```
                1    TITLE 'ALCS'
                2   * * * * * * * * * * * * * * * * * * * * *
                3   *                                       *
                4   *  ASSEMBLY LANGUAGE SUPPORT FOR 99/4   *
                5   *                                       *
                6   *  LOAD, INIT, PEEK : JDH  08/21/80     *
                7   *  LINK, CHARPAT                        *
                8   * * * * * * * * * * * * * * * * * * * * *
                9   *
               10   * FORMAT FOR LOAD:
               11   *    CALL LOAD open load-directive (comma load-directive)*
               12   *              close
               13   *       load-directive = file-name / address ( comma data )*
               14   *                      ( null      file-name )
               15   *       file-name      = string-expression
               16   *       address        = numeric-expression
               17   *       data           = numeric-expression
               18   *
               19   *    FILE TYPE = FIXED 80, DISPLAY , SEQUENTIAL FILE
               20   *
               21   * FUNCTION:
               22   *    LOADS ASSEMBLY LANGUAGE CODE INTO EXPANSION RAM
               23   *    ADDRESSES: 2000H - 3FFFH, RELOCATING
               24   *    RELOCATABLE CODE INTO AVAILABLE MEMORY, ABSOLUTE CODE
               25   *    IS LOADED
               26   *    INTO ITS ABSOLUTE ADDRESS.  ENTRY POINTS ARE DEFINED BY
               27   *    'DEF' STATEMENTS, AND ARE LOADED INTO HIGH END OF ERAM
               28   *
               29   *    RELOCATABLE OR ABSOLUTE CODE MAY BE STORED ON A FILE IN
               30   *    9900 OBJECT CODE FORMAT.
               31   *       VALID TAGS = 0,5,6,7,8,9,A,B,C,F,:
               32   *             TAGS 1,2,I,M ARE IGNORED
               33   *    THE SYMT OPTION IS NOT SUPPORTED.
               34   *
               35   *    ABSOLUTE CODE MAY BE LOADED DIRECTLY FROM THE PROGRAM
               36   *    BY SPECIFYING AN ADDRESS INSTEAD OF A FILE NAME,
               37   *    FOLLOWED BY THE DATA TO BE LOADED (WHICH IS PUT IN THE
               38   *    RANGE 0 -> 255
               39   *    THE RANGE OF THE ADDRESS OR DATA IS LIMITED TO
               40   *    +32767 DOWNTO -32768
               41   *    MULTIPLE DIRECT LOADS CAN BE IN THE SAME LOAD COMMAND
               42   *    PROVIDED THEY ARE SEPARATED BY EITHER A FILENAME
               43   *    OR A NULL STRING.
               44   *
               45   *    MVUP WAS USED TO TRANSFER DATA FROM CPU RAM TO ERAM
               46   *    SINCE IT WAS NOT KNOWN AT FIRST THAT THE MOVE
               47   *    INSTRUCTION COULD TRANSFER FROM CPU RAM TO ERAM
               48   *     (PROVIDED THAT >8300 IS SUBTRACTED FROM THE ADDRESSES)
               49   *
               50   * L I N K A G E   A N D   H E A D E R
               51   *
               52            GROM   6
               53            ORG    0
C000 AA        54            DATA   >AA
C001 000000    55            DATA   0,0,0
```

```
0004 000000     56          DATA   0,0,0,0,0,0
0007 000000
000A 0000       57          DATA   #0000
000C 000000     58          DATA   0,0,0,0
000F 00
0010 C01904     59  LINK1   DATA   #LINK2,4,:LINK:,#LINK
0013 4C494E
0016 4BC32A
0019 C02204     60  LINK2   DATA   #LINK3,4,:LOAD:,#LOAD
001C 4C4F41
001F 44C040
0022 C02804     61  LINK3   DATA   #LINK4,4,:INIT:,#INIT
0025 494E49
0028 54C2BA
002B C03404     62  LINK4   DATA   #LINK5,4,:PEEK:,#PEEK
002E 504545
0031 4BC2CA
0034 000007     63  LINK5   DATA   #0,7,:CHARPAT:,#GETCHR
0037 434841
003A 525C41
003D 54C439
                64  *
                65  *
                66  *              V A R I A B L E S
                67  *
0084            68  RAMTOP  EQU >84          =0 IF ERAM NOT PRESENT
0010            69  TAG     EQU >10          TAG FIELD
0011            70  FIELD   EQU >11          VALUE AFTER TAG FIELD, 4 BYTES.
                71  *                        (MUST FOLLOW TAG !!)
0015            72  INDEX   EQU >15          BYTE INDEX FOR COMPUTING CHECK
005E            73  SYMBOL  EQU >5E          LABEL OR PROGRAM ID - 8 BYTES
0002            74  CHKSUM  EQU >02          CHECK SUM WORD
0004            75  PC      EQU >04          ADDRESS IN ERAM TO LOAD NEXT VALU
0006            76  OFFSET  EQU >06          OFFSET OF RELOCATABLE PROGRAMS
                77  *                        LOADED INTO ERAM
0008            78  FRESTA  EQU >08          START OF FREE MEM IN ERAM
                79  *                        THE END OF THE RELOCATABLE PROGR
                80  *                        (START OF NEXT PROGRAM) IS STORE
                81  *                        IN FRESTA ONCE A "0" TAG IS FOUN
000A            82  FREEND  EQU >0A          END OF FREE MEM IN ERAM - POINTS
                83  * 1ST CHARACTER OF LAST ENTRY INTO ROUTINE NAME TABLE.
                84  * MUST FOLLOW FRESTA !!!!!!!
000E            85  BUFPNT  EQU >0E          I/O BUFFER POINTER
000C            86  BYTE    EQU >0C          STRING LENGTH FOR GETSTR - WORD
001C            87  SREF    EQU >1C          LOCATION OF ALLOCATED STRING FOR
0042            88  CHAT    EQU >42          CURRENT PROGRAM CHARACTER
004A            89  FAC     EQU >4A          FLOATING POINT ACCUMULATOR
004C            90  FAC2    EQU FAC+2
004E            91  FAC4    EQU FAC+4
0050            92  FAC6    EQU FAC+6
006E            93  VSPTR   EQU >6E
0022            94  ERRCOD  EQU >22
002C            95  PGMPTR  EQU >2C          PROGRAM POINTER
                96  ***   EQUATES FOR: MVUP : TRANSFER DATA FROM CPU RAM TO E
0000            97  VARO    EQU >00          DESTINATION ADDRESS
```

```
0016      98  VAR9   EQU >16            SOURCE ADDRESS
005C      99  ARG    EQU >5C            BYTE COUNT - WORD
         100  ***   EQUATES FOR ERR$$ ROUTINE
0040     101  FREPTR EQU >40            FREE POINTER
0004     102  PABPTR EQU >04            DUMMY PAB POINTER
         103  *
         104  *     E X T E R N A L   G P L   R O U T I N E S
         105  *
6A78     106  CHKEND EQU >6A78          CHECK END OF STATEMENT
0012     107  RPL    EQU >0012          RETURN TO CALLER
6A82     108  WARN$$ EQU >6A82          Warning routine call address
6A84     109  ERR$$  EQU >6A84          ERROR ROUTINE CALL ADDRESS
0010     110  DSR    EQU >0010          DEVICE SERVICE ROUTINE
A01C     111  LNKRTN EQU >A01C
         112  *
         113  *     M A C H I N E   L A N G U A G E   X M L S
         114  *
008B     115  GVWITE EQU >8B
0020     116  ALSUP  EQU >20            XML TO USER ASSEMBLY ROUTINE
00E0     117  CIF    EQU >80            CONVERT FAC TO INTEGER
007A     118  SYM    EQU >7A            GET SYMBOL NAME
007B     119  SMB    EQU >7B            GET SYMBOL POINTER
007C     120  ASSGNV EQU >7C            ASSIGN FAC TO VARIABLE
         121
0071     122  GETSTR EQU >71            ALLOACATE BASIC STRING SPACE
0089     123  MVUP   EQU >89            MOVE UP, CPU RAM -> CPU RAM
0074     124  PARSE  EQU >74            PARSE INPUT STRING
0012     125  CFI    EQU >12            CONVERT FAC TO INTEGER
0077     126  VPUSH  EQU >77            VARIABLE STACK PUSH OPERATION
0078     127  VPOP   EQU >78            VARIABLE STACK POP OPERATION
0079     128  PGMCH  EQU >79            GET NEXT PROGRAM CHARACTER
         129  * * * * * * * * * * * * * * * *
         130  *      C O N S T A N T S    *
         131  * * * * * * * * * * * * * * * *
C600     132  ALCEND EQU >C600          END OF THIS ROUTINE
         133  * START OF SUPPORT IN GROM
0600     134  SUPLEN EQU >600           LENGTH OF SUPPORT ROUTINE
0000     135  DEBUG  EQU >0             IF DEBUG ON THEN EQU >C000 ELSE >0
0065     136  STRING EQU >65            STRING ID # FOR FAC
8300     137  CPUOFF EQU >8300          CPU RAM OFFSET
2002     138  FSLOC  EQU >2002+DEBUG    FREE START LOCATION IN ERAM,
         139  *                          FREE END MUST FOLLOW IT.
2006     140  INITF  EQU >2006+DEBUG    INIT FLAG ADDRESS. INIT HAS BEEN
         141  *                         CALLED IF ERAM(INITF)=AA55 HEX
         142  ** FREE END INITIALIZED TO 4000, (FFF8 FOR DEBUGGER)
         143  ** FREE START IS INITIALIZED TO THE
         144  ** FIRST USEABLE MEMORY LOACTION FOR ASSEMBLY LANGUAGE CODE
         145
         146  *
         147  *     B A S I C   K E Y W O R D   V A L U E S
         148  *
0083     149  COMMA$ EQU >B3             ","
0086     150  RPAR$  EQU >B6             ")"
0087     151  LPAR$  EQU >B7             "("
         152  *
```

```
                153  *     B A S I C   P A B   O F F S E T S
                154  *
0001            155  FLG     EQU  1          FLAG BYTE ENTRY
0002            156  BUF     EQU  2          BUFFER ENTRY
0004            157  LEN     EQU  4          RECORD LENGTH ENTRY
0005            158  CHRCNT  EQU  5          CHARACTER COUNT
0006            159  RNM     EQU  6          RECORD NUMBER
0008            160  SCR     EQU  8          SCREEN OFFSET ENTRY
0009            161  NLEN    EQU  9          NAME LENGTH
000A            162  PABLEN  EQU  10         ACTUAL PAB LENGTH
                163
```

```
                    165  *
                    166  *          L O A D  - L D P 1 -  L D P 4 - L D P 5
                    167  *
                    168  LOAD
                    169  ** CHKSUM IS ALSO USED AS A FLAG TO TEST IF A FILE HAS
                    170  ** BEEN OPENED (SO THAT IT GETS CLOSED)
                    171  ** IT IS INITIALIZED TO 0001 AND WILL BE CHANGED TO
                    172  ** SOME OTHER VALUE IF A FILE IS USED
C040 BF0200         173    DST  0001,@CHKSUM     {INITIALIZE FILE FLAG}
C043 01
C044 06C1EB         174    CALL CHKIN            {CHECK IF INIT HAS BEEN CALLED}
C047 D642B7         175    $IF @CHAT .NE. LPAR$ GOTO ERRSY1  {SYNTAX ERROR IF NO "("}
C04A 453A
C04C 0F79           176    XML  PGMCH            {SKIP OVER "("}
                    177  * MAIN PARSE LOOP *
                    178  * CHECK FOR FILE-NAME OR ADDRESS
                    179  LDP1
C04E 0F74           180    XML   PARSE
C050 B6             181    DATA  RPAR$ -         {PARSE UPTO ")" OR ","}
C051 D64C65         182    $IF @FAC+2 .EQ. STRING GOTO LDP2    * {PROCESS FILE NAME}
C054 6090
                    183  * OTHERWISE IT IS AN ADDRESS
                    184  * CONVERT ADDRESS TO INTEGER, SAVE IN @PC
C056 0F12           185    XML   CFI             {CONVERT FAC TO INTEGER}
C058 D65403         186    $IF @FAC+10 .EQ. 03 GOTO ERRNO1      {CHECK FOR OVERFLOW}
C05B 6533
C05D BD044A         187    DST   @FAC,@PC        {SAVE IN ERAM LOACTION POINTER}
                    188
```

```
                    190  * CHECK FOR "," - IF THERE THEN DATA SHOULD FOLLOW
                    191  *                 ELSE END OF LOAD STATEMENT - GOTO LDR
                    192  LDP4
C060 D642B3         193    $IF @CHAT .NE. COMMA$ GOTO LDP5
C063 40B5
                    194  * DATA FOLLOWS OR A STRING IF NO MORE DATA
C065 CF79           195       XML   PGMCH          {SKIP ","}
C067 CF74           196       XML   PARSE  {GET DATA VALUE OR STRING IF END OF DA
C069 B6             197       DATA  RPAR$          {PARSE UPTO ")" OR ","}
C06A D64C65         198       $IF @FAC+2 .EQ. STRING GOTO LDP2  {NO MORE DATA}
C06D 6090
                    199  * FAC CONTAINS A NUMERIC
C06F 0F12           200            XML   CFI                          {FAC -> INTEGER}
C071 D65403         201            $IF @FAC+10 .EQ. 03 GOTO ERRNO1 {CHECK FOR OVERFL
C074 6533
                    202
                    203  * MOVE CPU RAM -> ERAM  (MOVE LOW BYTE)
C076 BF1683         204            DST   CPUOFF+FAC+1,@VAR9  {@FAC+1 -> (@PC) IN ERA
C079 4B
C07A 8D0004         205            DST   @PC,@VARO                    {DESTINATION}
C07D BF5000         206            DST   0001,@ARG                    {MOVE 1 BYTE}
C080 01
C081 0F89           207            XML   MVUP                       {MOVE CPU RAM -> ERA
                    208
C083 9104           209            DINC  @PC                          {INC ERAM ADDRESS}
C085 05C060         210            $GOTO LDP4                      {CONTINUE WITH NEXT BYT
                    211
                    212  * CHECK FOR ")": IF THERE RETURN ELSE SYNTAX ERROR
                    213  LDP5
C088 D642B6         214       $IF @CHAT .EQ. RPAR$ GOTO LDRET   {RETURN}
C08B 61E1
C08D 05C53A         215       $GOTO ERRSY1                       {SYNTAX ERROR}
```

```
                    217  *
                    218  *       L D P 2
                    219  *
                    220  * PROCESS FILE NAME
                    221  LDP2
0090 EE3161         222     $IF @FAC+7 .EQ. 0 GOTO LDNE2   {CHECK FOR NULL STRING}
0093 32
0094 06C225         223     CALL OPENIT           {OPEN FILE, FILE NAME IN FAC}
                    224  *
                    225  ****************************************
                    226  *       LOAD DATA INTO ERAM            *
                    227  ****************************************
                    228  *
                    229  * LOAD FRESTA, FREEND FROM ERAM
0097 BF1620         230          DST  FSLOC,@VAR9                 'SOURCE
009A 02
009B BF0083         231          DST  FRESTA+CPUOFF,@VAR0         'DESTINATION
009E 08
009F BF5C00         232          DST  0004,@ARG                   '# BYTES TO MOVE
00A2 04
00A3 0F89           233          XML  MVUP                        'LOAD
                    234  * INITIALIZE PC,OFFSET INCASE OF NO "0" TAG
00A5 BD0408         235          DST  @FRESTA,@PC
00A8 BD0608         236          DST  @FRESTA,@OFFSET  ' BASE ADDRESS FOR LOAD MODULE
                    237  * READ IN ONE RECORD, EVALUATE THE TAG FIELD
                    238  *
                    239  *   L D R D    - L D T G
                    240  *
00AB BF0200         241  LDRD   DST  0000,@CHKSUM                 'CLEAR CHECK SUM
00AE 00
00AF 06C263         242          CALL READIT                      'READ IN A RECORD
00B2 350005         243  LDTG   MOVE 5 FROM RAM(@BUFPNT) TO @TAG   'GET TAG & FIELD
00B5 10B00E
00B8 06C200         244          CALL LDIPCS              ' ADD 5 TO BUFPNT, ADD ASCII
00BB 05             245          DATA 05            ' VALUE OF CHARS. READ TO CHECKSUM
                    246  * CONVERT @FIELD TO NUMERIC (FROM ASCII HEX VALUE)
                    247  * STORE RESLUT: HIGH BYTE -> FIELD , LOW BYTE -> FIELD + 1
                    248  *  CONVERT HIGH BYTE FIRST: @FIELD & @FIELD+1
                    249  *  STORE RESULT IN FIELD
00BC A61130         250          SUB  >30,@FIELD                  ' 30 = "0"
00BF CE1109         251          $IF @FIELD .GT. 09 THEN  'SUBTRACT ASCII DIFFERENCE
00C2 40C7
                    252  *                                          BETWEEN "9" AND "A"
00C4 A61107         253              SUB 07,@FIELD
                    254          $END
00C7 E21104         255          SLL  @FIELD,04                   ' FIELD = FIELD * 32
00CA A61230         256          SUB  >30,@FIELD+1
00CD CE1209         257          $IF @FIELD+1 .GT. 09 THEN
00D0 40D5
00D2 A61207         258              SUB 07,@FIELD+1
                    259          $END
00D5 A01112         260          ADD  @FIELD+1,@FIELD             ' ADD TO HIGH BYTE
                    261  *    NOW CONVERT LOW BYTE: @FIELD+2 & @FIELD+3
                    262  *    STORE RESULT IN LOW BYTE OF FIELD -> FIELD + 1
00D8 A61330         263          SUB  >30,@FIELD+2
```

```
00E8 0E1309    264          $IF @FIELD+2 .GT. 09 THEN
00EA 40E0
00EC A61307    265              SUB 07,@FIELD+2
               266          $END
00EE 801213    267          ST   @FIELD+2,@FIELD+1  ' STORE IN LOW BYTE OF RESU
00F1 E21204    268          SLL  @FIELD+1,04            ' FIELD+1 = FIELD+1 * 1
00F4 A61430    269          SUB  >30,@FIELD+3
00F6 0E1409    270          $IF @FIELD+3 .GT. 09 THEN
00F8 40F4
00FA A61407    271              SUB 07,@FIELD+3
               272          $END
00FD A01214    273          ADD  @FIELD+3,@FIELD+1      ' ADD TO LOW BYTE
```

```
                    275  *
                    276  * BRANCH TO EVALUATION PROCEDURE FOR TAG
C1F7 A61030         277          SUB  >30,@TAG                    '>30 = "0"
C1FA D21000         278          $IF @TAG .LT. 00 GOTO ERRUC1     'IF TAG<"0" ILLEG. CHA
C1FD 45CF
C1FF CE1005         279          $IF @TAG .LE. >0A THEN           'TAGS "0" -> ":"
C202 611C
C204 8A10           280           CASE @TAG
C206 4148           281           BR   TAG0                  '"0" RELOCATABLE LENGTH
C208 40B2           282           BR   LDTG                  'IGNORE "1" TAG
C20A 40B2           283           BR   LDTG                  'IGNORE "2" TAG
C20C 45CF           284           BR   ERRUC1                'NO EXTERNAL REF. "3"
C20E 45CF           285           BR   ERRUC1                'NO EXTERNAL REF. "4"
C210 415D           286           BR   TAG5                  ' "5" RELOCATABLE ENTRY - DEF
C212 4160           287           BR   TAG6                  ' "6" ABSOLUTE ENTRY -    DEF
C214 4192           288           BR   TAG7                  ' "7" CHECK SUM
C216 40B2           289           BR   LDTG                  ' "8" IGNORE CHECK SUM
C218 419F           290           BR   TAG9                  ' "9" ABSOLUTE LOAD ADDRESS
C21A 41BA           291           BR   LDDNE                 ' ":" END OF FILE
                    292          $END
C21C A61011         293          SUB >11,@TAG                  ' SUBTRACT OFFSET SO
                    294  *                                       THAT "A" IS = 0
C21F D21000         295          $IF @TAG .LT. 0  GOTO ERRUC1 <";" -> "@" ILLEGAL CHA
C222 45CF
                    296
                    297  * SKIP OVER 'I' TAG - 8 CHAR. PROGRAM ID THAT FOLLOWS
C124 D61008         298          $IF @TAG .EQ.  8 GOTO LDTG2
C127 6156
                    299
                    300  * SKIP OVER 'M' TAG - 10 CHARACTERS THAT FOLLOW
C129 D6100C         301          $IF @TAG .NE. 12 GOTO LDTG3
C12C 4135
C12E 06C200         302          CALL LDIPCS
C131 0A             303          DATA 10
C132 05C0B2         304          $GOTO  LDTG
                    305  LDTG3
                    306
C135 CE1005         307          $IF @TAG .GT. 05 GOTO ERRUC1 <TAGS "G" -> ARE ILLEGA
C138 65CF
C13A 8A10           308            CASE @TAG
C13C 419C           309            BR TAGA              ' "A" RELOCATABLE PROG. ADDRESS
C13E 41A8           310            BR TAGB              ' "B" ABSOLUTE VALUE
C140 41A5           311            BR TAGC              ' "C" RELATIVE ADDRESS
C142 45CF           312            BR ERRUC1            ' "D" ERROR
C144 45CF           313            BR ERRUC1            ' "E" ERROR - UNDEFINED
C146 40A8           314            BR LDRD              ' "F" END OF RECORD
                    315  *
                    316  *       T A G 0  ->  T A G B
                    317  *
                    318
                    319  * EVALUTATE TAG FIELDS
                    320
C148 BD0608         321  TAG0    DST  @FRESTA,@OFFSET      ' NEW BASE ADDRESS
C14B BD0408         322          DST  @FRESTA,@PC          ' NEW PC
C14E A10811         323          DADD @FIELD,@FRESTA       ' ADD LENGTH TO FIND END OF
```

```
                    324  *                                  RELOCATABLE PROGRAM
                    325  *                                  WHICH IS START OF NEXT PROG
                    326  * MAKE SURE WE WON'T RUN INTO ROUTINE NAME TABLE NOW, SO
                    327  *   DON'T HAVE TO CHECK EVERY TIME WE LOAD A VALUE INTO E
                    328  *   ROUTINE TABLE MUST MAKE SURE IT DOESN'T RUN INTO
                    329  *   RELOCATABLE ASSEMBLY LANGUAGE CODE THOUGH
C151 C9080A         330          $IF @FRESTA .DHE. @FREEND GOTO ERRMF1  ;OUT OF MEM
C154 655D
                    331  * SKIP OVER PROGRAM ID - 8 BYTES
                    332
                    333  * ENTRY POINT TO SKIP OVER PROGRAM ID FOR 'I' TAG
                    334  LDTG2
                    335
C156 060200         336          CALL LDIPCS
C159 08             337          DATA 08                  ' INC BUFPNT, COMPUTE CHECKS
C15A 05C0B2         338          $GOTO LDTG
                    339
C15D A11106         340  TAG5    DADD @OFFSET,@FIELD              ' ADD STARTING OFFSE
                    341
                    342  * TAG6 IS AN ABSOLUTE ADDRESS SO DO NOT NEED TO ADD OFFSE
                    343
C160 350006         344  TAG6    MOVE 6 FROM RAM(@BUFPNT) TO @SYMBOL  'GET SMBOL NA
C163 5E800E
C166 060200         345          CALL  LDIPCS             ' INC BUPNT, COMPUTE CHECKS
C169 06             346          DATA  06                       ' WE READ 6 CHARS.
                    347
                    348  * ADD SYMBOL AND ITS ADDRESS - STORED IN FIELD - TO THE
                    349  *   ROUTINE ENTRY TABLE.  IT IS PUT AT THE END OF THE TAB
                    350  *   (THE END OF THE TABLE IS TOWARDS THE LOW END OF MEM
                    351  *   SINCE THE TABLE IS SEARCHED FROM THE END FIRST, IF
                    352  *   THERE ARE ANY DUPLICATE LABELS THE LAST ONE ENTERED W
                    353  *   HAVE PRECEDENCE OVER THE EARLY ONE(S).
                    354  *
C16A 970A           355          DDECT @FREEND                  ' SET TO ADDRESS FIE
                    356  * LOAD ADDRESS (STORED IN FIELD IN CPU RAM) INTO ROUTINE
                    357  *   NAME TABLE WHICH IS IN EXPANSION RAM.
C16C BF1683         358          DST  FIELD+CPUOFF,@VAR9          ' SOURCE
C16F 11
C170 BD000A         359          DST  @FREEND,@VAR0               ' DESTINATION
C173 BF5C00         360          DST  0002,@ARG                   ' # BYTES TO MOVE
C176 02
C177 0F89           361          XML  MVUP                        ' CPUR RAM -> ERAM
                    362  * LOAD SYMBOL INTO ROUTINE NAME TABLE
C179 A70A00         363          DSUB 06,@FREEND                  ' SET TO SYMBOL FIEL
C17C 06
C17D BF1683         364          DST  SYMBOL+CPUOFF,@VAR9         ' SOURCE
C180 5E
C181 BD000A         365          DST  @FREEND,@VAR0               ' DEST.
C184 BF5C00         366          DST  0006,@ARG                   ' MOVE 6 BYTES
C187 06
C188 0F89           367          XML  MVUP                        ' CPU RAM -> ERAM
                    368  * CHECK TO SEE IF WE'VE RUN INTO ASSEMBLY LANGUAGE CODE
C18A C9080A         369          $IF @FRESTA .DHE. @FREEND GOTO ERRMF1  ' OUT OF MEM
C18D 655D
C18F 05C0B2         370          $GOTO LDTG                       ' IF NOT THEN CONT
```

```
                    371
                    372  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                    373  * ROUTINE NAME TABLE ENTRY                                              *
                    374  *                                                                       *
                    375  *                          0   1   2   3   4   5   6   7              *
                    376  *                        -------------------------------------------  *
                    377  *          FREEND  ->    ! S ! Y ! M ! B ! O ! L ! ADDRESS !          *
                    378  *        (AFTER ENTRY)   -------------------------------------------  *
                    379  *          FREEND  ->    !   !   !   !   !   !   !           !          *
                    380  *       (BEFORE ENTRY)   -------------------------------------------  *
                    381  *                                                                       *
                    382  *    FREEND IS INITIALIZED TO 4000H BY INIT, ADDRESS IS AT              *
                    383  *      A HIGHER MEMORY LOCATION THEN SYMBOL.                            *
                    384  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                    385
C192 B311           386  TAG7    DNEG    @FIELD                      ' CHECKSUM IS 1'S COMPL
C194 D50211         387          $IF @CHKSUM .DNE. @FIELD GOTO ERRDE1 'CHECK SUM ERROR
C197 4588
C199 05C0B2         388          $GOTO   LDTG
                    389
C19C A11106         390  TAGA    DADD    @OFFSET,@FIELD              ' PC:= OFFSET + FIELD
                    391
                    392  * TAG 9 IS AN ABSOLUTE ADDRESS SO NO NEED TO ADD OFFSET
                    393
C19F BD0411         394  TAG9    DST     @FIELD,@PC
C1A2 05C0B2         395          $GOTO   LDTG
                    396
C1A5 A11106         397  TAGC    DADD    @OFFSET,@FIELD
                    398
                    399  * TAG B IS AN ABSOLUTE ENTRY SO NO NEED TO ADD OFFSET
                    400
                    401  * RELOCATABLE CODE IS CHECKED TO SEE IF IT WILL RUN INTO
                    402  *   ROUTINE NAME TABLE WHEN "0" TAG FOUND, SO THERE
                    403  *   IS NO NEED TO CHECK NOW.
                    404  *   ABSOLUTE CODE CAN GO ANYWHERE.
                    405
                    406  * LOAD FIELD INTO EXPANSION RAM USING MVUP ROUTINE
C1A8 BD0004         407  TAGB    DST     @PC,@VAR0                   ' DESTINATION
C1AB BF1683         408          DST     FIELD+CPUOFF,@VAR9          ' SOURCE
C1AE 11
C1AF BF5C00         409          DST     0002,@ARG                   ' MOVE 2 BYTES
C1B2 02
C1B3 0F89           410          XML     MVUP                        ' CPU RAM -> ERAM
C1B5 9504           411          DINCT   @PC                         ' WE LOADED 2 BYTES
C1B7 05C0B2         412          $GOTO   LDTG
```

```
                    414  ***********************************
                    415  * END OF LOAD FOR CURRENT FILE    *
                    416  ***********************************
                    417  *
                    418  * FRESTA & FREEND ARE STORED IN CPU RAM ( >8308 )
                    419  * WHILE LOADING A FILE INTO EXPANSION RAM.
                    420  * SO IF THE VALUES OF FRESTA OR FREEND ARE TO BE
                    421  * CHANGED THEN WORD LOCATIONS >8308 AND >830A MUST BE
                    422  * CHANGED AND NOT EXPANSION RAM.
                    423  *
                    424  *       L D D N E    - L D N E 2
                    425  *
                    426  *   DONE WITH LOAD
                    427  * PUT FRESTA, FREEND BACK INTO EXPANSION RAM
                    428  * IF FRESTA IS ODD THEN MAKE IT EVEN
                    429  *   SO THAT THE NEXT PROGRAM STARTS ON AN EVEN BOUNDARY
C1BA DA0901         430  LDDNE  $IF .BITO @FRESTA+1 .EQ. 1 THEN     {LOW BYTE ODD
C1BD 61C1
C1BF 91C8           431             DINC @FRESTA          {FORCE TO NEXT EVEN BOUNDAR
                    432          $END IF
C1C1 BF1E83         433          DST    FRESTA+CPUOFF,@VAR9
C1C4 08
C1C5 BF0020         434          DST    FSLOC,@VARO                ' DEST.
C1C8 02
C1C9 BF5000         435          DST    0004,@ARG                  ' LOAD 4 BYTES
C1CC 04
C1CD 0F89           436          XML    MVUP                       ' CPU RAM -> ERAM
                    437
                    438
C1CF 06C277         439          CALL   CLSIT                      ' CLOSE FILE
                    440  * CHECK FOR END OF LOAD COMMAND ")"
                    441  LDNE2
C1D2 D642B6         442    $IF @CHAT .EQ. RPAR$ GOTO LDRET       {CHECK FOR ")"}
C1D5 61E1
C1D7 D642B3         443    $IF @CHAT .NE. COMMA$ GOTO ERRSY1 {SYNTAX ERROR}
C1DA 453A
C1DC 0F79           444    XML   PGMCH      {SKIP COMMA}
C1DE 05C04E         445    $GOTO LDP1       {CONTINUE IN MAIN LOOP}
                    446  *
                    447  *          L D R E T  -  L D R E T 2
                    448  *
                    449  *  RETURN TO CALLING ROUTINE
                    450  LDRET
C1E1 0F79           451    XML   PGMCH             {SKIP OVER ")"}
                    452  * ENTRY POINT FOR INIT
                    453  LDRET2
C1E3 066A78         454    CALL  CHKEND            {CHECK FOR END OF STATEMENT}
C1E6 453A           455    BR    ERRSY1            {IF NOT END THEN SYNTAX ERROR}
C1E8 060012         456    CALL  RPL               {RETURN TO CALLER}
                    457
                    458  PAGE
                    459  *
                    460  *          C H K I N
                    461  *
                    462  * CHECK FOR INIT-FLAG =AA55 HEX
```

```
                    463  CHKIN
                    464  ** MOVE ERAM(INITF) -> CPU RAM(FAC)
C1EB BF0083         465    DST   FAC+CPUOFF,@VAR0          (DESTINATION)
C1EE 4A
C1EF 3F1680         466    DST   INITF,@VAR9               (SOURCE)
C1F2 06
C1F3 BF5000         467    DST   0002,@ARG                 (2 BYTES)
C1F6 02
C1F7 0F89           468    XML   MVUP                      (MOVE IT)
                    469
C1F9 D74AAA         470    $IF @FAC .DNE. >AA55 GOTO ERRSYN  (SYNTAX ERROR)
C1FC 55453D
                    471  * NO FILES HAVE BEEN OPENED SO IF THERE IS A SYNTAX
                    472  * ERROR GOTO ERRSYN !
C1FF 00             473    RTN
                    474
```

```
                    476  *
                    477  *   F I L E   R O U T I N E S
                    478  *
                    479
                    480  ************************************************************
                    481  * INCREMENT BUFFER POINTER by value after call statement
                    482  *  ADD VALUES READ TO CHECKSUM unless the first characte
                    483  *  is a "7" = 37 hex, then add only "7" char. to checksu
                    484  *  (other value is the checksum)
                    485  *
                    486  *              L D I P C S
                    487  LDIPCS
C200 8815           488    FETCH @INDEX                {INDEX= NO. OF BYTES READ }
C202 D6B00E         489    $IF RAM(@BUFPNT) .EQ. :7: THEN
C205 374213
C208 A30200         490       DADD  >37,@CHKSUM      {ADD VALUE OF "7" TO CHECKSUM}
C20B 37
C20C A30E00         491       DADD   05,@BUFPNT      {1 FOR "7", 4 FOR CHECSKUM   }
C20F 05
C210 050224         492    $ELSE
                    493       $REPEAT
C213 9C48B0         494          ST    RAM(@BUFPNT),@FAC+1  {CONVERT TO 2-BYTE VAL
C216 0E
C217 864A           495          CLR   @FAC                 ----------
C219 A1024A         496          DADD  @FAC,@CHKSUM         {ADD CHAR. TO CHECKSUM
C21C 910E           497          DINC  @BUFPNT
C21E 9215           498          DEC   @INDEX               {DO IT INDEX # OF TIME
C220 8E1542         499       $UNTIL @INDEX .EQ. 0
C223 13
                    500    $END
C224 00             501    RTN
                    502
                    503  *
                    504  *       O P E N I T
                    505  *
     C225           506  OPENIT EQU $
C225 BD0C50         507    DST  @FAC+6,@BYTE          Store actual spec. length
C228 A30C00         508    DADD PABLEN+80,@BYTE  Add in the PAB len. and buffer le
C22B 5A
C22C 0F77           509    XML  VPUSH              Push possible temp. string
C22E 0F71           510    XML  GETSTR               and try to allocate space
C230 0F78           511    XML  VPOP               Restore original string data
                    512  *
                    513  *          THE FOLLOWING VARIABLES CONTAIN IMPORTANT INFO
                    514  *
                    515  *          FAC+4,5   Start address of original device spe
                    516  *          FAC+6,7   Length of original device specificat
                    517  *          SREF      Location of PAB in VDP memory
                    518  *          BYTE      Length of entire PAB, including spec
                    519  *
C232 3450E0         520    MOVE @FAC+6 FROM RAM(@FAC+4) TO RAM(PABLEN(SREF))
C235 0A1CB0
C238 4E
C239 86B01C         521    CLR  RAM(@SREF)                Clear the entire PAB
C23C 350009         522    MOVE PABLEN-1 FROM RAM(@SREF) TO RAM(1(SREF))
```

```
C23F EC011C
C242 B01C
C244 BCE009    523     ST    @FAC+7,RAM(NLEN(SREF)) Copy spec. length
C247 1C51
C249 BEE008    524     ST    >60,RAM(SCR(SREF))     Screen offset
C24C 1C60
C24E BEE001    525     ST    &00000100,RAM(FLG(SREF)) Disp, fixed, seq, input
C251 1C04
C253 A1501C    526     DADD  @SREF,@FAC+6       Calculate the address of the
C256 A35000    527     DADD  PABLEN,@FAC+6          buffer
C259 0A
C25A BDE002    528     DST   @FAC+6,RAM(BUF(SREF))  Store buffer address in PAB
C25D 1C50
C25F 06C27B    529     CALL  CALDSR
C262 00        530     RTN
               531  ********************************************************************
     C263      532  READIT EQU $
C263 BD0EE0    533     DST   RAM(BUF(SREF)),@BUFPNT      Init buffer pointer
C266 021C
C268 BEB01C    534     ST    >02,RAM(@SREF)     Prepare to read
C26B 02
C26C BCE005    535     ST    RAM(LEN(SREF)),RAM(CHRCNT(SREF))
C26F 1CE004
C272 1C
C273 06C27B    536     CALL  CALDSR
C276 00        537     RTN
               538  *
               539  *        C L S I T
               540  *
     C277      541  CLSIT EQU $
C277 BEB01C    542     ST    >01,RAM(@SREF)     Prepare to close
C27A 01
               543  *
               544  *        C A L D S R   -   D S K E R R
               545  *
     C27B      546  CALDSR EQU $
C27B BD561C    547     DST   @SREF,@FAC+12      Compute start address of spec.
C27E A35600    548     DADD  NLEN,@FAC+12       Ready to call DSR routine
C281 09
C282 060010    549     CALL  DSR                Call DSR through program link
C285 08        550     DATA  8                  Type = DSR (8)
C286 6290      551     BS    DSKERR             Couldn't find the DSR
C288 DAE001    552     CLOG  >E0,RAM(FLG(SREF))  Set condition bit if no errors
C28B 1CE0
C28D 4290      553     BR    DSKERR
C28F 00        554     RTN
               555
     C290      556  DSKERR EQU $
C290 BD0440    557     DST    @FREPTR,@PABPTR          {SET UP DUMMY PAB}
C293 A70400    558     DSUB   06,@PABPTR               {MAKE IT STANDARD SIZE}
C296 06
C297 BDE004    559     DST    RAM(@SREF),RAM(4(PABPTR))  {STORE ERROR CODE}
C29A 04B01C
C29D 06C2A4    560     CALL   CLSNOE                   {CLOSE FILE}
C2A0 066A84    561     CALL   ERR$$                    {ISSUE I/O ERROR}
```

```
C2A3 24          562     DATA  36
                 563
                 564   *
                 565   *        C L S N O E
                 566   *
                 567   * TRY TO CLOSE THE CURRENT FILE
                 568   * IGNORE ANY ERRORS FROM THE CLOSING OF THE FILE.
                 569   * SINCE THE PAB IS NOT IN THE NORMAL PAB LIST
                 570   * THEN WE HAVE TO CLOSE THE FILE IN THE LOAD ROUTINE.
                 571   * ERR$$ WILL CLOSE THE REST OF THE FILES.
                 572
                 573   CLSNOE
                 574
                 575   ** CLOSE IT ONLY IF IT HAS BEEN OPENED
C2A4 D70200      576      $IF @CHKSUM .DNE. >0001 THEN  {CHECK FILE FLAG}
C2A7 0162B9
C2AA BEB01C      577       ST   01,RAM(@SREF)     {STORE CLOSE FILE CODE}
C2AD 01
C2AE BD561C      578       DST  @SREF,@FAC+12     {COMPUTE START ADDR. OF SPEC.
C2B1 A05600      579       DADD NLEN,@FAC+12      {READY TO CALL DSR}
C2B4 09
C2B5 06A010      580       CALL DSR               {CALL DSR THROUGH PROGRAM LIN
C2B8 08          581       DATA 8                 {"8" IS TYPE FOR DSR}
                 582      $END
C2B9 00          583      RTN
                 584   ********************************************************
```

```
                    586  * * * * * * * * * * * * * * * * * * * * *
                    587  *   *     I N I T         JDH  09/02/80  *
                    588  * * * * * * * * * * * * * * * * * * * * *
                    589
                    590  INIT
                    591
                    592  * CHECK IF EXPANSION RAM PRESENT
                    593  * LOAD SUPPORT INTO EXPANSION RAM FROM GROM
C2BA BE8064         594    $IF @RAMTOP .EQ. 0 GOTO ERRSYN  <IF NO ERAM, SYNTAX ERROR
C2BD 653D
                    595  ** LOAD AL HEADER, SUPPORT ROUTINES **
C2BF 310600         596    MOVE SUPLEN FROM ROM(ALCEND) TO @>2000+DEBUG->8300
C2C2 8F9D00
C2C5 C600
C2C7 05C1E3         597    $GOTO LDRET2
                    598  ********************************************************************
```

```
                    600  * * * * * * * * * * * * * * * * * * * * *
                    601  *                                         *
                    602  *   PEEK INSTRUCTION     JDH  09/04/80  *
                    603  *                                         *
                    604  * * * * * * * * * * * * * * * * * * * * *
                    605  *
                    606  * FORMAT:
                    607  *  CALL PEEK open address (comma numeric-variable)* close
                    608  * FUNCTION:
                    609  *   RETURNS THE VALUE AT address IN ERAM INTO
                    610  *   numeric-variable.   IF MORE THAN ONE numeric-variable I
                    611  *   SPECIFIED THEN address IS INCREMENTED AND THE VALUE
                    612  *   IN ERAM AT THE NEW address IS ASSIGNED TO THE NEXT
                    613  *   VARIABLE AND SO ON.
                    614  *
                    615
                    616  PEEK
                    617
C2CA D642B7         618    $IF @CHAT .NE. LPAR$ GOTO ERRSYN    {CHAT = "(" ?}
C2CD 453D
C2CF 0F79           619    XML PGMCH                           {SKIP "("}
C2D1 0F74           620    XML PARSE       {GET VALUE OF address}
C2D3 B6             621    DATA RPAR$
C2D4 D64C65         622    $IF @FAC+2 .EQ. STRING GOTO ERRSNM {address MUST BE NUME
C2D7 654D
C2D9 0F12           623    XML  CFI     {CONVERT FAC TO INTEGER}
C2DB D65403         624    $IF @FAC+10 .EQ. 03 GOTO ERRNO    {OVERFLOW?}
C2DE 6536
C2E0 BD044A         625    DST  @FAC,@PC           {SAVE PEEK ADDRESS}
C2E3 D642B3         626    $IF @CHAT .NE. COMMA$ GOTO ERRSYN  {CHAT = "," ?}
C2E6 453D
                    627
                    628  PEEK2
C2E8 0F79           629    XML  PGMCH      {SKIP ","}
C2EA CA4280         630    $IF @CHAT .HE. >80 GOTO ERRSYN {DON'T ALLOW TOKEN}
C2ED 653D
C2EF 0F7A           631    XML  SYM        {GET SYMBOL NAME}
C2F1 0F7B           632    XML  SMB        {GET VALUE POINTER}
C2F3 0F77           633    XML  VPUSH      {SAVE FAC ON STACK FOR ASSGNV}
C2F5 8E4C45         634    $IF @FAC+2 .NE. 0 GOTO ERRSNM  {MUST BE NUMERIC}
C2F8 4D
C2F9 864A           635    CLR  @FAC
C2FB 350007         636    MOVE 07 FROM @FAC TO @FAC+1   {CLEAR FAC}
C2FE 4B4A
                    637  ** GET PEEK VALUE FROM ERAM INTO @FAC+1
C300 BD1604         638    DST  @PC,@VAR9              {SOURCE}
C303 BF0083         639    DST  CPUOFF+FAC+1,@VAR0     {DESTINATION}
C306 4B
C307 BF5C00         640    DST  0001,@ARG              {MOVE 1 BYTE}
C30A 01
C30B 0F89           641    XML  MVUP
                    642  **
C30D 0F80           643    XML  CIF    {CONVERT FAC TO FLOATING POINT VALUE}
C30F 0F7C           644    XML  ASSGNV    {ASSIGN TO NUMERIC-VARIABLE}
C311 D642B3         645    $IF @CHAT .NE. COMMA$ GOTO PEEK5
```

```
0314 431D
0316 9104      646    DINC @PC    <INC POINTER TO NEXT ERAM ADDRESS>
0318 05C2E8    647    $GOTO PEEK2
               648
               649  PEEK5
               650  * CHECK FOR ")" AND END OF STATEMENT
               651  * IF ALL O.K. THEN RETURN TO CALLER
               652  * GETCHR ALSO RETURNS TO HERE
               653
031B D642B6    654    $IF @CHAT .NE. RPAR$ GOTO ERRSYN
031E 453D
0320 0F79      655    XML  PGMCH       <SKIP ")">
0322 066A78    656    CALL CHKEND
0325 453D      657    BR   ERRSYN
0327 060012    658    CALL RPL          <RETURN TO CALLER>
               659
```

```
                  661  ***********************************************
                  662  *                                             *
                  663  *  ASSEMBLY LANGUAGE SUPPORT FOR 99/4         *
                  664  *                                             *
                  665  *  LINK INSTRUCTION : 5E September, 1980      *
                  666  *                                             *
                  667  ***********************************************
                  668  *
                  669  *
                  670  *
                  671  *  FORMAT:
                  672  *
                  673  *  CALL LINK("file-name",parameter1,parameter2,.....)
                  674  *
                  675  *  LINK ROUTINE READS THE FILE NAME SPECIFIED BY THE USER
                  676  *  AND SAVE THE ADDRESS OF THE NAME FOR LATER USE.
                  677  *  THE FILE WILL BE SEARCHED IN UTILITY CODE LATER ON.
                  678  *
                  679  *  PARAMETERS ARE PASSED EITHER BY REFERENCE OR BY VALUE.
                  680  *  NUMERIC OR STRING VARIABLES AND NUMERIC OR STRING ARRAYS
                  681  *  ARE PASSED BY REFERENCE AND ALL OTHERS INCLUDING A USER
                  682  *  DEFINED FUNCTION ARE PASSED BY VALUE.
                  683  *
                  684  *  PARAMETER INFORMATION IS STORED IN CPU 00 THROUGH >10
                  685  *  THAT GIVES A PARAMETER TYPE CODE OF EACH PARAMETER.
                  686  *    CODE 0 ....   Numeric expression
                  687  *    CODE 1 ....   String expression
                  688  *    CODE 2 ....   Numeric variable
                  689  *    CODE 3 ....   String variable
                  690  *    CODE 4 ....   Numeric array
                  691  *    CODE 5 ....   String array
                  692  *
                  693  *  IF A PARAMETER IS PASSED AS A NUMERIC EXPRESSION ITS
                  694  *  ACTUAL VALUE GETS PUSHDED INTO THE VALUE STACK.
                  695  *  IN CASE OF A STRING EXPRESSION, ITS VALUE STACK CONTAIN
                  696  *  AN ID(>65), POINTER TO THE VALUE SPACE AND ITS LENGTH.
                  697  *  IF A PARAMETER GETS PASSED AS A REFERENCE THE PRODUCT
                  698  *  OF XML SYM AND XML SMB IN THE @FAC AREA GETS PUSHED INT
                  699  *  STACK.
                  700  *
                  701  *  AFTER AN ASSEMBLY LANGUAGE SUBPROGRAM IS EXECUTED LINK
                  702  *  ROUTINE WILL POP THE STACK TO GET RID OF PARAMETER
                  703  *  INFORMATION.  CONTROL WILL BE TRANSFERED TO THE BASIC
                  704  *  MAIN PROGRAM AFTERWARDS.
                  705  *
                  706  ****  DATA  AREA  ****
                  707  *  CPU RAM FREE SPACE USED IN THIS ROUTINE
                  708  *
                  709  * FLAG BITS STORED IN LOCATIONS 00 TO >0F
0010              710  OLDS   EQU   >10
0012              711  COUNT  EQU   >12
0014              712  STORE  EQU   >14
0016              713  TEMP   EQU   >16
                  714
```

```
                    716  ****************************************
                    717  *  CALL LINK program                   *
                    718  ****************************************
                    719  *
                    720  LINK
C32A C6C1E8         721     CALL CHKIN              Check if INIT has been called
C32D BD1C6E         722     DST @VSPTR,@OLDS        Save VSPTR for later use.
C330 D642B7         723     $IF @CHAT .NE. LPAR$ GOTO ERRSYN Check for "(".
C333 453D
C335 0F79           724     XML PGMCH               Advance program pointer.
C337 0F74           725     XML PARSE               Get the routine name.
C339 B6             726     DATA RPAR$              Read up to ")".
C33A D64C65         727     $IF @FAC2 .NE. >65 GOTO ERRBA   Should be a string.
C33D 45A4
C33F BF5065         728     $IF @FAC6 .DEQ. 0 GOTO ERRBA   Don't accept null string.
C342 A4
C343 C65106         729     $IF @FAC6+1 .H. 6 GOTO ERRBA   Should be less than 6 cha
C346 65A4
C348 0F77           730     XML VPUSH               Push to make it semi-permanent.
C34A B612           731     CLR @COUNT              Initialize parameter counter.
                    732  *
                    733  ****************************************
                    734  *  PARAMETERS get evaluated here       *
                    735  ****************************************
                    736  PAR01
C34C D642B6         737     $IF @CHAT .EQ. RPAR$ GOTO EXE01  No arg. So execute it.
C34F 640A
C351 D642B3         738     $IF @CHAT .NE. COMMA$ GOTO ERRSYN Should have a comma.
C354 453D
                    739  *
C356 BD222C         740     DST @PGMPTR,@ERRCOD  Save text pointer.
C359 0F79           741     XML PGMCH               Get the character
C35B CA4280         742     $IF @CHAT .HE. >80 GOTO VAL01 Must be an expression.
C35E 63CC
                    743  *  $IF @CHAT .EQ. LPAR$ Pass by expression.
C360 06C431         744     CALL CLRFAC             Clear FAC entry for SYM.
C363 0F7A           745     XML SYM                 Read in the symbol table info.
                    746  *
                    747  *  After XML SYM @FAC area contains a pointer to
                    748  *  symbol table.
                    749  *
                    750  *  Below statement checks if it is a UDF.
C365 DAB04A         751     $IF .BIT6 RAM(@FAC) .EQ. 1 GOTO VAL01 Pass by value.
C368 4043CC
C36B D642B3         752     $IF @CHAT .EQ. COMMA$ GOTO REF01   Pass by reference.
C36E 63EA
C370 D642B6         753     $IF @CHAT .EQ. RPAR$ GOTO REF01    Pass by reference.
C373 63EA
C375 D642B7         754     $IF @CHAT .EQ. LPAR$ GOTO ARRAY    An array.
C378 6381
C37A CA4280         755     $IF @CHAT .HE. >80 GOTO VAL01      Pass by value.
C37D 63CC
C37F 453D           756     BR  ERRSYN
                    757  *
                    758  ****************************************
```

```
                    759  *  ARRAY case gets checked here          *
                    760  ***************************************
                    761  *  should look like A(,,) etc.
                    762  *  Stack entry for an arry will look like
                    763  *  ---------------------------------------------------------
                    764  *  | Pointer to      |DOC  |      | Pointer to  |           |
                    765  *  | symbol table    | or  |      | dim info in |           |
                    766  *  | entry           |D65  |      | real v.s.   |           |
                    767  *  FAC-------------FAC2-------FAC4---------FAC6---------
                    768  *
                    769  *
                    770  ARRAY
C381 0F79           771      XML PGMCH                  Get the next character.
C383 D642B6         772      $IF @CHAT .EQ. RPAR$ GOTO ARRAY2    Pass by reference.
C386 6394
C388 D642B3         773      $IF @CHAT .EQ. COMMA$ GOTO ARRAY    More array informati
C38B 6381
C38D 932C           774      DDEC @PGMPTR                       Adjust the pointer.
C38F BE42B7         775      ST  LPAR$,@CHAT
C392 42EA           776      BR  REF01                          Pass by reference.
                    777  *
                    778  ARRAY2
                    779  *
                    780  *  In array cases the symbol table address gets stored
                    781  *  at FAC area, and the pointer to the value space
                    782  *  (dimension info.) goes into FAC4
                    783  *
C394 0F79           784      XML PGMCH                  Advance the program pointer.
C396 DAB04A         785      $IF .BIT7 RAM(@FAC) .NE. 1 THEN    Test string bit.
C399 8043A2
C39C BE9012         786       ST  4,*COUNT              Numeric array.
C39F 04
C3A0 43A6           787      $SELSE
C3A2 BE9012         788       ST  5,*COUNT              String array case.
C3A5 05
                    789      $END IF
                    790  *  Check if array is being shared
                    791  *  If it is then go back through the linkage to get
                    792  *  the actuals symbol table pointer.
                    793  *  Put the pointer to the value space (dimension info.)
                    794  *  into FAC4.
C3A6 DAB04A         795      $IF .BIT5 RAM(@FAC) .EQ. 1 THEN          Shared array?
C3A9 2063C3
C3AC 350002         796          MOVE 2 FROM RAM(6(FAC)) TO @FAC4     If so, get point
C3AF 4EE006
C3B2 4A
C3B3 DAEFFF         797          $IF .BIT5 RAM(-6(FAC4)) .EQ. 1 THEN  Shared also?
C3B6 FA4E20
C3B9 63C1
C3BB 350002         798              MOVE 2 FROM RAM(@FAC4) TO @FAC4  Get pointer
C3BE 4EB04E
                    799          $END IF
C3C1 43CA           800      $SELSE
C3C3 BD4E4A         801          DST   @FAC,@FAC4       Array is not shared.
C3C6 A34E00         802          DADD  6,@FAC4          Point to value space
```

```
C3CF 05
                    803      $END IF
                    804
C3CA 43FF           805      BR  PUSH
                    806  *
                    807  ****************************************
                    808  *  VALUE                               *
                    809  *  Passing the parameter by value      *
                    810  ****************************************
                    811  VAL01
C3CC 9D2C22         812      DST @ERRCOD,@PGMPTR  Restore program pointer.
C3CF CF79           813      XML PGMCH            Skip the first character.
C3D1 BD160C         814      DST @BYTE,@TEMP      In case of passing a string.
C3D4 CF74           815      XML PARSE            Parsing up to comma.
C3D6 86             816      DATA RPAR$
C3D7 BD0C16         817      DST @TEMP,@BYTE      Restore the value in >0C area.
                    818  *
                    819  *  After parsing @FAC area contains
                    820  *  its actual numeric value in a numeric case, and the
                    821  *  following information in a string case.
                    822  *  ---------------------------------------------------
                    823  *  | >0010          | >65 |      | Pointer to | Length of |
                    824  *  | or value pointer |    |      | string     | string    |
                    825  *  | address        |     |      |            |           |
                    826  *  FAC--------------FAC2-------FAC4--------FAC6---------
                    827  *
                    828  *
C3DA CE4C63         829      $IF @FAC2 .GT. >63 THEN      If more than 99 then
C3DD 43E5
C3DF BE9012         830        ST 1,*COUNT        Store flag for string expression.
C3E2 01
C3E3 43E8           831      $SELSE
C3E5 869012         832        CLR *COUNT         Otherwise it is a numeric express
                    833      $END IF
C3E8 43FF           834      BR  PUSH             Push into stack.
                    835  *
                    836  ****************************************
                    837  *  REFERENCE                           *
                    838  *  Passing the parameter by reference  *
                    839  ****************************************
                    840  *  Variables, array element and whole array passing.
                    841  *
                    842  *
                    843  *  After SMB @FAC Entry should look like
                    844  *  ---------------------------------------------------
                    845  *  | Pointer to     | >00 |      | Pointer to |           |
                    846  *  | symbol table   |     |      | value space|           |
                    847  *  | entry          |     |      |            |           |
                    848  *  FAC--------------FAC2-------FAC4--------FAC6---------
                    849  *  for a numeric case, and
                    850  *  ---------------------------------------------------
                    851  *  | Pointer to     | >65 |      | Pointer to | String    |
                    852  *  | value space    |     |      | string     |  length   |
                    853  *  | entry          |     |      |            |           |
                    854  *  FAC--------------FAC2-------FAC4--------FAC6---------
```

```
                    855  *  for a string case.
                    856  *
                    857  REF01
C3EA 0F78           858      XML SMB                    Get the location.
C3EC 3A4288         859      $IF @CHAT .HE. >88 GOTO VAL01 Pass array expression.
C3EF 6300
C3F1 8E4043         860      $IF @FAC2 .EQ. 00 THEN
C3F4 F6
C3F5 3E9012         861        ST 2,*COUNT              Must be a numeric variable.
C3F8 02
C3F9 43FF           862      $ELSE
C3FB 8E9012         863        ST 3,*COUNT              Must be a string variable.
C3FE 03
                    864      $END IF
                    865  *
                    866  ******************************************
                    867  *  PUSH routine                           *
                    868  *  Pushes @FAC entry into a value stack*
                    869  ******************************************
                    870  PUSH
C3FF 9012           871      INC @COUNT
C401 0E1210         872      $IF @COUNT .GT. 16 GOTO ERRBA  Too many parameters.
C404 65A4
                    873  *
C406 0F77           874      XML VPUSH
C408 434C           875      BR  PAR01                  Get the next argument.
                    876  *
                    877  ******************************************
                    878  *  EXECUTE routine                       *
                    879  *  Restore file name info and transfer *
                    880  *  control over to ALC                   *
                    881  ******************************************
                    882  EXE01
C40A 8E4A20         883      ST   >20,@FAC              Store blank in the FAC area.
C40D 350005         884      MOVE 5 FROM @FAC TO @FAC+1
C410 484A
C412 350004         885      MOVE 4 FROM RAM(12(OLDS)) TO @STORE  Get file name info.
C415 14E00C
C418 10
C419 34164A         886      MOVE @STORE+2 FROM RAM(@STORE) TO @FAC   Move to FAC.
C41C B014
C41E 8722           887      DCLR @ERRCOD               Clear program pointer for error code
C420 0F20           888      XML ALSUP                  Go to CPU at >2000 to execute.
C422 64D3           889      BS  ERROR                  Error found.
                    890  *                              If no error, start checking stack.
                    891  *
                    892  ******************************************
                    893  *  RETURN to the BASIC main program.    *
                    894  ******************************************
                    895  NOERR
C424 C56E10         896      $WHILE @VSPTR .DH. @OLDS
C427 442E
C429 0F78           897        XML VPOP                 Pop the stack.
C42B 05C424         898      $END WHILE                 Keep popping till no stack for
                    899  *
```

```
C42E 05A01C    900      B  LNKRTN                    Check ")" and end of statement.
               901  *
               902  ****************************************
               903  *  SUBROUTINES used in this file.      *
               904  ****************************************
               905  CLRFAC.
C431 B64A      906      CLR @FAC
C433 350007    907      MOVE 7 FROM @FAC TO @FAC+1
C436 484A
C438 00        908      RTN
               909
```

```
                    911
                    912  * * * * * * * * * * * * * * * * * * * * * * * *
                    913  *                                               *
                    914  * CHARPAT ROUTINE                              *
                    915  *                                               *
                    916  * 99/4   - JDH   10/01/80                       *
                    917  * * * * * * * * * * * * * * * * * * * * * * * *
                    918  *
                    919  * FORMAT:
                    920  *  CALL CHARPAT open ( <numeric expression> comma
                    921  *                      <string expression> )* close
                    922  * FUNCTION:
                    923  *  RETURNS THE CHARACTER DEFINTION PATTERN FOR CHARCTER
                    924  *  NUMBER <numeric expression> INTO <string expression>.
                    925  *
                    926  *
                    927
                    928  * Temporaries used in the routne
        0010        929  TBLPTR EQU >10      Table pointer
        0012        930  STRPTR EQU >12      String pointer
                    931
                    932  *
                    933  *  G E T C H R   -  G E T C H R 2
                    934  *
                    935  GETCHR
   C439 D642B7      936    $IF @CHAT .NE. LPAR$ GOTO ERRSYN
   C43C 453D
                    937
                    938  GCHR2
   C43E 0F79        939    XML PGMCH
   C440 0F74        940    XML PARSE
   C442 B6          941    DATA RPAR$
   C443 D64C65      942    $IF @FAC+2 .EQ. STRING GOTO ERRSNM   Can't be a string
   C446 654D
   C448 0F12        943    XML CFI       Convert FAC to integer
   C44A D65403      944    $IF @FAC+10 .EQ. 3 GOTO ERRBA      (range 32 -> 143)
   C44D 65A4
   C44F D34A00      945    $IF @FAC .DLT. #32  GOTO ERRBA          "   "    "
   C452 2045A4
   C455 CF4A00      946    $IF @FAC .DGT. #143 GOTO ERRBA          "   "    "
   C458 8F65A4
   C45B E34A00      947    DSLL  @FAC,3          (8 bytes / entry  so multiply by 8 )
   C45E 03
   C45F BF1003      948    DST   >300,@TBLPTR    (Base of character table less 32*8 )
   C462 00
   C463 A1104A      949    DADD  @FAC,@TBLPTR     (Add in arg offset)
                    950
   C466 BF0C00      951    DST  16,@BYTE         Get a 16 byte string in string space
   C469 10
   C46A 0F71        952    XML  GETSTR
   C46C BD121C      953    DST  @SREF,@STRPTR    Save pointer to string
   C46F BE1508      954    ST   8,@INDEX         Loop counter
                    955
                    956    $REPEAT
   C472 BCB012      957         ST  RAM(@TBLPTR),RAM(@STRPTR)
```

```
C475 B010
C477 E6B012   958          SRL RAM(@STRPTR),4          Get rid of low nibble    1
C47A C4
C47B A2B012   959          ADD >30,RAM(@STRPTR)        Add ASCII "0"            1
C47E 30
C47F CEB012   960          $IF RAM(@STRPTR) .LE. >39 GOTO GCHR3   >39=ASCII("9"):
C482 394489
C485 A2B012   961              ADD 7,RAM(@STRPTR)      Value "A" -> "F"         1
C488 07
              962  GCHR3
C489 9112     963          DINC @STRPTR                                         1
C48B BCB012   964          ST  RAM(@TBLPTR),RAM(@STRPTR)                        1
C48E B010
C490 B2B012   965          AND OF,RAM(@STRPTR)                                  1
C493 OF
C494 A2B012   966          ADD >30,RAM(@STRPTR)        Add ASCII "0"            1
C497 30
C498 CEB012   967          $IF RAM(@STRPTR) .LE. >39 GOTO GCHR4                 1
C49B 3944A2
C49E A2B012   968              ADD 7,RAM(@STRPTR)      Value "A" -> "F"
C4A1 07
              969  GCHR4
C4A2 9110     970          DINC @TBLPTR                                         1
C4A4 9112     971          DINC @STRPTR                                         1
C4A6 9215     972          DEC  @INDEX                                          1
C4A8 8E1544   973        $UNTIL @INDEX .EQ. 0
C4AB 72
              974
              975  * NOW ASSIGN THE STRING JUST CREATED TO THE STRING
              976  * VARIABLE FOLLOWING
              977
C4AC OF79     978        XML  PGMCH          Skip comma
C4AE CA4280   979        $IF @CHAT .HE. >80 GOTO ERRSYN Do not allow token.
C4B1 653D
C4B3 OF7A     980        XML  SYM            Get symbol table info for next arg.
C4B5 OF7B     981        XML  SMB
C4B7 OF77     982        XML  VPUSH          Save on stack for ASSGNV
C4B9 D64C65   983        $IF @FAC+2 .NE. STRING GOTO ERRSNM   Must be a string var
C4BC 454D
C4BE BF4A00   984        DST  >001C,@FAC     Temp string so use SREF as address
C4C1 1C
C4C2 BD4E1C   985        DST  @SREF,@FAC+4   Pointer to string
C4C5 BF5000   986        DST  16,@FAC+6      String length
C4C8 10
C4C9 OF7C     987        XML  ASSGNV         Assign to string variable
C4CB D642B3   988        $IF @CHAT .EQ. COMMA$ GOTO GCHR2
C4CE 643E
C4D0 05C31B   989        $GOTO PEEK5         Check for ")", and end of statement
              990  *                        return to caller
              991
```

```
                    993  * * * * * * * * * * * * * * * * * * * *
                    994  * ERROR BRANCH TABLE FOR LINK          *
                    995  * * * * * * * * * * * * * * * * * * * *
                    996
                    997
                    998  ERROR
C4D3 8A22           999     CASE @ERRCOD
C4D5 4424          1000     BR   NOERR
C4D7 4424          1001     BR   NOERR
C4D9 4536          1002     BR   ERRNO          2   Numeric Overflow
C4DB 453D          1003     BR   ERRSYN         3   Syntax Error
C4DD 4541          1004     BR   ERRIBS         4   Illegal after subprogram
C4DF 4545          1005     BR   ERRNQS         5   Unmatched quotes
C4E1 4549          1006     BR   ERRNTL         6   Name too long
C4E3 454D          1007     BR   ERRSNM         7   String-num mismatch
C4E5 4551          1008     BR   ERROBE         8   Option base error
C4E7 4555          1009     BR   ERRMUV         9   Improperly used name
C4E9 4559          1010     BR   ERRIM         10   Image error
C4EB 4560          1011     BR   ERRMEM        11   Memory full
C4ED 4564          1012     BR   ERRSO         12   Stack overflow
C4EF 4568          1013     BR   ERRNWF        13   Next without for
C4F1 456C          1014     BR   ERRFNN        14   For next nesting
C4F3 4570          1015     BR   ERRSNS        15   Must be in subprogram
C4F5 4574          1016     BR   ERRRSC        16   Recursive subprogram call
C4F7 4578          1017     BR   ERRMS         17   Missing subend
C4F9 457C          1018     BR   ERRRWG        18   Return without gosub
C4FB 4580          1019     BR   ERRST         19   String truncated
C4FD 4584          1020     BR   ERRBS         20   Bad subscript
C4FF 4588          1021     BR   ERRSSL        21   Speech string too long
C501 458C          1022     BR   ERRLNF        22   Line not found
C503 4590          1023     BR   ERRBLN        23   Bad line number
C505 4594          1024     BR   ERRLTL        24   Line too long
C507 4598          1025     BR   ERRCC         25   Can't continue
C509 459C          1026     BR   ERRCIP        26   Command illegal in program
C50B 45A0          1027     BR   ERROLP        27   Only legal in a program
C50D 45A4          1028     BR   ERRBA         28   Bad argument
C50F 45A8          1029     BR   ERRNPP        29   No program present
C511 45AC          1030     BR   ERRBV         30   Bad value
C513 45B0          1031     BR   ERRIAL        31   Incorrect argument list
C515 45B4          1032     BR   ERRINP        32   Input error
C517 45B8          1033     BR   ERRDAT        33   Data error
C519 45BF          1034     BR   ERRFE         34   File error
C51B 4424          1035     BR   NOERR
C51D 45C3          1036     BR   ERRIO         36   I/O error
C51F 45C7          1037     BR   ERRSNF        37   Subprogram not found
C521 4424          1038     BR   NOERR
C523 45CB          1039     BR   ERRPV         39   Protection violation
C525 45D2          1040     BR   ERRIVN        40   Unrecognized character
C527 45D6          1041     BR   WRNNO         41   Numeric overflow
C529 45DC          1042     BR   WRNST         42   String truncated
C52B 45E2          1043     BR   WRNNPP        43   No program present
C52D 45E8          1044     BR   WRNINP        44   Input error
C52F 45EE          1045     BR   WRNIO         45   I/O error
C531 45F4          1046     BR   WRNLNF        46   Line not found
                   1047  *
```

```
                1048  ****************************************
                1049  *  ERROR HANDLING SECTION              *
                1050  ****************************************
                1051  ****
C533 06C2A4     1052  ERRNO1 CALL CLSNOE        *ENTRY FOR LOAD
C536 066A84     1053  ERRNO  CALL ERR$$         *Numeric overflow
C539 02         1054         DATA 2
                1055  *
C53A 06C2A4     1056  ERRSY1 CALL CLSNOE        *ENTRY FOR LOAD
C53D 066A84     1057  ERRSYN CALL ERR$$         *Syntax error
C540 03         1058         DATA 3
                1059  *
C541 066A84     1060  ERRIBS CALL ERR$$         *Illegal after subprogram
C544 04         1061         DATA 4
                1062  *
C545 066A84     1063  ERRNQS CALL ERR$$         *Unmatched quotes
C548 05         1064         DATA 5
                1065  *
C549 066A84     1066  ERRNTL CALL ERR$$         *Name too long
C54C 06         1067         DATA 6
                1068  *
C54D 066A84     1069  ERRSNM CALL ERR$$         *String Number mismatch
C550 07         1070         DATA 7
                1071  *
C551 066A84     1072  ERROBE CALL ERR$$         *Option Base error
C554 08         1073         DATA 8
                1074  *
C555 066A84     1075  ERRMUV CALL ERR$$         *Improperly used name
C558 09         1076         DATA 9
                1077  *
C559 066A84     1078  ERRIM  CALL ERR$$         *Image error
C55C 0A         1079         DATA 10
                1080  *
C55D 06C2A4     1081  ERRMF1 CALL CLSNOE        *ENTRY FOR LOAD
C560 066A84     1082  ERRMEM CALL ERR$$         *Memory full
C563 0B         1083         DATA 11
                1084  *
C564 066A84     1085  ERRSO  CALL ERR$$         *Stack overflow
C567 0C         1086         DATA 12
                1087  *
C568 066A84     1088  ERRNWF CALL ERR$$         *Next without for
C56B 0D         1089         DATA 13
                1090  *
C56C 066A84     1091  ERRFNN CALL ERR$$         *For-next nesting
C56F 0E         1092         DATA 14
                1093  *
C570 066A84     1094  ERRSNS CALL ERR$$         *Must be in subprogram
C573 0F         1095         DATA 15
                1096  *
C574 066A84     1097  ERRRSC CALL ERR$$         *Recursive subprogram call
C577 10         1098         DATA 16
                1099
C578 066A84     1100  ERRMS  CALL ERR$$         *Missing subend
C57B 11         1101         DATA 17
                1102  *
```

```
C57C 066A84  1103  ERRRWG CALL ERR$$      *Return without gosub
C57F 12      1104  *    DATA 18
             1105  *
C580 066A84  1106  ERRST  CALL ERR$$      *String truncated
C583 13      1107        DATA 19
             1108  *
C584 066A84  1109  ERRBS  CALL ERR$$      *Bad subscript
C587 14      1110        DATA 20
             1111  *
C588 066A84  1112  ERRSSL CALL ERR$$      *Speech string too long
C58B 15      1113        DATA 21
             1114  *
C58C 066A84  1115  ERRLNF CALL ERR$$      *Line not found
C58F 16      1116        DATA 22
             1117  *
C590 066A84  1118  ERRBLN CALL ERR$$      *Bad line number
C593 17      1119        DATA 23
             1120  *
C594 066A84  1121  ERRLTL CALL ERR$$      *Line too long
C597 18      1122        DATA 24
             1123  *
C598 066A84  1124  ERRCC  CALL ERR$$      *Can't continue
C59B 19      1125        DATA 25
             1126  *
C59C 066A84  1127  ERRCIP CALL ERR$$      *Command illegal in program
C59F 1A      1128        DATA 26
             1129  *
C5A0 066A84  1130  ERROLP CALL ERR$$      *Only legal in a program
C5A3 1B      1131        DATA 27
             1132  *
C5A4 066A84  1133  ERRBA  CALL ERR$$      *Bad argument
C5A7 1C      1134        DATA 28
             1135  *
C5A8 066A84  1136  ERRNPP CALL ERR$$      *No program present
C5AB 1D      1137        DATA 29
             1138  *
C5AC 066A84  1139  ERRBV  CALL ERR$$      *Bad value
C5AF 1E      1140        DATA 30
             1141  *
C5B0 066A84  1142  ERRIAL CALL ERR$$      *Incorrect argument list
C5B3 1F      1143        DATA 31
             1144  *
C5B4 066A84  1145  ERRINP CALL ERR$$      *Input error
C5B7 20      1146        DATA 32
             1147  *
C5B8 06C2A4  1148  ERRDE1 CALL CLSNOE     *ENTRY FOR LOAD
C5BB 066A84  1149  ERRDAT CALL ERR$$      *Data error/check sum error
C5BE 21      1150        DATA 33
             1151  *
C5BF 066A84  1152  ERRFE  CALL ERR$$      *File error
C5C2 22      1153        DATA 34
             1154  *
C5C3 066A84  1155  ERRIO  CALL ERR$$      *I/O error
C5C6 24      1156        DATA 36
             1157  *
```

```
C5C7 066A84  1158 ERRSNF CALL ERR$$        *Subprogram not found
C5CA 25      1159        DATA 37
             1160 *
C5CB 066A84  1161 ERRPV  CALL ERR$$        *Protection violation
C5CE 27      1162        DATA 39
             1163 *
C5CF 06CDA4  1164 ERRUC1 CALL CLSNOE       *ENTRY FOR LOAD
C5D2 066A84  1165 ERRIVN CALL ERR$$        *Unrecognized character/illegal ta
C5D5 28      1166        DATA 40
             1167 *
C5D6 066A82  1168 WRNNO  CALL WARN$$       *Numeric overflow
C5D9 02      1169        DATA 2
C5DA 4424    1170        BR   NOERR
             1171 *
C5DC 066A82  1172 WRNST  CALL WARN$$       *String truncated
C5DF 13      1173        DATA 19
C5E0 4424    1174        BR   NOERR
             1175 *
C5E2 066A82  1176 WRNNPP CALL WARN$$       *No program present
C5E5 1D      1177        DATA 29
C5E6 4424    1178        BR   NOERR
             1179 *
C5E8 066A82  1180 WRNINP CALL WARN$$       *Input error
C5EB 20      1181        DATA 32
C5EC 4424    1182        BR   NOERR
             1183 *
C5EE 066A82  1184 WRNIO  CALL WARN$$       *I/O error
C5F1 23      1185        DATA 35
C5F2 4424    1186        BR   NOERR
             1187 *
C5F4 066A82  1188 WRNLNF CALL WARN$$       *Line not found
C5F7 26      1189        DATA 38
C5F8 4424    1190        BR   NOERR
             1191 * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
             1192 *                                                       *
             1193 *  S U P P O R T  R O U T I N E  C O D E  F O L L O W S *
             1194 *                                                       *
             1195 * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
             1196     END
```

ERRORS= 0

LENGTH= 1530 (>05FA)

186 SYMBOLS USED

SYMBOL VALUE DEF REFERENCE TABLE

	C14B	318									
ALCEND	C600	132	596								
ALSUP	0020	116	888								
ARG	005C	99	206	232	360	366	409	435	467	640	
ARRAY	C381	770	754	773							
ARRAY2	C394	778	772								
ASSGNV	007C	120	644	987							
BUF	0002	156	528	533							
BUFPNT	000E	85	243	344	489	491	494	497	533		
BYTE	000C	86	507	508	814	817	951				
CALDSR	C27B	546	529	536							
CB	0008	1									
CFI	0012	125	185	200	623	943					
CHAR	0004	1									
CHAT	0042	88	175	193	214	442	443	618	626	630	645 654
			723	737	738	742	752	753	754	755	772 773
			775	859	936	979	988				
CHKEND	5A78	106	454	656							
CHKIN	C1EB	463	174	721							
CHKSUM	0003	74	173	241	387	490	496	576			
CHRCNT	0005	158	535								
CIF	0080	117	643								
CLRFAC	C431	905	744								
CLSIT	C277	541	439								
CLSNDE	C2A4	573	560	1052	1056	1081	1148	1164			
COMMA$	00B3	149	193	443	626	645	738	752	773	988	
COUNT	0012	711	731	786	788	830	832	861	863	871	872
CPUOFF	8300	137	204	231	358	364	408	433	465	639	
DEBUG	0000	135	138	140	596						
DISPLA	0009	1									
DSKERR	C290	556	551	553							
DSR	0010	110	549	580							
ERR$$	6A84	109	561	1053	1057	1060	1063	1066	1069	1072	1075 1078
			1082	1085	1088	1091	1094	1097	1100	1103	1106 1109
			1112	1115	1118	1121	1124	1127	1130	1133	1136 1139
			1142	1145	1149	1152	1155	1158	1161	1165	
ERRBA	C5A4	1133	727	728	729	872	944	945	946	1028	
ERRBLN	C590	1118	1023								
ERRBS	C584	1109	1020								
ERRBV	C5AC	1139	1030								
ERRCC	C598	1124	1025								
ERRCIP	C59C	1127	1026								
ERRCOD	0022	94	740	812	887	999					
ERRDAT	C5BB	1149	1033								
ERRDE1	C5B8	1148	387								
ERRFE	C5BF	1152	1034								
ERRFNN	C56C	1091	1014								
ERRIAL	C5B0	1142	1031								
ERRIBS	C541	1060	1004								
ERRIM	C559	1078	1010								
ERRINP	C5B4	1145	1032								
ERRIO	C5C3	1155	1036								
ERRIVN	C5D2	1165	1040								
ERRLNF	C58C	1115	1022								
ERRLTL	C594	1121	1024								

SYMBOL	VALUE	DEF	REFERENCE TABLE
ERRMEM	C560	1082	1011
ERRMF1	C55D	1081	330 369
ERRMS	C578	1100	1017
ERRMUV	C555	1075	1009
ERRNO	C536	1053	624 1002
ERRNO1	C533	1052	186 201
ERRNPP	C548	1136	1029
ERRNQS	C545	1063	1005
ERRNTL	C549	1066	1006
ERRNWF	C568	1088	1013
ERROBE	C551	1072	1008
ERROLP	C5A0	1130	1027
ERROR	C4D3	998	889
ERRPV	C5C8	1161	1039
ERRRSC	C574	1097	1016
ERRRWG	C57C	1103	1018
ERRSNF	C5C7	1158	1037
ERRSUM	C54D	1069	622 634 942 983 1007
ERRSNS	C570	1094	1015
ERRSC	C564	1085	1012
ERRSSL	C588	1112	1021
ERRST	C580	1106	1019
ERRSY1	C53A	1056	175 215 443 455
ERRSYN	C53D	1057	470 594 618 626 630 654 657 723 738 756 936 979 1003
ERRUC1	C5CF	1164	278 284 285 295 307 312 313
EXEC1	C40A	882	737
FAC	004A	89	90 91 92 182 186 187 198 201 204 222 465 470 494 495 496 507 520 520 523 526 527 528 547 548 578 579 622 624 625 634 635 636 636 639 751 785 795 796 801 883 884 884 886 906 907 907 942 944 945 946 947 949 983 984 985 986
FAC2	004C	90	727 829 860
FAC4	004E	91	796 797 798 798 801 802
FAC6	0050	92	728 729
FIELD	0011	70	250 251 253 255 256 257 258 260 260 263 264 265 267 267 268 269 270 271 273 273 323 340 358 386 387 390 394 397 408
FIGURE	0002	1	
FLG	0001	155	525 552
FREEND	000A	82	330 355 359 363 365 369
FREPTR	0040	101	557
FRESTA	0008	78	231 235 236 321 322 323 330 369 430 431 433
FSLOC	2002	138	230 434
GCHR2	C43E	938	988
GCHR3	C489	962	960
GCHR4	C4A2	969	967
GETCHR	C439	935	63
GETSTR	0071	122	510 952
GVWITE	008B	115	
INDEX	0015	72	488 498 499 954 972 973
INIT	C2BA	590	61
INITF	2006	140	466

SYMBOL	VALUE	DEF	REFERENCE TABLE
LDDNE	C1BA	430	291
LDIPCS	C200	487	244 302 336 345
LDNE2	C1D2	441	222
LDP1	C04E	179	445
LDP2	C090	221	182 198
LDP4	C060	192	210
LDP5	C088	213	193
LDRD	C0AB	241	314
LDRET	C1E1	450	214 442
LDRET2	C1E3	453	597
LDTG	C0B2	243	282 283 289 304 338 370 388 395 412
LDTG2	C156	334	298
LDTG3	C135	305	301
LEN	0004	157	535
LINK	C32A	720	59
LINK1	C010	59	
LINK2	C019	60	59
LINK3	C022	61	60
LINK4	C02B	62	61
LINK5	C034	63	62
LNKRTN	A01C	111	900
LOAD	C040	168	60
LPAR$	0087	151	175 618 723 754 775 936
MVUP	0089	123	207 233 361 367 410 436 468 641
NLEN	0009	161	523 548 579
NOERR	C424	895	1000 1001 1035 1038 1170 1174 1178 1182 1186 1190
OFFSET	0006	76	236 321 340 390 397
OLDS	0010	710	722 885 896
OPENIT	C225	506	223
PABLEN	000A	162	508 520 522 527
PABPTR	0004	102	557 558 559
PAGE	C1EB	458	
PAR01	C34C	736	875
PARSE	0074	124	180 196 620 725 815 940
PC	0004	75	187 205 209 235 322 394 407 411 625 638 646
PEEK	C2CA	616	62
PEEK2	C2E8	628	647
PEEK5	C31B	649	645 989
PGMCH	0079	128	176 195 444 451 619 629 655 724 741 771 784 813 939 978
PGMPTR	002C	95	740 774 812
POP	0005	1	
PUSH	C3FF	870	805 834
RAM	0001	1	243 344 489 494 520 520 521 522 522 523 524 525 528 533 534 535 535 542 552 559 559 577 751 785 795 796 797 798 885 886 957 957 958 959 960 961 964 964 965 966 967 968
RAMTOP	0084	68	594
READIT	C263	532	242
REF01	C3EA	857	752 753 776
RNM	0006	159	
ROM	000A	1	596
RPAR$	00B6	150	181 197 214 442 621 654 726 737 753 772

SYMBOL	VALUE	DEF	REFERENCE TABLE
			816 941
APL	0012	107	456 658
SCR	0008	160	524
SMB	0079	119	632 858 981
SPRITE	0006	1	
SREF	001C	87	520 521 522 522 523 524 525 526 528 533
			534 535 535 542 547 552 559 577 578 953
			985
STORE	0014	712	885 886 886
STRING	0065	136	182 198 622 942 983
STRPTR	0012	930	953 957 958 959 960 961 963 964 965 966
			967 968 971
SUPLEN	0600	134	596
SYM	007A	118	631 745 980
SYMBOL	005E	73	344 364
TABLE	000B	1	
TAG	0010	69	243 277 278 279 280 293 295 298 301 307
			308
TAG3	C148	321	281
TAG5	C15D	340	286
TAG6	C160	344	287
TAG7	C192	386	288
TAG9	C19F	394	290
TAGA	C19C	390	309
TAGB	C1A8	407	310
TAGC	C1A5	397	311
TBLPTR	0010	929	948 949 957 964 970
TEMP	0016	713	814 817
TOP	0003	1	
VAL01	C3CC	811	742 751 755 859
VAR0	0000	97	205 231 359 365 407 434 465 639
VAR9	0016	98	204 230 358 364 408 433 466 638
VDP	000C	1	
VEL	0007	1	
VPOP	0078	127	511 897
VPUSH	0077	126	509 633 730 874 982
VSPTR	006E	93	722 896
WARN$$	6A82	108	1168 1172 1176 1180 1184 1188
WRNINP	C5E8	1180	1044
WRNIO	C5EE	1184	1045
WRNLNF	C5F4	1188	1046
WRNNO	C5D6	1168	1041
WRNNPP	C5E2	1176	1043
WRNST	C5DC	1172	1042
XPT	000E	1	
YPT	000D	1	