFF1             @FFF1:Byte count              1
A89A 02
A89B BD542E         1644           DST @EXTRAM,@DDD1       @DDD1:Source addr. in ERAM    1
A89E 9754           1645           DDECT @DDD1                                           1
A8A0 0F8C           1646           XML GREAD1              Read data from ERAM           1
A8A2 BD5E58         1647           DST @EEE1,@ARG2         @EEE1:Destination addr. on CP
                    1648        $END IF
A8A5 B25E7F         1649        AND >7F,@ARG2              Reset the breakpt if any
A8A8 056A7C         1650        B DISO
                    1651
                    1652
                    1653
                    1654  ************************************************************
                    1655  *       Code to decode error returned from ALC              *
                    1656  ************************************************************
A8AB 8A22           1657  EROR$  CASE @ERRCOD              DECODE ERROR FROM INTERPRETER
A8AD 4D53           1658         BR    ERRSYN                 0 SYNTAX ERROR
A8AF 4D5F           1659         BR    ERRMEM                 1 MEMORY FULL
A8B1 4D67           1660         BR    ERRBV                  2 BAD VALUE
A8B3 4D7F           1661         BR    ERRLNF                 3 LINE NOT FOUND
A8B5 4D53           1662         BR    ERRSYN                 4 SYNTAX
A8B7 4D7D           1663         BR    ERRBS                  5 BAD SUBSCRIPT
A8B9 4D57           1664         BR    ERRSNM                 6 STRING-NUMBER MISMATCH
```

```
A8BB 4D63   1665          BR   ERRSO          7 STACK OVERFLOW
A8BD 4D83   1666          BR   ERRBA          8 BAD ARGUMENT
A8BF 4D77   1667          BR   ERRRWG         9 RETURN WITHOUT GOSUB
A8C1 4D8B   1668          BR   ERRIAL         A INCORRECT ARGUMENT LIST
A8C3 4D6B   1669          BR   ERRFNN         B FOR/NEXT NESTING
A8C5 4D67   1670          BR   ERRNWF         C NEXT WITHOUT FOR
A8C7 4D5B   1671          BR   ERRMUV         D IMPROPERLY USED NAME
A8C9 4D8B   1672          BR   ERRIAL         E INCORRECT ARGUMENT LIST
A8CB 4D73   1673          BR   ERRRSC         F RECURSIVE SUBPROGRAM CALL
A8CD 4D8F   1674          BR   ERRSNF        10 SUBPROGRAM NOT FOUND
A8CF 4699   1675          BR   ERROLP        11 ONLY LEAGAL IN A PROGRAM
A8D1 4D6F   1676          BR   ERRSNS        12 MUST BE IN SUBPROGRAM
            1677  *
```

```
                    1679  *
                    1680  ***************************************************************
                    1681  *       SUBROUTINE TO GET LINE # FOLLOWING
                    1682  *       'BREAK', 'UNBREAK', 'RESTORE'.
                    1683  ***************************************************************
A8D3 D642C9         1684  LINE    $IF @CHAT .NE. LN$ GOTO ERRSYN   Should be line # re
A8D6 4D53
A8D8 0F79           1685          XML   PGMCHR                   GET HIGH ORDER LINE #
A8DA BC4A42         1686          ST    @CHAT,@FAC               BUILD RESULT IN FAC,FAC1
A8DD 0F79           1687          XML   PGMCHR
A8DF BC4B42         1688          ST    @CHAT,@FAC1              LOW ORDER LINE #
A8E2 0F79           1689          XML   PGMCHR                   GET TOKEN FOLLOWING LINE #
A8E4 00             1690          RTN
                    1691
                    1692
                    1693
                    1694
                    1695
A8E5 8654           1696  CONV1   CLR   @FAC10
A8E7 0F10           1697          XML    CSNUM                   Convert String to Number
                    1698  ***************************************************************
A8E9 BCA3BA         1699          ST    @FAC10,RAM(CSNTP1)
A8EC 54
A8ED BDA390         1700          DST   @FAC12,RAM(CSNTMP) Save those in temporary,
A8F0 56
                    1701  *         because in ERROV : WARNING routine have             *
                    1702  *         FAC12 and FAC10 values changed *
                    1703  ***************************************************************
A8F1 8E5468         1704  WRNNO   $IF   @FAC10 .NE. 0 THEN         Numeric overflow
A8F4 F9
A8F5 066A82         1705  WRNNO2 CALL WARN$$
A8F8 02             1706          DATA 2
                    1707          $END IF
A8F9 00             1708          RTN
```

```
                    1710
                    1711  ********************************************************
                    1712  *              SUBROUTINE FOR 'GCHAR'
                    1713  ********************************************************
A8FA 06AC3D         1714  GCHAR  CALL GPHV                 Get X,Y values
A8FD 06A9BD         1715         CALL NUMVAR               Get pointer to return variable
A900 310008         1716         MOVE 8 FROM ROM(#FLT1) TO @FAC   Clear FAC
A903 4AA3F3
A906 BC4B7D         1717         ST   CB,@FAC1             Get the character
A909 A64B60         1718         S    OFFSET,@FAC1         Remove screen offset
A90C CA4B64         1719         $IF  @FAC1 .HE. 100 THEN                          1
A90F 4919
A911 C04C4B         1720           EX   @FAC1,@FAC2                                2
A914 AE4B64         1721           DIV  100,@FAC1                                  1
A917 904A           1722           INC  @FAC                                       1
                    1723         $END IF
A919 0F7C           1724         XML  ASSGNV               Assign the value to the symbol
A91B 44F9           1725         BR   XPTRTN
                    1726
                    1727
                    1728  ********************************************************
                    1729  *              SUBROUTINE FOR 'COLOR'
                    1730  ********************************************************
A91D 0F7E           1731  COLOR  XML  SPEED                Must be
A91F 00             1732         DATA SYNCHK                 at a
A920 B7             1733         DATA LPAR$                     left parenthesis
A921 D642FD         1734         $IF @CHAT .EQ. NUMBE$ THEN If sprite number specified
A924 4936
A926 06AD98         1735           CALL SPNUM3             Check sprite number
                    1736  COL10
A929 06A969         1737           CALL SPCOL              Put the color in SAL.
A92C D642B2         1738           $IF @CHAT .NE. COMMA$ GOTO LNKRTN  More color chan
A92F 4AFC
A931 06AD9A         1739           CALL SPNUM2           Skip and get sprite number.
A934 4929           1740           BR   COL10
                    1741         $END IF
                    1742  *  This part for regular color change routine.
A936 0F7E           1743  COL20  XML  SPEED                Parse the character
A938 01             1744         DATA PARCOM                 set and insure a comma
A939 0F7E           1745         XML  SPEED                Insure in range of
A93B 02             1746         DATA RANGE                  0<= x <= 14
A93C 00000E         1747         DATA 0,#14
A93F A34A08         1748         DADD >80F,@FAC            Color table addr(>810 - >81D)
A942 0F
A943 0F77           1749         XML  VPUSH                Push table set address
A945 0F7E           1750         XML  SPEED                Parse the foreground color
A947 01             1751         DATA PARCOM                 and insure a comma
A948 06AC34         1752         CALL RAN16                Error if >16 or < 1
A94B BC0E4B         1753         ST   @FAC1,@VAR4          Save it
A94E E20E04         1754         SLL  @VAR4,4              Foreground color in 4 MSBits
A951 0F74           1755         XML  PARSE                Get background color
A953 B6             1756         DATA RPAR$
A954 06AC34         1757         CALL RAN16                Error if >16 or <1
A957 B40C4B         1758         OR   @FAC1,@VAR4          Background colro in 4 LSBits
A95A 0F7E           1759         XML   VPOP                Get color table address
```

```
A95C BCB04A  1760          ST   @VAR4,RAM(@FAC)   Load the colors into the table
A95F 0E
             1761  *
A960 D642B3  1762          $IF @CHAT .NE. COMMA$ GOTO LNKRTN  End of call. Go ba
A963 4AFC
A965 0F79    1763          XML  PGMCHR             Skip ","
A967 4936    1764          BR   COL20              Take care of the next set.
             1765  *
             1766  *       CALL SPCOL -- Changes color of sprite.
             1767  *                     Called also from SPRITE.
             1768  *
             1769  SPCOL
A969 0F74    1770          XML  PARSE
A96B B6      1771          DATA RPAR$              Get the color number.
A96C 06AC34  1772          CALL RAN16              Check range 1 - 16.
A96F BCE003  1773          ST   @FAC1,RAM(3(SPSAL))  Store in SAL.
A972 084B
A974 00      1774          RTN
```

```
                1776  ********************************************************************
                1777  *                                                                  *
                1778  *       INTARG - Insures that the value in FAC is a                *
                1779  *            numeric, converts it to integer, issues               *
                1780  *            error messages if necessary or returns.               *
                1781  *                                                                  *
                1782  ********************************************************************
                1783  FTINT
A975 C64C63     1784  INTARG $IF  @FAC2 .H. >63 GOTO ERRSNM  If string - error
A978 6D57
A97A B654       1785         CLR  @FAC10             ASSUME NO ERROR OR WARNING
A97C B76C       1786         DCLR @FPERAD
A97E 0F12       1787         XML  FLTINT
A980 8E544D     1788         $IF  @FAC10 .NE. 0 GOTO ERRBV   If error
A983 87
A984 D24A00     1789         $IF  @FAC .LT. 0 GOTO ERRBV    CAN'T BE < ZERO
A987 4D87
A989 00         1790         RTN
                1791
                1792
                1793  *       FAC IS SET UP WITH F.P. 1
A98A BC4B00     1794  JOYXY  ST   @VAR0,@FAC1
A98D 8E0049     1795         $IF  @VAR0 .EQ. 0 THEN If <> 0
A990 95
A991 B64A       1796           CLR @FAC              (>0000000000000000)
A993 499D       1797         $SELSE
A995 D20000     1798           $IF  @VAR0 .LT. 0 THEN
A998 699D
A99A BE4ABF     1799             ST   >BF,@FAC
                1800           $END IF
                1801         $END IF
A99D 0F7C       1802         XML  ASSGNV             Assign the value
A99F 00         1803         RTN
                1804
                1805
A9A0 BC004B     1806  KEYJOY ST   @FAC1,@VAR0       Keyboard selection
A9A3 06A9BD     1807         CALL NUMVAR            Get variable for key-code
A9A6 D642B3     1808         $IF  @CHAT .NE. COMMA$ GOTO ERRSYN  If not comma-err
A9A9 4D53
A9AB 0F79       1809         XML  PGMCHR            Get next character
A9AD 06A9BD     1810         CALL NUMVAR            Get variable for key-status
A9B0 BC7400     1811         ST   @VAR0,@KEYBD      Keyboard selection
A9B3 310008     1812         MOVE 8 FROM ROM(#FLT1) TO @FAC  Set up float 1
A9B6 4AA3F3
A9B9 03         1813         SCAN                   SCAN the keyboard
A9BA B674       1814         CLR  @KEYBD            Clear key code
                1815  *                               (Doesn't affect stat)
A9BC 01         1816         RTNC                   Return scan condition code
                1817
A9BD CA4280     1818  NUMVAR $IF @CHAT .HE. >80 GOTO ERRSYN Do not allow a token
A9C0 6D53
A9C2 0F7A       1819         XML  SYM               Get the symbol name
A9C4 DAB04A     1820         CLOG >C0,RAM(@FAC)     Can't be string or function
A9C7 C0
A9C8 4D5D       1821         BR   ERRMUV            It is - IMPROPERLY USED NAME
```

```
A9CA 0F7B     1822          XML   SMB                 Get value pointer
A9CC 0F77     1823          XML   VPUSH               Put on stack for ASSGNV
A9CE 00       1824          RTN                       And return
              1825
              1826
A9CF 8000A0   1827  ATTREG  DATA  #>8000,#>A000,#>C000,>9F,>BF,>DF,>FF,>00,>06
A9D2 00C000
A9D5 9FBFDF
A9D8 FF0006
              1828
              1829
              1830
              1831
A9DB D642B7   1832  COMB    .$IF  @CHAT .NE. LPAR$ GOTO ERRSYN  If not '(' - error
A9DE 4D53
A9E0 00       1833          RTN
              1834  *
              1835  ***
              1836  *
A9E1 350008   1837  SQUISH  MOVE  8 FROM RAM(@FAC8) TO @FAC  Sneak it out
A9E4 4AB052
A9E7 BD586E   1838          DST   @VSPTR,@FAC14       Now move stack to squish it
A9EA A55852   1839          DSUB  @FAC8,@FAC14           out - # of bytes to move
A9ED 69F8     1840          BS    SQU05               If none to move
A9EF 3458EF   1841          MOVE  @FAC14 FROM RAM(8(FAC8)) TO RAM(-16(FAC8))
A9F2 FFF052
A9F5 E00852
A9F8 A76E00   1842  SQU05   DSUB  8,@VSPTR           Reset top of stack
A9FB 08
A9FC 00       1843          RTN
```

```
                    1845  *
                    1846  ***************************************************************
                    1847  *                    Subprogram for 'CLEAR'
                    1848  ***************************************************************
                    1849  CLEAR
A9FD 0780           1850          ALL   :  :+OFFSET          Clear the screen
A9FF BE7F03         1851          ST    3,@XPT               Initialize screen pointer
AA02 4B00           1852          BR    LNKRT2               Return to caller
                    1853
                    1854  ***************************************************************
                    1855  *                    Subprogram for VERSION                   *
                    1856  ***************************************************************
AA04 06A9DB         1857  VERS    CALL  COMB                 Insure have left paren
AA07 06AD43         1858          CALL  ERRC05               Get symbol information
AA0A 310008         1859          MOVE  8 FROM ROM(#FLT1)  TO @FAC   Floating 1
AA0D 4AA3F3
AA10 904A           1860          INC   @FAC                 Now floating 100!
AA12 4C06           1861          BR    KEY2A                Assign and return to caller
                    1862
                    1863
                    1864  *       INITIALIZATION DATA FOR SOUND
                    1865
AA14 420812         1866  FLTS    DATA >42,>0B,>12,>22,0,0,0,0
AA17 220000
AA1A 0000
                    1867
AA1C 01FF01         1868  SNDREG  DATA >01,>FF,>01,>04,>9F,>BF,>DF,>FF,>00
AA1F 049FBF
AA22 DFFF00
```

```
                1870  ********************************************************
                1871  *                  Subprogram for 'SOUND'              *
                1872  *       BUILDS 2 BLOCKS IN VDP RAM                     *
                1873  *                                                      *
                1874  *  1ST BLOCK: >01,<ATTENUATION FOR NOISE>,<INTERRUPT COUNT>*
                1875  *  2ND BLOCK: >04,>9F,>BF,>DF,>FF,>00                   *
                1876  ********************************************************
                1877  SOUND  $REPEAT                    Insure previous sound started
AA25 D780CC     1878         $UNTIL @>CC .DNE. VRMSND
AA28 03796A
AA2B 25
AA2C 310009     1879         MOVE 9 FROM ROM(#SNDREG) TO RAM(VRMSND)
AA2F A379AA
AA32 1C
AA33 06AC28     1880         CALL LPAR                 Duration in milliseconds
AA36 D24A00     1881         $IF  @FAC .LT. 0 THEN     Don't wait for completion
AA39 6A40
AA3B 834A       1882           DNEG @FAC                  of previous sound
AA3D 8780CE     1883           DCLR @PRTNFN            Make GPL interp stop previous
                1884         $END IF
AA40 0F7E       1885         XML  SPEED                Insure duration
AA42 02         1886         DATA RANGE                   is in range
AA43 01109A     1887         DATA 1,#4250                    of 1 - 4250.
                1888  *      Convert duration into 1/60s of a second
AA46 AB4A00     1889         DMUL 6,@FAC               Duration *6
AA49 06
AA4A AF4A00     1890         DDIV 100,@FAC             (duration * 6) / 100
AA4D 64
AA4E 8E4B4A     1891         $IF  @FAC1 .EQ. 0 THEN    If duration =0
AA51 54
AA52 904B       1892           INC  @FAC1              Set it to 1/60 of a second
                1893         $END IF
AA54 BCA37B     1894         ST   @FAC1,RAM(VRMSND+2) 3rd byte of the 1st block
AA57 4B
                1895  *                                      : INTERRUPT COUNT
                1896  ********************************************************
                1897  *      SOUND TABLE OF 10 BYTES IN CPU RAM (>00 - >09)  *
                1898  *  >00 ->05 : FREQUENCY CONTROL                        *
                1899  *  >06 ->08 : ATTENUATION CONTROL                      *
                1900  *  >09      : NOISE CONTROL(non-zero = noise encountered)*
                1901  *  >0A      : POINTER FOR CURRENT FREQUENCY CONTROL     *
                1902  *  >0B      : POINTER FOR CURRENT ATTENUATION CONTROL   *
                1903  *                          >00 ,>01  FOR REG. 0;       *
                1904  *                          >02 ,>03  FOR REG. 1;       *
                1905  *                          >04 ,>05  FOR REG. 2        *
                1906  *  Reg0:>8000, Reg1:>A000, Reg2:>C000                   *
                1907  *  INITIALIZE ATTENUATION CONTROL:                      *
                1908  *  Reg0:>9F, Reg1:>BF, Reg2:>DF                        *
                1909  ********************************************************
AA58 31000C     1910         MOVE 12 FROM ROM(#ATTREG) TO @>00
AA5B 00A9CF
AA5E 0F7E       1911  SOUND1 XML  SPEED                Parse the frequency value
AA60 01         1912         DATA PARCOM                  and insure a comma
AA61 06A3ED     1913         CALL CKSTNM               Must be a numeric
AA64 D24A00     1914         $IF  @FAC .LT. 0 GOTO SOUND2   Noise if negative
```

```
AA67 4AA6
AA69 310008  1915          MOVE 8 FROM ROM(#FLTS) TO @ARG  Constant 111834
AA6C 5CAA14
AA6F 0F09    1916          XML  FDIV                 P = 111834/FREQUENCY
AA71 0F7E    1917          XML  SPEED                Insure in range
AA73 02      1918          DATA RANGE
AA74 0303FF  1919          DATA 3,#1023              Range:  3 - 1023
             1920  *  GET THE L. S. 4 BITS AND M. S. 6 BITS OF 'P'
AA77 EB4A00  1921          DSRC @FAC,4
AA7A 04
AA7B E64A04  1922          SRL  @FAC,4
AA7E B5900A  1923          DOR  @FAC,*STADDR 1ST BYTE OF FREQUENCY CONTROL BYTES
AA81 4A
             1924  *                            BIT  7   6   5   4   3   2   1   0
             1925  *                                 1  <REG.>  0   < L.S. 4 OF P>
             1926  *
             1927  *                            2ND BYTE OF FREQUENCY CONTROL BYTES
             1928  *                                 0   0   <   M.S. 6 OF 'P'    >
             1929  *
AA82 940A    1930          INCT @STADDR              Advance pointer for next time
AA84 06A85B  1931          CALL ATTNUT               Get attenuation
             1932  *                          BIT  7   6   5   4   3   2   1
             1933  *                               1  <REG.>  1   0   0   0
AA87 B0900B  1934          AND  @FAC1,*VAR2            1  <REG.>  1   < ATTN/2 DB>
AA8A 4B
AA8B 900B    1935          INC  @VAR2                Advance pointer for next time
             1936
             1937  *  CHECK FOR END OF SOUND CALL
AA8D D642B6  1938  SOUND3 $IF  @CHAT .EQ. RPAR$ GOTO SOUND5   End of stmt ?
AA90 6AC4
AA92 0F7E    1939          XML  SPEED                If not right parenthesis
AA94 00      1940          DATA SYNCHK                  then must be at
AA95 B3      1941          DATA COMMA$                     a comma
AA96 D60A06  1942          $IF  @STADDR .NE. >06 GOTO SOUND1 If not 3 regs yet
AA99 4A5E
             1943  *       3 sound regs already - so must be noise control
AA9B 0F7E    1944          XML  SPEED                Get frequency(should be noise)
AA9D 01      1945          DATA PARCOM                  and insure a comma
AA9E 06A3ED  1946          CALL CKSTNM               Must be a numeric value
AAA1 D24A00  1947          $IF  @FAC .GE. 0 GOTO ERRBV  If not noise-error
AAA4 6D87
             1948  *                                                Bad value error.
             1949  *  NOISE CONTROL
AAA6 D609FF  1950  SOUND2 $IF  @>09 .NE. >FF GOTO ERRBA * BAD ARGUMENT ERROR
AAA9 4D83
AAAB 834A    1951          DNEG @FAC                 -(FREQUENCY)
AAAD 0F7E    1952          XML  SPEED                Insure in range
AAAF 02      1953          DATA RANGE                   of 1 - 8.
AAB0 010008  1954          DATA 1,#8
AAB3 924B    1955          DEC  @FAC1                0-7 (2ND BIT: 'T';
             1956  *                                              OTH, 1ST BITS: 'S'
AAB5 BC094B  1957          ST   @FAC1,@>09
AAB8 B609E0  1958          OR   >E0,@>09             Noise control byte:
             1959  *                            BIT  7   6   5   4   3   2   1   0
             1960  *                                 1   1   1   0   0  <T>  < S >
```

```
                    1961
                    1962  *  PUT ATTENUATION IN THE 2ND BYTE OF 1ST BLOCK
AABB 06AB5B         1963          CALL  ATTNUT              Get attenuation
AABE BCA37A         1964          ST    @FAC1,RAM(VRMSND+1)
AAC1 4B
                    1965  *                                  1   1   1   1   < ATTN/2  DB>
AAC2 4ABD           1966          BR    SOUND3              Go check for end of list
                    1967
AAC4 8610           1968  SOUND5  CLR   @VAR5               Pointer to sound table
AAC6 8E80CE         1969  SND05   $IF   @PRTNFN .EQ. 0 GOTO SOUND6  Wait until previous
AAC9 6AD5
AACB 03             1970          SCAN                            Is finished and
AACC 4AC6           1971          BR    SND05                        look for a break-key
AACE D67502         1972          $IF   @RKEY .NE. BREAK GOTO SND05   If not break-key
AAD1 4AC6
AAD3 4127           1973          BR    EXEC6C              If BREAK-KEY encountered
                    1974
                    1975  *  LOAD SOUND TABLE
                    1976
AAD5 BC8100         1977  SOUND6  ST    *VAR5,@>0100
AAD8 9010
AADA 9010           1978          INC   @VAR5               Next byte in table
AADC D6100A         1979          $IF   @VAR5 .NE. >0A GOTO SOUND6 If not finished
AADF 4AD5
AAE1 BF4A03         1980          DST   VRMSND,@FAC         Where the 2 blocks are
AAE4 79
AAE5 F64A01         1981          I/O   @FAC,1              Start sound from VDP list
AAE8 4AFC           1982          BR    LNKRTN              Return to caller
```

```
                    1984  ********************************************************
                    1985  *                Subprogram for 'HCHAR'
                    1986  ********************************************************
                    1987  HCHAR
AAEA 06AC6A         1988          CALL HVCHR          Get X,Y values,character,# of chars
AAED 8F4A6A         1989          $IF  @FAC .DEQ. 0 GOTO XPTRTN   If 0 characters
AAF0 F9
AAF1 08E000         1990  HCHAR1  FMT  '@VARO'              Display horizontally on screen
AAF4 FB
AAF5 934A           1991          DDEC @FAC                 Done yet?
AAF7 4AF1           1992          BR   HCHAR1               No - finish it
AAF9 BC7F02         1993  XPTRTN  ST   @MNUM,XPT            Restore x-pointer
AAFC 0F7E           1994  LNKRTN  XML  SPEED                Must be at
AAFE 00             1995          DATA SYNCHK                  a right
AAFF B6             1996          DATA RPAR$                      parenthesis
AB00 066A78         1997  LNKRT2  CALL CHKEND               Check end of statement
AB03 4D53           1998          BR   ERRSYN               If not end-of-stmt - ERROR
AB05 060012         1999          CALL RPL                  Return to caller
                    2000
                    2001
                    2002  ********************************************************
                    2003  *                Subprogram for 'VCHAR'
                    2004  ********************************************************
                    2005  VCHAR
AB08 06AC6A         2006          CALL HVCHR          GET X,Y VALUES,CHARACTER,# OF CHRS
AB0B 8F4A6A         2007          $IF  @FAC .DEQ. 0 GOTO XPTRTN    IF 0 CHARACTERS
AB0E F9
AB0F 08E000         2008  VCHAR1  FMT  '@VARO',31<          Display the character
AB12 9EFB
AB14 934A           2009          DDEC @FAC                 Done yet?
AB16 6AF9           2010          BS   XPTRTN               Yes - return
AB18 8E7E4B         2011          $IF  YPT .NE. 0 GOTO VCHAR1  If not at start of col
AB1B 0F
AB1C 907F           2012          INC  XPT                  Move X-ptr to right one col
AB1E 05AB0F         2013          B    VCHAR1
```

```
                    2015  ************************************************************
                    2016  *                  Subprogram FOR 'CHAR'
                    2017  ************************************************************
AB21 06A9DB         2018  CHARLY CALL COMB
                    2019
AB24 0F79           2020  CHAR5  XML  PGMCHR        Skip "(" or ",".
AB26 0F7E           2021         XML  SPEED         Get the first value.
AB28 01             2022         DATA PARCOM           and insure a comma
AB29 0F7E           2023         XML  SPEED         Insure in range
AB2B 02             2024         DATA RANGE            of 32-143
AB2C 20008F         2025         DATA 32,#143
                    2026  *
AB2F E34A00         2027         DSLL @FAC,3        Convert chr number to address.
AB32 03
AB33 A34A03         2028         DADD >300,@FAC     CORRECT FOR OFFSET
AB36 00
AB37 BD044A         2029         DST  @FAC,@VARY    SAVE IT
                    2030  *
AB3A 0F74           2031         XML  PARSE         GET STRING
AB3C B6             2032         DATA RPAR$
AB3D D&4065         2033         $IF  @FAC2 .NE. >65 GOTO ERRSNM   MUST BE STRING
AB40 4D57
AB42 350004         2034         MOVE 4 FROM @FAC4 TO @VAR5  VAR5 pointer to string v
AB45 104E
                    2035  *
                    2036  *      Start defining character description.
                    2037  *           VARY   Address of RAM for character description.
                    2038  *           VAR5   Pointer to string value.
                    2039  *           VAR7   Length of string value.
                    2040  *           VAR9   Temporary counter.
                    2041  *           VAR9+1 Temporary counter.
                    2042  *
AB47 C71200         2043         $IF  @VAR7 .DH. 64 THEN    Max 4 charcters at a time.
AB4A 404B51
AB4D BF1200         2044           DST  64,@VAR7            IGNORE THE EXCESSES
AB50 40
                    2045         $END IF
                    2046  CHAR4C
AB51 CB0407         2047         $IF @VARY .DHE. SPRVB GOTO CHARL4 Don't have space f
AB54 806BCE
AB57 BE4A30         2048         ST   ZERO,@FAC    FLOATING POINT ACCUMULATOR (>30)
AB5A 35000F         2049         MOVE 15 FROM @FAC TO @FAC1
AB5D 4B4A
                    2050  *
AB5F 8F126B         2051         $IF  @VAR7 .DEQ. 0 GOTO CHAR5O  Fill with zero.
AB62 80
AB63 CB1200         2052         $IF @VAR7 .DL. 16 THEN
AB66 106B72
AB69 34124A         2053           MOVE @VAR7 FROM RAM(@VAR5) TO @FAC Move whatever i
AB6C B010
AB6E 8712           2054           DCLR @VAR7
AB70 4B80           2055         $ELSE
AB72 350010         2056           MOVE 16 FROM RAM(@VAR5) TO @FAC   Move one charact
AB75 4AB010
AB78 A71200         2057           DSUB 16,@VAR7      Less num of bytes to move.
```

```
AB7B 10
AB7C A31000   2058            DADD 16,@VAR5      Move pointer.
AB7F 10
              2059          $END IF
              2060  *
              2061  CHAR50
AB80 BE144A   2062          ST   FAC,@VAR9     START TO LOAD THE CHARACTERS
AB83 BE1501   2063          $FOR @VAR9+1 = 1 TO 8
AB86 05AB8B
AB89 9015CE
AB8C 15086B
AB8F CA
AB90 B60C     2064            CLR  @BYTE             Clear dot-building byte
AB92 E20C04   2065  CHARL2    SLL  @BYTE,4           For loop(2 chars per byte)
AB95 BC5C90   2066            ST   *VAR9,@ARG
AB98 14
AB99 CA5C30   2067            $IF  @ARG .L. ZERO GOTO ERRBV  If < 0
AB9C 4D87
AB9E CE5C39   2068            $IF  @ARG .LE. NINE GOTO CHARL3 If in 0-9
ABA1 4BAD
ABA3 CA5C41   2069            $IF  @ARG .L. A GOTO ERRBV  If > 9 but < A
ABA6 4D87
ABA8 C65C46   2070            $IF  @ARG .H. F GOTO ERRBV  If > F
ABAB 6D87
ABAD A65C30   2071  CHARL3    S    ZERO,@ARG         Character - >30
ABB0 C65C0A   2072            $IF  @ARG .H. 10 THEN   If in A-F
ABB3 4BB8
ABB5 A65C07   2073              S    7,@ARG             Correct for that too
              2074            $END IF
ABB8 B40C5C   2075            OR   @ARG,@BYTE        Dot expression
ABBB 9014     2076            INC  @VAR9
ABBD DA1401   2077            TBR  @VAR9,0           1st half of row finished?
ABC0 4B92     2078            BR   CHARL2            Yes - do 2nd half
              2079  *                                (each takes half byte)
ABC2 BCB004   2080            ST   @BYTE,RAM(@VARY) Load characters
ABC5 0C
ABC6 9104     2081           DINC @VARY
ABC8 4B89     2082          $END FOR                 Load characters on next row
ABCA 8F124B   2083          $IF @VAR7 .DNE. 0 GOTO CHAR40 More char to describe.
ABCD 51
              2084  CHARL4
ABCE D642B3   2085          $IF @CHAT .EQ. COMMA$ GOTO CHAR5  More specified?
ABD1 6B24
ABD3 4AFC     2086          BR   LNKRTN            Return.
```

```
                2088  ************************************************************
                2089  *                  Subprogram for "KEY"                    *
                2090  ************************************************************
ABD5 06AC28     2091  KEY     CALL LPAR           GET KEY UNIT
ABD8 0F7E       2092          XML  SPEED          Insure in range
ABDA 02         2093          DATA RANGE            of 0 - 5.
ABDB 000005     2094          DATA 0,#5
ABDE 06A9A0     2095          CALL KEYJOY         GET VARIABLES FOR CODE AND STATUS
                2096  *                             AND SCAN KEYBOARD
                2097  *                           KEYJOY RETURNS KEY STATUS
ABE1 6BEC       2098          BS   KEY1B          KEY STATUS = 1
ABE3 834A       2099          DNEG @FAC                 Assume status = -1
ABE5 D675FF     2100          $IF  @RKEY .EQ. >FF THEN  But correct if = 0        1
ABE8 4BEC
ABEA 874A       2101            DCLR @FAC         KEY STATUS = 0                  1
                2102          $END IF
ABEC 0F7C       2103  KEY1B   XML  ASSGNV         ASSIGN VALUE TO VARIABLE
ABEE BF4A40     2104          DST  >4001,@FAC     RE-STORE F.P. 1 IN FAC
ABF1 01
ABF2 8E756C     2105          $IF  @RKEY .EQ. 0 GOTO KEY2 If key-code=0
ABF5 04
ABF6 C67563     2106          $IF  @RKEY .H. >63 GOTO KEY1C  If key-code > >63
ABF9 6C00
ABFB BC4B75     2107          ST   @RKEY,@FAC1    FLOATING FORMAT(>40__000000000000)
ABFE 4C06       2108          BR   KEY2A
AC00 834A       2109  KEY1C   DNEG @FAC           KEY CODE ASSIGNED TO -1
AC02 4C06       2110          BR   KEY2A
AC04 874A       2111  KEY2    DCLR @FAC           (>0000000000000000)
AC06 0F7C       2112  KEY2A   XML  ASSGNV         ASSIGN VALUE TO VARIABLE
AC08 4AFC       2113          BR   LNKRTN
```

```
                    2115  ************************************************************
                    2116  *       SUBROUTINE FOR 'JOYSTICK'
                    2117  ************************************************************
AC0A 06AC28         2118  JOYST   CALL LPAR           KEY UNIT
AC0D 0F7E           2119          XML  SPEED          Insure in range
AC0F 02             2120          DATA RANGE             of 1 - 4
AC10 010004         2121          DATA 1,#4
AC13 06A9A0         2122          CALL KEYJOY         GET VARIABLES FOR X, Y
                    2123  *                              AND SCAN KEYBOARD
AC16 BC0076         2124          ST   @JOYY,@VARO    JOYSTICK Y POSITION
AC19 06A98A         2125          CALL JOYXY          -4 - +4
AC1C BF4A40         2126          DST  >4001,@FAC      RE-STORE F.P. 1 IN FAC
AC1F 01
AC20 BC0077         2127          ST   @JOYX,@VARO    JOYSTICK X POSITION
AC23 06A98A         2128          CALL JOYXY          -4 - +4
AC26 4AFC           2129          BR   LNKRTN
```

```
                2131  ************************************************************
                2132  *       INSURE LEFT PARENTHESIS AND THEN PARSE TO A COMMA   *
                2133  ************************************************************
AC28 0F7E       2134  LPAR    XML   SPEED              Must be
AC2A 00         2135          DATA  SYNCHK                at a
AC2B B7         2136          DATA  LPAR$                    left parenthesis
AC2C 0F74       2137          XML   PARSE              Do the parse
AC2E B3         2138          DATA  COMMA$             Stop on a comma
AC2F 0F7E       2139          XML   SPEED              Must be
AC31 00         2140          DATA  SYNCHK                at a
AC32 B3         2141          DATA  COMMA$                   comma
AC33 00         2142          RTN                           .
                2143
                2144  ************************************************************
                2145  *       SUBROUTINE FOR 'RANGE' USED IN ALL SOUND AND GRAPHICS
                2146  ************************************************************
                2147  *
AC34 0F7E       2148  RAN16   XML   SPEED              Insure in range
AC36 02         2149          DATA  RANGE                 of 1 to 16
AC37 010010     2150          DATA  1,#16
AC3A 924E       2151          DEC   @FAC1              Adjust to internal range.
AC3C 00         2152          RTN
```

```
                2154  ********************************************************************
                2155  *       SUBROUTINE TO GET ROW,COLUMN VALUES
                2156  ********************************************************************
AC3D 06AC2B     2157  GPHV    CALL LPAR                 Insure '(', parse, insure ')'
AC40 0F7E       2158          XML  SPEED                Insure in range
AC42 02         2159          DATA RANGE                   of 1 - 24.
AC43 010018     2160          DATA 1,#24
AC46 924B       2161          DEC  @FAC1                Adjust to internal range
AC48 BC027F     2162          ST   XPT,@MNUM
AC4B BC7E4B     2163          ST   @FAC1,YPT            Set row pointer
AC4E 0F7E       2164          XML  SPEED                Get column value
AC50 01         2165          DATA PARCOM                  and insure a comma
AC51 0F7E       2166          XML  SPEED                Insure in range
AC53 02         2167          DATA RANGE                   of 1 to 32
AC54 010020     2168          DATA 1,#32
AC57 924B       2169          DEC  @FAC1                Internal range: 0 - 31
AC59 BC7F4B     2170          ST   @FAC1,XPT            Set column pointer
AC5C 00         2171          RTN
```

```
                2173  *
                2174  *          Subprogram to control border color
                2175  *        Character background is also affected
                2176  *              since transparent is used.
                2177  *
AC5D 06A565     2178  BORDER CALL PARFF              Insure '(', and parse
AC60 06AC34     2179         CALL RAN16              Check 1-16 & put in internal
AC63 3D0001     2180         MOVE 1 FROM @FAC1 TO VDP(7)  Load VDP register
AC66 0748
AC68 4B00       2181         BR   LNKRT2             Return to BASIC program
```

```
                    2183  *       GET ROW, COLUMN VALUES AND NUMBER OF CHARACTERS
                    2184
AC6A 06AC3D         2185  HVCHR   CALL GPHV                  GET X,Y VALUES
AC6D 0F74           2186          XML  PARSE
AC6F B6             2187          DATA RPAR$
AC70 06A975         2188          CALL FTINT
AC73 A24D60         2189          ADD  OFFSET,@FAC1
AC76 BC004B         2190          ST   @FAC1,@VAR0          SAVE THE CHARACTER
AC79 BF4A00         2191          DST  1,@FAC               ASSUME 1 CHARACTER
AC7C 01
AC7D D642B6         2192          $IF @CHAT .NE. RPAR$ THEN  If not right parenthesis i
AC80 6C8C
AC82 0F7E           2193            XML  SPEED              Must be                    i
AC84 00             2194            DATA SYNCHK                at a                    i
AC85 B3             2195            DATA COMMA$                   comma                i
AC86 0F74           2196            XML  PARSE              # OF CHARACTERS            i
AC88 B6             2197            DATA RPAR$                                         i
AC89 06A975         2198            CALL FTINT              FLOATING TO INTEGER        i
                    2199          $END IF
AC8C 00             2200          RTN
```

```
                2202  *
                2203  ***************************************************************
                2204  *                                                             *
                2205  *       ERRWXY - Is the subroutine for CALL ERR(W,X,Y,Z)       *
                2206  *       The parameters indicate:                              *
                2207  *                                                             *
                2208  *       W - The error code # of the error                     *
                2209  *       X - Indicates whether execution(-1) error or          *
                2210  *           I/O (0-255) error on LUNO 0-255                   *
                2211  *       Y - Indicates the severity code of the error          *
                2212  *       Z - Line number of the error                          *
                2213  *                                                             *
                2214  *       ERR Can be called with 2 forms:                       *
                2215  *            CALL ERR(W,X,Y,Z) and CALL ERR(W,X)              *
                2216  *       If ERR is called and no error has occurred then       *
                2217  *            all values returned are zero.                    *
                2218  *                                                             *
                2219  ***************************************************************
                2220  ERRWXY
                2221  *
                2222  *
ACBD BD526E     2223          DST  @VSPTR,@FAC8      Get a temp VSPTR
AC90 C55224     2224          $WHILE @FAC8 .DH. @STVSPT While not a bottom of stac
AC93 4CC7
AC95 BC5CE0     2225            ST  RAM(2(FAC8)),@ARG   Keep ID code in ARG area.
AC98 0252
AC9A D65C69     2226            $IF @ARG .EQ. >69 THEN       *** ERROR entry.
AC9D 4CA6
AC9F 06A9E1     2227               CALL SQUISH         Squish it out of the stack
ACA2 0F77       2228               XML  VPUSH          Put permanent copy of error
                2229  *                                    entry on stack
ACA4 4CD3       2230               BR    ERR10         Jump out now
                2231  *            Jump always
                2232            $END IF
ACA6 D65C67     2233            $IF  @ARG .EQ. >67 THEN      *** FOR entry.
ACA9 4CB1
ACAB A75200     2234               DSUB  32,@FAC8      Skip it
ACAE 20
ACAF 4CC5       2235            $SELSE
ACB1 D65C66     2236               $IF  @ARG .EQ. >66 THEN  *** GOSUB entry.
ACB4 4CBC
ACB6 A75200     2237                  DSUB  8,@FAC8    Skip it
ACB9 08
ACBA 4CC5       2238               $SELSE                     *** SUBPROGRAM entry
ACBC D65C6A     2239                 $IF @ARG .NE. >6A GOTO ERRSYN  * SYNTAX ERROR
ACBF 4D53
ACC1 A75200     2240                 DSUB  16,@FAC8   Skip it.
ACC4 10
                2241               $END IF
                2242            $END IF
ACC5 4C90       2243          $END WHILE
ACC7 BF4A00     2244          DST  >0080,@FAC         No error entry there so
ACCA 80
ACCB BF4C69     2245          DST  >6900,@FAC2           fake one
ACCE 00
```

```
ACCF 874E     2246          DCLR @FAC4
ACD1 8750     2247          DCLR @FAC6
ACD3 0F77     2248  ERR10   XML  VPUSH              Push the temporary entry or
              2249  *                                  top of stack
              2250  *       Code to get "W" in
              2251  *
ACD5 06A9DB   2252          CALL COMB               Check for left par.
ACD8 06AD43   2253          CALL ERRC05             Pick up user's symbol
ACDB BC4BEF   2254          ST   RAM(-8(VSPTR)),@FAC1  Get error code
ACDE FFFB6E
ACE1 0F80     2255          XML  CIF                Convert it to floating
ACE3 0F7C     2256          XML  ASSGNV             Assign it
              2257  *
              2258  *       Code to get "X" in
ACE5 06AD3E   2259          CALL ERRCOM          Check syntax & get user's symbol
ACE8 DAEFFF   2260          $IF  .BIT7 RAM(-7(VSPTR)) .EQ. 0 THEN  If execution i
ACEB F96E80
ACEE 4CFA
ACF0 310008   2261               MOVE 8 FROM ROM(#FLT1) TO @FAC  Make it such    1
ACF3 4AA3F3
ACF6 824A     2262               DNEG  @FAC         Make it a negative          1
ACF8 4DC2     2263          $SELSE                                               1
ACFA BC4BEF   2264               ST  RAM(-5(VSPTR)),@FAC1   Get I/O LUNO number  1
ACFD FFFB6E
AD00 0F80     2265               XML CIF            Convert it to floating      1
              2266          $END IF
AD02 0F7C     2267          XML  ASSGNV
              2268  *
              2269  *       Code to get "Y" in
              2270  *
AD04 D642B6   2271          $IF  @CHAT .NE. RPAR$  If long form of CALL ERR      1
AD07 6D39
AD09 06AD3E   2272           CALL  ERRCOM      Check syntax & get user's symbol  1
AD0C BC4BEF   2273           ST    RAM(-7(VSPTR)),@FAC1  Get severity code     1
AD0F FFF96E
AD12 B24D7F   2274           RB    @FAC1,7          Reset execution / I/O flag  1
AD15 0F80     2275           XML   CIF              Convert it                  1
AD17 0F7C     2276           XML   ASSGNV           Assign it                   1
              2277  *
              2278  *       Code to get "Z" in
              2279  *
AD19 06AD3E   2280           CALL  ERRCOM           Check syntax & get symbol   1
AD1C BD4CEF   2281           DST   RAM(-2(VSPTR)),@FAC2  Get line pointer       1
AD1F FFFE6E
AD22 BD4A4C   2282           DST   @FAC2,@FAC                                   1
AD25 8F4C6D   2283           $IF @FAC2 .DNE. 0 THEN   If line number exists     2
AD28 35
AD29 974C     2284             DDECT @FAC2          Point to the ln #           2
AD2B 066034   2285             CALL GRSUB1          Read ln # (2 bytes) from VDP 2
              2286  *                                   or ERAM (use GREAD)
AD2E 4C       2287             DATA FAC2            @FAC2:Source addr. on ERAM/VD
AD2F BD4A5C   2288             DST @EEE,@FAC        Put the ln # in FAC         2
AD32 B24A7F   2289             AND >7F,@FAC         Reset the breakpt if any    2
              2290           $END IF
AD35 0F80     2291           XML   CIF              Convert it
```

```
AD37 0F7C     2292           XML   ASSGNV          Assign it
              2293          $END IF
AD39 0F78     2294          XML   VPOP             Trash the temporary entry
AD3B 05AAFC   2295          B     LNKRTN           Return from subprogram
              2296  *       Must be long branch because of AND above
              2297  ********************************************************
              2298  *
              2299  *
              2300  ERRCOM
AD3E D642B3   2301          $IF @CHAT .NE. COMMA$ GOTO ERRSYN   Check for comma
AD41 4D53
              2302  ERRC05
AD43 0F79     2303          XML   PGMCHR            Get the next character.
AD45 CA4280   2304          $IF @CHAT .HE. >80 GOTO ERRSYN  Do not allow token.
AD48 6D53
AD4A 0F7A     2305          XML   SYM               Collect name& s.t. entry
AD4C 0F78     2306          XML   SMB               Get value space
AD4E 0F77     2307          XML   VPUSH             Push it
& CLR@FAC
Set
AD50 B64A     2308          CLR   @FAC              Set up for conversion
AD52 00       2309          RTN
              2310  *
              2311  * CHANGE IN ADDRESS OF THE ERROR CALLS WILL AFFECT
              2312  * THE FILE SUBS.....
              2313  *       ERROR mesasges called form this file
              2314  *
AD53 066A84   2315  ERRSYN CALL ERR$$             * SYNTAX ERROR
AD56 03       2316          DATA 3                  (Shared by SUBS)
              2317  *
AD57 066A84   2318  ERRSNM CALL ERR$$             * STRING-NUMBER MISMATCH
AD5A 07       2319          DATA 7                  (Shared by SUBS)
              2320  *
AD5B 066A84   2321  ERRMUV CALL ERR$$             * IMPROPERLY USED NAME
AD5E 09       2322          DATA 9
              2323  *
AD5F 066A84   2324  ERRMEM CALL ERR$$             * MEMORY FULL
AD62 0B       2325          DATA 11
              2326  *
AD63 066A84   2327  ERRSO  CALL ERR$$             * STACK OVERFLOW
AD66 0C       2328          DATA 12
              2329  *
AD67 066A84   2330  ERRNWF CALL ERR$$             * NEXT WITHOUT FOR
AD6A 0D       2331          DATA 13
              2332  *
AD6B 066A84   2333  ERRFNN CALL ERR$$             * FOR/NEXT NESTING
AD6E 0E       2334          DATA 14
              2335  *
AD6F 066A84   2336  ERRSNS CALL ERR$$             * MUST BE IN SUBPROGRAM
AD72 0F       2337          DATA 15
              2338  *
AD73 066A84   2339  ERRRSC CALL ERR$$             * RECURSIVE SUBPROGRAM CALL
AD76 10       2340          DATA 16
              2341  *
AD77 066A84   2342  ERRRWG CALL ERR$$             * RETURN WITHOUT GOSUB
AD7A 12       2343          DATA 18
              2344  *
```

```
AD7B 066A84  2345  ERRBS  CALL ERR$$          * BAD SUBSCRIPT
AD7E 14      2346         DATA 20
             2347  *
AD7F 066A84  2348  ERRLNF CALL ERR$$          * LINE NOT FOUND
AD82 16      2349         DATA 22
             2350  *
AD83 066A84  2351  ERRBA  CALL ERR$$          * BAD ARGUMENTS
AD86 1C      2352         DATA 28
             2353  *
AD87 066A84  2354  ERRBV  CALL ERR$$          * BAD VALUE
AD8A 1E      2355         DATA 30               (Shared by SUBS)
             2356  *
AD8B 066A84  2357  ERRIAL CALL ERR$$          * INCORRECT ARGUMENT LIST
AD8E 1F      2358         DATA 31               (Shared by SUBS)
             2359  *
AD8F 066A84  2360  ERRSNF CALL ERR$$          * SUBPROGRAM NOT FOUND
AD92 25      2361         DATA 37
             2362  *
             2363  *   Other error messages appear in this program
             2364  *
             2365  * ERRRDY     * READY                          DATA 0
             2366  * ERRBRK     * BREAK POINT                    DATA 1
             2367  * ERROLP     * ONLY LEGAL IN A PROGRAM        DATA 27
             2368  *
             2369  * WRNNO1     * NUMERIC OVERFLOW               DATA 2
             2370  * WRNNO2
             2371  * WRNST1     * STRING TRUNCATED               DATA 19
             2372  * WRNST2
             2373  * WRNLNF     * LINE NOT FOUND                 DATA 38
             2374  *
             2375         END
```

ERRORS= 0

LENGTH= 3475 (>0D93)

497 SYMBOLS USED

SYMBOL	VALUE	DEF	REFERENCE TABLE
A	0041	382	2069
ABS$	00CB	331	
ACCEP$	00A4	291	
ACCEPT	B01E	86	565
ACCEPX	A19E	565	542
ALL$	00EC	354	
AND$	00BB	314	
APPEN$	00F9	367	
ARG	005C	194	195 196 197 198 199 201 202 203 719 742 795 797 800 804 816 823 826 831 834 841 876 880 1013 1066 1072 1077 1078 1079 1080 1091 1263 1274 1276 1279 1321 1323 1325 1330 1332 1333 1335 1336 1358 1366 1384 1395 1469 1476 1532 1535 1915 2066 2067 2068 2069 2070 2071 2072 2073 2075 2225 2226 2233 2236 2239
ARG2	005E	195	1068 1070 1078 1085 1371 1372 1373 1375 1378 1389 1394 1396 1463 1641 1647 1649
ARG3	005F	196	
ARG4	0060	197	1025 1274
ARG6	0062	198	1019 1268 1271 1275
ARG8	0064	199	1275 1280
ASC	ABBA	1639	395 494
ASC01	A51A	1134	544
ASSGNV	007C	113	1542 1724 1802 2103 2112 2256 2267 2276 2292
AT$	00F0	358	
ATN$	00CC	332	
ATTNUT	AB5B	1600	1931 1963
ATTREG	A9CF	1827	1910
BASE	0043	169	
BASE$	00F1	359	
BEEP$	00EE	356	
BORDER	AC5D	2178	421
BREAK	0002	375	1972
BREAK$	00BE	269	
BRKBIT	0006	214	672 676
BUFLEV	0046	172	511
BYTE	000C	139	951 958 960 963 1002 1003 1004 1005 1009 1016 1080 1087 1121 1163 1164 1165 1169 1205 1206 1209 1212 1219 1220 1221 1229 1260 1261 1266 1269 1270 1282 1283 1287 1292 1317 1322 1325 1332 1334 1335 1336 1338 2064 2065 2075 2080
CALL$	009D	284	
CB	0008	1	1717
CHAR	0004	1	
CHAR40	AB51	2046	2083
CHAR5	AB24	2020	2085
CHAR50	AB80	2061	2051
CHARL2	AB92	2065	2078
CHARL3	ABAD	2071	2068
CHARL4	ABCE	2084	2047
CHAPLY	AB21	2018	418
CHAT	0042	168	600 603 607 631 634 636 652 671 674 1196 1465 1507 1684 1686 1688 1734 1738 1762 1808 1818 1832 1938 2085 2192 2271 2301 2304
CHKEND	6A78	69	613 863 1359 1367 1399 1997

SYMBOL	VALUE	DEF	REFERENCE TABLE
CHR$$	00D6	342	
CHR$01	A4FF	1119	552
CHRTAB	601C	59	476
CIF	0080	118	1106 2255 2265 2275 2291
CIRCU$	00C5	324	
CKSTNM	A3ED	912	872 1603 1913 1946
CLEAR	A9FD	1849	413
CLOSE	800A	81	564
CLOSE$	00A0	287	
CLOSEX	A19B	564	537
CLRRTN	A116	498	470
CLSALL	8012	84	
CNS	0073	104	1158
COL10	A929	1736	1740
COL20	A936	1743	1764
COLON$	00B5	308	
COLOR	A91D	1731	414
COM3	A9DB	1832	1857 2018 2252
COMMA$	00B3	306	1403 1738 1762 1808 1941 2085 2138 2141 2195 2301
COMPCT	0070	102	
CONC$	00B8	311	1000
CONCOS	A48E	1029	1019 1024
CONCAT	A438	996	461
CONPI	A250	710	707
CONT	0075	106	390 616 708 721 728 849 884 888 899 1030 1090 1107 1126 1170 1190 1345 1352 1414 1563
CONV1	ABE5	1696	399
COS$	00CD	333	
CPL	0010	53	698
CRNBUF	0820	243	441
CSNTMP	0390	233	1700
CSNTP1	03BA	234	1699
CSNUM	0010	100	1228 1697
DATA	0034	161	1621 1629
DATA$	0093	274	
DATAST	AB6E	1617	394 1627
DATST1	AC89	1630	1623
DDD1	0054	188	966 1644 1645
DEF$	0089	264	
DELET	8002	77	567
DELET$	0099	280	
DELETX	A1A4	567	549
DELINK	A80A	1566	398 1559
DIGIT$	00E9	351	
DIM$	008A	265	
DISO	6A7C	70	1650
DISPL$	00A2	289	
DISPL1	8000	76	566
DISPLA	0009	1	
DISPLX	A1A1	566	550
DIVI$	00C4	323	
EDGECH	007F	385	
EEE	0050	187	2288
EEE1	0058	190	644 964 1377 1383 1385 1389 1428 1621 1647
ELSE$	0081	256	

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

END$    008B   266
ENDSCR  02FE   220
ENLN    0032   160     637 1371 1420
ENTO9   6A80    72    1514
ENTER   6A7E    71    1520
EOF     801C    85     556
EOFX    A183   556     543
EQUAL$  00BE   317    1507
ERASE$  00EF   357
ERDR$   A8AB  1657     456
ERR$$   6A84    74     480  517 1363 2315 2318 2321 2324 2327 2330 2333
                      2336 2339 2342 2345 2348 2351 2354 2357 2360
ERR10   ACD3  2248    2230
ERRBA   AD83  2351    1136 1187 1189 1666 1950
ERRBRK  A143   516
ERRBS   AD7B  2345    1663
ERRBV   AD87  2354    1058 1259 1660 1788 1789 1947 2067 2069 2070
ERRCO5  AD43  2302    1858 2253
ERRCOD  0022   152     452  498  615  997 1462 1657
ERRCOM  AD3E  2300    2259 2272 2280
ERRFNN  AD6B  2333    1669
ERRIAL  AD8B  2357    1487 1668 1672
ERRLN   038A   232     650  653
ERRLNF  AD7F  2348     645 1661
ERRMEM  AD5F  2324    1659
ERRMUV  AD5B  2321    1671 1821
ERRNWF  AD67  2330    1670
ERROLP  A699  1363    1459 1675
ERROR$  00A5   292
ERRRDY  A0E9   479
ERRRSC  AD73  2339    1673
ERRRWG  AD77  2342    1667
ERRSNF  AD8F  2360     693  701 1674
ERRSNM  AD57  2318     734  738  912  983 1001 1104 1135 1156 1186 1551
                      1553 1664 1784 2033
ERRSNS  AD6F  2336    1676
ERRSO   AD63  2327     465 1503 1665
ERRSYN  AD53  2315     607  614  652  674 1196 1658 1662 1684 1808 1818
                      1832 1998 2239 2301 2304
ERRWXY  AC8D  2220     423
EXEC    A09C   434     392
EXEC1   A0A9   438     396
EXEC20  A0BC   452     443
EXEC6C  A127   508    1362 1973
EXEC6D  A12B   509     397
EXECBK  A11A   503     454
EXECG   0076   107     450
EXECND  A0DF   475     453
EXECTR  A0ED   484     455
EXP$    0076   208
EXP$$   00CE   334
EXTRAM  002E   158     435  436  504  509 1641 1644
F       0046   383    2070
FAC     004A   174     175  176  177  178  179  180  181  182  183  184
                      185  186  187  188  189  190  192  634  644  707
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
			719 742 791 792 808 813 814 837 873 874
			879 883 886 887 893 909 954 1013 1058 1066
			1068 1105 1138 1259 1260 1263 1273 1276 1278 1283
			1311 1377 1469 1474 1476 1489 1528 1530 1532 1533
			1535 1537 1538 1584 1588 1686 1716 1722 1748 1760
			1789 1796 1799 1812 1820 1837 1859 1860 1881 1882
			1889 1890 1914 1921 1922 1923 1947 1951 1980 1981
			1989 1991 2007 2009 2027 2028 2029 2048 2049 2053
			2056 2062 2099 2101 2104 2109 2111 2126 2191 2244
			2261 2262 2282 2288 2289 2308
FAC1	004B	175	636 1122 1137 1312 1607 1611 1688 1717 1718 1719
			1720 1721 1753 1758 1773 1794 1806 1891 1892 1894
			1934 1955 1957 1964 2049 2107 2151 2161 2163 2169
			2170 2180 2189 2190 2254 2264 2273 2274
FAC10	0054	182	1227 1696 1699 1704 1785 1788
FAC11	0055	183	1157 1159 1160 1165
FAC12	0056	184	697 1161 1164 1222 1223 1224 1229 1700
FAC14	0058	185	1838 1839 1841
FAC15	0059	186	694 695 1513
FAC2	004C	176	637 638 643 645 646 649 650 734 738 904
			905 912 956 983 1001 1199 1226 1315 1317 1464
			1478 1479 1486 1487 1490 1536 1550 1583 1720 1784
			2033 2245 2281 2282 2283 2284 2287
FAC3	004D	177	907 908 1479 1492
FAC4	004E	178	894 955 1015 1084 1137 1203 1216 1269 1273 1289
			1326 1494 1538 1585 1586 2034 2246
FAC6	0050	179	793 815 950 951 1002 1016 1021 1022 1026 1071
			1073 1077 1087 1105 1125 1136 1169 1187 1204 1205
			1265 1266 1270 1271 1288 1321 1330 1495 2247
FAC7	0051	180	1461 1471 1473
FAC8	0052	181	504 505 901 902 909 952 959 971 1017 1088
			1217 1327 1416 1419 1420 1425 1428 1530 1533 1837
			1839 1841 1841 2223 2224 2225 2234 2237 2240
FADD	0006	94	798 827
FCOMP	000A	97	744
FDIV	0009	96	1916
FFF1	0056	189	963 1643
FIGURE	0002	1	
FIXED$	00FA	368	
FLAG	0045	171	513 601 604 605 608 609 672 676 1344 1351
			1505 1523
FLT1	A3F3	915	1716 1812 1859 2261
FLTINT	0012	101	1787
FLTS	AA14	1866	1915
FMUL	0008	95	796 801 805 817 824 832 835 842 846 847
			877 881
FOR$	008C	267	
FPERAD	006C	205	1786
FREPTR	0040	167	1495
FTINT	A975	1783	2188 2198
GCHAR	A8FA	1714	415
GETSTR	0071	103	953 1011 1082 1214 1318
GO$	0085	260	
GOSUB$	0087	262	
GOTO$	0086	261	

SYMBOL	VALUE	DEF	REFERENCE TABLE
GPHV	AC3D	2157	1714 2185
GPLCAL	A22F	693	463
GREAD1	008C	122	1646
GREAT	003E	381	495
GREAT$	00C0	319	
GRINT	0022	55	802 833 878
GRSUB1	6034	62	2285
GRSUB2	802C	89	1618
GRSUB3	802E	90	642 1374 1423
GVWITE	008B	121	967
GWSUB	6036	63	1387 1426
HCHAR	AAEA	1987	416
HCHAR1	AAF1	1990	1992
HVCHR	AC6A	2185	1988 2006
IF$	0084	259	
IMAGE$	00A3	290	
INPUT	8006	79	562
INPUT$	0092	273	
INPUTX	A195	562	535
INT$	00CF	335	
INTARG	A975	1784	1057 1065 1120 1258 1310
INTRND	A28D	748	402 437
IOSTRT	003C	165	
JOYST	AC0A	2118	420
JOYX	0077	128	2127
JOYXY	A98A	1794	2125 2128
JOYY	0076	127	2124
KEY	ABD5	2091	419
KEY1B	ABEC	2103	2098
KEY1C	AC00	2109	2106
KEY2	AC04	2111	2105
KEY2A	AC06	2112	1861 2108 2110
KEYBD	0074	125	1811 1814
KEYJOY	A9A0	1806	2095 2122
LEN$	00D5	341	
LEN01	A4F3	1103	551
LEN02	A4FB	1106	1139 1284
LESS	003C	380	492
LESS$	00BF	318	
LET$	008D	268	
LINE	ABD3	1684	393 1370
LINEGP	A6A4	1370	1360 1404
LINK1	A026	412	
LINK2	A030	413	412
LINK3	A03A	414	413
LINK4	A044	415	414
LINK5	A04E	416	415
LINK6	A058	417	416
LINK7	A062	418	417
LINKA	A06B	419	418
LINKB	A073	420	419
LINKC	A07D	421	420
LINKD	A088	422	421
LINKE	A094	423	422
LINKS1	AD9E	67	423

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

LINPUT  8030    91     568
LINPUX  A1A7   568     532
LITS05  A3FB   950     391
LITS06  A42C   971    1124 1168
LITS07  A404   953     972
LITS08  A406   954    1018 1089 1328
LITS09  A411   958    1218
LN$     00C9   329     631 1684
LNBUF   0036   162    1617 1619 1624 1625 1626
LNGP1   A6AE  1373    1379
LNGP2   A6C2  1381    1377
LNGP2B  A6E6  1399    1409
LNGP4   A6FA  1414    1400
LNKRT2  AB00  1997    1852 2181
LNKRTN  AAFC  1994     404 1738 1762 1982 2086 2113 2129 2295
LOG$    00D0   336
LPAR    AC28  2134     733 1053 1248 1303 1880 2091 2118 2157
LPAR$   00E7   310    1196 1465 1733 1832 2136
LSUBP   0048   173     512
M$EDIT  6010    45      58   59   60   61   62   63
M$FLMG  8000    47      76   77   78   79   80   81   82   83   84   85
                        86   87   88   89   90   91
M$MON   0010    43      53   54   55   56
M$PSCN  6A70    44      69   70   71   72   73   74
M$SUB   AD98    46      65   66   67
MAXMIN  A26D   733     717  726
MINUS$  00C2   321
MNUM    0002   134    1993 2162
MSGBRK  6048    60
MULT$   00C3   322
NBREAK  A68B  1358     525
NEXT$   0096   277     607  674
NINE    0039   379    2068
NLNADD  02E2   219     487  488  490  496
NMAX    A258   716     539
NMAX$1  A25E   719     727
NMIN    A265   726     540
NOT$    00BD   316
NPI     A248   706     538
NREAD   800E    83     559
NREADX  A18C   559     529
NRND    A29F   791     553
NRNDMZ  A37F   863     534
NTRACE  A681  1344     527
NUDTAB  0028   155     444
NUDTB   A147   523     444
NUM$    00C8   328
NUMBE$  00FD   371    1734
NUMER$  00E8   350
NUMVAR  A9BD  1818    1715 1807 1810
NUNBRK  A69D  1366     526
NUNTRC  A686  1351     528
OFFSET  0060   250     385  492  495 1718 1850 2189
ON$     009B   282
ONBRK   A218   669     460
```

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

ONECHR  0378   226    1122 1123
ONERR   A1DA   629     458
ONERR2  A1F2   641     647
ONERR4  A206   648     644
ONWARN  A1AA   598     462
ONWRN5  A1CF   612     655  679
ONWRN7  A1D1   613     700
OPEN    8008    80     563
OPEN$   009F   286
OPENX   A198   563     536
OPTIO$  009E   285
OR$     00BA   313
OUTPU$  00F7   365
PARCOM  0001   116    1056 1251 1744 1751 1912 1945 2022 2165
PARFF   A565  1196    1103 1119 1134 1155 1185 2178
PARSE   0074   105     736  870  999 1060 1197 1253 1305 1467 1544 1601
                      1755 1770 2031 2137 2186 2196
PERMA$  00FB   369
PGMCHR  0079   110     599  612  630  633  635  670 1502 1522 1541 1562
                      1685 1687 1689 1763 1809 2020 2303
PGMPTR  002C   157     441  508  695  696  697 1361 1489 1501 1515 1560
                      1561
PLUS$   00C1   320
POP     0005     1
POS$    00D9   345
POS01   A5BD  1248     545
POS02   A5F8  1271    1290
POS04   A606  1276    1281
POS06   A615  1282    1268
POS08   A617  1283    1293
POS10   A61C  1287    1277
POS12   A624  1292    1265 1267 1272
PRGFLG  0044   170     434  475  503 1362 1458
PRINT   8004    78     560
PRINT$  009C   283     600
PRINTX  A18F   560     530
PRTNFN  00CE    49    1883 1969
PUSSTR  A430   983     998 1054 1249 1252 1304
RAM     0001     1     435  492  495  508  509  510  511  512  513  514
                       650  653  748  791  808  813  837  879  883  886
                       887  909  960  960 1003 1025 1025 1122 1137 1165
                      1211 1221 1223 1276 1276 1311 1338 1338 1394 1396
                      1478 1501 1504 1516 1524 1525 1526 1526 1538 1551
                      1553 1560 1569 1573 1574 1576 1578 1586 1641 1699
                      1700 1760 1773 1820 1837 1841 1841 1879 1894 1964
                      2053 2056 2080 2225 2254 2260 2264 2273 2281
RAMFLG  0089   215     693 1640
RAMFRE  0086   211    1593
RAMTOP  0084   210     952 1382 1459 1592
RAN16   AC34  2148    1752 1757 1772 2179
RANDOM  0078   129     904  907
RANGE   0002   117    1605 1746 1886 1918 1953 2024 2093 2120 2149 2159
                      2167
READ$   0097   278
REC     8022    88     558
```

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

RECX    A189   558     524
REM$    009A   281
RESTO$  0094   275
RESTOR  800C    82     561
RESTOX  A192   561     533
RETUR$  0088   263
RKEY    0075   126    1972 2100 2105 2106 2107
RND$    00D7   343
RNDA1   A357   854     795  816
RNDA2   A34F   853     823
RNDC1   A367   856     797
RNDC2   A35F   855     826
RNDEM   A377   858     800  831  841  876
RNDEP   A36F   857     804  834  880
RNDM1   A3C0   893     864
RNDM2   A3D2   901     895  897
RNDX1   03A5   238     791  808  883  887  896
RNDX2   03A0   237     748  813  837  879  886  898
ROM     000A     1     707  748  795  797  800  804  816  823  826  831
                       834  841  876  880 1716 1812 1859 1879 1910 1915
                      2261
RPAR$   00B6   309     737  741 1061 1064 1254 1257 1306 1309 1602 1756
                      1771 1938 1996 2032 2187 2192 2197 2271
RPL     0012    54    1999
RPT$01  A628  1303     541
RPT$02  A667  1331    1339
RSTK    0088   216
RTNB    0082   119     499
RTNG    0026   154     443
RTNSET  A5B9  1231    1210
RUN$    00A9   296
S$RUN   601E    61     557
S$RUNX  A186   557     531
SADD    000B    98     825  828  848
SAVEVP  0388   231     510
SBUFLV  0384   229     511
SCROLL  0083   120     489
SEG$$   00D8   344
SEG$01  A490  1053     548
SEG$06  A4D4  1080    1076 1092
SEG$08  A4EF  1091    1071
SEMIC$  00B4   307
SEQUE$  00F6   364
SEXTRM  0386   230     509
SFLAG   0398   236     513  514
SGN$$   00D1   337
SIGN$   0075   207
SIN$    00D2   338
SIZE$   00EB   353
SLSUBP  0396   235     512
SMB     007B   112    1529 1822 2306
SMTSRT  001E   150
SNDCS   AAC6  1969    1971 1972
SNDREG  AA1C  1868    1879
SOUND   AA25  1877     412
```

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

SOUND1  AA5E  1911     1942
SOUND2  AAA6  1950     1914
SOUND3  AA8D  1938     1966
SOUND5  AAC4  1968     1938
SOUND6  AAD5  1977     1969 1979
SPACE   0020   384     1159 1221 1223
SPCOL   A969  1769      405 1737
SPEED   007E   114      739 1055 1062 1250 1255 1307 1401 1604 1731 1743
                       1745 1750 1885 1911 1917 1939 1944 1952 1994 2021
                       2023 2092 2119 2134 2139 2148 2158 2164 2166 2193
SPGMPT  0382   228      508
SPNUM2  AD9A    66     1739
SPNUM3  AD98    65     1735
SPRITE  0006     1
SPRSAL  0300   222
SPRVB   0780   242     2047
SPSAL   0008   136     1773
SQR$    00D3   339
SQUCS   A9F8  1842     1840
SQUISH  A9E1  1837      400 2227
SRDATA  8020    87     1622
SREF    001C   149      954  955  960  964 1020 1219 1222 1334 1338 1537
                       1588
SSEP$   0082   257
SSUB    000C    99      807  836  882
STACK   0072   123
STADDR  000A   137     1923 1930 1942
START   0372   224      435
STEP$   00B2   305
STLN    0030   159      645 1373 1416 1625
STOP$   0098   279      603  652  671
STR$$   00DB   347
STR$01  A52B  1155      547
STREND  001A   148
STRIN$  00C7   326
STRSP   0018   147
STVSPT  0024   153     2224
SUB$    00A1   288
SUBND$  00A8   295
SUBSTK  0073   124      465 1503
SUBTAB  003A   164
SUBXT$  00A7   294
SYM     007A   111     1819 2305
SYMBOL  0376   225     1526
SYMTAB  003E   166     1494 1526 1528 1567 1569 1577 1584
SYNCHK  0000   115      740 1063 1256 1308 1402 1732 1940 1995 2135 2140
                       2194
TAB$    00FC   370
TABLE   000B     1
TAN$    00D4   340
TEMP1   0056   192     1070 1071 1072 1073 1074 1075 1576 1577
TEMP4   0064   201     1020 1021 1025 1474 1478 1501
TEMP5   0066   202      960  966 1015 1084 1085 1086 1123 1167 1203 1204
                       1208 1211 1216 1326 1338 1473 1487 1567 1573 1574
                       1574 1575 1576 1578
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
TEMP$	0068	203	1009 1022 1025 1026
THEN$	00B0	303	
TIMER	0079	130	
TO$	00B1	304	
TONE2	0036	56	
TOP	0003	1	
TOPL15	6012	58	477
TRACE$	0090	271	
TREM$	0083	258	871 1545
UALPH$	00EA	352	
UBSUB	A6FC	1416	406 1413
UBSUB1	A70C	1422	506 1418
UDF	A717	1458	459
UNBKO1	A6F7	1413	1368
UNBRE$	008F	270	
UNQST$	00C8	327	328
UNTRA$	0091	272	
UPDAT$	00F8	366	
USING$	00ED	355	
VAL$	00DA	346	
VALC1	A555	1185	546
VALCD	A571	1202	401 1188
VALID$	00FE	372	
VARC	0000	133	1794 1795 1798 1806 1811 1990 2008 2124 2127 2190
VAR2	000B	138	1934 1935
VAR4	000E	140	1753 1754 1758 1760
VAR5	0010	141	1968 1977 1978 1979 2034 2053 2056 2058
VAR7	0012	142	2043 2044 2051 2052 2053 2054 2057 2083
VAR9	0014	143	2062 2063 2066 2076 2077
VARA	002A	156	
VARW	0020	151	485 486 487 488 490 492 493 495 496 497
VARY	0004	135	2029 2047 2080 2081
VCHAR	A808	2005	417
VCHAR1	A80F	2008	2011 2013
VDP	000C	1	2180
VEL	0007	1	
VERS	AA04	1557	422
VPOP	0078	109	743 821 845 1012 1014 1067 1069 1083 1215 1262 1264 1319 1470 1531 1759 2294
VPUSH	0077	108	735 794 799 803 809 819 830 843 875 984 1010 1059 1081 1213 1466 1475 1477 1496 1534 1749 1823 2228 2248 2307
VRAMVS	0958	244	
VRMSND	0379	227	1878 1879 1894 1964 1980
VROA$	03C0	240	1165 1167 1504 1524
VSPTR	006E	206	510 806 818 820 822 829 844 1003 1311 1493 1497 1516 1525 1551 1553 1560 1569 1597 1838 1842 2223 2254 2260 2264 2273 2281
WARN$	00A6	293	
WARN%$	6A82	73	468 1006 1313 1407 1705
WARN0$	A0D4	465	457
WRNLNF	A6F1	1406	1373
WRNNO	ABF1	1704	1230
WRNNO1	A0D9	468	
WRNNO2	ABF5	1705	

SYMBOL	VALUE	DEF	REFERENCE TABLE
WRNPRT	0001	212	604 608
WRNST1	A457	1006	
WRNST2	A641	1313	
WRNSTP	0002	213	605 609
X1SEED	A29A	751	
X2	0003	249	439
X2SEED	A295	750	748
XFLAG	0016	144	1506
XMLCON	A684	1345	1333
XOR$	00BC	315	
XPT	000E	1	439 486 497 1851 1993 2012 2162 2170
XPTRTN	AAF9	1993	1725 1989 2007 2010
YPT	000D	1	2011 2163
ZERO	0030	378	2048 2067 2071