```
6896 DA4820  1039      $IF .BIT(ASRC) @FLAG .EQ. 1 GOTO START
6899 4079

LAST ERROR FOUND AT LINE  1015

***** ERROR IN IMMEDIATE FIELD  LINE= 1040  COLUMN= 40

***** ERROR IN ADDRESS FIELD  LINE= 1040
689B DA4901  1040      $IF .BIT(LD) @FLAG2 .EQ. 1 GOTO LOAD
689E 6079
68A0 DA4908  1041      $IF .BIT(OTHR) @FLAG2 .EQ. 1 GOTO BEGIN
68A3 4025
68A5 4133    1042      BR  EDIT                          Go back to editor
             1043   ****
             1044   CLSALL
68A7 BF1C10  1045      DST  PAB1.@SREF       Close first file
68AA 00
68AB 0668C4  1046      CALL CLSNO
68AE BF1C11  1047      DST  PAB2.@SREF       Close second file
68B1 00
68B2 0668C4  1048      CALL CLSNO
68B5 BF1C12  1049      DST  PAB3.@SREF       Close third file
68B8 00
68B9 0668C4  1050      CALL CLSNO
68BC BF1C13  1051      DST  PAB4.@SREF       Close fourth file used in COPY state
68BF 00
68C0 0668C4  1052      CALL CLSNO
68C3 00      1053      RTN
             1054   ****
             1055   CLSNO
68C4 BD561C  1056      DST  @SREF.@FAC12     Pointer to name length
68C7 A35600  1057      DADD PABLEN-1.@FAC12
68CA 09
68CB BEB01C  1058      ST   1,RAM(@SREF)     Close file
68CE 01
68CF 060010  1059      CALL DSRLNK           Call DSR -- no worry about erro
68D2 08      1060      DATA 8
68D3 86E009  1061      CLR  RAM(NLEN(SREF))  Clear name len to avoid prob wi
68D6 1C
68D7 00      1062      RTN
             1063   ****
             1064   ************************************************************
             1065   *      WARNING HANDLING SECTION                            *
             1066   ************************************************************
             1067   WARN
68D8 06694B  1068      CALL CLNMSG Clean up message area
68DB 310008  1069      MOVE 11 FROM ROM(#WRNM) TO RAM(>2A2)
68DE A2A269
68E1 77
68E2 48EE    1070      BR   WRNENT
             1071
             1072   ************************************************************
             1073   *      ERROR HANDLING SECTION                              *
             1074   ************************************************************
             1075   ERROR
68E4 06694B  1076      CALL  CLNMSG      Cleanup message area
68E7 310009  1077      MOVE 9 FROM ROM(#ERRM) TO RAM(>2A2)
68EA A2A269
68ED 57
```

```
                 1078 . WRNENT
68EE 884A        1079          FETCH @FAC        FETCH MESSAGE ADDRESS.
68F0 8848        1080          FETCH @FAC+1
68F2 864C        1081          CLR  @FAC+2
68F4 330001      1082          MOVE 1 FROM ROM(@FAC) TO @FAC+3
68F7 4D0000
68FA 4A
68FB 324CA2      1083          MOVE @FAC+2 FROM ROM(1(FAC)) TO RAM(>2C2)
68FE C20001
6901 4A
6902 D74A69      1084          $IF @FAC .DEQ. #IO THEN          If an I/O error     1
6905 8E490C
6908 BCA2D2      1085            ST  @FAC4.RAM(>2D2)            Display error code  1
690B 4E
                 1086          $END IF
690C D74A69      1087          $IF @FAC .DEQ. #XMLERR THEN                          1
690F BF493B
6912 E72200      1088            DSRL @ERRCOD,4                 Change error code to1
6915 04
6916 E62304      1089            SRL  @ERRCOD+1.4                                    1
6919 C62209      1090            $IF @ERRCOD .H. 9 THEN         Take care of A - F  2
691C 4926
691E C6220F      1091              $IF @ERRCOD .H. 15 GOTO CONT Number too big so s2
6921 6938
6923 A22207      1092              ADD 7,@ERRCOD                                     2
                 1093            $END IF
6926 C62309      1094            $IF @ERRCOD+1 .H. 9 THEN       Take care of A - F  2
6929 4933
692B C6230F      1095              $IF @ERRCOD+1 .H. 15 GOTO CONT Skip this number 2
692E 693B
6930 A22307      1096              ADD 7,@ERRCOD+1                                   2
                 1097            $END IF
6933 A32230      1098            DADD >3030,@ERRCOD                                  1
6936 30
6937 BDA2D0      1099            DST  @ERRCOD,RAM(>2D0)         Display it on screen1
693A 22
                 1100          $END IF
                 1101  CONT
693B 310017      1102          MOVE 23 FROM ROM(#PRESS) TO RAM(>2E2)
693E A2E269
6941 60
                 1103  SCAN40
6942 03          1104          SCAN
6943 4942        1105          BR   SCAN40       Wait till hit enter.
6945 D67500      1106          $IF @KEY .NE. ETR GOTO SCAN40
6948 4942
694A 00          1107          RTN
                 1108  ****
                 1109  CLNMSG
694B BEA2A0      1110          ST    SPACE,RAM(>2A0)  CLEAN UP MSG AREA.
694E 20
694F 3500SF      1111          MOVE >5F FROM RAM(>2A0) TO RAM(>2A1)
6952 A2A1A2
6955 A0
6956 00          1112          RTN
```

```
                1113  *
                1114  ********
                1115  *       DATA AREA
                1116  ********
6957 2A2045     1118  ERRM    DATA :* ERROR *:
6960 505245     1119  PRESS   DATA :PRESS ENTER TO CONTINUE:
6977 2A2057     1120  WRNM    DATA :* WARNING *:
6982 0B4045     1121  MEM     DATA 11,:MEMORY FULL:
698E 0E492F     1122  IO      DATA 14,:I/O ERROR CODE:
699D 134E4F     1123  RAMERR  DATA 19,:NO MEMORY EXPANSION:
69B1 0D4E41     1124  NAM     DATA 13,:NAME TOO LONG:
69BF 0A4552     1125  XMLERR  DATA 10,:ERROR CODE:
69CA 19434F     1126  CHRWRN  DATA 25,:CONTROL CHARACTER REMOVED:
69E4 0B494C     1127  ILLTAG  DATA 11,:ILLEGAL TAG:
69F0 0E4348     1128  CSMERR  DATA 14,:CHECKSUM ERROR:
69FF 144455     1129  DUPDEF  DATA 20,:DUPLICATE DEFINITION:
6A14 14554E     1130  UNRREF  DATA 20,:UNRESOLVED REFERENCE:
6A29 115052     1131  PRGNF   DATA 17,:PROGRAM NOT FOUND:
                1132  ******
6A3B 707070     1133  CDATA   DATA >70,>70,>70,>70,>70,>70,>70,>70
6A43 007E42     1134          DATA 00,>7E,>42,>42,>42,>42,>7E,00
6A4B E0000E     1135  VDPRG2  DATA >E0,>0,>0E,>1,06,0,>F5
```

```
                 *       TITLE   BASIC
                 ***************************************************
                 *       Sept 09, 1981    Author S.Endo            *
                 ***************************************************
                 *       BASIC SUPPORT FUNCTIONS                   *
                 ***************************************************
                 *****
                 *       This program takes care of interface to BASIC.
                 *       Routines callable from console BASIC are listed
                 *       in the link list below.
                 *****
                 *******
                 **      XML EQUATES    ***
                 ******
0012             CFI     EQU  >12    Convert floating point to integer
0013             SYM     EQU  >13
0014             SMB     EQU  >14
0015             ASSGNV  EQU  >15    Assign value to numeric.
0017             VPUSH   EQU  >17    Push to stack
0018             VPOP    EQU  >18    Pop off the stack.
001B             PGMCH   EQU  >1B    Advance one character
                 *
                 ****    GPL CALLS      ***
                 *
0038             GETSTR  EQU  >38    GET ENOUGH SPACE FOR STRING
001A             WARN$$  EQU  >1A    Warning handling routine
001C             ERR$$   EQU  >1C    Error handling routine
                 ******
                 ***     CPU RAM EQUATES  ***
                 *****
0004             PC      EQU  >04    Address pointer
000C             BYTE    EQU  >0C    Number of bytes to move
0010             OLDS    EQU  >10
0012             CNT     EQU  >12    Counter.
0016             PCODE   EQU  >16    ERAM address to store parm information
                 *
200A             PARM    EQU  >200A->8300 PARAMETER COUNT INFORMATION
                 *
002C             PGMPTR  EQU  >2C    Program pointer
                 *
0040             FREPTR  EQU  >40    Free space pointer
0042             CHAT    EQU  >42    Character code
006E             VSPTR   EQU  >6E    Value stack pointer
                 ****
00B3             COMMA$  EQU  >B3    Token for ","
00B6             RPAR$   EQU  >B6    Token for ")"
00B7             LPAR$   EQU  >B7    Token for "("
0065             STRING  EQU  >65    String ID byte
                 ***********************************************************
                 **   LINK LIST -- list of callable subprograms            *
                 ***********************************************************
6A52 6A5B6A      LINK01  DATA #LINK02,#INIT,4,:INIT:
6A55 960449
6A58 4E4954
6A5B 6A646A      LINK02  DATA #LINK03,#BLOAD,4,:LOAD:
```

```
6A5E EC044C
6A61 4F4144
6A64 6A6D6C  1191  LINK03 DATA #LINK04,#BLINK,4,:LINK:
6A67 08044C
6A6A 494E4B
6A6D 6A766B  1192  LINK04 DATA #LINK05,#PEEK,4,:PEEK:
6A70 830450
6A73 45454B
6A76 6A806B  1193  LINK05 DATA #LINK06,#PEEKV,5,:PEEKV:
6A79 7E0550
6A7C 45454B
6A7F 56
6A80 6A8A6A  1194  LINK06 DATA #LINK07,#POKEV,5,:POKEV:
6A83 E70550
6A86 4F4B45
6A89 56
6A8A 000060  1195  LINK07 DATA #0,#GETCHR,7,:CHARPAT:
6A8D 120743
6A90 484152
6A93 504154
             1196  ************************************************************
             1197  *   INIT -- initialization routine                         *
             1198  ************************************************************
             1199  INIT
6A96 864808  1200      SB   @FLAG,BSC Set BASIC flag on.
6A99 A32C00  1201      DADD 5,@PGMPTR Skip name len and name 'INIT'
6A9C 05
6A9D 066AAA  1202      CALL INIT3     Force UTIL load
             1203  *
6AA0 4001    1204      BR   RTBSC2    Go back to BASIC interpreter
             1205  ***
             1206  * - Routine to load UTIL subroutines
             1207  ***
             1208  INIT2
6AA2 D78F90  1209      $IF @IDCODE .DNE. >A55A THEN     Check if UTIL is thet
6AA5 00A55A
6AA8 6AD1
             1210  INIT3
6AAA 066504  1211      CALL CHKRAM                      Is E-RAM there?      1
6AAD BE4A03  1212      ST   3,@FAC                      Force UTIL load      1
6AB0 BF4C70  1213      DST  ALCODE,@FAC2                                     1
6AB3 00
6AB4         1214  INITLP EQU $
6AB4 330004  1215      MOVE 4 FROM ROM(@FAC2) TO @FAC4  Load utility routine1
6AB7 4E0000
6ABA 4C
6ABB A34C00  1216      DADD 4,@FAC2                                          1
6ABE 04
6ABF 324ECF  1217      MOVE @FAC4 FROM ROM(@FAC2) TO @->8300(FAC6)           1
6AC2 700050
6AC5 00004C
6AC8 A14C4E  1218      DADD @FAC4,@FAC2                                      1
6ACB 924A    1219      DEC  @FAC                                             1
6ACD 4AB4    1220      BR   INITLP                                           1
6ACF 4549    1221      BR   CLRTOP        Clear editor information and return1
```

```
                1222        $END IF
6AD1 00         1223        RTN
                1224   ***
                1225   ****
                1226   SKIPN
6AD2 864A       1227        CLR  @FAC                Count size of routine name
6AD4 BC4880     1228        ST   RAM(@PGMPTR),@FAC+1  Copy size
6AD7 2C
6AD8 A12C4A     1229        DADD @FAC,@PGMPTR        Advance pointer to end of name
6ADB 912C       1230        DINC @PGMPTR             Advance pointer to char after name
6ADD 0F1B       1231        XML  PGMCH               Get next token
6ADF D642B7     1232        $IF @CHAT .NE. LPAR$ GOTO ERRIS  Has to be "(".
6AE2 4E07
6AE4 0F1B       1233        XML  PGMCH               Advance pointer one more time
6AE6 00         1234        RTN
```

```
                 1236  ***
                 1237  ***********************************************************
                 1238  ***     CALL LOAD and POKEV from BASIC program     ********
                 1239  ***********************************************************
                 1240  POKEV
6AE7 B64801      1241      SB   @FLAG,VFLG      Turn on VDP flag
6AEA 4AF1        1242      BR   LDP1            Join LOAD routine
                 1243  BLOAD
6AEC 8748        1244      DCLR @FLAG           Delete all flags
6AEE B64808      1245      SB   @FLAG,BSC       Called from BASIC program.
                 1246  *****
                 1247  LDP1
6AF1 066AD2      1248      CALL SKIPN           Skip name first
                 1249  LDP3
6AF4 0EB6        1250      PARSE RPAR$  Parse up to ")".
                 1251  **  Process file name if it is a string.  **
6AF6 D64C65      1252      $IF @FAC2 .EQ. STRING GOTO LDP2
6AF9 6B3D
                 1253  **  Otherwise, it is an absolute address  **
6AFB 0F12        1254      XML  CFI        Convert to integer
6AFD D65403      1255      $IF @FAC10 .EQ. 03 GOTO ERRNTB     Check for overflow
6B00 6E1B
6B02 B0044A      1256      DST  @FAC,@PC  Save in ERAM(or VRAM) address pointer
                 1257  LDP4
6B05 D642B3      1258      $IF @CHAT .NE. COMMA$ GOTO RTBSC  RETURN TO BASIC.
6B08 4CFC
6B0A 0F18        1259      XML   PGMCH     Skip ","
6B0C 0EB6        1260      PARSE RPAR$     Parse up to ")"
6B0E D64C65      1261      $IF @FAC2 .EQ. STRING THEN        No more data?      1
6B11 4B19
6B13 8F506B      1262        $IF @FAC6 .DEQ. 0 GOTO BEXTLD   Null string?       1
6B16 7S
6B17 4E25        1263        BR  ERRBA                       Bad argument       1
                 1264      $END IF
                 1265  **  Numeric in FAC area
6B19 0F12        1266      XML  CFI        Convert to integer
6B1B D65403      1267      $IF @FAC10 .EQ. 03 GOTO ERRNTB    Overflow error
6B1E 6E1B
                 1268  ***
6B20 0A4801      1269      $IF .BIT(VFLG) @FLAG .EQ. 1 THEN                     1
6B23 6B2B
6B25 BC8004      1270        ST  @FAC+1,RAM(@PC)     POKE INTO VDP RAM LOCATION 1
6B28 4B
6B29 4B38        1271      $SELSE                  Poke into CPU RAM              1
6B2B A70483      1272        DSUB OFFSET,@PC                                    1
6B2E 00
6B2F BCC000      1273        ST  @FAC+1,@0(PC)       Store it in ERAM           1
6B32 044B
6B34 A30483      1274        DADD OFFSET,@PC                                    1
6B37 00
                 1275      $END IF
6B38 9104        1276      DINC @PC                Increment ERAM address.
6B3A 056B05      1277      B    LDP4               Continue loop
                 1278  *****
                 1279  LDP2
```

```
6B30 8F506B  1280      $IF @FAC6 .DEQ. 0 GOTO BEXTLD  Skip null string.
6B40 75
6B41 DA4801  1281      $IF .BIT(VFLG) @FLAG .EQ. 1 GOTO ERRSNM Not for VDP poke
6B44 4E20
             1282  *
             1283  *   Following section prepares PAB area for object file    *
             1284  *
6B46 B00C50  1285      DST  @FAC6,@BYTE        Store actual spec. length
6B49 A30CC0  1286      DADD PABLEN+20,@BYTE    Add in the PAB len. and buffer len
6B4C 5A
6B4D 0F17    1287      XML  VPUSH               Push possible temp. string
6B4F 066AA2  1288      CALL INIT2               make sure Utility is there
6B52 060038  1289      CALL GETSTR              and try to allocate space
6B55 0F18    1290      XML  VPOP                Restore original string data
             1291  *
             1292  *   Variable specification
             1293  *    FAC+4,5    Start address of original device spec.
             1294  *    FAC+6,7    LENGTH OF ORIGINAL DEVICE SPECIFICATIO
             1295  *    SREF       Location of PAB in VDP memory
             1296  *    BYTE       Length of entire PAB, including spec.
             1297  *
6B57 0663FE  1298      CALL CLRPAB              Clear the entire PAB
6B5A 0663C3  1299      CALL COPYNM              Copy device name and length
6B5D BEE008  1300      ST   >60,RAM(SCR(SREF))      Screen offset
6B60 1C60
6B62 BEE001  1301      ST   &00000100,RAM(FLG(SREF)) Disp, fixed, seq, input
6B65 1C04
6B67 A1501C  1302      DADD @SREF,@FAC6         Calculate the address of the
6B6A A35000  1303      DADD PABLEN,@FAC6        buffer
6B6D 0A
6B6E B0E002  1304      DST  @FAC6,RAM(BUF(SREF))   Store buffer address in PAB
6B71 1C50

LAST ERROR FOUND AT LINE  1040

***** ERROR IN IMMEDIATE FIELD  LINE= 1305  COLUMN= 16

***** ERROR IN BRANCH ADDRESS  LINE= 1305
6B73 0000    1305      BR   BENTLD              Enter object file Loader
             1306  ****
             1307  ********                                          *******
             1308  **  Entry from object file loader  CALL LOAD        ****
             1309  ********                                          *******
     6B75    1310  BEXTLD EQU  $
             1311  *
6B75 D642B3  1312      $IF @CHAT .NE. COMMA$ GOTO RTBSC   Go back to BASIC.
6B78 4CFC
6B7A 0F1B    1313      XML  PGMCH                       (SKIP COMMA)
6B7C 4AF4    1314      BR   LDP3                        (CONTINUE IN MAIN LOOP)
             1315  ********
```

```
                 ********                                                  ******
                 ************************************************************
                 *                PEEK AND PEEKV ROUTINE                    *
                 ************************************************************
                 *
                 * FORMAT:
                 *   CALL PEEK  (address, numeric variable,....)
                 *   CALL PEEKV (Address, numeric variable.....)
                 * FUNCTION:
                 *   RETURNS THE VALUE AT address IN ERAM INTO
                 *   numeric-variable.  IF MORE THAN ONE numeric-variable IS
                 *   SPECIFIED THEN address IS INCREMENTED AND THE VALUE
                 *   IN ERAM AT THE NEW address IS ASSIGNED TO THE NEXT
                 *   VARIABLE AND SO ON.
                 *
                 PEEKV
6B7E B64801      SB   @FLAG.VFLG   VDP FLAG SET
6B81 4B85        BR   PEEKO        ENTRY POINT
                 PEEK
6B83 8748        DCLR @FLAG        CLEAR FLAG
                 PEEKO
6B85 066AD2      CALL SKIPN        SKIP NAME FIRST
                 * CALL INIT2      CHECK UTIL IS THERE AND LOAD
                 PEEK1
6B88 0EB6        PARSE RPAR$       GET VALUE OF ADDRESS
6B8A D64C65      $IF @FAC2 .EQ. STRING GOTO PEEK3   SKIP NULL STRING
6B8D 6BAC
6B8F 0F12        XML  CFI     [CONVERT FAC TO INTEGER]
6B91 D65403      $IF @FAC10 .EQ. 03 GOTO ERRNTB     [OVERFLOW?]
6B94 6E1B
6B96 BD044A      DST  @FAC,@PC                      [SAVE PEEK ADDRESS]
6B99 D642B3      $IF @CHAT .NE. COMMA$ GOTO RTBSC   [GO BACK TO BASIC?
6B9C 4CFC
                 *
                 PEEK2
6B9E 0F1B        XML  PGMCH                         SKIP ","
6BA0 D642C7      $IF @CHAT .EQ. >C7 THEN            STRING                1
6BA3 4BB9
6BA5 0EB6        PARSE RPAR$       GET STRING VALUE                       1
6BA7 D64C65      $IF @FAC2 .EQ. STRING THEN SKIP NULL STRING              2
6BAA 4BB9
                 PEEK3
6BAC 8F504E      $IF @FAC6 .NE. 0 GOTO ERRSNM SHOULD BE NULL              2
6BAF 20
6BB0 D642B3      $IF @CHAT .NE. COMMA$ GOTO RTBSC RETURN TO BASIC         2
6BB3 4CFC
6BB5 0F1B        XML PGMCH                          SKIP COMMA            2
6BB7 4B88        BR  PEEK1                          GO BACK TO THE LOOP   2
                 $END IF
                 $END IF
6BB9 CA4280      $IF @CHAT .NE. >80 GOTO RTBSC GO BACK TO BASIC?
6BBC 6CFC
6BBE 0F13        XML  SYM                           GET SYMBOL NAME
6BC0 0F14        XML  SMB                           GET VALUE POINTER
6BC2 0F17        XML  VPUSH                         SAVE FAC AREA
```

```
                 1364  *
                 1365  ** GET VALUE FROM ERAM OR RAM INTO FAC+1
                 1366  *
6BC4 DA4801      1367    $IF .BIT(VFLG) @FLAG .EQ. 1 THEN                    1
6BC7 6BCF
6BC9 BC5DB0      1368      ST  RAM(@PC),@ARG+1      GET VALUE FROM VDP RAM     1
6BCC 04
6BCD 4BDC        1369    $ELSE                      GET VALUE FROM CPU RAM     1
6BCF A70483      1370      DSUB OFFSET,@PC          TAKE OUT OFFSET            1
6BD2 00
6BD3 BC5DC0      1371      ST   @0(PC),@ARG+1       MOVE VALUE FROM ERAM       1
6BD6 0004
6BD8 A30483      1372      DADD OFFSET,@PC                                     1
6BDB 00
                 1373    $END IF
                 1374  *
                 1375  **      CONVERT INTEGER TO FLOATING INTO FAC AREA
                 1376  *
6BDC 066D0A      1377    CALL CLRFAC                CLEAR FAC AREA FIRST
6BDF 8E5D6B      1378    $IF @ARG+1 .EQ. 0 GOTO SKIP  IF ZERO ...
6BE2 FC
6BE3 BE4A40      1379    ST   >40,@FAC              IF LESS THAN 100
6BE6 865C        1380    CLR  @ARG                  PREPARE FOR WORD OPERATION
6BE8 AE5C64      1381    DIV  100,@ARG              DIVIDE BY 100
6BEB BC4B5C      1382    ST   @ARG,@FAC+1           VALUE IF OVER 100
6BEE BC4C5D      1383    ST   @ARG+1,@FAC+2         Move to the third digit.
6BF1 8E4B4B      1384    $IF @FAC+1 .EQ. 0 THEN     Value is less than 100      1
6BF4 FA
6BF5 C04C4B      1385      EX   @FAC+1,@FAC+2       Move third byte to second   1
6BF8 4BFC        1386    $ELSE                      If over 100                 1
6BFA 904A        1387      INC  @FAC                Increment exponent value    1
                 1388    $END IF
                 1389  *
                 1390  *
                 1391  SKIP
6BFC 0F15        1392    XML  ASSGNV    (ASSIGN TO NUMERIC-VARIABLE)
6BFE D642B3      1393    $IF @CHAT .NE. COMMA$ GOTO RTBSC
6C01 4CFC
6C03 9104        1394    DINC @PC    (INC POINTER TO NEXT ERAM ADDRESS)
6C05 056B9E      1395    B    PEEK2
                 1396  *
                 1397  *****
```

```
1399  ******************************************
1400  *                                        *
1401  *  LINK INSTRUCTION : SE September, 1980 *
1402  *                                        *
1403  ******************************************
1404  *
1405  *  FORMAT:
1406  *
1407  *  CALL LINK("file-name",parameter1,parameter2,.....)
1408  *
1409  *  LINK ROUTINE READS THE FILE NAME SPECIFIED BY THE USER
1410  *  AND SAVE THE ADDRESS OF THE NAME FOR LATER USE.
1411  *  THE FILE WILL BE SEARCHED IN UTILITY CODE LATER ON.
1412  *
1413  *  PARAMETERS ARE PASSED EITHER BY REFERENCE OR BY VALUE.
1414  *  NUMERIC OR STRING VARIABLES AND NUMERIC OR STRING ARRAYS
1415  *  ARE PASSED BY REFERENCE AND ALL OTHERS INCLUDING A USER
1416  *  DEFINED FUNCTION ARE PASSED BY VALUE.
1417  *
1418  *  PARAMETER INFORMATION IS STORED IN ERAM >200A THRU >201A
1419  *  THAT GIVES A PARAMETER TYPE CODE OF EACH PARAMETER.
1420  *    CODE 0 ....   Numeric expression
1421  *    CODE 1 ....   String expression
1422  *    CODE 2 ....   Numeric variable
1423  *    CODE 3 ....   String variable
1424  *    CODE 4 ....   Numeric array
1425  *    CODE_5 ....   String array
1426  *
1427  *  IF A PARAMETER IS PASSED AS A NUMERIC EXPRESSION ITS
1428  *  ACTUAL VALUE GETS PUSHDED INTO THE VALUE STACK.
1429  *  IN CASE OF A STRING EXPRESSION, ITS VALUE STACK CONTAINS
1430  *  AN ID(>65). POINTER TO THE VALUE SPACE AND ITS LENGTH.
1431  *  IF A PARAMETER GETS PASSED AS A REFERENCE THE PRODUCT
1432  *  OF XML SYM AND XML SMB IN THE @FAC AREA GETS PUSHED INTO
1433  *  STACK.
1434  *
1435  *  AFTER AN ASSEMBLY LANGUAGE SUBPROGRAM IS EXECUTED LINK
1436  *  ROUTINE WILL POP THE STACK TO GET RID OF PARAMETER
1437  *  INFORMATION. CONTROL WILL BE TRANSFERED TO THE BASIC
1438  *  MAIN PROGRAM AFTERWARDS.
1439  *
1440  ****  DATA  AREA  ****
1441  *  CPU RAM FREE SPACE USED IN THIS ROUTINE
1442  *
1443  *  FLAG BITS STORED IN LOCATIONS 00 TO >0F
1444  ****
1445
```

```
                  1447  ****************************************
                  1448  *  CALL LINK program                   *
                  1449  ****************************************
                  1450  *
                  1451  BLINK
6C08 B64808       1452     SB   @FLAG,BSC       CALLED FROM BASIC PROGRAM.
6C0B 066AD2       1453     CALL SKIPN           SKIP NAME FIRST
6C0E 066504       1454     CALL CHKRAM          Make sure E-RAM is there
6C11 D78F9D       1455     $IF @IDCODE .DNE. >A55A GOTO ERRPN No program
6C14 00A55A
6C17 4E02
6C19 B64808       1456     SB   @FLAG,BSC       Set bit for BASIC flag
6C1C BD106E       1457     DST @VSPTR,@OLDS     Save VSPTR for later use.
6C1F 0E66         1458     PARSE RPAR$          Get the routine name.
                  1459  *                       Read up to ")".
6C21 D64C65       1460     $IF @FAC2 .NE. STRING GOTO ERRSNM Should be a string.
6C24 4E20
6C26 C65106       1461     $IF @FAC6+1 .H. 6 GOTO ERRBA   Should be less than 6 char
6C29 6E25
6C2B 0F17         1462     XML VPUSH            Push to make it semi-permanent.
6C2D 8612         1463     CLR @CNT             Initialize parameter counter.
6C2F BF169D       1464     DST PARM,@PCODE      Set initial address for parmeter
6C32 0A
                  1465  *
                  1466  ****************************************
                  1467  *  PARAMETERS get evaluated here       *
                  1468  ****************************************
                  1469  PARO1
6C33 D642B6       1470     $IF @CHAT .EQ. RPAR$ GOTO EXEO1  No arg. So execute it.
6C36 6CD5
6C38 D642B3       1471     $IF @CHAT .NE. COMMA$ GOTO ERRIS  Should have a comma.
6C3B 4E07
                  1472  *
6C3D BD222C       1473     DST @PGMPTR,@ERRCOD  Save text pointer.
6C40 0F1B         1474     XML PGMCH            Get the character
6C42 CA4280       1475     $IF @CHAT .HE. >80 GOTO VALO1 Must be an expression.
6C45 6C98
                  1476  *  $IF @CHAT .EQ. LPAR$ Pass by expression.
6C47 066D0A       1477     CALL CLRFAC          Clear FAC entry for SYM.
6C4A 0F13         1478     XML SYM              Read in the symbol table info.
                  1479  *
                  1480  *  After XML SYM @FAC area contains a pointer to
                  1481  *  symbol table.
                  1482  *
                  1483  *  Below statement checks if it is a UDF.
6C4C 0A804A       1484     $IF .BIT6 RAM(@FAC) .EQ. 1 GOTO VALO1 Pass by value.
6C4F 404C98
6C52 D642B3       1485     $IF @CHAT .EQ. COMMA$ GOTO REFO1   Pass by reference.
6C55 6CB1
6C57 D642B6       1486     $IF @CHAT .EQ. RPAR$ GOTO REFO1    Pass by reference.
6C5A 6CB1
6C5C D642B7       1487     $IF @CHAT .EQ. LPAR$ GOTO ARRAY    An array.
6C5F 6C68
6C61 CA4280       1488     $IF @CHAT .HE. >80 GOTO VALO1      Pass by value.
6C64 6C98
```

```
6C66 4E07    1489      BR  ERRIS
             1490   *
             1491   ****************************************
             1492   *  ARRAY case gets checked here        *
             1493   ****************************************
             1494   *  should look like A(,,) etc.
             1495   *  Stack entry for an arry will look like
             1496   *  ---------------------------------------------------------
             1497   * | Pointer to      |>00  |      | Pointer to  |           |
             1498   * | symbol table    | or  |      | dim info in |           |
             1499   * | entry           |>65  |      | real v.s.   |           |
             1500   * FAC---------------FAC2--------FAC4----------FAC6---------
             1501   *
             1502   *
             1503   ARRAY
6C68 0F1B    1504      XML PGMCH              Get the next character.
6C6A D642B6  1505      $IF @CHAT .EQ. RPAR$ GOTO ARRAY2    Pass by reference.
6C6D 6C7B
6C6F D642B3  1506      $IF @CHAT .EQ. COMMA$ GOTO ARRAY    More array informatio
6C72 6C68
6C74 932C    1507      DDEC @PGMPTR                        Adjust the pointer.
6C76 BE42B7  1508      ST  LPAR$.@CHAT
6C79 4CB1    1509      BR  REF01                           Pass by reference.
             1510   *
             1511   ARRAY2
             1512   *
             1513   *  In array cases the symbol table address gets stored
             1514   *  at FAC area, and the pointer to the value space
             1515   *  (dimension info.) goes into FAC4
             1516   *
6C7B 0F1B    1517      XML PGMCH              Advance the program pointer.
6C7D DA804A  1518      $IF .BIT7 RAN(@FAC) .NE. 1 THEN  Test string bit.      1
6C80 804C8A
6C83 BEC000  1519      ST  4,@0(PCODE)        Numeric array.                  1
6C86 1604
6C88 4C8F    1520      $ELSE                                                  1
6C8A BEC000  1521      ST  5,@0(PCODE)        String array case.              1
6C8D 1605
             1522      $END IF
             1523   *
             1524   *  Put the pointer to the value space (dimension info.)
             1525   *  into FAC4.
6C8F BD4E4A  1526      DST  @FAC,@FAC4        Array is not shared.
6C92 A34E00  1527      DADD  6,@FAC4          Point to value space
6C95 06
             1528   *
6C96 4CC8    1529      BR  PUSH
             1530   *
             1531   ****************************************
             1532   *  VALUE                               *
             1533   *  Passing the parameter by value      *
             1534   ****************************************
             1535   VAL01
6C98 BD2C22  1536      DST @ERRCOD,@PGMPTR    Restore program pointer.
6C9B 0F1B    1537      XML PGMCH              Skip the first character.
```

```
6C9D 0EB6   1538      PARSE RPARS          Parsing up to comma.
            1539  *
            1540  *   After parsing @FAC area contains
            1541  *   its actual numeric value in a numeric case, and the
            1542  *   following information in a string case.
            1543  *
            1544  *  | >001C              | >65 |      | Pointer to | Length of |
            1545  *  | or value pointer   |     |      | string     | string    |
            1546  *  | address            |     |      |            |           |
            1547  *  FAC----------------FAC2--------FAC4---------FAC6---------
            1548  *
            1549  *
6C9F D64C65 1550      $IF @FAC2 .EQ. STRING THEN  If it is a string          1
6CA2 4CAB
6CA4 BEC000 1551        ST 1,@0(PCODE)     Store flag for string expression.  1
6CA7 1601
6CA9 4CAF   1552      $ELSE                                                   1
6CAB 86C000 1553        CLR @0(PCODE)      Otherwise it is a numeric express  1
6CAE 16
            1554      $END IF
6CAF 4CC8   1555      BR  PUSH             Push into stack.
            1556  *
            1557  ****************************************
            1558  *   REFERENCE                          *
            1559  *   Passing the parameter by reference *
            1560  ****************************************
            1561  *   Variables, array element and whole array passing.
            1562  *
            1563  *
            1564  *   After SMB @FAC Entry should look like
            1565  *
            1566  *  | Pointer to         | >00 |      | Pointer to |          |
            1567  *  | symbol table       |     |      | value space |          |
            1568  *  | entry              |     |      |            |          |
            1569  *  FAC----------------FAC2--------FAC4---------FAC6---------
            1570  *  for a numeric case, and
            1571  *
            1572  *  | Pointer to         | >65 |      | Pointer to | String    |
            1573  *  | value space        |     |      | string     | length    |
            1574  *  | entry              |     |      |            |           |
            1575  *  FAC----------------FAC2--------FAC4---------FAC6---------
            1576  *  for a string case.
            1577  *
            1578  REF01
6CB1 0F14   1579      XML SMB              Get the location.
6CB3 CA42B8 1580      $IF @CHAT .HE. >B8 GOTO VALC1 Pass array expression.
6CB6 6C98
6CB8 8E4C4C 1581      $IF @FAC2 .EQ. 00 THEN                                  1
6CBB C3
6CBC BEC000 1582        ST 2,@0(PCODE)     Must be a numeric variable.        1
6CBF 1602
6CC1 4CC8   1583      $ELSE                                                   1
6CC3 BEC000 1584        ST 3,@0(PCODE)     Must be a string variable.         1
6CC6 1603
            1585      $END IF
```

```
                 1586  *
                 1587  ****************************************
                 1588  *  PUSH routine                        *
                 1589  *  Pushes @FAC entry into a value stack*
                 1590  ****************************************
                 1591  PUSH
6CC8 9012        1592     INC @CNT.            Increment number of parameter.
6CCA C61210      1593     $IF @CNT .H. 16 GOTO ERR3A  Too many parameters.
6CCD 6E25
6CCF 9116        1594     DINC @PCODE          Increment parm info address
                 1595  *
6CD1 0F17        1596     XML VPUSH
6CD3 4C33        1597     BR  PARG1            Get the next argument.
                 1598  *
                 1599  ****************************************
                 1600  *  EXECUTE routine                     *
                 1601  *  Restore file name info and transfer *
                 1602  *  control over to GLC                  *
                 1603  ****************************************
                 1604  EXEC1
6CD5 BE4A20      1605     ST   SPACE.@FAC      Store blank in the FAC area.
6CD8 350005      1606     MOVE 5 FROM @FAC TO @FAC+1
6CDB 4B4A
6CDD 350004      1607     MOVE 4 FROM RAM(12(OLDS)) TO @TEMP
6CE0 00E00C
6CE3 10

LAST ERROR FOUND AT LINE  1305

***** ERROR IN IMMEDIATE FIELD  LINE= 1608  COLUMN= 33

***** ERROR IN ADDRESS FIELD  LINE= 1608
6CE4 8F0240      1608     $IF @TEMP+2 .DEQ. 0 GOTO EX10  Do not move 0 bytes.
6CE7 00
6CE8 34024A      1609     MOVE @TEMP+2 FROM RAM(@TEMP) TO @FAC Move name to FAC.
6CEB B000
6CED BD5002      1610     DST  @TEMP+2,@FAC6   Copy name length

LAST ERROR FOUND AT LINE  1608

***** ERROR IN IMMEDIATE FIELD  LINE= 1611  COLUMN= 13

***** ERROR IN BRANCH ADDRESS  LINE= 1611
6CF0 0000        1611     BR  EX10             Join execution code.
                 1612  *
                 1613  ****************************************
                 1614  *  RETURN to the BASIC main program.   *
                 1615  ****************************************
                 1616  NOERR
6CF2 C56E10      1617     $WHILE @VSPTR .DH. @OLDS                          1
6CF5 4CFC
6CF7 0F18        1618       XML VPOP                Pop the stack.          1
6CF9 D56CF2      1619     $END WHILE                Keep popping till no stack for
                 1620  *
                 1621  RTBSC
6CFC D642B6      1622     $IF @CHAT .NE. RPAR$ GOTO ERR15    Check ")".
6CFF 4E07
                 1623  RTBSC2
6D01 0F1B        1624     XML PGMCH                 Advance to the next character.
6D03 8E424E      1625     $IF @CHAT .NE. CC GOTO ERR15    Check end of statement.
6D06 07
6D07 0F0012      1626     CALL DTL                  Go back to BASIC
```

```
                1627  *
                1628  ****************************************
                1629  *  SUBROUTINES used in this file.      *
                1630  ****************************************
                1631  CLRFAC
600A 864A       1632      CLR @FAC
600C 350007     1633      MOVE 7 FROM @FAC TO @FAC+1
600F 484A
6011 00         1634      RTN
                1635
```

```
                 1637
                 1638  ****************************************
                 1639  *                                      *
                 1640  * CHARPAT ROUTINE                      *
                 1641  *                                      *
                 1642  ****************************************
                 1643  *
                 1644  * FORMAT:
                 1645  *   CALL CHARPAT open ( <numeric expression> comma
                 1646  *                      <string expression> )* close
                 1647  * FUNCTION:
                 1648  *  RETURNS THE CHARACTER DEFINTION PATTERN FOR CHARCTER
                 1649  *  NUMBER <numeric expression> INTO <string expression>.
                 1650  *
                 1651  ****
     0010        1652  TBLPTR EQU  >10    Table pointer
     0012        1653  STRPTR EQU  >12    String pointer
                 1654  ****
                 1655  GETCHR
6D12 066AD2      1656   CALL SKIPN     Skip subroutine name
                 1657  GCHR2
6D15 0EB6        1658   PARSE RPAR$
6D17 D64C65      1659   $IF @FAC2 .EQ. STRING GOTO ERRSNM   Can't be a string
6D1A 6E20
6D1C 0F12        1660   XML CFI      Convert FAC to integer
                 1661   *  Note that 32 - 159 char are supported in Console BASIC
6D1E D65403      1662   $IF @FAC10 .EQ. 3 GOTO ERRNTB    (range 32 -> 159)
6D21 6E18
6D23 D34A00      1663   $IF @FAC .DLT. #32  GOTO ERRBA
6D26 204E25
6D29 CF4A00      1664   $IF @FAC .DGT. #159 GOTO ERRBA
6D2C 9F6E25
6D2F E34A00      1665   DSLL  @FAC,3     ...   (8 bytes / entry so multiply by 8 )
6D32 03
6D33 BF1003      1666   DST   >300,@TBLPTR  (Base of character table less 32*8 )
6D36 00
6D37 A1104A      1667   DADD  @FAC,@TBLPTR  (Add in arg offset)
                 1668
6D3A BF0C00      1669   DST  16,@BYTE        Get a 16 byte string in string space
6D3D 10
6D3E 060038      1670   CALL GETSTR
6D41 BD121C      1671   DST  @SREF,@STRPTR   Save pointer to string
6D44 BE0408      1672   ST  8,@TEMP4         Loop counter
                 1673  **
                 1674  GETLP
6D47 BC0080      1675      ST  RAM(@TBLPTR),@TEMP
6D4A 10
6D4B E70000      1676      DSRL @TEMP,4                Shift 4 bits to right.
6D4E 04
6D4F A20030      1677      ADD >30,@TEMP               Add ASCII "0"
6D52 CE0039      1678      $IF @TEMP .LE. >39 GOTO GCHR3   >39=ASCII("9")
6D55 4D5A
6D57 A20007      1679      ADD 7,@TEMP                 Value "A" -> "F"
                 1680  GCHR3
6D5A E60104      1681      SRL @TEMP+1,4               Shift 4 bits for next byte.
```

```
605D A20130  1682        ADD >30,@TEMP+1              Add ASCII "0"
6060 CE0139  1683        $IF @TEMP+1 .LE. >39 GOTO GCHR4
6063 4068
6065 A20107  1684        ADD 7,@TEMP+1                Value "A" -> "F"
             1685  GCHR4
6068 BDB012  1686        DST  @TEMP,RAM(@STRPTR)      Now put in RAM
606B 00
606C 9110    1687        DINC @TBLPTR                 Next byte
606E 9512    1688        DINCT @STRPTR                Move pointer two bytes
6070 9204    1689        DEC  @TEMP4                  Count down
6072 4047    1690        BR   GETLP                   Continue loop
             1691  *
             1692  * NOW ASSIGN THE STRING JUST CREATED TO THE STRING
             1693  * VARIABLE FOLLOWING
             1694  ***                                                  ***
             1695  ***                                                  ***
6074 0F1B    1696    XML  PGMCH      (SKIP ",")
6076 CA4280  1697    $IF @CHAT .NE. >80 GOTO ERRIS  (DON'T ALLOW TOKEN)
6079 6E07
607B 0F13    1698    XML  SYM        (GET SYMBOL NAME)
607D 0F14    1699    XML  SMB        (GET VALUE POINTER)
607F 0F17    1700    XML  VPUSH      (SAVE FAC ON STACK FOR ASSGNV)
             1701  ***
6081 D64C65  1702    $IF @FAC2 .NE. STRING GOTO ERRSNM   Must be a string var.
6084 4E20
6086 BF4A00  1703    DST  >001C,@FAC    Temp string so use SREF as address
6089 1C
608A BD4E1C  1704    DST  @SREF,@FAC4   Pointer to string
608D BF5000  1705    DST  16,@FAC6      String length
6090 10
6091 0F15    1706    XML  ASSGNV        Assign to string variable
6093 D64283  1707    $IF @CHAT .NE. COMMA$ GOTO RTBSC  Finish looping!
6096 4CFC
6098 0F1B    1708    XML  PGMCH         Advance pointer. Skip ","
609A 4015    1709    BR   GCHR2         Repeat the process
             1710
```

```
                 *****
                 ************************************************************
                 * ERROR BRANCH TABLE FOR LINK                              **
                 ************************************************************
                 *
                 *  ERROR MESSAGE EQUATES (msg are in console GROM)
                 *
 202C            MSG1    EQU  >202C        Incorrect statement
 2040            MSG4    EQU  >2040        Bad name
 2049            MSG5    EQU  >2049        Memory full
 2055            MSG6    EQU  >2055        Can't continue
                 MSG3
                 MSG14
 2064            MSG7    EQU  >2064        Bad value
 206E            MSG8    EQU  >206E        Number too big
 207D            MSG11   EQU  >207D        String-number missmatch
 2094            MSG12   EQU  >2094        Bad argument
 20A1            MSG13   EQU  >20A1        Bad subscript
 20AF            MSG15   EQU  >20AF        Name conflict
                 MSG2
 20BD            MSG17   EQU  >20BD        Can't do that
 2009            MSG2LN  EQU  >2009        Bad line number
 20F9            MSG18   EQU  >20F9        For-next error
 2113            MSG21   EQU  >2113        I/O error
 211D            MSG22   EQU  >211D        File error
 2128            MSG23   EQU  >2128        Input error
 2134            MSG24   EQU  >2134        Data error
 213F            MSG19   EQU  >213F        Line too long
                 *
                 ****   ERROR BRANCHES   ******
                 *
                 ERRBSC
 609C 9673               DECT  @SUBSTK          Adjust subroutine stack
 609E CA2208             $IF @ERRCOD .L. 8 GOTO ERRDSR   I/O error
 60A1 4E61
 60A3 CA220F             $IF @ERRCOD .HE. >F GOTO ERRORB  Do not close the file
 60A6 6DAB
 60A8 0668C4             CALL  CLSNO            Close the file
                 ERRORB
 60AB C62221             $IF @ERRCOD .H. >21 GOTO ERRUEC Unknown error
 60AE 6E5C
 60B0 A62208             SUB  8,@ERRCOD
 60B3 8A22               CASE @ERRCOD
 60B5 4DE9               BR  ERRMM              8   Memory full
 60B7 4E07               BR  ERRIS              9   Control char -- not issued
 60B9 4DEE               BR  ERRTAG             A   Bad tag
 60BB 4DF3               BR  ERRCHK             B   Checksum error
 60BD 4DF8               BR  ERRDD              C   Duplicate definition
 60BF 4DFD               BR  ERRUR              D   Unresolved reference
 60C1 4E07               BR  ERRIS              E   No entry address -- not issued
 60C3 4E02               BR  ERRPN              F   Program not found
 60C5 4E07               BR  ERRIS             10   Incorrect statement
 60C7 4E0C               BR  ERRBN             11   Bad name
 60C9 4E11               BR  ERRCC             12   Can't continue
 60CB 4E16               BR  ERRBV             13   Bad value
```

```
6DCD 4E18   1764      BR  ERRNTB          14   Number too big
6DCF 4E20   1765      BR  ERRSNM          15   String-number mismatch
6DD1 4E25   1766      BR  ERRBA           16   Bad argument
6DD3 4E2A   1767      BR  ERRBS           17   Bad subscript
6DD5 4E2F   1768      BR  ERRNC           18   Name conflict
6DD7 4E34   1769      BR  ERRCDT          19   Can't do that
6DD9 4E39   1770      BR  ERRBLN          1A   Bad line number
6DDB 4E3E   1771      BR  ERRFNE          1B   For-next error
6DDD 4E68   1772      BR  ERRIOB          1C   I/O error
6DDF 4E43   1773      BR  ERRFE           1D   File error
6DE1 4E48   1774      BR  ERRINP          1E   Input error
6DE3 4E4D   1775      BR  ERRDAT          1F   Data error
6DE5 4E52   1776      BR  ERRLTL          20   Line too long
6DE7 4DE9   1777      BR  ERRMM           21   Memory full without
                  1778  *                       file being open
                  1779  ****************************************
                  1780  *  ERROR HANDLING SECTION              *
                  1781  ****************************************
                  1782  ****
6DE9 06001C       1783  ERRMM  CALL ERR$$      * Memory full
6DEC 2049         1784         DATA #MSG5
                  1785  *
6DEE 06001C       1786  ERRTAG CALL ERR$$      * Bad tag
6DF1 6E60         1787         DATA #MSGTAG
                  1788  *
6DF3 06001C       1789  ERRCHK CALL ERR$$      * Checksum error
6DF6 6E79         1790         DATA #MSGCHK
                  1791  *
6DF8 06001C       1792  ERRDD  CALL ERR$$      * Duplicate definition
6DFB 6E88         1793         DATA #MSGDD
                  1794  *
6DFD 06001C       1795  ERRUR  CALL ERR$$      * Unresolved references
6E00 6E9D         1796         DATA #MSGUR
                  1797  *
6E02 06001C       1798  ERRPN  CALL ERR$$      * PROGRAM NOT FOUND
6E05 6EB2         1799         DATA #MSGPN
                  1800  *
6E07 06001C       1801  ERRIS  CALL ERR$$
6E0A 202C         1802         DATA #MSG1      * Incorrect statement
                  1803  *
6E0C 06001C       1804  ERRBN  CALL ERR$$
6E0F 2040         1805         DATA #MSG4      * BAD NAME
                  1806  *
6E11 06001C       1807  ERRCC  CALL ERR$$
6E14 2055         1808         DATA #MSG6      * Can't continue
                  1809  *
6E16 06001C       1810  ERRBV  CALL ERR$$
6E19 2064         1811         DATA #MSG7      * BAD VALUE
                  1812  *
6E1B 06001C       1813  ERRNTB CALL ERR$$
6E1E 206E         1814         DATA #MSG8      * NUMBER TOO BIG
                  1815  *
6E20 06001C       1816  ERRSNM CALL ERR$$
6E23 2070         1817         DATA #MSG11     * String Number mismatch
                  1818  *
```

```
6E25 06001C  1819  ERRBA   CALL ERR$$
6E28 2094    1820          DATA #MSG12      * BAD ARGUMENT
             1821  *
6E2A 06001C  1822  ERRBS   CALL ERR$$
6E2D 20A1    1823          DATA #MSG13      * BAD SUBSCRIPT
             1824  *
6E2F 06001C  1825  ERRNC   CALL ERR$$
6E32 20AF    1826          DATA #MSG15      * NAME CONFLICT
             1827  *
6E34 06001C  1828  ERRCDT  CALL ERR$$
6E37 20BD    1829          DATA #MSG17      * CAN'T DO THAT
             1830  *
6E39 06001C  1831  ERRBLN  CALL ERR$$
6E3C 2009    1832          DATA #MSGBLN     * Bad line number
             1833  *
6E3E 06001C  1834  ERRFNE  CALL ERR$$
6E41 20F9    1835          DATA #MSG18      * For-next error
             1836  *
6E43 06001C  1837  ERRFE   CALL ERR$$
6E46 211D    1838          DATA #MSG22      * File error
             1839  *
6E48 06001C  1840  ERRINP  CALL ERR$$
6E4B 2128    1841          DATA #MSG23      * Input error
             1842  *
6E4D 06001C  1843  ERRDAT  CALL ERR$$
6E50 2134    1844          DATA #MSG24      * Data error
             1845  *
6E52 06001C  1846  ERRLTL  CALL ERR$$
6E55 213F    1847          DATA #MSG19      * Line too long
             1848  *
6E57 06001C  1849  ERRRMB  CALL ERR$$
6E5A 6EC4    1850          DATA #MSGRAM     * No memory expansion
             1851  *
6E5C 06001C  1852  ERRUEC  CALL ERR$$
6E5F 6ED8    1853          DATA #MSGUEC     * Unknown error code
             1854  *
             1855  ********
             1856  ************************************************************
             1857  *  ERRDSR -- routine to handle IO errors for BASIC         *
             1858  ************************************************************
             1859  ERRDSR
6E61 B0041C  1860          DST  @SREF,@TEMP4        Copy PAB address
6E64 A70400  1861          DSUB 04,@TEMP4           BASIC assumes offset by 4
6E67 04
             1862  *
6E68 06001C  1863  ERRIOB  CALL ERR$$
6E6B 2113    1864          DATA #MSG21      * I/O error
             1865  **
             1866  ********
             1867  ********
             1868  ****  Error messages
             1869  *******
             1870          BASE 0,0,0,0,0,0,>60
6E6D 0BA9AC  1872  MSGTAG DATA 11.:ILLEGAL TAG:
6E79 0EA3A8  1873  MSGCHK DATA 14.:CHECKSUM ERROR:
```

```
6E88 14A4B5  1874  MSGDD  DATA 20,:DUPLICATE DEFINITION:
6E9D 14B5AE  1875  MSGUR  DATA 20,:UNRESOLVED REFERENCE:
6EB2 11B0B2  1876  MSGPN  DATA 17,:PROGRAM NOT FOUND:
6EC4 13AEAF  1877  MSGRAM DATA 19,:NO MEMORY EXPANSION:
6ED8 12B5AE  1878  MSGUEC DATA 18,:UNKNOWN ERROR CODE:
     7000    1880  ALCODE EQU >7000
             1881  *    END
```

```
SOURCE ACCESS NAME=      PCD2.ASSM.SRC.BSCSUP2
OBJECT ACCESS NAME=      PCD2.ASSM.OBJ.BSCSUP2
LISTING ACCESS NAME=     PCD2.ASSM.LST.BSCSUP2
ERROR ACCESS NAME=       PCD2.ATA.LST.ERRS
OPTIONS=                 XREF.BUNLST,TUNLST,DUNLST
MACRO LIBRARY PATHNAME=
```

```
0001                    IDT  'BSCSUP'
0002                 *
0003                 *     BASIC SUPPORT ROUTINES: contain numeric reference and
0004                 *     assignment (NUMREF,NUMASG), string reference and assig
0005                 *     (STRREF,STRASG), and error reporting (ERR).
0006                 *
0007        00CE     SET   EQU  >00CE        Works with console, not debugger cons
0008                 *
0009                 *     SYSTEM EQUATES
0010                 *
0011        000E     SCNKEY EQU  >000E
0012        6010     EXBASX EQU  >6010        EXTENDED BASIC XML TABLES (BASE)
0013        200A     PARM   EQU  >200A        Information for parameter type.
0014        8300     PAD    EQU  >8300
0015        830C     BYTE   EQU  >830C
0016        8310     OLDS   EQU  >8310        OLD VALUE PTR BEFORE EVALUATE SUB
0017                 *                        ARGUMENT
0018        8312     COUNT  EQU  >8312        NO. OF ARGUMENT
0019        831C     SREF   EQU  >831C
0020        8322     ERRCOD EQU  >8322        PLACE TO RETURN ERROR CODE
0021        8343     BASE   EQU  >8343
0022        8348     FLAG   EQU  >8348        Flag used in GPL main program.
0023        834A     FAC    EQU  >834A
0024        8354     SCTEMP EQU  >8354
0025        8355     SCLEN  EQU  >8355
0026        8356     SCNAME EQU  >8356
0027        8350     FAC6   EQU  FAC+6
0028        836E     VSPTR  EQU  >836E        VALUE STACK PTR
0029        8370     MAXVDP EQU  >8370
0030        837C     STATUS EQU  >837C
0031        83CC     SNDADD EQU  >83CC
0032        83CE     STFLGS EQU  >83CE
0033        8300     CRULST EQU  >8300
0034        83E0     GPLWS  EQU  >83E0        GPL/EXTENDED BASIC WORKSPACE
0035                 *
0036        4000     WRVDP  EQU  >4000        Write enable for VDP
0037        8800     VDPRD  EQU  >8800        VDP read data address
0038        8C00     VDPWD  EQU  >8C00        VDP write data address
0039        8C02     VDPWA  EQU  >8C02        VDP write address address
0040                 *
0041                 *     ERROR EQUATES
0042                 *
0043        1000     ERRIS  EQU  >1000        Incorrect statement
0044        1100     ERRBN  EQU  >1100        Bad name
0045        2100     ERRMM  EQU  >2100        Memory full
0046        1200     ERRCC  EQU  >1200        Can't continue
0047        1300     ERRBV  EQU  >1300        Bad value
0048        1400     ERRNTB EQU  >1400        Number too big
0049        1500     ERRSNM EQU  >1500        String number mismatch
0050        1600     ERRBA  EQU  >1600        Bad argument
0051        1700     ERRBS  EQU  >1700        Bad subscript
0052        1800     ERRNC  EQU  >1800        Name conflict
0053        1900     ERRCDT EQU  >1900        Can't do that
0054        1A00     ERRBLN EQU  >1A00        Bad line number
0055        1B00     ERRFNE EQU  >1B00        For next error
0056        1C00     ERRIOB EQU  >1C00        I/O error
0057        1D00     ERRFE  EQU  >1D00        File error
0058        1E00     ERRINP EQU  >1E00        Input error
0059        1F00     ERRDAT EQU  >1F00        Data error
0060        2000     ERRLTL EQU  >2000        Line too long
```

```
0061        2200  ERRSYN EQU  >2200          Syntax error
0062        2300  ERRNO  EQU  >2300          Numeric overflow
0063        2400  ERRUC  EQU  >2400          Unrecognized character
0064        0F00  ERRSNF EQU  >0F00          Program not found
0065        2500  ERRST  EQU  >2500          String truncated
0066              *
0067              *
0068              *      UTILITY BLWP VECTORS
0069              *
0070                     REF  GPLLNK,UTLTAB
0071                     DEF  NUMASG,NUMREF,STRASG,STRREF,ERR
0072 0000 0014'  NUMASG DATA UTILWS,NAENTR     Numeric Assignment
0073 0004 0014'  NUMREF DATA UTILWS,NRENTR     Numeric reference
0074 0008 0014'  STRASG DATA UTILWS,SAENTR     String assignment
0075 000C 0014'  STRREF DATA UTILWS,SRENTR     String reference
0076 0010 0014'  ERR    DATA UTILWS,ERENTR     Return error code to BASIC
0077              *
0078              *      UTILTIY WORKSPACE
0079              *
0080 0014 0000   UTILWS DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0081        0019' R2LB   EQU   UTILWS+5
0082        001D' R4LB   EQU   UTILWS+9
0083              *
0084              ** D A T A   C O N S T A N T S
0085              *
0086 0034    65   CBH65  BYTE >65
0087 0035    20   EQBIT  BYTE >20
0088              *
```

```
0090                 *
0091                 *        NUMERIC ASSIGNMENT UTILITY
0092                 *
0093                 *
0094                 *        UTILITY ROUTINE NAENTR
0095                 *          A utility to allow a numeric value to be assigned
0096                 *          to a numeric variable or a numeric array element
0097                 *          passed in a parameter list of LINK call.
0098                 *          The numeric value to be assigned should be in FAC a
0099                 *          The stack has info. for each argument.
0100                 *          For a simple numeric variale, the stack entry looks
0101                 *          like this :
0102                 *               ----------------------------------------
0103                 *               |Ptr to      |>00|  |Ptr to        |      |
0104                 *               |S.T. entry  |   |  |Value space  |      |
0105                 *               ----------------------------------------
0106                 *          For an array element, the stack entry looks like:
0107                 *               ----------------------------------------
0108                 *               | Ptr to     |  |  |Ptr to       |      |
0109                 *               | S.T. entry |  |  |Dim. info.   |      |
0110                 *               ----------------------------------------
0111                 *          REGISTER USAGE
0112                 *            R0 : Array element no. or zero
0113                 *            R1 : Argument number
0114                 *            R0,R1,R2,R3,R4,R5,R6,R7,R8 are used in this
0115                 *            routine
0116                 *
0117                 *            5 types of arguments :
0118                 *        Identifier 0 : Numeric Expression
0119                 *        Identifier 1 : String Expression
0120                 *        Identifier 2 : Numeric Variable
0121                 *        Identifier 3 : String Variable
0122                 *        Identifier 4 : Numeric Array
0123                 *        Identifier 5 : String Array
0124                 *
0125                 *
0126 0036 C010  NAENTR   MOV   *R13,R0     For the change of the work space
0127 0038 C060           MOV   @2(R13),R1  For the change of the work space
     003A 0002
0128 003C 06A0           BL    @CKARG      Check valid argument no. and
     003E 007E
0129                 ***************  set up stack entry ptr in R5 ***************
0130                 *******  Load parameter information address >200A ***********
0131                 ****Check  SIMPLE NUMERIC VAR. OR AN ARRAY REF.  ***********
0132 0040 0003           MOVB  *R3,R3      Get the one byte identifier in
0133                 *                      CPU
0134 0042 1368           JEQ   NUERR2      Can't do the assignment to a
0135                 *                      numeric expression argument
0136 0044 0983           SRL   R3,9        Make it double
0137 0046 0643           DECT  R3          Flag 2 is for NUMERIC VARIABLE
0138 0048 1617           JNE   NUM04       No - go to the code for array
0139                 ******  FOR ASSIGMENT TO A SIMPLE NUMERIC VAR.     *********
0140 004A C000           MOV   R0,R0       In this case, R0 must contain 0
0141 004C 1663           JNE   NUERR2      ERROR : BAD ARGUMENT
0142 004E C0C5           MOV   R5,R3       Get the ptr to the stack entry
0143 0050 05C3           INCT  R3          Point to the ID byte of the
0144                 *                      stack entry
0145 0052 06A0           BL    @GETV1      Get 1 byte (ID) from VDP        |
     0054 036C
0146                 *                      Numeric ID byte must be 0       |
```

```
0147 0056 165A          JNE   NUERR1     ERROR:STRING-NO. MISMATCH
0148 0058 05C3          INCT  R3         Point to the value space ptr
0149 005A 06A0          BL    @GET1      Get value address
     005C 0330'
0150 005E C0C1          MOV   R1,R3      Just copy
0151 0060 C103 NUM00    MOV   R3,R4      PUT routine uses R4
0152 0062 0203          LI    R3,FAC     Load FAC address
     0064 834A
0153 0066 0202 NUM01    LI    R2,4       8 bytes to copy
     0068 0004
0154 006A C073 NUM02    MOV   *R3+,R1    Get data from FAC area
0155 006C 06A0          BL    @PUT1      Move two bytes at a time.
     006E 034C'
0156 0070 05C4          INCT  R4         Increment VDP address
0157 0072 0602          DEC   R2         Decrement the counter - done?
0158 0074 15FA          JGT   NUM02      No - loop for more
0159 0076 0320          RTWP             Yes - return to the caller
0160              ***** FOR ASSIGNMENT TO ONE ELEMENT OF AN ARRAY ***********
0161              ***** R0 : ARRAY ELEMENT NO. R5: PTR TO STACK ENTRY ******
0162 0078 06A0 NUM04    BL    @ARYBND    Check array element  w/in bound
     007A 00A2'
0163              *                      and find the ptr to element in ERAM
0164 007C 10F1          JMP   NUM00      Go to the general code to move
0165              *                      from FAC to ERAM
0166              ***********************************************************
0167              *******  CHECK VALID ARGUMENT NO. AND SET UP STACK ENTRY **
0168              ********  PTR IN R5                                  *****
0169 007E C041 CKARG    MOV   R1,R1      Argument no. can't be 0
0170 0080 1349          JEQ   NUERR2     ERROR : BAD ARGUMENT
0171 0082 0A81          SLA   R1,8       Make one byte
0172 0084 9060          CB    @COUNT,R1  Valid argument?
     0086 8312
0173 0088 1145          JLT   NUERR2     ERROR:BAD ARGUMENT
0174 008A 0981          SRL   R1,8       Make it two byte again
0175 008C C141          MOV   R1,R5      Get the argument no.
0176 008E 0A35          SLA   R5,3       8 bytes per stack entry
0177 0090 0225          AI    R5,8       Skip the 8 bytes file name
     0092 0008
0178              *                      on top of stack
0179 0094 A160          A     @OLDS,R5   Add to the old stack ptr
     0096 8310
0180              *                      R5 : Ptr to the stack entry
0181 0098 C0C1          MOV   R1,R3      Copy argument number
0182 009A 0603          DEC   R3         Offset by 1
0183 009C 0223          AI    R3,PARM    Get address of parm info.
     009E 200A
0184 00A0 045B          RT
0185              ***********************************************************
0186              ********* CHECK THE ARRAY ELEMENT NO. IS WITHIN BOUND  *****
0187 00A2 C24B ARYBND   MOV   R11,R9     Save rtn addr.
0188 00A4 0643          DECT  R3         Flag 4 is for ARRAY
0189 00A6 1632          JNE   NUERR1     NO-Assume string arg.
0190              *                      Error:STRING-NO MISMATCH
0191 00A8 C0C5          MOV   R5,R3      Get the stack entry ptr
0192 00AA 06A0          BL    @GET1      Get the S.T. entry ptr
     00AC 0330'
0193 00AE C0C1          MOV   R1,R3      R1 : Data req. for GET1
0194              *                      R3 : Addr. req. for GET1
0195 00B0 06A0          BL    @GETV1     Get the 1st byte of S.T. entry
     00B2 034C'
```

```
0196 00B4 112B              JLT   NUERR1      String bit is set - ERROR
0197                 *                        STRING-NO MISMATCH
0198 00B6 06A0              BL    @ARY1       Go through this general routine
     00B8 00C2
0199                 *** R3 has address of the value space *****
0200 00BA 6004              S     R4,R0       Check the BASE : 0 : O.K.
0201                 *                                BASE : 1 : offset 1
0202 00BC 0A30              SLA   R0,3        6 bytes per element
0203 00BE A0C0              A     R0,R3       Add to the address of value space
0204 00C0 0459              B     *R9
0205                 ************************************************************
0206 00C2 C28B       ARY1   MOV   R11,R10     Save RTN addr.
0207 00C4 0A51              SLA   R1,5        R1 still contains 1st byte from
0208                 *
0209                 *                        func bits
0210 00C6 09D1              SRL   R1,13       Shift back to use as a double
0211                 ******  R1:NO. OF DIMENSION. R5: STACK ENTRY PTR   ********
0212 00C8 C201              MOV   R1,R8       Save no. of dimension for later
0213 00CA D120              MOVB  @BASE,R4    Check the BASE
     00CC 8343
0214 00CE 0984              SRL   R4,8        Make it double
0215 00D0 1303              JEQ   NUM08       If BASE=0,INDEX=0 -- it's o.k.
0216 00D2 0600              DEC   R0          Decrement the INDEX
0217 00D4 1123              JLT   NUERR3      BASE=1, INDEX=0---ERROR
0218                 *                        BAD SUBSCRIPT
0219 00D6 0580              INC   R0          Restore INDEX
0220 00D8 0206       NUM08  LI    R6,1        Initial value for accumulator
     00DA 0001
0221 00DC C0C5              MOV   R5,R3       Put the stack entry ptr in R3
0222 00DE 0223              AI    R3,4        Try to get the dim. info. ptr
     00E0 0004
0223 00E2 06A0              BL    @GET1       Go to get the dim. info. ptr
     00E4 0330
0224 00E6 C0C1              MOV   R1,R3       Put it in R3
0225 00E8 0643              DECT  R3          For the following INCT instruct.
0226 00EA 05C3       NUM10  INCT  R3          Point to dimension infor. entry
0227                 *                        2-byte maxima per dimension
0228 00EC 06A0              BL    @GET1       Get the dimension maxima in R1
     00EE 0330
0229 00F0 0581              INC   R1          BASE=0, add 1 offset
0230 00F2 6044              S     R4,R1       BASE=1, O.K.
0231 00F4 3981              MPY   R1,R6       Get the max. array index
0232                 *                        R6 : accumulator
0233 00F6 C186              MOV   R6,R6       First 2 bytes must be 0 here
0234 00F8 1611              JNE   NUERR3      else BAD SUBSCRIPT
0235 00FA C187              MOV   R7,R6       Get last two bytes result
0236 00FC 0608              DEC   R8          Decrement the counter
0237                 *                        R8 : no. of dimension
0238 00FE 15F5              JGT   NUM10       Loop for more
0239 0100 0606              DEC   R6          BASE=0:element no. must range
0240                 *                        from 0 to (R6-1)
0241 0102 A184              A     R4,R6       BASE=1:element no. range 1 -- R6
0242 0104 8180              C     R0,R6       User's INDEX must <= max. INDEX
0243 0106 150A              JGT   NUERR3      NO - ERROR:BAD SUBSCRIPT
0244 0108 05C3              INCT  R3          Contains value space address in
0245                 *                        VDP RAM.
0246 010A 045A              B     *R10        RTN
0247                 ************************************************************
0248 010C 0200       NUERR1 LI    R0,ERRSNM   STRING-NO. MISMATCH
     010E 1500
```

```
0249 0110 0460            B      @ERROR      Error out
     0112 039E'
0250 0114 0200  NUERR2    LI     R0,ERRBA    BAD ARGUMENT
     0116 1600
0251 0118 0460            B      @ERROR      Error out
     011A 039E'
0252 011C 0200  NUERR3    LI     R0,ERRBS    BAD SUBSCRIPT
     011E 1700
0253 0120 0460            B      @ERROR
     0122 039E'
```

```
                *
                *       NUMERIC REFERENCE UTILITY
                *
                *
                *         UTILITY ROUTINE NRENTR
                *          A utility to allow reference to a numeric exp.
                *          , numeric variable or a numeric array element
                *          passed in a parameter list of a LINK call.
                *          The numeric value will be placed in the FAC
                *          area after calling NRENTR.
                *          The stack has info. for each argument.
                *          For a simple numeric variale, the stack entry looks
                *          like this :
                *               ----------------------------------------
                *               |Ptr to      |>001|  |Ptr to        |        |
                *               |S.T. entry  |    |  |Value space   |        |
                *               ----------------------------------------
                *
                *          For an array reference, the stack entry looks
                *          like this :
                *             ----------------------------------------
                *             | Ptr to      |  |  |Ptr to       |        |
                *             | S.T. entry  |  |  |Dim. info.   |        |
                *             ----------------------------------------
                *
                *          For a numeric exp., the stack entry has the
                *          8 bytes floating points value passed by a parse
                *          routine.
                *         REGISTER USAGE
                *          R0 : Array element no. or 0
                *          R1 : Argument number
                *          R0,R1,R2,R3,R4,R5,R6,R7,R8 are used in this
                *          routine
                *
0124 C01D  NRENTR  MOV   *R13,R0     For the change of the work space
0126 C06D          MOV   @2(R13),R1  For the change of the work space
0128 0002
012A 06A0          BL    @CKARG      Check valid argument no.
012C 007E'
                *                    and set up stack ptr and parm add
                *******  R5 : Ptr. to the stack entry, R1 : Argument no.
                *******  CHECK THE ARGUMENT IDENTIFIER IN CPU RAM **********
                ***** A NUMERIC EXP., SIMPLE NUMERIC VAR. OR AN ARRAY ******
012E D003          MOVB  *R3,R3      Get the one byte identifier in
                *                    CPU
0130 0983          SRL   R3,8        Make it double
0132 160E          JNE   REF00       Not a numeric expression
                ******  A NUMERIC EXP., MOVE THE 8 BYTES STACK ENTRY *******
                ******  FROM VDP TO FAC.                              ******
0134 C000          MOV   R0,R0       R0 in this case must contain 0
0136 1629          JNE   RFERR2      ERROR : BAD ARGUMENT
0138 0202          LI    R2,8        8 bytes to move
013A 0008
013C 0204          LI    R4,FAC      Destination addr.
013E E34A
0140 C0C5          MOV   R5,R3       Source addr.: stack entry ptr
0142 06A0  REF03   BL    @GET1       Get 2 bytes from VDP
0144 0330'
0146 CD01          MOV   R1,*R4+     Put 2 bytes in FAC
0148 0503          INCT  R3          Update source addr.
014A 0642          DECT  R2          Update the counter
014C 15FA          JGT   REF03       More to move
```

```
0310 014E 0380                RTWP              Return to caller
0311                 ********  A SIMPLE NUMERIC VAR. OR AN ARRAY ***************
0312 0150 0643  REF00         DECT  R3          Flag 2 is for NUMERIC VARIABLE
0313 0152 1616                JNE   REF04       NO -- go to the code for array
0314                 ****** FOR REFERENCE TO A SIMPLE NUMERIC VAR.  *********
0315 0154 C000                MOV   R0,R0       In this case, R0 must contain 0
0316 0156 1619                JNE   RFERR2      ERROR : BAD ARGUMENT
0317 0158 C0C5                MOV   R5,R3       Get the ptr to the stack entry
0318 015A 05C3                INCT  R3          Point to the ID byte of the
0319                 *                          stack entry
0320 015C 06A0                BL    @GETV1      Get 1 byte (ID) from VDP
     015E 036C'
0321                 *                          Numeric ID byte must be 0
0322 0160 1612                JNE   RFERR1      ERROR:STRING-NO. MISMATCH
0323 0162 05C3                INCT  R3          Points to value stack
0324 0164 06A0                BL    @GET1       Get value space address
     0166 0330'
0325 0168 C0C1                MOV   R1,R3       Copy for later use
0326                 *                                                          |
0327 016A 0204  REF01         LI    R4,FAC      Destination addr. is in FAC     |
     016C 834A
0328 016E 0202                LI    R2,4        Move 8 bytes value              |
     0170 0004
0329 0172 06A0  REF02         BL    @GET1       Move 2 bytes from VDP to R1     |
     0174 0330'
0330 0176 C001                MOV   R1,*R4+     Move 2 bytes into FAC area      |
0331 0178 05C3                INCT  R3          Increment VDP address pointer   |
0332 017A 0602                DEC   R2                                          |
0333 017C 15FA                JGT   REF02       Read more                       |
0334 017E 0380                RTWP                                              |
0335                 *                          return to caller
0336                 *****  FOR ASSIGNMENT TO ONE ELEMENT OF AN ARRAY ***********
0337                 *****  R0 : ARRAY ELEMENT NO.  R5 : PTR TO STACK ENTRY ******
0338 0180 06A0  REF04         BL    @ARYBND     Check array elem. is w/in bound
     0182 00A2'
0339                 *          and find the ptr to the array element in ERAM *
0340 0184 10F2                JMP   REF01       Go to the general code to move
0341                 *                          from ERAM to FAC
0342                 ***********************************************************
0343 0186 0460  RFERR1        B     @NUERR1     STRING-NO. MISMATCH
     0188 010C'
0344 018A 0460  RFERR2        B     @NUERR2     BAD ARGUMENT
     018C 0114'
```

```
0346                 ***********************************************************
0347                 *
0348                 *       STRING ASSIGNMENT UTILITY
0349                 *
0350                 *
0351                 *       UTILITY ROUTINE SAENTR
0352                 *         A utility to allow a string to be assigned to a
0353                 *         string variable or a string array element passed
0354                 *         in a parameter list of a LINK call.
0355                 *         The stack has info. for each argument.
0356                 *         For a string variale, the stack entry looks like
0357                 *         this ----------------------------------------
0358                 *              |Ptr to      |>65|  |Pointer  | String  |
0359                 *              |Value space|    |  |to String| Length  |
0360                 *              ----------------------------------------
0361                 *         For a string array, the stack entry looks like
0362                 *         this ----------------------------------------
0363                 *              |Ptr to      |    |  |Ptr to   |         |
0364                 *              |S. T. entry|    |  |Dim info.|         |
0365                 *              ----------------------------------------
0366                 *         USER'S REGISTER USAGE-- following registers must
0367                 *           be set up prior to the SAENTR call.
0368                 *           R0 : Array element no. or zero
0369                 *           R1 : Argument number
0370                 *           R2 : String address in ERAM. (the first byte
0371                 *                of the string is the length of the string.)
0372                 ***********************************************************
0373                 *       THIS ROUTINE REGISTER USAGE (UTILWS)
0374                 *           R0 : Array element
0375                 *           R1 : Argument number
0376                 *           R2 : Return address for VGASSN
0377                 *           R5 : Value stack address
0378                 *           R9 : String address copied from user's reg 3
0379                 *           R10: Return address for ARY1
0380                 ***********************************************************
0381 018E C01D SAENTR  MOV   *R13,R0        Array element from user
0382 0190 C06D         MOV   @2(R13),R1     Argument number
     0192 0002
0383 0194 C26D         MOV   @4(R13),R9     String addr. in ERAM
     0196 0004
0384                 *
0385 0198 06A0         BL    @CKARG         Check valid argument no. and
     019A 007E
0386                 *                      set up ptr to stack entry in R5
0387                 *       R3 has address of parm type info
0388 019C D0D3         MOVB  *R3,R3         Get the one byte identifier in
0389                 *                      CPU
0390 019E 0983         SRL   R3,8           Make it double
0391 01A0 0603         DEC   R3             Check if string expression
0392 01A2 1324         JEQ   STERR2         Can't do the assignment to a
0393                 *                      string expression argument
0394 01A4 0643         DECT  R3             Flag 3 is for string variable
0395 01A6 163A         JNE   STARY          NO : Is STRING ARRAY ?
0396                 ******  ASSIGNMENT TO STRING VARIABLE  *******
0397 01A8 C000         MOV   R0,R0          Array element has to be zero.
0398 01AA 1620         JNE   STERR2         ERROR : BAD ARGUMENT
0399 01AC C0C5         MOV   R5,R3          Addr.reg. for GETV1
0400 01AE 05C3         INCT  R3             Point to the ID byte of the
0401                 *                      stack entry
0402 01B0 06A0         BL    @GETV1         Get 1 byte (ID) from VDP
```

```
     01B2 036C'
0403 01B4 9801          CB    R1,@CBH65  Is this a string variable ?
     01B6 0034'
0404 01B8 1615          JNE   STERR1     No - STRING NO. MISMATCH ERROR
0405                 ************************************************
0406                 ******  PUSH THIS STACK ENTRY ON TOP OF THE STACK FOR ******
0407                 ******  ROUTINE "ASSGNV" TO USE                       ******
0408                 ************************************************
0409                 ****    MOVE THIS STRING VARIABLE ENTRY TO CPU FAC AREA ***
0410                 ****    IN ORDER TO DO A VPUSH                          ***
0411 01BA 0206          LI    R6,8       # of bytes to move
     01BC 0008
0412 01BE 0204          LI    R4,FAC     Destination addr. in CPU
     01C0 834A
0413 01C2 C0C5          MOV   R5,R3      Restore stack entry ptr
0414                 *                   R3 : addr. reg. for GET1
0415 01C4 06A0  STR02   BL    @GET1      Get 2 bytes from VDP
     01C6 0330'
0416 01C8 C001          MOV   R1,*R4+    Put to FAC area
0417 01CA 05C3          INCT  R3         Update source addr.
0418 01CC 0646          DECT  R6         Update the counter
0419 01CE 15FA          JGT   STR02      More to move
0420                 ******  DO THE VPUSH, GETSTR, COPY STRING TO STRING SPACE
0421                 ******  AND ASSIGNMENT                    ******************
0422 01D0 06A0          BL    @VGASSN
     01D2 0246'
0423                 **********  NOW UPDATES THE STACK ENTRY OF THIS  *******
0424                 **********  STRING VARIABLE                      *******
0425                 **********  R5 STILL POINTS TO THE OLD STACK ENTRY *******
0426                 **********  FOR THIS STRING VARIABLE             *******
0427                 **********  (length only, pointer is garbage anyway) *****
0428 01D4 C105          MOV   R5,R4
0429 01D6 0224          AI    R4,6       "STRING LENGTH" entry
     01D8 0006
0430 01DA C060          MOV   @BYTE,R1   Put the length into data reg.
     01DC 830C
0431                 *********  SRL   R1,8       Make it two bytes
0432 01DE 06A0          BL    @PUT1      Put 2 bytes of data to VDP
     01E0 034C'
0433 01E2            STRTN
0434 01E2 0380          RTWP             Return
0435                 ************************************************
0436 01E4 0200  STERR1  LI    R0,ERRSNM  STRING-NO. MISMATCH
     01E6 1500
0437 01E8 0460          B     @ERROR
     01EA 039E'
0438 01EC 0200  STERR2  LI    R0,ERRBA   BAD ARGUMENT
     01EE 1600
0439 01F0 0460          B     @ERROR     Error return
     01F2 039E'
0440                 ************************************************
0441                 ************ Common routine for STRASG & STRREF to check
0442                 ************ valid array element
0443 01F4 C08B  STARYC  MOV   R11,R2     Save RTN addr.
0444 01F6 0643          DECT  R3         Flag 5 is for string array
0445 01F8 16F5          JNE   STERR1     ERROR : STRING NO. MISMATCH
0446 01FA C0C5          MOV   R5,R3      Get the stack entry ptr
0447 01FC 06A0          BL    @GET1      Get the S.T. entry ptr
     01FE 0330'
0448 0200 C0C1          MOV   R1,R3      R1 : Data reg. for GET1
```

```
0449                *                                   R3 : Addr. req. for GET1
0450 0202 06A0                BL    @GETV1      Get the 1st byte of S.T. entry
     0204 036C'
0451 0206 1102                JLT   STARY1      O.K.
0452 0208 0460                B     @STERR1     ERROR : STRING-NO MISMATCH
     020A 01E4'
0453 020C           STARY1
0454                *  R0 has array element number
0455 020C 06A0                BL    @ARY1       Go through general routine to
     020E 00C2'
0456                *                 check array element is out of bound
0457                *  R4:BASE, R0:ARRAY ELEMENT, R3:PTR TO BEGINNING OF VALUE
0458                *                                                   SPACE
0459 0210 6004                S     R4,R0       Check the BASE : 0 : O.K.
0460                *                                       BASE : 1 : Offset 1
0461 0212 0A10                SLA   R0,1        2 bytes per element
0462 0214 A0C0                A     R0,R3       Ptr to the exect element we want
0463 0216 06A0                BL    @GET1       Get the string ptr
     0218 0330'
0464 021A 0452                B     *R2         RTN
0465                ***********************************************************
0466                ***********************************************************
0467 021C 06A0      STARY     BL    @STARYC     Go through general routine
     021E 01F4'
0468                *                             shared by STRREF
0469                *   NOW  R1 : PTR TO THE OLD STRING
0470                *        R3 : PTR TO THE VALUE SPACE
0471                *** SET UP FAC ENTRY AND PUSH IT ON STACK FOR ASSIGN ROUT. *
0472 0220 0206                LI    R6,FAC      Destination addr.
     0222 834A
0473 0224 CD83                MOV   R3,*R6+     Put ptr to value space in
0474 0226 D0A0                MOVB  @C8H65,*R6+ Put >65 ID in
     0228 0034'
0475 022A D084                MOVB  R4,*R6+     Clear the unuse byte
0476 022C CD81                MOV   R1,*R6+     Put the ptr to string in
0477 022E C0C1                MOV   R1,R3       Set up addr. reg. for GETV1
0478 0230 1602                JNE   STARY4      Not null ptr : o.k.
0479 0232 0406                CLR   *R6         Clear the length bytes
0480 0234 1005                JMP   STARY6      Go on
0481 0236 0603      STARY4    DEC   R3          Point to string length
0482 0238 06A0                BL    @GETV1      Get the length from VDP
     023A 036C'
0483 023C 0981                SRL   R1,8        Make it two bytes
0484 023E C581                MOV   R1,*R6      Put the length in FAC area
0485                ******  FAC ENTRY HAS BEEN SET UP NOW  *********************
0486 0240           STARY6
0487                *
0488 0240 06A0                BL    @VGASSN     Do the VPUSH, GETSTR AND ASSIGN
     0242 0246'
0489                *****   ASSIGNMENT WILL POP THE STACK, OLD STACK ENTRY ****
0490                *****   DOES NOT NEED TO BE UPDATED IN THIS CASE  ******
0491 0244 0380                RTWP              Return...
0492                ***********************************************************
0493                ***********************************************************
0494                ***********************************************************
0495 0246 C088      VGASSN    MOV   R11,R2          Save rtn addr.
0496                *********  VPUSH  routine in console ROM  ****************
0497                ***********************************************************
0498 0248 02E0                LWPI  GPLWS       Load GPL workspace.
     024A 83E0
```

```
0499 024C 020B          LI    R11,>1EAA    XML table addr. for VPUSH
     024E 1EAA
0500 0250 0698          BL    *R11         Do the VPUSH
0501 0252 02E0          LWPI  UTILWS       Reload utility workspace.
     0254 0014'
0502                 ************************************************************
0503                 *****  GETS THE STRING SPACE, COPIES THE STRING FROM ERAM **
0504                 *****  INTO VDP STRING SPACE, AND SETS UP THE FAC WITH A  **
0505                 *****  STRING ENTRY OF THE FOLLOWING FORM:                **
0506                 *****  ---------------------------------------------      **
0507                 *****  | >001C | >65 | XX |  POINTER  |  LENGTH  |       **
0508                 *****  |       |     |    | TO STRING | OF STRING |       **
0509                 *****  ---------------------------------------------      **
0510                 ***** FAC     +2     +3   +4          +6                  **
0511                 ************************************************************
0512 0256 0199          MOVB  *R9,R6       R9:Addr. of string (length byte
0513                 *                     & string) supplies by ALC
0514 0258 0986          SRL   R6,8         Set up the length byte
0515 025A 0806          MOVB  R6,@STATUS   MAKE SURE NO ACCIDENTAL ERROR EX
     025C 837C
0516 025E
0517 025E C806          MOV   R6,@BYTE     For GETSTR routine
     0260 830C
0518 0262 C806          MOV   R6,@FAC6     Set up length byte entry in FAC
     0264 8350
0519                 ************  GETSTR routine in console GROM ***************
0520                 ************************************************************
0521 0266 0420          BLWP  @GPLLNK      Call GETSTRG routine
     0268 0000
0522 026A 0038          DATA  >0038
0523                 *************  SET UP FAC ENTRY  ******************
0524 026C 0206          LI    R6,FAC       Optimize to save bytes
     026E 834A
0525 0270 0204          LI    R4,>001C     Get addr. of SREF
     0272 001C
0526 0274 C084          MOV   R4,*R6+      Indicate a temporary string
0527 0276 D0A0          MOVB  @CBH65,*R6+  Indicate a string
     0278 0034'
0528 027A D084          MOVB  R4,*R6+      Byte is not used
0529 027C C5A0          MOV   @SREF,*R6    Save ptr to the string.
     027E 831C
0530                 ***************  COPY THE STRING TO STRING SPACE  ***********
0531 0280 C220          MOV   @BYTE,R8     Get # of bytes to copy.
     0282 830C
0532 0284 1309          JEQ   STR04        If none to copy
0533 0286 C116          MOV   *R6,R4       Get pointer to destination
0534                 *                     R4:addr. reg. for PUTV1 below
0535 0288 C0C9          MOV   R9,R3        R9:Ptr to string (length byte)
0536 028A 0583          INC   R3           Skip the len byte-ptr to string
0537 028C D073  STR06   MOVB  *R3+,R1      Get one byte from ERAM
0538 028E 06A0          BL    @PUTV1       Put 1 byte in VDP
     0290 0380'
0539 0292 0584          INC   R4           Update the destination addr.
0540 0294 0608          DEC   R8           1 less to move
0541 0296 15FA          JGT   STR06        If not done-loop for more
0542                 *                     ASSGNV destroys all the reg.
0543                 *******   Call ASSNV routine in console ROM  **************
0544                 ********
0545 0298 02E0  STR04   LWPI  GPLWS        Load GPL workspace
     029A 83E0
```

```
0546 029C 020B          LI    R11,>1788   Access to ASSGN directly...
     029E 1788
0547 02A0 069B          BL    *R11        Call ASSGNV
0548 02A2 02E0          LWPI  UTILWS      Re-load utility workspace
     02A4 0014
0549            ********MOV   R2,R11      Restore RTN addr..
0550            ********RT
0551 02A6 0452          B     *R2
```

```
0553                *
0554                *
0555                *        STRING REFERENCE UTILITY
0556                *
0557                *
0558                *      UTILITY ROUTINE SRENTR
0559                *        A utility to allow a ref. to a string expression,
0560                *        a string variable or a string array element passed
0561                *        in a parameter list of LINK call.
0562                *
0563                *        The stack has info. for each argument.
0564                *         For a string expression, the stack entry looks
0565                *         likes this
0566                *              ---------------------------------------
0567                *              | >001C      |>65|  |Pointer    | String |
0568                *              |or Ptr to   |   |  |to String  | Length |
0569                *              |Value space|   |  |           |        |
0570                *              ---------------------------------------
0571                *              >001C : For a temporary string
0572                *              Ptr to Value Space : for a permanent string
0573                *      For a string variale, the stack entry looks like
0574                *      this ---------------------------------------
0575                *              |Ptr to      |>65|  |Pointer    | String |
0576                *              |Value space|   |  |to String| Length |
0577                *              ---------------------------------------
0578                *         NOTE: in this case, the pointer to the string may
0579                *         no longer be valid.
0580                *         For a string array, the stack entry looks likes
0581                *         this
0582                *              ---------------------------------------
0583                *              |Ptr to      |  |  |Ptr to     |        |
0584                *              |S.T. entry  |  |  |Dim info.  |        |
0585                *              ---------------------------------------
0586                *
0587                *      USER'S REGISTER USAGE -- the following registers mu
0588                *        be set up prior to the call.
0589                *        R0 : Array element or zero
0590                *        R1 : Argument number
0591                *        R2 : String-address in ERAM, (the first byte
0592                *              of the string is the length of the string.)
0593                *              Before calling SRENTR : The string len has
0594                *              the max. length which can be allocated for
0595                *              the string.
0596                *              After SRENTR call: The string len is the actu
0597                *              length of the string which is followed by the
0598                *              string itself.
0599                *
0600                ***************************************************************
0601 02A8 C01D  SRENTR     MOV    *R13,R0      For the change of the work space
0602                *                           Array element now is in R0
0603 02AA C06D             MOV    02(R13),R1   For the change of the work space
     02AC 0002
0604                *                           Argument no. is in R1
0605 02AE 06A0             BL     @CKARG       Check valid argument no. and
     02B0 007E
0606                *                           set up ptr to stack entry in R5
0607                * R3 points to parm type info address
0608 02B2 0003             MOVB   *R3,R3       Get one byte identifier in CPU
0609 02B4 0983             SRL    R3,8         Make it double
0610 02B6 0603             DEC    R3           Flag 1 is for string expression
```

```
0611 02B8 1302                JEQ   SRF01      String expression is o.k.
0612 02BA 0643                DECT  R3         Flag 3 is for string variable
0613 02BC 162C                JNE   STARYF     Not string exp. or var., must be
0614                *                          string array in this case.
0615                *************  REF. TO STRING EXP. OR STRING VAR. *********
0616 02BE C000  SRF01         MOV   R0,R0      R0 must be 0 in this case.
0617 02C0 1631                JNE   SRFER2     ERROR : BAD ARGUMENT
0618 02C2 C02D                MOV   @4(R13),R0 Now move string ptr into R0
     02C4 0004
0619 02C6 C0C5                MOV   R5,R3      Addr.reg. for GETV1
0620 02C8 05C3                INCT  R3         Point to the ID byte of the
0621                *                          stack entry
0622 02CA 06A0                BL    @GETV1     Get 1 byte (ID) from VDP
     02CC 036C'
0623 02CE 9801                CB    R1,@CBH65  Is this a string variable ?
     02D0 0034'
0624 02D2 1626                JNE   SRFER1     No - STRING NO. MISMATCH ERROR
0625 02D4 0643                DECT  R3         back to first word of entry
0626 02D6 06A0                BL    @GET1
     02D8 0330'
0627 02DA 0281                CI    R1,SREF-PAD  is this a temporary?
     02DC 001C
0628 02DE 1302                JEQ   SRF02      if so, pointer on stack is OK.
0629 02E0 C0C1                MOV   R1,R3      otherwise go back to symbol table
0630 02E2 1002                JMP   SRF025
0631 02E4 0223  SRF02         AI    R3,4       Points to "ptr to string"
     02E6 0004
0632 02E8 06A0  SRF025        BL    @GET1      Get the "ptr to string" from
     02EA 0330'
0633                *                          stack entry in VDP
0634 02EC C041  SRF03         MOV   R1,R1      Is it null ptr?
0635 02EE 1307                JEQ   SRF05      Yes - put 0 in length byte
0636 02F0 C181                MOV   R1,R6      R6 now contains "ptr to string"
0637 02F2 0601                DEC   R1         Points to the length byte of
0638                *                          this string in string space
0639 02F4 C0C1                MOV   R1,R3      Set up addr. reg. for GETV1
0640 02F6 06A0                BL    @GETV1     Get the 1 byte actual length
     02F8 036C'
0641                *                          of this string in R1
0642 02FA 9050                CB    *R0,R1     Compare the max. len. user
0643                *                          provided with the actual len.
0644 02FC 1A15                JL    SRFER3     Not enough space - ERROR
0645                *                          STRING TRUNCATED ERROR
0646 02FE DC01  SRF05         MOVB  R1,*R0+    Enough space - o.k.
0647                *                          Put the actual len. into the
0648                *                          length byte
0649 0300 1309                JEQ   SRF08      0 byte length - don't move
0650                ****** R0 : Points to the allocated string space in ERAM ***
0651                *                          Destination addr. for the
0652                *                          following move
0653 0302 C0C6                MOV   R6,R3      Set up addr. reg. for GETV1
0654                *                          R3 now contains "ptr to string"
0655                *                          Source addr. for following move
0656 0304 0981                SRL   R1,8       Make actual len. byte a double
0657 0306 C141                MOV   R1,R5      R5 : # of bytes to move
0658 0308 06A0  SRF07         BL    @GETV1     Get 1 byte from VDP
     030A 036C'
0659 030C DC01                MOVB  R1,*R0+    Put 1 byte into ERAM
0660 030E 0583                INC   R3         Update the source addr.
0661 0310 0605                DEC   R5         Update the counter
```

```
0662 0312 15FA            JGT   SRF07      More to move - loop back
0663 0314 0380  SRF08     RTWP             Return the caller
0664                      ******************************************************
0665                      ************  REF. TO STRING ARRAY      ******************
0666 0316 06A0  STARYF    BL    @STARYC    Check array element is within
     0318 01F4'
0667                      *     bound and find the string element ptr in value space
0668 031A C020            MOV   @4(R13),R0 Put provided string addr.in ERAM
     031C 0004
0669                      *  by the user into R0 now. (for the change of work space)
0670 031E 10E6            JMP   SRF03      Go to the general code to move
0671                      *                the string from VDP to ERAM
0672                      ******************************************************
0673 0320 0460  SRFER1    B     @NUERR1    STRING-NO. MISMATCH
     0322 010C'
0674 0324 0460  SRFER2    B     @NUERR2    BAD ARGUMENT
     0326 0114'
0675 0328 0200  SRFER3    LI    R0,ERRST   STRING TRUNCATED
     032A 2500
0676 032C 0460            B     @ERROR     Error out
     032E 039E'
```

```
0678                ***************************************************
0679                ***     UTILITY ROUTINES TO ACESS VDP           ***
0680                ***************************************************
0681                ****** GET1 : Get two bytes of data from VDP
0682                ******      R3 : Address in VDP
0683                ******      R1 : Where the two bytes data stored
0684 0330 06C3      GET1        SWPB  R3
0685 0332 D803                  MOVB  R3,@VDPWA
     0334 8C02
0686 0336 06C3                  SWPB  R3
0687 0338 D803                  MOVB  R3,@VDPWA
     033A 8C02
0688 033C 1000                  NOP
0689 033E D060                  MOVB  @VDPRD,R1
     0340 8800
0690 0342 06C1                  SWPB  R1
0691 0344 D060                  MOVB  @VDPRD,R1
     0346 8800
0692 0348 06C1                  SWPB  R1
0693 034A 045B                  RT
0694                ****** PUT1 : Put two bytes of data into VDP
0695                ******      R4 : Address on VDP
0696                ******      R1 : Data
0697 034C 06C4      PUT1        SWPB  R4
0698 034E D804                  MOVB  R4,@VDPWA
     0350 8C02
0699 0352 06C4                  SWPB  R4
0700 0354 0264                  ORI   R4,WRVDP
     0356 4000
0701 0358 D804                  MOVB  R4,@VDPWA
     035A 8C02
0702 035C 1000                  NOP
0703 035E D801                  MOVB  R1,@VDPWD
     0360 8C00
0704 0362 06C1                  SWPB  R1
0705 0364 D801                  MOVB  R1,@VDPWD
     0366 8C00
0706 0368 06C1                  SWPB  R1
0707 036A 045B                  RT
0708                ****** GETV1 : Get 1 byte of data from VDP
0709                ******      R3 : Address in VDP
0710                ******      R1 : Where the 1 byte data stored
0711 036C 06C3      GETV1       SWPB  R3
0712 036E D803                  MOVB  R3,@VDPWA
     0370 8C02
0713 0372 06C3                  SWPB  R3
0714 0374 D803                  MOVB  R3,@VDPWA
     0376 8C02
0715 0378 1000                  NOP
0716 037A D060                  MOVB  @VDPRD,R1
     037C 8800
0717 037E 045B                  RT
0718                ****** PUTV1 : Put 1 bytes of data into VDP
0719                ******      R4 : Address on VDP
0720                ******      R1 : Data
0721 0380 06C4      PUTV1       SWPB  R4
0722 0382 D804                  MOVB  R4,@VDPWA
     0384 8C02
0723 0386 06C4                  SWPB  R4
0724 0388 0264                  ORI   R4,WRVDP
```

```
        038A 4000
 0725   038C 0804          MOVB  R4,BVDPWA
        038E 8C02
 0726   0390 1000          NOP
 0727   0392 0801          MOVB  R1,BVDPWD
        0394 8C00
 0728   0396 045B          RT
```

```
0730                 ********            ********
0731                 **     ERROR ROUTINE      **
0732                 ********            ********
0733 039E C81D  ERENTR MOV  *R13.@ERRCOD
     03A0 8322
0734 03A2 1002         JMP  ERR2
0735 03A4 C800  ERROR  MOV  R0.@ERRCOD
     03A6 8322
0736 03A8 02E0  ERR2   LWPI GPLWS
     03AA 83E0
0737 03AC 020B         LI   R11.UTLTAB
     03AE 0000
0738 03B0 022B         AI   R11.14
     03B2 000E
0739 03B4 C2DB         MOV  *R11.R11   restore GPL return address
0740 03B6 0460         B    @SET
     03B8 00CE
0741                   END
NO ERRORS,    NO WARNINGS
```

LABEL	VALUE	DEFN	REFERENCES								
ARY1	00C2'	0206	0198	0455							
ARYBND	00A2'	0187	0162	0338							
BASE	8343	0021	0213								
BYTE	830C	0015	0430	0517	0531						
CBH65	0034'	0086	0403	0474	0527	0623					
CKARG	007E'	0169	0126	0289	0385	0605					
COUNT	8312	0018	0172								
CRULST	8300	0033									
EOBIT	0035'	0087									
ERENTR	0398'	0733	0076								
ERR D	0010'	0076	0071								
ERR2	03A2'	0736	0734								
ERRBA	1600	0050	0250	0438							
ERRBLN	1A00	0054									
ERRBN	1100	0044									
ERRBS	1700	0051	0252								
ERRBV	1300	0047									
ERRCC	1200	0046									
ERRCDT	1900	0053									
ERRCOD	8322	0020	0733	0735							
ERRDAT	1F00	0059									
ERRFE	1000	0057									
ERRFNE	1B00	0055									
ERRINP	1E00	0058									
ERRIOB	1C00	0056									
ERRIS	1000	0043									
ERRLTL	2000	0060									
ERRNM	2100	0045									
ERRNC	1800	0052									
ERRNO	2300	0062									
ERRNTB	1400	0048									
ERROR	039E'	0735	0249	0251	0253	0437	0439	0676			
ERRSNF	0F00	0064									
ERRSNM	1500	0049	0248	0436							
ERRST	2500	0065	0675								
ERRSYN	2200	0061									
ERRUC	2400	0063									
EXBASX	6010	0012									
FAC	834A	0023	0027	0152	0303	0327	0412	0472	0524		
FAC6	8350	0027	0518								
FLAG	8348	0022									
GET1	0330'	0684	0149	0192	0223	0228	0305	0324	0329	0415	0447
			0463	0626	0632						
GETV1	036C'	0711	0145	0195	0320	0402	0450	0482	0622	0640	0658
GPLLNK R	0268'	0070	0521								
GPLNS	83E0	0034	0498	0545	0736						
MAXVDP	8370	0029									
NAENTR	0036'	0126	0072								
NRENTR	0124'	0287	0073								
NUERR1	010C'	0248	0147	0189	0196	0343	0673				
NUERR2	0114'	0250	0134	0141	0170	0173	0344	0674			
NUERR3	011C'	0252	0217	0234	0243						
NUM00	0060'	0151	0164								
NUM01	0066'	0153									
NUM02	006A'	0154	0158								
NUM04	0078'	0162	0138								
NUM08	0008'	0220	0215								
NUM10	00EA'	0226	0238								
NUMASG D	0000'	0072	0071								
NUMREF D	0004'	0073	0071								

LABEL	VALUE	DEFN	REFERENCES								
OLDS	8310	0016	0179								
PAD	8300	0014	0627								
PARM	200A	0013	0183								
PUT1	034C'	0697	0155	0432							
PUTV1	0360'	0721	0538								
R0	0000		0126	0140	0140	0200	0202	0203	0216	0219	0242
			0248	0250	0252	0287	0300	0300	0315	0315	0361
			0397	0397	0436	0438	0459	0461	0462	0601	0616
			0616	0618	0642	0646	0659	0668	0675	0735	
R1	0001		0127	0150	0154	0169	0169	0171	0172	0174	0175
			0181	0193	0207	0210	0212	0224	0229	0230	0231
			0288	0306	0325	0330	0382	0403	0416	0430	0448
			0476	0477	0483	0484	0537	0603	0623	0627	0629
			0634	0634	0636	0637	0639	0642	0646	0656	0657
			0659	0689	0690	0691	0692	0703	0704	0705	0706
			0716	0727							
R10	000A		0206	0246							
R11	000B		0187	0206	0443	0495	0499	0500	0546	0547	0737
			0738	0739	0739						
R13	0000		0126	0127	0287	0285	0381	0382	0383	0601	0603
			0615	0646	0733						
R2	0002		0153	0157	0302	0308	0326	0332	0443	0464	0495
			0551								
R2LB	0019'	0081									
R3	0003		0132	0132	0136	0137	0142	0143	0148	0150	0151
			0152	0154	0181	0182	0183	0188	0191	0193	0203
			0221	0222	0224	0225	0226	0244	0294	0294	0296
			0304	0307	0312	0317	0318	0323	0325	0331	0388
			0388	0390	0391	0394	0399	0400	0413	0417	0444
			0446	0448	0462	0473	0477	0481	0535	0536	0537
			0608	0608	0609	0610	0612	0619	0620	0625	0629
			0631	0639	0653	0660	0684	0685	0686	0687	0711
			0712	0713	0714						
R4	0004		0151	0156	0200	0213	0214	0230	0241	0303	0306
			0327	0330	0412	0416	0428	0429	0459	0475	0525
			0526	0528	0533	0539	0697	0698	0699	0700	0701
			0721	0722	0723	0724	0725				
R4LB	0010'	0082									
R5	0005		0142	0175	0176	0177	0179	0191	0221	0304	0317
			0399	0413	0428	0446	0619	0657	0661		
R6	0006		0220	0231	0233	0233	0235	0239	0241	0242	0411
			0416	0472	0473	0474	0475	0476	0479	0484	0512
			0514	0515	0517	0518	0524	0526	0527	0528	0529
			0533	0636	0653						
R7	0007		0235								
R8	0008		0212	0236	0531	0540					
R9	0009		0167	0204	0383	0512	0535				
REF00	0150'	0312	0297								
REF01	016A'	0327	0340								
REF02	0172'	0329	0333								
REF03	0142'	0305	0309								
REF04	0180'	0338	0313								
RFERR1	0186'	0343	0322								
RFERR2	018A'	0344	0301	0316							
SAENTR	018E'	0381	0074								
SCLEN	8355	0025									
SCNAME	8356	0026									
SCNKEY	000E	0011									
SCTEMP	8354	0024									
SET	00CE	0007	0740								

LABEL	VALUE	DEFN	REFERENCES								
SNDADD	83CC	0031									
SREF	831C	0019	0529	0627							
SRENTR	02A8'	0601	0075								
SRFO1	02BE'	0616	0611								
SRFO2	02E4'	0631	0628								
SRFO2S	02E8'	0632	0630								
SRFO3	02EC'	0634	0670								
SRFO5	02FE'	0646	0635								
SRFO7	0308'	0658	0662								
SRFO8	0314'	0663	0649								
SRFER1	0320'	0673	0624								
SRFER2	0324'	0674	0617								
SRFER3	0328'	0675	0644								
STARY	021C'	0467	0395								
STARY1	020C'	0453	0451								
STARY4	0236'	0481	0478								
STARY6	0240'	0486	0480								
STARYC	01F4'	0443	0467	0666							
STARYF	0316'	0666	0613								
STATUS	837C	0030	0515								
STERR1	01E4'	0436	0404	0445	0452						
STERR2	01EC'	0438	0392	0398							
STFLGS	83CE	0032									
STRO2	01C4'	0415	0419								
STRO4	0298'	0545	0532								
STRO6	028C'	0537	0541								
STRASG D	0008'	0074	0071								
STRREF D	000C'	0075	0071								
STRTN	01E2'	0433									
UTILWS	0014'	0080	0072	0073	0074	0075	0076	0081	0082	0501	0548
UTLTAB R	03A8'	0070	0737								
VDPRD	8800	0037	0689	0691	0716						
VDPWA	8C02	0039	0685	0687	0698	0701	0712	0714	0722	0725	
VDPWD	8C00	0038	0703	0705	0727						
VGASSN	0246'	0495	0422	0488							
VSPTR	836E	0028									
WRVDP	4000	0036	0700	0724							

```
SOURCE ACCESS NAME=      SD2.ASSM.SRC.SAVE2
OBJECT ACCESS NAME=      SD2.ASSM.OBJ.SAVE2
LISTING ACCESS NAME=     SD2.ASSM.LST.SAVE2
ERROR ACCESS NAME=
OPTIONS=                 XREF.BUNLST,TUNLST.DUNLST
MACRO LIBRARY PATHNAME=
```

```
0001                    IDT  'SAVE'
0002                *-----------------------------------------------------
0003                *
0004                *       Save memory image to cassette or disk
0005                *
0006                *       The user must load his own program in to cou
0007                *       ram, and then load this utility.  He must put
0008                *       a DEF START.STOP in his program.  This utility will
0009                *       promct for a file name.
0010                *
0011                *-----------------------------------------------------
0012                        REF  VSBW.VSBR.VMBW.GPLLNK
0013                        REF  DSRLNK.KSCAN
0014                        REF  GRMWA.GRMRA.GRMRD.GPLWS
0015                        DEF  SAVE
0016                        REF  SFIRST.SLAST.SLOAD.VMBR
0017 2800                   AORG >2800
0018                *
0019 2800   28              TEXT '(C) COPYRIGHT 1981 BY TEXAS INSTRUMENTS. INC.'
0020 282E                   EVEN
0021                *
0022        8356    SCNAME EQU  >8356
0023        837C    STATUS EQU  >837C
0024        8375    KEYCOD EQU  >8375
0025                *
0026 282E   20      MENU   TEXT '                                        '  1
0027 284E   20             TEXT '  SAVE MEMORY IMAGE                    '  2
0028 286E   20      BLANK  TEXT '                                        '  3
0029 288E   20             TEXT ' 1. PROGRAM TO BE SAVED MUST           '  4
0030 28AE   20             TEXT '    ALREADY BE LOADED.                 '  5
0031 28CE   20             TEXT '                                       '  6
0032 28EE   20             TEXT ' 2. THE FIRST WORD IN THE              '  7
0033 290E   20             TEXT '    PROGRAM MUST BE EXECUTABLE.        '  8
0034 292E   20             TEXT '                                       '  9
0035 294E   20             TEXT ' 3. PROGRAMS WHICH ARE BIGGER          '  10
0036 296E   20             TEXT '    THAN 8K WILL CREATE NEW            '  11
0037 298E   20             TEXT '    FILE NAMES BY ADDING A ONE         '  12
0038 29AE   20             TEXT '    TO THE LAST BYTE OF THE            '  13
0039 29CE   20             TEXT '    CURRENT FILE NAME.                 '  14
0040 29EE   20             TEXT '                                       '  15
0041 2A0E   20             TEXT ' CURRENT FILE :                        '  16
0042        0202    SADDR  EQU  $-MENU+2
0043 2A2E   20             TEXT '                                       '  17
0044 2A4E   20             TEXT ' STATUS:                               '  18
0045        0242    EADDR  EQU  $-MENU+2
0046 2A6E   20             TEXT '                                       '  19
0047 2A8E   20             TEXT '                                       '  20
0048 2AAE   20             TEXT '                                       '  21
0049 2ACE   20             TEXT '                                       '  22
0050 2AEE   20             TEXT '  ENTER FILE NAME?                     '  23
0051 2B0E   20             TEXT '                                       '  24
0052        0300    MENLEN EQU  $-MENU
0053        02C2    PADDR  EQU  22*32+2
0054        02E2    FADDR  EQU  (23*32)+2
0055 2B2E   4353    CS     DATA 'CS'
0056 2B30   01      ONE    BYTE 1
0057 2B31   00             BYTE 0
0058 2B32                  EVEN
0059        2B31    FILEN  EQU  $-1
0060 2B32           FILE   BSS  32          MUST BE ON EVEN WORDBOUNDARY
```

```
0061      2B52  FILEND EQU  $
0062            *RR1   TEXT ' SFIRST IS LESS THAN >A000
0063 2B52   20  ERR2   TEXT ' SLAST IS INVALID
0064 2B72   20  ERR3   TEXT ' I/O ERROR
0065 2B92   20  ERR4   TEXT ' BAD FILE NAME
0066 2BB2 0000  ERRCOD DATA 0
0067 2BB4 0600  CPAB   DATA >0600,>1000.0
0068 2BBA 2000  FSIZE  DATA >2000.0
0069      0F80  PAB    EQU  >F80
0070      0F89  PABFL  EQU  >F80+9
0071      0F8A  PABFN  EQU  >F80+10
0072 2BBE 0000  HEADER DATA 0.0,0
0073      0300  SAB    EQU  >300
0074 2BC4               EVEN
0075 2BC4   B7  INITS  BYTE >B7,>F,>80,>F,>D0
0076      2BC5  POS    EQU  INITS+1
0077      2BC6  OFFS   EQU  $-1
0078 2BCA               EVEN
0079 2BCA   B7  ON     BYTE >B7,>F
0080 2BCC   00  CURSOR BYTE 0.0,0,0,0,0,>7C,>7C
0081 2BD4 0000  LOAD   DATA 0
```

```
0083                *-------------------------------------------------------
0084                *
0085                *       SCANS KEY BOARD FOR FILE NAME INPUT
0086                *
0087                *-------------------------------------------------------
0088 2BD6           SAVE
0089 2BD6 04C0              CLR  R0
0090 2BD8 0201              LI   R1,MENU
     2BDA 282E
0091 2BDC 0202              LI   R2,MENLEN
     2BDE 0300
0092 2BE0 0420              BLWP @VMBW
     2BE2 0000
0093 2BE4 0200      SAVE05  LI   R0,>400        SET UP SPRITE
     2BE6 0400
0094 2BE8 0201              LI   R1,CURSOR
     2BEA 2BCC
0095 2BEC 0202              LI   R2,8
     2BEE 0008
0096 2BF0 0420              BLWP @VMBW
     2BF2 2BE2
0097 2BF4 C820              MOV  @ON,@INITS     TURN SPRITES ON
     2BF6 2BCA
     2BF8 2BC4
0098                *
0099 2BFA           SAV20
0100 2BFA 0203              LI   R3,FADDR       R3 -> CURRENT CHAR POSITION
     2BFC 02E2
0101 2BFE C820              MOV  @ON,@INITS     INITIALIZE SPRITES
     2C00 2BCA
     2C02 2BC4
0102 2C04           SAV30
0103 2C04 06A0              BL   @WSPRIT        WRITE SPRITE
     2C06 2C6C
0104 2C08 0420              BLWP @KSCAN         WAIT FOR A KEY
     2C0A 0000
0105 2C0C D0A0              MOVB @STATUS,R2     IS THIS A NEW KEY?
     2C0E 837C
0106 2C10 1603              JNE  SAV40          YES
0107 2C12 10F8              JMP  SAV30          NO, WAIT
0108 2C14 0460      EXIT    B    @DONE
     2C16 2DF8
0109                *
0110 2C18           SAV40
0111 2C18 04C1              CLR  R1
0112 2C1A D060              MOVB @KEYCOD,R1     R1 -> KEY
     2C1C 8375
0113 2C1E 0281              CI   R1,>0800       NO, BACK SPACE?
     2C20 0800
0114 2C22 1317              JEQ  BACKSP
0115 2C24 0281              CI   R1,>0000       ENTER KEY?
     2C26 0000
0116 2C28 132A              JEQ  ENTER
0117 2C2A 0281              CI   R1,>0500
     2C2C 0500
0118 2C2E 13F2              JEQ  EXIT
0119 2C30 0281              CI   R1,>0F00
     2C32 0F00
0120 2C34 13EF              JEQ  EXIT
0121 2C36 0281              CI   R1,>2000
```

```
        2C38 2000
0122    2C3A 1AE4          JL   SAV30
0123    2C3C C003          MOV  R3,R0          WRITE KEY ON SCREEN
0124    2C3E 0420          BLWP @VSBW
        2C40 0000
0125    2C42 0583          INC  R3             BUMP CURSOR ADDRESS
0126    2C44 0283          CI   R3,24*32-1
        2C46 02FF
0127    2C48 1B08          JH   SAV20
0128    2C4A B820          AB   @EIGHT,@POS    MOV SPRITE OVER
        2C4C 2C5C
        2C4E 2BC5
0129    2C50 1009          JMP  SAV30          NO. GET NEXT KEY
0130    2C52       BACKSP
0131    2C52 0283          CI   R3,FADDR       AT THE BEGINNING?
        2C54 02E2
0132    2C56 1206          JLE  SAV30          YES, IGNORE KEY
0133    2C58 0603          DEC  R3             NO, BACK UP
0134    2C5A 0201          LI   R1,>0800
        2C5C 0800
0135         2C5C  EIGHT   EQU  $-2
0136    2C5E 7801          SB   R1,@POS        ADJUST SPRITE
        2C60 2BC5
0137    2C62 1000          JMP  SAV30          GET NEXT KEY
0138    2C64 0201  ENAME   LI   R1,ERR4
        2C66 2B92
0139    2C68 0460          B    @EI010
        2C6A 2D9A
0140    2C6C       WSPRIT
0141    2C6C 0200          LI   R0,SAB
        2C6E 0300
0142    2C70 0201          LI   R1,INITS
        2C72 2BC4
0143    2C74 0202          LI   R2,5
        2C76 0005
0144    2C78 0420          BLWP @VMBW
        2C7A 2BF2
0145    2C7C 045B          RT
0146    2C7E       ENTER
0147    2C7E D820          MOVB @OFFS,@INITS
        2C80 2BC8
        2C82 2BC4
0148    2C84 06A0          BL   @WSPRIT
        2C86 2C6C
0149    2C88 0200          LI   R0,FADDR
        2C8A 02E2
0150    2C8C 0201          LI   R1,FILE
        2C8E 2B32
0151    2C90 0202          LI   R2,32
        2C92 0020
0152    2C94 0420          BLWP @VMBR          READ FILE NAME
        2C96 0000
```

```
0154                *-----------------------------------------------------
0155                *
0156                *       SCAN FILE NAME FOR LENGTH
0157                *
0158                *       SET UP R6 -> CURRENT START ADDRESS
0159                *              R7 -> CURRENT #BYTES LEFT TO SAVE
0160                *              R8 -> LAST BYTE IN CPU FILE NAME BUFFER
0161                *              R4 -> LENGTH OF THE FILE NAME + 1
0162                *
0163                *-----------------------------------------------------
0164 2C98 0203              LI    R3,FILE
     2C9A 2832
0165 2C9C 0283      SAV45   CI    R3,FILEND     SCAN FOR END OF FILE NAME.
     2C9E 2852
0166 2CA0 15E1              JGT   ENAME
0167 2CA2 9833              CB    *R3+,@BLANK
     2CA4 286E
0168 2CA6 16FA              JNE   SAV45
0169 2CA8 C203              MOV   R3,R8         R8 -> LAST BYTE IN FILE NAME
0170 2CAA 0648              DECT  R8
0171 2CAC 0204              LI    R4,FILE
     2CAE 2832
0172 2CB0 60C4              S     R4,R3         CALCULATE LENGTH OF FILE NAME
0173 2CB2 C103              MOV   R3,R4         R4 -> # BYTES TO MOVE INTO VDP RAM
0174 2CB4 0603              DEC   R3            R3 = LENGTH BYTE
0175 2CB6 0283              CI    R3,3
     2CB8 0003
0176 2CBA 11D4              JLT   ENAME         BAD DEVICE NAME
0177 2CBC 06C3              SWPB  R3
0178 2CBE D803              MOVB  R3,@FILEN
     2CC0 2831
0179                *
0180 2CC2 0206              LI    R6,SLOAD      PROGRAM'S LOAD ADDRESS
     2CC4 0000
0181 2CC6 C806              MOV   R6,@LOAD
     2CC8 2B04
0182 2CCA 0206              LI    R6,SFIRST     GET PROGRAM'S START ADDRESS
     2CCC 0000
0183                *       CI    R6,>A000      VALID?
0184                *       JL    ESTART        NO
0185 2CCE 0207              LI    R7,SLAST      GET PROGRAMS STOP ADDRESS
     2CD0 0000
0186 2CD2 61C6              S     R6,R7         FILE SIZE
0187 2CD4 115B              JLT   ESTOP         BAD FILE SIZE
```

```
0189                *-----------------------------------------------------------
0190                *
0191                *       MAIN LOOP TO SAVE EACH FILE IN MEMORY IMAGE FORMAT
0192                *            CALCUATES FILE SIZE
0193                *            BUILDS HEADER
0194                *            WRITES FAB
0195                *            CHECKS TO SEE IF NEED TO WRITE ANOTHER FILE
0196                *
0197                *-----------------------------------------------------------
0198 2CD6 C087  MAIN    MOV  R7,R2          #BYTES TO MOVE
0199 2CD8 0287          CI   R7,>2000-6     BIGGER THAN 8K?
     2CDA 1FFA
0200 2CDC 1102          JLT  MAIN10         NOPE
0201 2CDE 0202          LI   R2,>2000-6
     2CE0 1FFA
0202 2CE2 C282  MAIN10  MOV  R2,R10         R10 = # BYTES TO MOVE
0203 2CE4 0222          AI   R2,6           MAKE ROOM FOR HEADER
     2CE6 0006
0204 2CE8 C802          MOV  R2,@FSIZE
     2CEA 2BBA
0205 2CEC 0200          LI   R0,PAB
     2CEE 0F80
0206 2CF0 0201          LI   R1,CPAB
     2CF2 2BB4
0207 2CF4 0202          LI   R2,9
     2CF6 0009
0208 2CF8 0420          BLWP @VMBW          WRITE 10 BYTES OF PAB
     2CFA 2C7A
0209                *
0210 2CFC C084          MOV  R4,R2          #BYTES IN FILE NAME
0211 2CFE 0200          LI   R0,PABFL
     2D00 0F89
0212 2D02 0201          LI   R1,FILEN       MOVE FILENAME INTO PAB
     2D04 2B31
0213 2D06 0420          BLWP @VMBW
     2D08 2CFA
0214 2D0A 04C0          CLR  R0
0215 2D0C 0201          LI   R1,MENU
     2D0E 282E
0216 2D10 0202          LI   R2,MENLEN      RESTORE SCREEN
     2D12 0300
0217 2D14 0420          BLWP @VMBW
     2D16 2D08
0218 2D18 0200          LI   R0,SADDR
     2D1A 0202
0219 2D1C 0201          LI   R1,FILE
     2D1E 2B32
0220 2D20 C084          MOV  R4,R2
0221 2D22 0602          DEC  R2
0222 2D24 0420          BLWP @VMBW
     2D26 2D16
0223                *
0224 2D28 04E0          CLR  @HEADER
     2D2A 2BBE
0225 2D2C 81CA          C    R10,R7         MORE DATA TO WRITE?
0226 2D2E 1302          JEQ  MAIN20         NO
0227 2D30 0720          SETO @HEADER        YES, SET FLAG TO INDICATE THIS
     2D32 2BBE
0228 2D34       MAIN20
0229 2D34 C820          MOV  @FSIZE,@HEADER+2   LOAD FILE SIZE
```

```
      2D36 2BBA
      2D38 2BC0
0230  2D3A C820          MOV  @LOAD,@HEADER+4       LOAD START ADDRESS
      2D3C 2BD4
      2D3E 2BC2
0231  2D40 0200          LI   R0,>1000
      2D42 1000
0232  2D44 0201          LI   R1,HEADER
      2D46 2BBE
0233  2D48 0202          LI   R2,6
      2D4A 0006
0234  2D4C 0420          BLWP @VMBW         WRITE HEADER IN PABBUF
      2D4E 2D26
0235                *
0236  2D50 0200          LI   R0,>1006      WRITE AFTER THE HEADER
      2D52 1006
0237  2D54 C08A          MOV  R10,R2        R10=SIZE
0238  2D56 C046          MOV  R6,R1         R6=START
0239  2D58 0420          BLWP @VMBW         MOVE PROGRAM INTO VDP
      2D5A 2D4E
0240                *
0241  2D5C 6820          C    @FILE,@CS     CASSETTE?
      2D5E 2B32
      2D60 2B2E
0242  2D62 1323          JEQ  CASSIO
0243  2D64        DISKIO
0244  2D64 0201          LI   R1,PABFL      DO A MEMORY IMAGE SAVE
      2D66 0F89
0245  2D68 C801          MOV  R1,@SCNAME
      2D6A 8356
0246  2D6C 0420          BLWP @DSRLNK
      2D6E 0000
0247  2D70 0008          DATA 8
0248  2D72 130F          JEQ  EIO           DISK ERROR
0249  2D74 06A0          BL   @CKERR
      2D76 2E0A
0250  2D78 C060          MOV  @HEADER,R1
      2D7A 2BBE
0251  2D7C 1330          JEQ  DONE
0252  2D7E A18A          A    R10,R6        CALCULATE NEW START
0253  2D80 61CA          S    R10,R7        CALCULATE NEW LENGTH
0254  2D82 A80A          A    R10,@LOAD     CALCULATE NEW LOAD ADDRESS
      2D84 2BD4
0255  2D86 B620          AB   @ONE,*R8      CHANGE FILE NAME
      2D88 2B30
0256  2D8A 10A5          JMP  MAIN
0257                *
0258  2D8C 0201   ESTOP  LI   R1,ERR2       BAD STOP ADDRESS
      2D8E 2B52
0259  2D90 1004          JMP  EIO10
0260              *START LI   R1,ERR1       BAD START ADDRESS
0261              *      JMP  EIO10
0262  2D92 06A0   EIO    BL   @CKERR
      2D94 2E0A
0263  2D96 0201   EIO05  LI   R1,ERR3       IO ERROR
      2D98 2B72
0264  2D9A 0200   EIO10  LI   R0,EADDR
      2D9C 0242
0265  2D9E 0202          LI   R2,32
      2DA0 002C
```

```
0266 2DA2 0420          BLWP @VMBW
     2DA4 205A
0267 2DA6 0460          B    @SAVEOS
     2DA8 2BE4
0268                 *
0269 2DAA            CASSIO
0270 2DAA 0201          LI   R1,PAB+13     POINT TO END OF "CS1"
     2DAC 0F6D
0271 2DAE C801          MOV  R1,@SCNAME
     2DB0 8356
0272 2DB2 0201          LI   R1,3
     2DB4 0003
0273 2DB6 C801          MOV  R1,@>8354     NAME IS 3 BYTES LONG
     2DB8 8354
0274 2DBA 0201          LI   R1,>800       OFFSET FOR SEARCH
     2DBC 0800
0275 2DBE D801          MOVB R1,@>8360
     2DC0 8360
0276 2DC2 0200          LI   R0,PABFN      MOVE DEVICE NAME TO FAC
     2DC4 0F8A
0277 2DC6 0201          LI   R1,>834A
     2DC8 834A
0278 2DCA 0202          LI   R2,3
     2DCC 0003
0279 2DCE 0420          BLWP @VMBR
     2DD0 2C96
0280                 *
0281 2DD2 04E0          CLR  @>83D0        CRULST, INITIALIZE SEARCH
     2DD4 83D0
0282 2DD6 D820          MOVB @>83D0,@>837C
     2DD8 83D0
     2DDA 837C
0283 2DDC 0420          BLWP @GPLLNK
     2DDE 0000
0284 2DE0 003D          DATA >3D
0285                 *
0286 2DE2 06A0          BL   @CKERR
     2DE4 2E0A
0287 2DE6 C060          MOV  @HEADER,R1    FINISHED?
     2DE8 2BBE
0288 2DEA 1306          JEQ  DONE          YES
0289 2DEC A18A          A    R10,R6        UPDATE START
0290 2DEE A80A          A    R10,@LOAD
     2DF0 2BD4
0291 2DF2 61CA          S    R10,R7        UPDATE # BYTES LEFT
0292 2DF4 0460          B    @MAIN
     2DF6 2CD6
0293                 *----------------------------------------------------
0294 2DF8            DONE
0295 2DF8 04C0          CLR  R0
0296 2DFA D800          MOVB R0,@>837C     CLEAR STATUS
     2DFC 837C
0297 2DFE 02E0          LWPI GPLWS
     2E00 0000
0298 2E02 0300          LIMI 2
     2E04 0002
0299 2E06 06A0          BL   @>70          RETURN TO GPL
     2E08 0070
0300                 *
0301 2E0A            CKERR
```

```
0302 2E0A 0200        LI   R0,PAB+1      STATUS BYTE
     2E0C 0F81
0303 2E0E 0420        BLWP @VSBR         READ 1 BYTE FROM VDP
     2E10 0000
0304 2E12 09D1        SRL  R1,13         RIGHT JUSTIFY ERROR CODE
0305 2E14 C041        MOV  R1,R1
0306 2E16 1306        JEQ  CKRT          ERROR=0
0307 2E18 06C1        SWPB R1
0308 2E1A 0221        AI   R1,>3000      CONVERT ERROR TO ASCII
     2E1C 3000
0309 2E1E D801        MOVB R1,@ERR3+11   PUT IN ERROR MESSAGE
     2E20 2B70
0310 2E22 10B9        JMP  E1005
0311 2E24 0458  CKRT  RT
0312                  END
NO ERRORS.     NO WARNINGS
```

```
$          2E26          0042  0045  0052  0059  0061  0077  0135
BACKSP     2C52   0130   0114
BLANK      256E   0028   0167
CASSIO     2DAA   0269   0242
CKERR      2E0A   0301   0249  0262  0266
CRT        2E24   0311   0306
CPAB       2B84   0067   0206
CS         2B2E   0055   0241
CURSOR     2BCC   0080   0094
DISKIO     2D64   0243
DONE       2DF8   0294   0108  0251  0268
DSRLNK  R  2D6E   0013   0245
EADDR      0242   0045   0264
EIGHT      2C5C   0135   0128
EIO        2D92   0262   0248
EIO05      2D96   0263   0310
EIO10      2D9A   0264   0139  0259
ENAME      2C64   0138   0166  0176
ENTER      2C7E   0146   0116
ERR2       2B52   0063   0256
ERR3       2B72   0064   0263  0309
ERR4       2B92   0065   0138
ERRCOD     2B82   0066
ESTOP      2D8C   0256   0187
EXIT       2C14   0108   0118  0120
FADDR      02E2   0054   0100  0131  0149
FILE       2B32   0060   0150  0164  0171  0219  0241
FILEN      2B31   0059   0178  0212
FILEND     2B52   0061   0165
FSIZE      2BBA   0066   0204  0229
GPLLNK  R  2D0E   0012   0283
GPLWS   R  2E00   0014   0297
GRMRA   R         0014
GRMRD   R         0014
GRMWA   R         0014
HEADER     2B9E   0072   0224  0227  0229  0230  0232  0250  0287
INITS      2BC4   0075   0076  0097  0101  0142  0147
KEYCOD     8375   0024   0112
KSCAN   R  2C0A   0013   0104
LOAD       2BD4   0081   0181  0230  0254  0290
MAIN       2CD6   0198   0256  0292
MAIN10     2CE2   0202   0200
MAIN20     2D34   0228   0226
MENLEN     0300   0052   0091  0216
MENU       282E   0026   0042  0045  0052  0090  0215
OFFS       2BC8   0077   0147
ON         2BCA   0079   0097  0101
ONE        2B30   0056   0255
PAB        0F80   0069   0205  0270  0302
PABFL      0F89   0070   0211  0244
PABFN      0F8A   0071   0276
PADDR      02C2   0053
POS        2BC5   0076   0128  0136
R0         0000          0089  0093  0123  0141  0149  0205  0211  0214  0218
                         0231  0236  0264  0276  0295  0296  0302
R1         0001          0090  0094  0111  0112  0113  0115  0117  0119  0121
                         0134  0136  0138  0142  0150  0206  0212  0215  0219
                         0232  0238  0244  0246  0250  0258  0263  0270  0271
                         0272  0273  0274  0275  0277  0287  0304  0305  0306
                         0307  0308  0309
```

```
R10      000A          0202  0225  0237  0252  0253  0254  0269  0290  0291
R2       0002          0091  0095  0105  0143  0151  0198  0201  0202  0203
                       0204  0207  0210  0216  0220  0221  0233  0237  0265
                       0278
R3       0003          0100  0123  0125  0126  0131  0133  0164  0165  0167
                       0169  0172  0173  0174  0175  0177  0178
R4       0004          0171  0172  0173  0210  0220
R6       0006          0180  0181  0182  0186  0198  0252  0269
R7       0007          0185  0186  0198  0199  0225  0253  0291
R8       0008          0169  0170  0255
SAB      0300    0073  0141
SADDR    0202    0042  0218
SAV20    28FA    0099  0127
SAV30    2C04    0102  0107  0122  0129  0132  0137
SAV40    2C18    0110  0106
SAV45    2C9C    0165  0166
SAVE   D 2B06    0068  0015
SAVEOS   2BE4    0093  0267
SCNAME   8356    0022  0245  0271
SFIRST R 2CCC    0016  0182
SLAST  R 2CD0    0016  0185
SLOAD  R 2CC4    0016  0180
STATUS   837C    0023  0105
VMBR   R 2DD0    0016  0152  0279
VMBW   R 20A4    0012  0092  0096  0144  0208  0213  0217  0222  0234  0239
                       0266
VSBR   R 2E10    0012  0303
VSBW   R 2C40    0012  0124
WSPRIT   2C6C    0140  0103  0148
```