Please do not upload this copyright pdf document to any other website. Breach of copyright may result in a criminal conviction.

This Acrobat document was generated by me, Colin Hinson, from a document held by me. I requested permission to publish this from Texas Instruments (twice) but received no reply. It is presented here (for free) and this pdf version of the document is my copyright in much the same way as a photograph would be. If you believe the document to be under other copyright, please contact me.

The document should have been downloaded from my website https://blunham.com/Radar, or any mirror site named on that site. If you downloaded it from elsewhere, please let me know (particularly if you were charged for it). You can contact me via my Genuki email page: https://www.genuki.org.uk/big/eng/YKS/various?recipient=colin

You may not copy the file for onward transmission of the data nor attempt to make monetary gain by the use of these files. If you want someone else to have a copy of the file, point them at the website. (https://blunham.com/Radar). **Please do not point them at the file itself as it may move or the site may be updated.**

It should be noted that most of the pages are identifiable as having been processed by me.

I put a lot of time into producing these files which is why you are met with this page when you open the file.

In order to generate this file, I need to scan the pages, split the double pages and remove any edge marks such as punch holes, clean up the pages, set the relevant pages to be all the same size and alignment. I then run Omnipage (OCR) to generate the searchable text and then generate the pdf file.

Hopefully after all that, I end up with a presentable file. If you find missing pages, pages in the wrong order, anything else wrong with the file or simply want to make a comment, please drop me a line (see above).

It is my hope that you find the file of use to you personally – I know that I would have liked to have found some of these files years ago – they would have saved me a lot of time !

Colin Hinson
In the village of Blunham, Bedfordshire.

```
0001                    IDT  'INTERP'
0002                *                >>> GRAPHICS LANGUAGE INTERPRETER <<<      *
0003                ***********************************************************
0004                *                                                         *
0005                *          GRAPHICS PROGRAMMING LANGUAGE INTERPRETER      *
0006                *                                                         *
0007                ***********************************************************
0008                *          STATUS BLOCK CONTENTS RELATIVE TO PAD          *
0009                *       >70->71                  LAST VDP ADDRESS FROM MONITOR*
0010                *       >72          STKDAT      DATA STACK POINTER         *
0011                *       >73          STKADD      ADDRESS STACK POINTER      *
0012                *       >74          PLAYER      PLAYER NUMBER              *
0013                *       >75          KEYBRD      CODE FOR KEY PUSHED        *
0014                *       >76          JOYY        Y VALUE OF JOYSTICK        *
0015                *       >77          JOYX        X VALUE OF JOYSTICK        *
0016                *       >78          RANDOM      RANDOM NUMBER              *
0017                *       >79          TIME        TIMING VARIABLE            *
0018                *       >7A          MOTION      MOTION PARAMETER           *
0019                *       >7B          VDPST       VDP STATUS                 *
0020                *       >7C          STATUS      STATUS REGISTER            *
0021                *       >7D          CHRBUF      CHARACTER BUFFER           *
0022                *       >7E          YPT         Y POINTER IN VIDEO DISPLAY *
0023                *       >7F          XPT         X POINTER IN VIDEO DISPLAY *
0024                ***********************************************************
0025                *                EXTERNAL DEFINITIONS
0026                       DEF  NEXT,VDPRD,VDPWD,WRVDP,VDPWA
0027                       DEF  GRMWA,GRMRA,GRMWD,GRMRD,H20
0028                       DEF  SET,RESET,ENTRY,PUTSTK,GETSTK
0029                       DEF  SROM,SGROM,C2,C1000
0030                       REF  PARSEG,FLTTAB,WRITE,READ,TIMER,XTAB
0031                       REF  CONTG,EXECG,RTNG,VERIFY,CHTAB
0032                *                I/O DEVICE ADDRESSES
0033        8000    VDPREG EQU  >8000
0034        4000    WRVDP  EQU  >4000
0035        8C02    VDPWA  EQU  >8C02              LOAD VDP ADDRESS
0036        FC00    VDPSTA EQU  >8802-VDPWA        READ VDP STATUS
0037        FFFE    VDPWD  EQU  >8C00-VDPWA        WRITE DATA TO VDP
0038        FBFE    VDPRD  EQU  >8800-VDPWA        READ DATA FROM VDP
0039        9800    GRMRD  EQU  >9800              READ GROM DATA
0040        0402    GRMWA  EQU  >9C02-GRMRD        LOAD GROM ADDRESS
0041        0002    GRMRA  EQU  >9802-GRMRD        READ GROM ADDRESS
0042        0400    GRMWD  EQU  >9C00-GRMRD        WRITE DATA TO GRAM
0043        8400    SGCADR EQU  >8400              LOAD SOUND CHIP ADDRESS
0044        9400    SPKCMD EQU  >9400              SPEECH WRITE
0045        9000    SPKSTA EQU  >9000              SPEECH READ
0046                *------------------------------ ------------------------------*
0047                *
0048                *                STATUS BLOCK DEFINITIONS
0049         300    PAD    EQU  > 300
0050        834A    FAC    EQU  PAD+>4A
0051        8355    SCLEN  EQU  PAD+>55
0052        836C    TEMP2  EQU  PAD+>6C
0053        836D    TYPE   EQU  PAD+>6D
0054        8372    STKDAT EQU  PAD+>72
0055        8373    STKADD EQU  PAD+>73
0056        8374    PLAYER EQU  PAD+>74
0057        8375    KEYBRD EQU  PAD+>75
0058        8376    JOYY   EQU  PAD+>76
0059        8377    JOYX   EQU  PAD+>77
0060        8378    RANDOM EQU  PAD+>78
```

```
0061      8379  TIME   EQU  PAD+>79
0062      837A  MOTION EQU  PAD+>7A
0063      837B  VDPST  EQU  PAD+>7B
0064      837C  STATUS EQU  PAD+>7C
0065      837D  CHRBUF EQU  PAD+>7D
0066      837E  YPT    EQU  PAD+>7E
0067      837F  XPT    EQU  PAD+>7F
0068            *
0069            *                  WORKSPACE DEFINITIONS
0070      83C0  INTWSP EQU  PAD+>C0          INTERRUPT WORKSPACE AREA
0071      83E0  WORKSP EQU  PAD+>E0          RESERVE WORKSPACE AREA
0072            *
0073            *
0074            *                  ABSOLUTE 9900 LOCATIONS
0075      83C0  RAND16 EQU  PAD+>C0      SEED FOR RANDOM NUMBER
0076      83C2  INTFLG EQU  PAD+>C2      Flags for VDP interrupt handler
0077      83C4  INTPTR EQU  PAD+>C4      Pointer to VDP interrupt handler
0078      83C6  KBDFLG EQU  PAD+>C6      KBD FLAG (0=99/4,1=PASCAL,2=TE II
0079      83C7  OLDMOD EQU  PAD+>C7      LAST MODIFIER FLAGS
0080      83C8  DBNCE  EQU  PAD+>C8      POSITION DEBOUNCE FOR CONSOLE KBD
0081      83C9  DBNC1  EQU  PAD+>C9      POSITION DEBOUNCE FOR SPLIT KBD 1
0082      83CA  DBNC2  EQU  PAD+>CA      POSITION DEBOUNCE FOR SPLIT KBD 2
0083      83CB  SAVEG  EQU  PAD+>CB      SAVE GROM ADDRESS OF HEADER
0084      83CC  SNDADD EQU  PAD+>CC      SOUND LIST ADDRESS
00 5      83CE  STFLGS EQU  PAD+>CE      NUMBER OF SOUND BYTES
00 6      83D0  CRULST EQU  PAD+>D0
0087      83D2  SADDR   QU  PAD+>D2
0088      83D4  SAVVDP EQU  PAD+>D4
0089      83D6  TIMOUT EQU  PAD+>D6
0090      83D   RSAVE  EQU  PAD+>D8      SAVE R11 IN SCAN ROUTINE
0091            *      EQU  PAD+>DA
0092            *      EQU  PAD+>DC
0093            *      EQU  PAD+>DE
0094            *
0095            *                  LOCATIONS OF LOW ORDER BYTES OF REGISTERS
0096      83 1  R0LB   EQU  PAD+>E1
0097      83 3  R1LB   EQU  PAD+>E3
009       83 5  R2LB   EQU  PAD+>E5
0099      83 7  R3LB   EQU  PAD+>E7
0100       3 9  R4LB   EQU  PAD+>E9
0101      83 D  R6LB   EQU  PAD+>ED
0102      83F1  R LB   EQU  PAD+>F1
0103       3 B  R5LB   EQU  PAD+>EB
0104      83 F  R7LB   EQU  PAD+>EF
0105      83F3  R9LB   EQU  PAD+>F3
0106       3F7  R11L    QU  PAD+>F7
0107       3F9  R12LB   QU  PAD+>F9
010        3D9  AR1 L   QU  PAD+>D9
0109      83FB  R13L    QU  PAD+>FB
0110            *-------------------------------------------------------
0111            *                  SPECIAL  QUATE VALUES
0112      0024  ROW A  EQU  >24           CRU base for row data - output
0113      0006  COLBAS EQU  >06           CRU base for col data - input
0114      16E0  BBJOY  EQU  >16E0         Joystick tables
0115      1700  KEYTAB EQU  >1700         Table of unmodified keycodes
0116      1730  KSHIFT EQU  >1730         TABLE OF SHIFTED KEYCODES
0117      1760  KFNCTN EQU  >1760         Table of function keycodes
0118      1790  KCNTRL EQU  >1790         Table of control keycodes
0119      17C0  KSPLIT EQU  >17C0         TABLE OF SPLIT KEYBOARD KEYCODES
0120      2000  VDELTA EQU  >2000               >01 - >E0 INVALID VERTICAL
```

```
0134                    *---------------------------------------------------------------
0135                    *       ***** ENTRY POINT OF EMULATOR *****
0136                    *
0137 0000               RORG 0
0138 0000 83E0          DATA WORKSP,ENTRY       RESET VECTOR
     0002 0024'
0139 0004 83C0          DATA INTWSP,REMOTE      ALL 9901 INTERRUPTS
     0006 0900'
0140 0008 83C0  INTTIM  DATA INTWSP,TIMIN
     000A 0A92'
0141                    *
0142 000C   30  CLOCK   BYTE >30                3 MHZ CLOCK
0143 000D   AA  HAA     BYTE >AA                VALID ROM ID
0144                    ***************************************************************
0145                    *       CALL SCAN FROM ROM                                    *
0146                    ***************************************************************
0147 000E 0460  SCNKEY  B     @KSCAN            ALWAYS BL HERE FROM ANY ROM
     0010 02B2'
0148      0012' H00     EQU   $
0149      0013' BIT4    EQU   $+1
0150 0012 0008  BIT12   DATA >0008              HERE TO MAINTAIN ADDRESSES
0151                    ***************************************************************
0152                    *
0153                    **      BRANCH TO RETURN FROM BREAKPOINT FROM 99/4 DSR
0154                    *
0155 0014 1E00          SBZ   0                 DISABLE 99/4 DSR
0156 0016 0460          B     @PROCSS           RE-ENTRY ON BP IN 99/4
     0018 007A'
0157 001A 1E00          SBZ   0                 DISABLE 99/4 DSR
0158 001C 0460          B     @TRYAGN           RE-ENTRY ON ILLEGAL OPCODE IN 99/
     001E 0078'
0159                    ***************************************************************
0160 0020 0460          B     @CHKBRK           CHECK FOR BASIC & RS232 "BREAK" K
     0022 04B2'
0161                    ***************************************************************
```

```
0452                    *=============================================
0453                    ***    KEYBOARD SCAN
0454                    *
0455                    *** Initialization
0456                    *
0457 02AE 020B  KEYSCN  LI    R11,NEXT          RETURN TO NEXT IN GPL INTERP
     02B0 0070'
0458 02B2 C80B  KSCAN   MOV   R11,@RSAVE        Save return address
     02B4 83D8
0459 02B6 06A0          BL    @PUTSTK           Save GROM address
     02B8 0864'
0460 02BA 04CC          CLR   R12
0461 02BC 1D15          SBO   21                P5 off - alpha lock line
0462 02BE D160          MOVB  @PLAYER,R5        Is it console keyboard?
     02C0 8374
0463 02C2 0985          SRL   R5,8              Right justify
0464 02C4 C185          MOV   R5,R6             Make a copy of keyboard numb
0465 02C6 1312          JEQ   KSCAN1            Scan console keyboard
0466      02CA' HOF     EQU   $+2
0467 02C8 0200          LI    R0,>0FFF          No, so assume keyboard 1
     02CA 0FFF
0468 02CC 0606          DEC   R6                Is it keyboard 1 ?
0469 02CE 1312          JEQ   JSCAN             YES
0470 02D0 0200          LI    R0,>F0FF          No, so assume keyboard 2
     02D2 F0FF
0471 02D4 0606          DEC   R6                Is it keyboard 2 ?
0472 02D6 130E          JEQ   JSCAN             Yes
0473 02D8 0606          DEC   R6                Adjust again
0474 02DA 8806          C     R6,@C2            Illegal keyboard number?
     02DC 0072'
0475 02DE 1851          JH    NOKEY             Yes, return no key
0476 02E0 D806          MOVB  R6,@PLAYER        Reset keyboard number to zer
     02E2 8374
0477 02E4 06C6          SWPB  R6                Get low byte for flag
0478 02E6 D806          MOVB  R6,@KBDFLG        Store new keyboard flag
     02E8 83C6
0479 02EA 04C5          CLR   R5
0480 02EC 04C0  KSCAN1  CLR   R0                Scan console keyboard
04 1 02EE 04C6          CLR   R6                ASSUME NO KEY DOWN
0482 02F0 101E          JMP   KSCAN2
0483                    *
04 4                    *** Fire button positions
0485                    *
0486      02F1' FIRE    EQU   $-1
0487 02F2   29          BYTE  41
048  02F3   25          BYT   37
04 9                    *
0490                    *** Scan Joystick
0491                    *
0492 02F4 020C  JSCAN   LI    R12,ROWBAS        CRU base for row selection
     02F6 0024
0493 02F8 30E5          LDCR  @LINE(R5),3       SELECT LINE TO SCAN
     02FA 0405'
0494 02FC 020C          LI    R12,COLBAS        CRU base to read data
     02FE 0006
0495 0300 04C3          CLR   R3
0496 0302 0704          SETO  R4                "Clean" register for read
0497 0304 3544          STCR  R4,5              Read line
0498 0306 0994          SRL   R4,9              Get the fire button
0499 0308 1803          JOC   JSCAN1            No fire button
```

```
      03B8 0498'
 0597 03BA D820          MOVB @R3LB,@DBNCE       Always debounce KBD 0
      03BC 83E7
      03BE 83C8
 0598 03C0 D960          MOVB @R3LB,@DBNCE(R5)   Debounce KBD being scanned
      03C2 83E7
      03C4 83C8
 0599 03C6 C145          MOV  R5,R5              Is this KBD 0 ?
 0600 03C8 160C          JNE  NEWMOD             No so done
 0601 03CA C303          MOV  R3,R12             Get offset
 0602 03CC 022C          AI   R12,-8             Is key on line 0 ?
      03CE FFF8
 0603 03D0 1108          JLT  NEWMOD             Yes
 0604 03D2 0201          LI   R1,2               Assume split KBD 2
      03D4 0002
 0605 03D6 093C          SRL  R12,3              Which split KBD ?
 0606 03D8 1801          JOC  DBKB2              KBD 2 is right
 0607 03DA 0601          DEC  R1                 No, KBD 1 is right
 0608 03DC D860  DBKB2   MOVB @R3LB,@DBNCE(R1)   Debounce appropriate split KBD
      03DE 83E7
      03E0 83C8
 0609 03E2 D807  NEWMOD  MOVB R7,@OLDMOD
      03E4 83C7
 0610            *
 0611            *** Got the key position. Now get the key code.
 0612            *
 0613 03E6 D1E0  MODIFY  MOVB @OLDMOD,R7
      03E8 83C7
 0614 03EA 0201          LI   R1,KSPLIT          Assume split keyboard
      03EC 17C0
 0615 03EE C145          MOV  R5,R5              Split keyboard?
 0616 03F0 160E          JNE  MAPIT              Yes
 0617 03F2 0201          LI   R1,KCNTRL          Assume control mode
      03F4 1790
 0618 03F6 0A27          SLA  R7,2               Is it control mode?
 0619 03F8 180A          JOC  MAPIT              Yes
 0620 03FA 0201          LI   R1,KFNCTN          Assume function mode
      03FC 1760
 0621 03FE 09F7          SRL  R7,15              Is it function mode?
 0622 0400 1806          JOC  MAPIT              Yes
 0623 0402 0201          LI   R1,KSHIFT          Assume shift mode
      0404 1730
 0624      0405' LINE    EQU  $-1
 0625 0406 0607          DEC  R7                 Is it shifted?
 0626 0408 1302          JEQ  MAPIT              Yes
 0627 040A 0201          LI   R1,KEYTAB          It's an unmodified key
      040C 1700
 0628 040E A043  MAPIT   A    R3,R1              Add table offset
 0629 0410 DB41          MOVB R1, GRMWA(R13)     Set up to read table
      0412 0402
 0630 0414 DB60          MOVB @R1LB,@GRMWA(R13)  Set up to read table
      0416 83E3
      0418 0402
 0631            ***** Add following NOP 7/29/81
 0632 041A 1000          NOP
 0633 041C D01D          MOVB *R13,R0            Get key code from table
 0634 041E C145          MOV  R5,R5              Is it split keyboard?
 0635 0420 162B          JNE  OLDCHR             Yes, no special mapping
 0636 0422 D820          MOVB @KBDFLG,@R3LB      Get keyboard flag
```

```
0121  0780  RSMOT  EQU  >0780   SPRITE MOTION LIST ADDRESS
0122  0480  GSAML  EQU  >0480   SML - SAL
0123  2200  SHIFTQ EQU  >2200   HHU SHIFT Q CODE
0124  400C  H400C  EQU  >400C   INTERRUPT ADDRESS VECTOR
0125  4000  H4000  EQU  >4000   ROM ID LOCATION
0126  0000  INTROM EQU  >0000   CRU ADDRESS FOR PERIPH. ROM
0127  2800  DSRWSP EQU  >2800   DSR WORKSPACE FOR 99/4 DEBUGGE
0128  280A  XOPWSP EQU  >280A   XOP WORKSPACE FOR AL BPS
0129  2836  SSFLG  EQU  >2836   SINGLE STEP FLAG
0130  2838  MALPC  EQU  >2838   MODIFIED ALPC FLAG
0131  4024  SSTEP  EQU  >4024   DSR ENTRY POINT FOR SNGL STEP
0132  4028  ALBRK  EQU  >4028   DSR ENTRY POINT AFTER AL BP
```

```
0163          0024' ENTRY  EQU  $
0164          0025' HOD    EQU  $+1
0165 0024 020D        LI   R13,GRMRD
     0026 9800
0166 0028 020E        LI   R14,>0100
     002A 0100
0167 002C 020F        LI   R15,VDPWA
     002E 8C02
0168          0030' H2     EQU  $
0169          0030' IMMOPS EQU  $
0170          0032' H20    EQU  $+2
0171 0030 0200        LI   R0,>20         FIRST  BYTE IN GROM
     0032 0020
0172 0034 1013        JMP  DGBADD
0173 0036 1000 C1000  DATA >1000          HERE TO MAINTAIN ADDRESSES
```

```
0175                    *===============================================
0176 0038 1E00          SBZ  0                      DISABLE 99/4 DSR
0177 003A 02E0          LWPI XOPWSP                 SELECT CORRECT WORKSPACE
     003C 280A
0178 003E 0380          RTWP                        RETURN TO AL PROGRAM
0179 0040 280A          DATA XOPWSP,PRCXOP          XOP VECTOR FOR AL BP
     0042 0C1C'
0180                    * PASCAL TIBUG XOP
0181                    **** THE FOLLOWING LINE IS CHANGED 7/29/81
0182                    *       DATA >FFE8,>FFF8
0183 0044 FFD8          DATA >FFD8,>FFF8
     0046 FFF8
0184                    * A SPARE XOP JUST FOR FUN
0185 0048 83A0          DATA >83A0,>8300
     004A 8300
0186 004C 1100  FNCEQ   DATA >1100          TEST BITS FOR "FUNCTION ="
0187                    *                   HERE TO MAINTAIN ADRESSES
0188                    *---------------------------------------------------------
0189                    *       ***  BEGIN EXECUTION OF GPL INSTRUCTIONS ***
0190                    *                LIBRARY CALL ROUTINE
0191 004E 06A0  DGBA    BL   @PUTSTK        SAVE RETURN ADDRESS
     0050 0864'
0192 0052 06A0          BL   @PUTSTK        INCREMENT STACK POINTER
     0054 0864'
0193 0056 C90D          MOV  R13,@PAD(R4)   SAVE GROM BASE ADDRESS
     0058 8300
0194 005A C342          MOV  R2,R13         NEW GROM BASE ADDRESS
0195 005C D11D  DGBADD  MOVB *R13,R4        SYNCHRONIZE YOUR GROMS
0196 005E C180          MOV  R0,R6          BRANCH TO ADDRESS IN GROMS
0197 0060 DB46  LDKADD  MOVB R6,@GRMWA(R13)
     0062 0402
0198 0064 DB60          MOVB @R6LB,@GRMWA(R13)  LOAD GROM ADDRESS
     0066 83ED
     006  0402
0199 006A 5820  RES T   SZCB @BIT2,@STATUS      RESET CONDITION BIT
     006C 011B'
     006E 837C
0200      0072' IN2      QU   $+2
0201      0072' C2       EQU  IN2
0202 0070 0300  NEXT    LIMI 2              ALLOW INTERRUPTS BETWEEN
     0072 0002
0203      0074' H03     EQU  $
0204 0074 0300          LIMI 0              GPL INSTRUCTIONS -
     0076 0000
0205 0078 D25D  TRYAGN  MOVB *R13,R9        LOAD INSTRUCTION FROM GAME ROM
0206 007A 1105  PROCSS  JLT  ABOPS          JUMP ON SIGN BIT
0207 007C D109          MOVB R9,R4          MOVE INST INTO WORK REGISTER
020  007  09C4          SRL  R4,12          SHIFT TO LEAVE TOP 3 BITS * 2
0209 0080 C164          M V   ITAB(R4),R5   GET BRANCH ADDRESS
     00 2 0C36'
0210 0084 0455          B    *R5            JUMP ON TOP THREE BITS
0211                    *---------------------------------------------------------
0212                    *          IN TRUCTIONS WITH A AND B OPERANDS
0213 0086 04C4  ABOPS   CLR  R4             CLEAR VDP RAM FLAGS
0214 0088 C149          MOV  R9,R5          LOAD R5 WITH DOUBLE FLAG
0215 008A 0245          ANDI R5,>100
     008C 0100
0216 008E 06A0          BL   @GETMAD        GET FIRST OPERAND
     0090 077A'
0217 0092 0604          SWPB R4
```

```
0218 0094 C0C1          MOV   R1,R3          SAVE VARIABLE ADDRESS
0219 0096 C080          MOV   R0,R2          SAVE VARIABLE VALUE
0220 0098 0289          CI    R9,>A000       SINGLE OPERAND ?
     009A A000
0221 009C 1A09          JL    AOPS           YES
0222 009E 2260          COC   @IMMOPS,R9     IMMEDIATE OR VARIABLE ?
     00A0 0030'
0223 00A2 160C          JNE   ABOPA          VARIABLE
0224 00A4 C04D          MOV   R13,R1         GET  IMMEDIATE VALUE
0225 00A6 D011          MOVB  *R1,R0         GET FIRST BYTE
0226 00A8 0601          DEC   R1             MODIFY FOR COMMON ROUTINE
0227 00AA 06A0          BL    @MADC2         GO TO COMMON ROUTINE
     00AC 07AA'
0228 00AE 1008          JMP   BOPS
0229 00B0 C209  AOPS    MOV   R9,R8          BRANCH THROUGH BRANCH TABLE
0230 00B2 0988          SRL   R8,8
0231 00B4 0700          SETO  R0
0232 00B6 C228          MOV   @ATAB(R8),R8
     00B8 0BFE'
0233 00BA 0458          B     *R8
0234 00BC 06A0  ABOPA   BL    @GETMAD        GET SECOND VARIABLE OPERAND
     00BE 077A'
0235 00C0 C209  BOPS    MOV   R9,R8          BRANCH THROUGH BRANCH TABLE
0236 00C2 0998          SRL   R8,9
0237 00C4 C228          MOV   @BTAB(R8),R8 LOAD BRANCH ADDRESS
     00C6 0C4E'
0238 00C8 8002          C     R2,R0          COMPARE FOR IF PRIMITIVES
0239 00CA 0458          B     *R8
 240                    *--------------------------------------------------------*
0241                    *              PERFORM IF COMPARISONS
0242 00CC 11CE  GREQ    JLT   RESET          A ALGEBRAIC .GE. B
0243 00CE F820  SET     SOCB  @BIT2,@STATUS      SET CONDITION BIT
     00D0 011B'
     00D2 837C
0244 00D4 10CD          JMP   NEXT
0245 00D6 1BFB  HIGH    JH    SET            A LOGICAL .GT. B
0246 00D8 10C8          JMP   RESET
0247 00DA 14F9  HIGHEQ  JHE   SET            A LOGICAL .GE. B
0248 00DC 10C6          JMP   RESET
0249 00DE 15F7  GREATR  JGT   SET            A ALGEBRAIC .GT. B
0250 00E0 10C4          JMP   RESET
0251 00E2 0540  IFAND   INV   R0             PERFORM LOGICAL AND
0252 00E4 4080          SZC   R0,R2
0253 00E6 13F3          JEQ   SET
0254 00E8 10C0          JMP   RESET
0255 00EA C082  IFZ     MOV   R2,R2          COMPARE TO ZERO
0256 00EC 02C4  EQUAL   STST  R4             STORE STATUS IN STATUS BYTE
0257 00EE D 04          MOV   R4,@STATUS
     00F0  37C
025  00F2 10BE          JMP   NEXT
0259 00F4 C009  TSTST   MOV   R9,R0          MOVE INST TO SHIFT REGISTER
0260 00F6 0AC0          SLA   R0,12          REMOVE HIGH ORDER BITS
0261 00F8 09D0          SRL   R0,13          POSITION 3 REMAINING BITS
 262 00FA D160          MOVB  @STATUS,R5     LOAD STATUS BYTE
     00FC 837C
0263 00FE 0A05          SLA   R5,R0          SHIFT TO TEST BIT
0264 0100 18E6          JOC   SET
0265 0102 10B3          JMP   RESET
0266                    *--------------------- ----- ----------------------------*
0267                    *         BRANCH INSTRUCTIONS
```

```
0268 0104 D19D  BLONG   MOVB *R13,R6       RECALL ADDRESS FROM GROM
0269 0106 1000          NOP
0270 0108 D81D          MOVB *R13,@R6LB
     010A 83ED
0271 010C 10A9          JMP  LDKADD        LOAD GAME ROM ADDRESS
0272 010E D120  BSET    MOVB @STATUS,R4    IS CONDITION SET ?
     0110 837C
0273 0112 0A24          SLA  R4,2
0274 0114 1106          JLT  BRAD          YES
0275 0116 D11D  NOBR    MOVB *R13,R4       INCREMENT PROGRAM COUNTER
0276 0118 10A8          JMP  RESET
0277      011B' BIT2    EQU  $+1           CONSTANT >20
0278 011A D120  BRES T  MOVB @STATUS,R4    IS CONDITION SET ?
     011C 837C
0279 011E 0A24          SLA  R4,2
0280 0120 11FA          JLT  NOBR          YES
0281 0122 D81D  BRAD    MOVB *R13,@R9LB    GET SECOND BYTE OF ADDRESS
     0124 83F3
0282      0128' H1FFF   EQU  $+2
0283 0126 0249          ANDI R9,>1FFF      MASK TO LEAVE BRANCH ADDRESS
     0128 1FFF
0284 012A D1AD          MOVB @GRMRA(R13),R6    READ OLD MSB OF GROM AD
     012C 0002
0285 012E 0246          ANDI R6,>E000      LEAVE TOP 3 BITS OF ADDRESS
     0130 E000
0286 0132 E189          SOC  R9,R6         COMBINE TO GIVE NEW ADDRESS
0287 0134 1095          JMP  LDKADD
0288                    *--------------------------- ----------------------
0289                    *          SINGLE OPERAND INSTRUCTIONS
0290 0136 0742  ABS     ABS  R2            ABSOLUTE VALUE
0291 0138 107A          JMP  TRAP
0292 013A 0502  NEG     NEG  R2            NEGATE VALUE
0293 013C 1078          JMP  TRAP
0294 013E 0702  CLR     SETO R2            SET TO ZERO
0295 0140 0542  INV     INV  R2            INVERT VALUE
0296 0142 1075          JMP  TRAP
0297 0144 C184  FETCH   MOV  R4,R6         SAVE ADDRESSING MODE FLAG
0298 0146 06A0          BL   @PUTSTK       SAVE CURRENT PROGRAM ADDRESS
     014  0 64'
0299 014A 0644          DECT R4            SET I TO OLD STACK POINTER
0300 014C 06A0          BL   @GTSTK        GET ADDRESS FROM STACK TOP
     014E 0 4 '
0301 0150 D09D          MOVB *R13,R2       LOAD BYTE FROM GROM
0302 0152 0882          SRA  R2,8          EXTEND SIGN
0303 0154 05A4          INC  @PAD(R4)      INCREMENT RETURN ADDRESS
     0156 8300
0304 015  05C4          INCT R4            SET I BACK TO NEW STACK POIN
0305 015A 06A0          BL    GTSTK1       RESTORE GROM ADDRESS
     015C 0 4C'
0306 015E C106          MOV  R6,R4         RESTORE ADDRESSING MODE FLAG
0307 0160 1066          JMP  TRAP
0308 0162 0602  CASE    DEC  R2            VARIABLE DISPLACEMENT
0309 0164 1782          JNC  RESET
0310 0166 D15D          MOVB *R13,R5       SKIP TO NEW ADDRESS
0311 0168 1000          NOP                WAIT FOR GROM TO CATCH UP
0312 016A D15D          MOVB *R13,R5       SKIP TO NEW ADDRESS
0313 016C 10FA          JMP  CASE
0314 016E B80E  PUSH    AB   R14,@STKDAT   LOAD STACK ADDRESS  R14=01X
     0170 8372
0315 0172 D1A0          MOVB @STKDAT,R6
```

```
      0174 8372
0316  0176 0986          SRL   R6,8
0317  0178 D9A0          MOVB  @R2LB,@PAD(R6)      PUSH LSBYTE
      017A 83E5
      017C 8300
0318  017E 0460          B     @NEXT
      0180 0070'
0319                     *--------------------------------------------------------*
0320                     *          TWO OPERAND INSTRUCTIONS
0321  0182 09E0  DECT    SRL   R0,14         DECREMENT VALUE BY 2
0322  0184 0600  INCT    DEC   R0            INCREMENT INSTRUCTIONS
0323  0186       INC
0324  0186 0500  MINUS   NEG   R0            SUBTRACT OPERATION
0325  0188       DEC
0326  0188 D145  ADD     MOVB  R5,R5         SINGLE OR DOUBLE
0327  018A 134B          JEQ   ADDB
0328  018C A080          A     R0,R2         ADD OPERATION
0329  018E 104C          JMP   FILLST
0330  0190 0540  AND     INV   R0            LOGICAL AND OPERATION
0331  0192 4080          SZC   R0,R2
0332  0194 1049          JMP   FILLST
0333  0196 E080  OR      SOC   R0,R2         LOGICAL OR OPERATION
0334  0198 1047          JMP   FILLST
0335  019A 2880  XOR     XOR   R0,R2         LOGICAL EXCLUSIVE OR OPERATION
0336  019C 1045          JMP   FILLST
0337  019E C080  STORE   MOV   R0,R2         STORE SOURCE INTO DESTINATION
0338  01A0 1046          JMP   TRAP
0339  01A2 C242  EXCH    MOV   R2,R9         EXCHANGE A AND B
 340  01A4 C080          MOV   R0,R2
0341  01A6 06A0          BL    @TRAPA
      01A8 0232'
0342  01AA 06C4          SWPB  R4
0343  01AC C0C1          MOV   R1,R3
0344  01AE 101B          JMP   MUL2
0345  01B0 0802  SRA     SRA   R2,R0         SHIFT RIGHT ARITHMETIC
0346  01B2 103D          JMP   TRAP
0347  01B4 0A02  SLL     SLA   R2,R0
0348  01B6 103B          JMP   TRAP
0349  01B8 D145  SRL     MOVB  R5,R5         SHIFT RIGHT LOGICAL
0350  01BA 1601          JN    SRL1          DOUBLE INSTRUCTION
0351  01BC 7082          SB    R2,R2
0352  01BE 0902  SRL1    SRL   R2,R0
0353  01C0 1036          JMP   TRAP
0354  01C2 D145  SRC     MOVB  R5,R5         SHIFT RIGHT CIRCULAR
0355  01C4 1602          JNE   SRC1          DOUBLE INSTRUCTION
0356  01C6 D0A0          MOVB   R2LB,R2
      01C8 83E5
0357  01CA 0B02  SRC1    SRC   R2,R0
035   01CC 1030          JMP   TRAP
0359  01CE C202  MUL     MOV   R2,R8         MULTIPLY
0360  01D0 D145          MOVB  R5,R5         SINGLE OR DOUBLE
0361  01D2 1601          JNE   MUL0          DOUBLE
0362  01D4 7208          SB    R ,R8         CLEAR EXTENDED SIGN BITS
 363  01D6 3A00  MUL0    MPY   R0,R8
0364  01D8 D145          MOVB  R5,R5         SINGLE OR DOUBLE ?
0365  01DA 1602          JNE   MUL1          DOUBLE
0366  01DC D809          MOVB  R9,@R8LB
      01DE 83F1
0367  01E0 C088  MUL1    MOV   R8,R2         STORE FIRST HALF OF RESULT
0368  01E2 06A0          BL    @TRAPA
```

```
      01E4 0232'
0369  01E6 C089  MUL2   MOV  R9,R2          STORE SECOND HALF OF RESULT
0370  01E8 1022        JMP  TRAP
0371  01EA D805  DIV    MOVB R5,@STATUS     CLEAR STATUS
      01EC 837C
0372  01EE C202        MOV  R2,R8
0373  01F0 C080        MOV  R0,R2
0374  01F2 C043        MOV  R3,R1          SET DESTINATION ADDRESS
0375  01F4 0581        INC  R1
0376  01F6 D145        MOVB R5,R5          SINGLE OR DOUBLE ?
0377  01F8 1301        JEQ  DIV1           SINGLE
037   01FA 05 1        INC  R1
0379  01FC D104  DIV1   MOVB R4,R4          CPU OR VDP ?
03 0  01FE 1303        JEQ  DIVC           CPU
0381  0200 06A0        BL   @MVDPA         GET REST OF DIVIDEND FROM VDP
      0202 07FA'
0382  0204 1002        JMP  DIVB
0383  0206 06A0  DIVC   BL   @MADC          GET REST OF DIVIDEND FROM CPU
      0208 07A8'
0384  020A C240  DIVB   MOV  R0,R9
0385  020C D145        MOVB R5,R5          SINGLE OR DOUBLE ?
0386  020E 1603        JNE  DIV2
0387  0210 D260        MOVB @R8LB,R9       DO SIGN EXTENSION FOR BYTE
      0212 83F1
03    0214 0888        SRA  R8,8           OPERANDS
03 9  0216 3E02  DIV2   DIV  R2,R8          PERFORM DIVISION
0390  0218 19E3        JNO  MUL1
0391  021A F820        SOCB @BIT4,@STATUS     SET DIVIDE OVERFLOW
      021C 0013'
      021E 837C
0392  0220 10DF        JMP  MUL1
0393               *              FILL IN STATUS BITS
0394  0222 B820  ADDB   AB   @R0LB,@R2LB
      0224 83E1
      0226 83E5
0395  0228 02CB  FILLST STST R11           STORE STATUS WORD
0396  022A D80B        MOVB R11,@STATUS    STORE STATUS BITS
      022C 837C
0397               *
039                *          *** TRAP OUT CHANGES IN CB, XPT, AND YPT **
0399               *
0400  022E 020B  TRAP   LI   R11,NEXT       LOAD R11 TO RETURN TO NEXT
      0230 0070'
0401  0232 D104  TRAPA  MOVB R4,R4          DESTINATION IS CPU OR VDP ?
0402  0234 130F        J Q   TCPU          CPU
0403  0236 D7E0        MOV   R3LB,*R15     LOAD VDP ADDRESS
      023  3E7
0404  023A 0263        ORI  R3,WRVDP       S T BIT TO WRITE TO VDP
      023C 4000
0405  023E D7C3        MOV  R3,*R15
0406  0240 D145        MOVB R5,R5          SINGLE OR DOUBLE
0407  0242 1303        JEQ  TRAP1          SINGLE
040   0244 DBC2        MOVB R2,@VDPWD(R15) MOVE 2 BYTES TO VDP
      0246 FFFE
0409  0248 0583        INC  R3
0410  024A DBE0  TRAP1  MOVB @R2LB,@VDPWD(R15) MOVE 1 BYTE TO VDP
      024C 83E5
      024E FFFE
0411  0250 0583        INC  R3
0412  0252 045B  TEXIT  RT
```

```
0413 0254 D145  STCPU  MOVB R5,R5         SINGLE OR DOUBLE ?
0414 0256 1301         JEQ  TRAP2         SINGLE
0415 0258 DCC2         MOVB R2,*R3+       STORE 2 BYTE RESULT
 416 025A 06C2  TRAP2  SWPB R2
0417 025C DCC2         MOVB R2,*R3+       STORE 1 BYTE RESULT
0418 025E 0283         CI   R3,PAD+>7E    IS CB THE DESTINATION ?
     0260 837E
0419 0262 16F7         JNE  TEXIT         NO
0420 0264 C18B         MOV  R11,R6
0421 0266 06A0         BL   @GETDA        GET VDP DISPLAY ADDRESS
     0268 0880'
0422 026A DBC2  RTN    MOVB R2,@VDPWD(R15) WRITE CHARACTER TO VDP
     026C FFFE
0423 026E 0456         B    *R6           RETURN
0424                   *--------------------------------------------------------*
0425                   *          MISCELLANEOUS INSTRUCTIONS
0426 0270 0A39  MISCLN SLA  R9,3          REMOVE TOP THREE BITS OF INST
0427 0272 09A9         SRL  R9,10         LEAVE BOTTOM FIVE BITS * 2
0428 0274 C129         MOV  @MSCTAB(R9),R4  GET BRANCH ADDRESS
     0276 0C3E'
0429 0278 0454         B    *R4
0430                   *
0431                   *--------------------------------------------------------*
0432                   *          GENERATE A RANDOM NUMBER
0433 027A 0204  RANDNO LI   R4,28645
     027C 6FE5
0434 027E 3920         MPY  @RAND16,R4
     0280 83C0
 435 0282 0225         AI   R5,31417
     0284 7AB9
0436 0286 C805         MOV  R5,@RAND16
     0288 83C0
0437 028A D19D         MOVB *R13,R6       LOAD MODULO VALUE
0438 028C 0986         SRL  R6,8          SHIFT TO LOW ORDER POSITION
0439 028E 0586         INC  R6
0440 0290 0 ^4         CLR  R4            CLEAR UPPER HALF OF DIVIDEND
0441 0292 06C5         SWPB R5            SWAP BYTES IN J
0442 0294 3D06         DIV  R6,R4         PERFORM DIVISION
0443 0296 D 20         MOVB @R5LB,@RANDOM      STORE REMAINDER
     029  83EB
     029A 8378
0444 029C 1006         JMP  BKGR1
0445                   *
0446                   *          SET BACKGROUND COLOR
0447 029E 0207  BKGRND LI   R7,>700+VDPREG     CHANGE BACKGROUND COLOR
     02A0  700
044  02A2 D 1D         MOVB *R13,@R7LB   IMMEDIATE  OPERAND
     02A4  3 F
0449 02A6 06A0         BL    S TVDP      OUTPUT ADDRESS TO VDP
     02A  0 9A'
0450 02AA 0460  BKGR1  B    @NEXT
     02AC 0070'
```

```
500 030A D825          MOVB @FIRE(R5),@R3LB    Get fire button
    030C 02F1'
    030E 83E7
501 0310 0A14  JSCAN1  SLA  R4,1               Times 2 = index into table
502 0312 0224          AI   R4,BBJOY           Add in address of table
    0314 16E0
503 0316 DB44          MOVB R4,@GRMWA(R13)
    0318 0402
504 031A DB60          MOVB @R4LB,@GRMWA(R13)
    031C 83E9
    031E 0402
505                    **** FOLLOWING NOP IS ADDED 7/29/81
506 0320 1000          NOP
507 0322 D81D          MOVB *R13,@JOYY         Read Y value into JOYY
    0324 8376
508 0326 D81D          MOVB *R13,@JOYX         Read X value into JOYX
    0328 8377
509 032A C0C3          MOV  R3,R3              Fire button down?
510 032C 163E          JNE  DEBOUN             Yes
511                    *
512                    *** Scan Keyboard for a Key
513                    *
514 032E 0201  KSCAN2  LI   R1,5                Set row = 0
    0330 0005
515 0332 04C2          CLR  R2                 Assume no key is down
516 0334 04C7          CLR  R7                 Assume no modifier keys
517 0336 020C  ROWLP   LI   R12,ROWBAS         CRU BASE FOR ROW SELECTION
    0338 0024
518 033A 06C1          SWPB R1
519 033C 30C1          LDCR R1,3               Output row scan to KB
520 033E 06C1          SWPB R1
521 0340 020C          LI   R12,COLBAS         Base for column input
    0342 0006
522 0344 0704          SETO R4                 "Clean" register for read
523 0346 3604          STCR R4,8               INPUT COLUMN DATA
524 0348 0544          INV  R4                 MAKE ONE'S REPRESENT KEYS DOWN
525                    * LINES ADDED ON 8/11/81 TO FIX BUG <><><><><><><><><><><>
526                    * ADDED BY ALLEN ACREE
527 034A C041          MOV  R1,R1              ROW 0?
052 034C 1603          JNE  NOTONE
529 034E D1C4          MOVB R4,R7
530 0350 0244          ANDI R4,>0F00
    0352 0F00
531 0354               NOTONE
532                    *  END OF INSERTION     <><><><><><><><><><><><><><><><><>
533 0354 4100          SZC  R0,R4              MASK OUT INAPPROPRIATE KEYS
534 0356 1311          JEQ  NXTROW             IF NO KEY DOWN, TRY NEXT LINE
535                    *
536                    *** GOT A K Y! DETERMINE WHICH ONE!
537                    *
538 0358 C041  GOTK Y  MOV  R1,R1              IS THIS LINE 0 ?
539 035A 1602          JNE  GOTK5              NO
540 035C C145          MOV  R5,R5              SCANNING SPLIT KEYBOARD?
541 035E 160D          JNE  NXTROW             YES SO NO KEYS ON THIS LINE
542                    * DELETE FOLLOWING LINES ON 8/11/81 TO FIX BUG <><><><><><>
543                    * BY ALLEN ACREE
544                    *      MOVB R4,R7       SAVE CNTRL AND/OR FNCT'N
545                    *      ANDI R4,>0F00    MASK OUT CNTRL AND/OR FNCT'N
546                    *      JEQ  NXTROW      NO OTHER KEYS SO NEXT LINE
547                    * END OF DELETION <><><><><><><><><><><><><><><><><><><><>
```

```
0548 0360 C082  GOTK5  MOV  R2,R2              IS THIS THE SECOND KEY ?
0549 0362 1603         JNE  NXTROW             YES, IGNORE IT
0550 0364 0702         SETO R2                 INDICATE A KEY DOWN
0551 0366 C0C1         MOV  R1,R3              GET ROW NUMBER
0552 0368 0A33         SLA  R3,3               Right-justify & MPY by 8
0553 036A 0603         DEC  R3                 ADJUST FOR ADDING IN COLUMNS
0554 036C 0583  CNTLP  INC  R3                 ADD IN ANOTHER COLUMN
0555 036E 0A14         SLA  R4,1               SHIFT NEXT BIT OUT INTO CARR
0556 0370 17FD         JNC  CNTLP              BIT = 1 IS KEY DEPRESSED
0557                   * FOLLOWING TWO LINES REMOVED ON 8/11/81 TO FIX BUG <><><>
055                    * BY ALLEN ACREE
0559                   *      MOV  R5,R5       SPLIT KEYBOARD ?
0560                   *      JNE  DEBOUN      YES GO WITH IT
0561                   * END OF DELETION <><><><><><><><><><><><><><><><><><><><>
0562 0372 C041         MOV  R1,R1              IS THIS LINE 0 ?
0563 0374 1302         JEQ  NXTROW
0564      0378' C1     EQU  $+2
0565 0376 0201         LI   R1,1               MOVE ON TO LAST LINE
     0378 0001
0566                   *
0567                   *** SELECT NEXT LINE TO POLL
0568                   *
0569 037A 0601  NXTROW DEC  R1                 Decrement row number
0570 037C 18DC         JOC  ROWLP              Not finished- scan next row
0571 037E C082         MOV  R2,R2              Was a key pressed?
0572 0380 1614         JNE  DEBOUN             Yes
0573                   *
0574                   *** No Keys Down
0575                   *
0576 0382 04C6  NOKEY  CLR  R6                 Condition reset
0577 0384 D806         MOVB R6,@OLDMOD         Clear old modifiers
     0386 83C7
0578 0388 0700         SETO R0                 Invalid keycode is >FF
0579 038A 9940         CB   R0,@DBNCE(R5)      Was key just released?
     03 C 83C8
0580 038  1302         JEQ  NOKEY2             No so no delay
0581 0390 06A0         BL   @KL10MS            Kill some time for debounce
     0392 0498'
0582 0394 D800  NOKEY2 MOVB R0,@DBNCE          Always clear KBD 0 debounce
     0396 83C8
0583 0398 D940         MOVB R0,@DBNCE(R5)      Always clear KBD being scann
     039A 83C8
05 4 039C C145         MOV  R5,R5              Is this KBD 0 ?
05 5 039E 166C         JN   OLDCHR             No so done
05 6 03A0 D800         MOVB R0,@DBNC1          Clear KBD 1 debounce
     03A2 83C9
05 7 03A4 D800         MOV  R0, DBNC2          Clear KBD 2 debounce
     03A6  3CA
05   03A  1067         JMP  OLDCHR
0589                   *
0590                   *** Debounce key
0591                   *
0592 03AA 9960  DEBOUN CB   @R3LB,@DBNCE(R5)   Same key position?
     03AC 83E7
     03AE 83C8
0593 03B0 131A         JEQ  MODIFY             Yes, go get old key code
0594      03B4' H020   EQU  $+2
0595 03B2 0206         LI   R6,>2000           Set status to new
     03B4 2000
0596 03B6 06A0         BL   @KL10MS            Kill some time for debounce
```

```
                0426 83E7
0637 0428 06A0          BL   @RNGCHK            Is it a lowercase letter
     042A 04A2'
0638 042C 617A          DATA 'az'
0639 042E 160A          JNE  WHCHKB             No so no alpha lock
0640 0430 04CC          CLR  R12                Prepare to check alpha lock
0641 0432 C0C3          MOV  R3,R3              Is this keyboard 0 ?
0642 0434 1304          JEQ  ALOCK              Yes so map to upper case
0643 0436 1E15          SBZ  21                 Select P5 - output
0644 0438 0BEC          SRC  R12,14             Kill 14 micro-seconds
0645 043A 1F07          TB   7                  Test alpha lock
0646 043C 1302          JEQ  RSTP5              No alpha lock
0647 043E 7020  ALOCK   SB   @H020,R0           Map to upper case
     0440 03B4'
0648 0442 1D15  RSTP5   SBO  21                 De-select P5 - alpha lock lir
0649 0444 C0C3  WHCHKB  MOV  R3,R3              Is it keyboard 0 ?
0650 0446 1607          JNE  WHCH2              No, so no mapping
0651 0448 06A0          BL   @RNGCHK            Bad key?
     044A 04A2'
0652 044C 101F          DATA >101F
0653 044E 1399          JEQ  NOKEY              Yes, return no key
0654 0450 9800          CB   R0,@H5F            Is it higher than "_" ?
     0452 0387'
0655 0454 1396          JH   NOKEY              Yes, return no key
0656 0456 0603  WHCH2   DEC  R3                 Is this keyboard 4 ?
0657 0458 160F          JNE  OLDCHR             No, it's 5. So return code.
0658 045A 9800          CB   R0,@HOD            Is it RETURN key?
     045C 0025'
0659 045E 130C          JEQ  OLDCHR             Yes so return it
0660 0460 9800          CB   R0,@HOF            Is it GPL control key (1-1
     0462 02CA'
0661 0464 1B03          JH   MAP4               No
0662 0466 F020          SOCB @H80,R0            Yes so set sign bit
     0468 0470'
0663 046A 1006          JMP  OLDCHR             Return it
0664 046C 06A0  MAP4    BL   @RNGCHK            Is it ASCII printable ?
     046E 04A2'
0665 0470 809F  H80     DATA >809F
0666 0472 1602          JNE  OLDCHR             Yes, return it
0667 0474 5020          SZCB @H 0,R0            Control code, reset sign bit
     0476 0470'
0668 0478 D800  OLDCHR  MOVB R0,@KEYBRD         Output ASCII character
     047A 8375
0669 047C 06A0          BL   @GETSTK            Restore GROM address
     047E 0 42'
0670 0480 D806          MOVB R6,@STATUS         Store status
     04 2  37C
0671 04 4 1306          JEQ  EXSCN              No new key
0672 0486 D7E0          MOVB @SAVVDP,*R15       Turn on the screen
     0488 83D4
0673 048A 04E0          CLR  @TIMOUT            Reset screen timer
     048C 83D6
0674 048E D7E0          MOVB @H 1,*R15          VDP register 1
     0490 0B61'
0675 0492 C2E0  EXSCN   MOV  @RSAVE,R11         Restore return address
     0494 83D8
0676 0496 045B          RT                      Return to caller
0677            *
0678            *** Kill time routine
```

```
0680 0498 020C  KL10MS LI   R12,1250          Load up 10 m-secs of counts
     049A 04E2
0681 049C 060C  KILTIM DEC  R12               Count it down
0682 049E 16FE         JNE  KILTIM            Not done yet
0683 04A0 045B         RT                     Return to caller
0684            *
0685            *** Range check routine
0686            *
0687 04A2 C33B  RNGCHK MOV  *R11+,R12          Fetch range
0688 04A4 9300         CB   R0,R12            Compare to low end of range
0689 04A6 1A04         JL   RNGRT             Out of range low
0690 04A8 9800         CB   R0,@R12LB         Compare to high end of range
     04AA 83F9
0691 04AC 1B01         JH   RNGRT             Out of range high
0692 04AE 9000         CB   R0,R0             In range so set EQ bit
0693 04B0 045B  RNGRT  RT                     Return to caller
0694            *
0695            *** Check for the BASIC and RS232 "BREAK" key
0696            *
0697 04B2 020C  CHKBRK LI   R12,>24           Select output for rows
     04B4 0024
0698 04B6 30E0         LDCR @H00,3            Select line 0
     04B8 0012'
0699 04BA 0B7C         SRC  R12,7             Kill some time
0700 04BC 020C         LI   R12,6             Select input for columns
     04BE 0006
0701 04C0 360C         STCR R12,8             Read data
0702 04C2 2720         CZC  @C1000,R12        Is proper key down?
     04C4 0036'
0703 04C6 160A         JNE  CHKBEX            No so exit
0704 04C8 020C         LI   R12,>24           Select output for rows
     04CA 0024
0705 04CC 30E0         LDCR @H03,3            Select line 3
     04CE 0074'
0706 04D0 0B7C         SRC  R12,7             Kill some time
0707 04D2 020C         LI   R12,6             Select input for columns
     04D4 0006
0708 04D6 360C         STCR R12,8             Read data
0709 04D8 2720         CZC  @C1000,R12        Is proper key down?
     04DA 0036'
0710 04DC 045B  CHKBEX RT                     Return condition to the caller
```

```
0712                    *---------------------------------------------------
0713                    *       FORMAT FOR STRING
0714 04DE 04C9  FORMAT  CLR   R9              RESET BLOCK COUNT TO 0
0715 04E0 04C3          CLR   R3              ZERO BIAS BYTE
0716 04E2 06A0          BL    @GETDA          ENCODE VDP ADDRESS
     04E4 0880'
0717 04E6 D21D  FUNCTN  MOVB  *R13,R8         GET NEXT BYTE FROM GAME ROM
0718 04E8 020C          LI    R12,STKADD
     04EA 8373
0719 04EC C148          MOV   R8,R5           SAVE DATA BYTE
0720 04EE 0A38          SLA   R8,3            SHIFT TO FIND FUNCTION TO PERFORM
0721 04F0 09B8          SRL   R8,11           MOVE TO LOW ORDER POSITION
0722 04F2 0548          INV   R8              LEAVE N IN RANGE OF MINUS 1 - 32
0723 04F4 09C5          SRL   R5,12           ISOLATE FUNCTION BITS
0724 04F6 C165          MOV   @FBTAB(R5),R5       GET BRANCH ADDRESS
     04F8 0CDC'
0725 04FA 0202          LI    R2,LOOP
     04FC 050A'
0726 04FE 0704          SETO  R4              SET I TO -1
0727 0500 0455          B     *R5
0728                    *                     REPEAT CHARACTER N TIMES DOWN THE SCREEN
0729 0502 0A54  RPTDWN  SLA   R4,5            SET I TO -32
0730                    *                     REPEAT CHARACTER N TIMES ACROSS THE SCREEN
0731 0504 8CB2  RPTACR  C     *R2+,*R2+       LOAD BRANCH ADDRESS FOR LOOPING
0732 0506 1001          JMP   LOOP
0733                    *                     STORE N CHARACTERS DOWN THE SCREEN
0734 0508 0A54  STRDWN  SLA   R4,5            SET I TO -32
0735                    *                     STORE N CHARACTERS ACROSS THE SCREEN
0736 050A       STRACR
0737                    *                     LOOP TO STORE CHARACTERS
0738 050A D19D  LOOP    MOVB  *R13,R6         GET NEXT BYTE FROM GAME ROM
0739 050C A183  LOOPB   A     R3,R6           ADD BIAS TO CHARACTER
0740 050E DBC6  LOOPA   MOVB  R6,@VDPWD(R15) MOVE BYTE INTO DISPLAY
     0510 FFFE
0741 0512 61C4          S     R4,R7           DISPLACEMENT TO NEXT CHARACTER
0742 0514 0287          CI    R7,>320         DO I WANT WRAPAROUND?
     0516 0320
0743 0518 1405          JHE   ZEND            NO
0744 051A 0287  UPXY    CI    R7,>300         DISPLAY ADDRESS OUT OF RANGE?
     051C 0300
0745 051E 1A02          JL    ZEND            NO
0746 0520 0227          AI    R7,->300        SUBTRACT LENGTH OF DISPLAY
     0522 FD00
0747 0524 06A0  ZEND    BL    @RESTOR
     0526 05B8'
074  052  06A0          BL    @GETDA
     052A 0 80'
0749 052C 058           INC   R8              INCREMENT LOOP COUNTER
0750 052E 13DB          J Q   FUNCTN          DONE WITH LOOP IF N = 0
0751 0530 0452          B     *R2             LOOP UNTIL N = 0
0752                    *                     SKIP N LINES DOWN THE SCREEN
0753 0532 0A5   SKPDWN  SLA   R8,5            MULTIPLY LINES TO SKIP BY 32
0754                    *                     SKIP N SPACES ACROSS THE SCREEN
0755 0534 61C   SKPACR  S     R8,R7           MOV  NUMBER OF SPACES TO SKIP I
0756 0536 070           SETO  R8              S T LOOP COUNTER TO -1
0757 0538 10F0          JMP   UPXY            GO PERFORM SKIP
0758                    *                     REPEAT BLOCK N TIMES
0759 053A 0589  RPTBLK  INC   R9              INCREMENT BLOCK COUNTER
0760 053C 05DC          INCT  *R12            INCREMENT STACK POINTER
0761 053E D19C          MOVB  *R12,R6         GET DATA STACK POINTER
```

```
0762 0540 0986          SRL  R6,8           MOVE TO LOW ORDER BYTE
0763 0542 D9A0          MOVB @R8LB,@PAD(R6)      PUSH N ONTO DATA STACK
     0544 83F1
     0546 8300
0764 0548 10CE          JMP  FUNCTN
0765                *                REPETITION OF BLOCK HAS BEEN COMPLETED
0766 054A 065C  FINISH  DECT *R12           DECREMENT DATA STACK POINTER R14=01)
0767 054C 0609          DEC  R9             DECREMENT BLOCK COUNT
0768 054E 10CB          JMP  FUNCTN
0769                *                END BLOCK FUNCTION
0770 0550 C249  ENDBLK  MOV  R9,R9          IS BLOCK COUNT ZERO ?
0771 0552 1330          JEQ  ENDFMT         YES - END OF FORMAT INSTRUCTION
0772 0554 D11D          MOVB *R13,R4        LOAD LOOP ADDRESS
0773 0556 D19C          MOVB *R12,R6        GET DATA STACK POINTER
0774 0558 D15D          MOVB *R13,R5
0775 055A 0986          SRL  R6,8
0776 055C B98E          AB   R14,@PAD(R6) DECREMENT REPITITION COUNT R14=01XX
     055E 8300
0777 0560 13F4          JEQ  FINISH         DONE WITH BLOCK IF COUNT IS ZERO
0778 0562 DB44          MOVB R4,@GRMWA(R13)      LOAD GROM WITH LOOP ADDRESS
     0564 0402
0779 0566 DB45          MOVB R5,@GRMWA(R13)
     0568 0402
0780 056A 10BD          JMP  FUNCTN
0781                *
0782                *                SPECIAL FORMAT FUNCTIONS
0783      056F' HE4     EQU  $+3
0784 056C 0288  SPECL   CI   R8,-28         WHICH FUNCTION ?
     056E FFE4
0785 0570 13EF          JEQ  ENDBLK         END BLOCK
0786 0572 1511          JGT  VARBLE         VARIABLE CHARACTERS
0787 0574 C04D          MOV  R13,R1
0788 0576 0288          CI   R8,-30         WHICH FUNCTION ?
     0578 FFE2
0789 057A 1309          JEQ  BIASV          SET COLOR BIAS
0790 057C 150A          JGT  BIASI          SET COLOR BIAS IMMEDIATE
0791 057E 06A0          BL   @RESTOR        SET XPT OR YPT
     0580 05B8'
0792 0582 0508          NEG  R8
0793      0587' H5F     EQU  $+3
0794 0584 DA1D          MOVB *R13,@PAD+>5F(R8) LOAD VALUE OF XPT OR YPT
     0586 835F
0795 0588 06A0          BL   @GETDA         COMPUTE VALUE OF ADDRESS
     058A 0880'
0796 058C 10AC          JMP  FUNCTN
0797                *                SET COLOR BIAS
0798 05 E 06A0  BIASV   BL   @GETMAB        GET MEMORY ADDRESS
     0590 0778'
0799 0592 D0D1  BIASI   MOVB *R1,R3         LOAD COLOR BIAS
0800 0594 10A8          JMP  FUNCTN
0801                *                VARIABLE CHARACTERS
0802 0596 06A0  VARBLE  BL   @GETMAB        GET MEMORY ADDRESS
     0598 0778'
0803 059A 0202          LI   R2,LOOPV       SET LOOP REGISTER
     059C 059E'
0804 059E D1B1  LOOPV   MOVB *R1+,R6        LOAD VARIABLE CHARACTER
0805 05A0 10B5          JMP  LOOPB
0806                *-------------------------------------------------------*
0807                *                CLEAR SCREEN
0808 05A2 D15D  ALL     MOVB *R13,R5        GET CHARACTER TO STORE
```

```
0809 05A4 06A0          BL   @SETV
     05A6 08A4'
0810 05A8 0207          LI   R7,768        NUMBER OF BYTES TO TRANSFER
     05AA 0300
0811 05AC DBC5  ALLOOP  MOVB R5,@VDPWD(R15) STORE BYTE IN DISPLAY
     05AE FFFE
0812 05B0 0607          DEC  R7
0813 05B2 16FC          JNE  ALLOOP
0814                    *                  RESTORE XPT AND YPT FROM DISPLAY ADDRESS
0815 05B4 020B  ENDFMT  LI   R11,NEXT      END OF FORMAT
     05B6 0070'
0816 05B8 0A37  RESTOR  SLA  R7,3          SHIFT YPT VALUE TO TOP BYTE
0817 05BA D807          MOVB R7,@YPT       STORE VALUE OF YPT
     05BC 837E
0818 05BE 0A87          SLA  R7,8          SHIFT OFF VALUE OF YPT
0819 05C0 0937          SRL  R7,3          LEAVE VALUE OF XPT
0820 05C2 D807          MOVB R7,@XPT       STORE VALUE OF XPT
     05C4 837F
0821 05C6 045B          RT
```

```
                *===============================================================*
                *       INPUT / OUTPUT INSTRUCTION
05C8 C080   INOUT   MOV   R0,R2          GET INDEX INTO I/O TABLE
05CA C043           MOV   R3,R1          MOVE LIST ADDRESS FOR CASSETTE
05CC A082           A     R2,R2
05CE C122           MOV   @IOTAB(R2),R4      LOAD ADDRESS OF I/O FUNCTION
05D0 0CEC'
05D2 04C9           CLR   R9             CLEAR R9 FOR CRU INSTRUCTIONS
05D4 0454           B     *R4            BRANCH TO I/O FUNCTION
                *          SOUND INSTRUCTION
05D6 024E   SOUND   ANDI  R14,>FFFE      CLEAR SOUND SOURCE FLAG
05D8 FFFE
05DA E380           SOC   R0,R14         SET SOUND SOURCE FLAG
05DC C813           MOV   *R3,@SNDADD    LOAD SOUND LIST ADDRESS
05DE 83CC
05E0 D80E           MOVB  R14,@STFLGS    SET TIME DELAY TO 1 UNIT R14=01XX
05E2 83CE
05E4 0460   GEXIT   B     @NEXT
05E6 0070'
                *          CRU INSTRUCTIONS                                   *
05E8 0589   CRUIN   INC   R9
05EA C331   CRUOUT  MOV   *R1+,R12       LOAD CRU BASE ADDRESS
05EC A30C           A     R12,R12
05EE 04C2           CLR   R2
05F0 D0B1           MOVB  *R1+,R2        LOAD NUMBER OF BITS TO MOVE
05F2 0A42           SLA   R2,4
05F4 E242           SOC   R2,R9          COMBINE NUMBER AND IN/OUT FLAG
05F6 0B69           SRC   R9,6           REPOSITION BITS
05F8 0269           ORI   R9,>3012       R9 IS NOW LDCR OR STCR
05FA 3012
05FC D091           MOVB  *R1,R2         LOAD CPU SOURCE/DEST ADDRESS
05FE 06C2           SWPB  R2
0600 0222           AI    R2,PAD
0602 8300
0604 0489           X     R9             EXECUTE INSTRUCTION IN R9
0606 10EE           JMP   GEXIT
                *===============================================================*
                *          EXECUTE 9900 CODE IN EXTERNAL ROM
0608 D25D   XML     MOVB  *R13,R9
060A C109           MOV   R9,R4
060C 09C9           SRL   R9,12
060E 0A19           SLA   R9,1
0610 0A44           SLA   R4,4
0612 09B4           SRL   R4,11
0614 A129           A     @XMLTAB(R9),R4
0616 0CFA'
0618 C114           MOV   *R4,R4
061A 0694           BL    *R4             GO TO SUBROUTINE
061C 10 3           JMP   GEXIT
```

```
0865                    *===============================================
0866                    *       MOVE DATA INSTRUCTION
0867 061E D14E  MOVDAT  MOVB  R14,R5        SET DOUBLE FLAG
0868 0620 0999          SRL   R9,9          IS LENGTH VAR OR IMMEDIATE ?
0869 0622 1804          JOC   LIMM          IMMEDIATE
0870 0624 06A0          BL    @GETMAD       GET VARIABLE LENGTH
     0626 077A'
0871 0628 C200          MOV   R0,R8
0872 062A 1004          JMP   MVSRC
0873 062C D21D  LIMM    MOVB  *R13,R8       LOAD IMMEDIATE LENGTH
0874 062E 0AF4          SLA   R4,15         STALL FOR LONG TIME
0875 0630 D81D          MOVB  *R13,@R8LB
     0632 83F1
0876 0634 04C4  MVSRC   CLR   R4
0877 0636 0AC9          SLA   R9,12
0878 0638 06A0          BL    @MVADDR       GET DEST ADDRESS
     063A 0758'
0879 063C C081          MOV   R1,R2         SAVE DEST ADDRESS
0880 063E B249          AB    R9,R9         WAS DEST VDP REGISTER ?
0881 0640 1702          JNC   DTYPE         NO
0882 0642 0224          AI    R4,3          DESTINATION IS VDP REGISTERS
     0644 0003
0883 0646 C1C4  DTYPE   MOV   R4,R7         SAVE DEST TYPE
0884 0648 04C4          CLR   R4
0885 064A 06A0          BL    @MVADDR       GET SOURCE ADDRESS
     064C 075 '
0886                    *          LOAD BRANCH REGISTERS
0887 064E A104          A     R4,R4         DOUBLE R4
0888 0650 C1A4          MOV   @MSRC(R4),R6 LOAD ADDRESS TO SOPURCE OF MOVE
     0652 0CCE'
0889 0654 A1C7          A     R7,R7         DOUBLE R7
0890 0656 C1E7          MOV   @MDST(R7),R7 LOAD ADDRESS TO DESTINATION
     0658 0CD4'
0891 065A 06A0          BL    @PUTSTK       SAVE PROGRAM COUNTER
     065C 0864'
0892 065E 0456  BSRC    B     *R6           BRANCH TO SOURCE
0893                    *          LOAD ONE BYTE FROM THE SOURCE OF THE MOVE
0894 0660 D2F1  CPUSRC  MOVB  *R1+,R11      LOAD BYTE FROM CPU ADDR SPACE
0895 0662 0457          B     *R7
0896 0664 D7E0  VDPSRC  MOVB  @R1LB,*R15    LOAD VDP WITH ADDRESS
     0666  3E3
0 97 066  D7C1          MOVB  R1,*R15
0 98 066A 0581          INC   R1            INCREMENT SOURCE ADDRESS
0899 066C D2EF          MOVB  @VDPRD(R15),R11 LOAD BYTE FROM VDP
     066E FBFE
0900 0670 0457          B     *R7
0901 0672 DB41  GRMSRC  MOVB  R1,@GRMWA(R13)    LOAD GROM WITH ADDRESS
     0674 0402
0902 0676 DB60          MOV   @R1LB,@GRMWA(R13)
     067  83E3
     067A 0402
0903 067C 05 1          INC   R1            INCREMENT SOURCE ADDRESS
0904 067E D2DD          MOVB  *R13,R11      LOAD BYTE FROM GROM
0905 0680 0457          B     *R7           BRANCH TO DESTINATION
0906                    *          DESTINATIONS FOR THE MOVE INSTRUCTION
0907 0682 DC8B  CPUDST  MOVB  R11,*R2+      STORE BYTE IN CPU ADDR SPACE
0908 0684 1022          JMP   TESTLP
0909 0686       GRMDST
0910 0686 DB42          MOVB  R2,@GRMWA(R13)    LOAD GROM WITH ADDRESS
     0688 0402
```

```
0911 068A DB60          MOVB @R2LB,@GRMWA(R13)
     068C 83E5
     068E 0402
 912 0690 0582          INC  R2             INCREMENT DESTINATION ADDRESS
0913 0692 DB4B          MOVB R11,@GRMWD(R13)   STORE BYTE IN GRAM
     0694 0400
0914 0696 1019          JMP  TESTLP
0915 0698 93A0  REGDST  CB   @R2LB,R14
     069A 83E5
0916 069C 1607          JNE  REGADJ
0917 069E 23A0          COC  @BIT12,R14     TEST VDP MEMORY TYPE
     06A0 0012'
0918 06A2 1602          JNE  REG100
0919 06A4 026B          ORI  R11,>8000      SET 16K MEMORY SELECT
     06A6 8000
0920 06A8 D80B  REG100  MOVB R11,@SAVVDP    SAVE VDP REGISTER
     06AA 83D4
0921 06AC D7CB  REGADJ  MOVB R11,*R15       STORE BYTE IN VDP REGISTER
0922 06AE 0262          ORI  R2,>80         SET FLAG FOR VDP REGISTER LOAD
     06B0 0080
0923 06B2 D7E0          MOVB @R2LB,*R15     STORE REGISTER NUMBER
     06B4 83E5
0924 06B6 0582          INC  R2             INCREMENT REGISTER NUMBER
0925 06B8 1008          JMP  TESTLP
0926 06BA       VDPDST
0927 06BA D7E0          MOVB @R2LB,*R15     LOAD VDP WITH DEST ADDRESS
     06BC 83E5
0928 06BE 0262          ORI  R2,WRVDP       SET FLAG TO WRITE TO VDP
     06C0 4000
0929 06C2 D7C2          MOVB R2,*R15
0930 06C4 0582          INC  R2             INCREMENT DESTINATION ADDRESS
0931 06C6 DBCB          MOVB R11,@VDPWD(R15) STORE BYTE IN VDP
     06C8 FFFE
0932            *          TEST END OF LOOP FOR THE MOVE INSTRUCTION
0933 06CA 0608  TESTLP  DEC  R8             DECREMENT NUMBER TO MOVE
0934 06CC 15C8          JGT  BSRC           LOOP UNTIL NUMBER IS ZERO
0935 06CE 0460  TSTRTN  B    @RETNC         RESTORE PROGRAM COUNTER
     06D0 083E'
```

```
0937                *=============================================================
0938                *************************************************************
0939                *                                                    4/14/78
0940                *            REVISED FROM COINC5 FOR GS REV E   8/09/75
0941                * COINCIDENCE ROUTINE FOR INSERTION INTO REL4 INTERPRETE
0942                * UPON ENTRANCE TO THIS ROUTINE AT LABEL 'COINC' THE
0943                *  REGISTERS ARE ASSUMED TO BE SET UP:
0944                *MSBY  R2 = Y2 IN MSBY AND X2 IN LSBY;
0945                *MSBY  R0 = Y1 IN MSBY AND X1 IN LSBY;
0946                *
0947                * IT IS ALSO ASSUMED THAT THE GROM'S INTERNAL ADDRESS IS S
0948                * UP PREPARED TO READ (FOLLOWING THE COINC INSTRUCTION   ):
0949                *      - A ONE BYTE GRANULARITY VALUE, FOLLOWED BY:
0950                *      - A TWO BYTE ADDRESS POINTING TO THE COINCIDENCE
0951                *           TABLE.  THE TABLE IS ASSUMED TO RESIDE IN GROM.
0952                *           AND HAVE THE FOLLOWING FORMAT:
0953                *
0954                *           BYTE 0- TV = VERTICAL BIT SIZE OF TABLE LESS 1.
0955                *           BYTE 1- TH = HORIZ.   BIT SIZE OF TABLE LESS 1.
0956                *           BYTE 2- V1 = VERTICAL DOT SIZE OF OBJECT 1/2**
0957                *           BYTE 3- H1 = HORIZ.   DOT SIZE OF OBJECT 1/2**
0958                *           BYTES 4 ON - THE BIT TABLE ITSELF;  THE BITS A
0959                *                ARRANGED SUCH THAT THE FIRST (TH+1) BITS
0960                *                REPRESENT BOOLEAN COINCIDENCE VALUES
0961                *                CORRESPONDING TO A DELTA Y (Y1-Y2) OF
0962                *                -V1 THRU -V1+TV AND DELTA X (X1-X2) VALUE
0963                *                -H1 THROUGH -H1+TH.
0964                *
0965 06D2 C200     COINC  MOV  R0,R8
0966 06D4 C0C8            MOV  R8,R3          FIRST GET DELTA Y AND DELTA X
0967 06D6 70C2            SB   R2,R3          R3 = Y1-Y2 = DELTA Y.
0968 06D8 06C8            SWPB R8             GET X1 IN MSBY.
0969 06DA 06C2            SWPB R2             GET X2 IN MSBY.
0970 06DC 7202            SB   R2,R8          R8 X1-X2 = DELTA X.
0971 06DE D01D            MOVB *R13,R0         SET RESOLUTION AND TABLE POINT
0972 06E0 0980            SRL  R0,8           R0 = GRAN.
0973 06E2 D15D            MOVB *R13,R5
0974 06E4 06C5            SWPB R5
0975 06E6 D15D            MOVB *R13,R5
0976 06E8 06C5            SWPB R5             R5 = TABLE PTR.
0977 06EA 06A0            BL   @PUTSTK           SAVE GROM PC.
     06EC 0864'
0978                *
0979                * NOW GET TV,TH,V1,H1 OUT OF THE FIRST 4 BYTES OF TABLE.
0980                *
0981 06EE DB45            MOVB R5,@GRMWA(R13)   PUT OUT TABLE PTR. LSBY.
     06F0 0402
0982 06F2 06C5            SWP  R5
0983 06F4 DB45            MOV  R5, GRMWA(R13)   PUT OUT TABLE PTR. MSBY.
     06F6 0402
0984 06F8 06C5            SWPB R5
09 5 06FA D09D            MOVB *R13,R2          R2 = TV (MSBY).
0986 06FC 1000            NOP
0987 06FE D05D            MOVB *R13,R1          R1 = TH (MSBY).
0988 0700 1000            NOP
0989 0702 D19D            MOVB *R13,R6          R6 = V1 (MSBY).
0990 0704 1000            NOP
0991 0706 D1DD            MOVB *R13,R7          R7 = H1 (MSBY).
0992                *
0993                * NOW ON WITH THE SHOW.  THE REGISTERS ARE NOW SET UP AS:
```

```
0994                    *
0995                    *       R0 = GRANULARITY;
0996                    *MSBY   R1 = TH = COINCIDENCE TABLE HORIZONTAL SIZE - 1.
0997                    *MSBY   R2 = TV = COINCIDENCE TABLE VERTICAL   SIZE - 1.
0998                    *MSBY   R3 = Y1 - Y2 = DELTA Y;
0999                    *MSBY   R8 = X1 - X2 = DELTA X;
1000                    *       R5 = POINTER TO COINCIDENCE TABLE IN GROM.
1001                    *MSBY   R6 = V1 = VERTICAL SIZE OF OBJECT ONE IN DOTS.
1002                    *MSBY   R7 = H1 = HORIZ.   SIZE OF OBJECT ONE IN DOTS.
1003                    *       R13= GRMRD.
1004                    *
1005 0708 C000                  MOV   R0,R0         IF GRANULARITY IS 0, DON'T SHIFT
1006 070A 1302                  JEQ   DNTSHF              BECAUSE 9900 SHIFT BY 0 IS FU
1007 070C 0803                  SRA   R3,R0       DIVIDE DELTA Y BY (2** GRAN).
1008 070E 0808                  SRA   R8,R0       DIVIDE DELTA X BY (2** GRAN).
1009 0710 B207  DNTSHF          AB    R7,R8       R8 = B = H1 + DELTA X.
1010 0712 111E                  JLT   NOCOIN
1011 0714 B0C6                  AB    R6,R3       R3 = A = V1 + DELTA Y.
1012 0716 111C                  JLT   NOCOIN
1013 0718 9083                  CB    R3,R2           A::TV
1014 071A 151A                  JGT   NOCOIN
1015 071C 9048                  CB    R8,R1           B::TH
1016 071E 1518                  JGT   NOCOIN        RANGE TEST PASSED?
1017 0720 0981                  SRL   R1,8          NOW COMPUTE TABLE INDEX.
1018 0722 0581                  INC   R1          R1 = TH +1.
1019 0724 0983                  SRL   R3,8        R3 = A.
1020 0726 3843                  MPY   R3,R1       R2 = A * (TH + 1).
1021 0728 0988                  SRL   R8,8        R8 = B.
1022 072A A088                  A     R8,R2       R2 = INDEX. COMPUTE TABLE AND BIT POSI
1023 072C C002                  MOV   R2,R0       R0 = INDEX ALSO.
1024 072E 0242                  ANDI  R2,>FFF8    R2 = ROUNDED DOWN TO LOWER MULT OF 8.
     0730 FFF8
1025 0732 6002                  S     R2,R0       R0 = BIT DISPLACEMENT (0 = LEFTMOST).
1026 0734 0832                  SRA   R2,3        R2 = BYTE INDEX INTO TABLE.
1027 0736 A085                  A     R5,R2       R2 = ACTUAL ADDRESS OF BYTE.
1028 0738 8C32                  C     *R2+,*R2+   INCREMENT PTR BY 4 FOR 4 BYTE HEADER
1029 073A DB42                  MOVB  R2,@GRMWA(R13)    PULL PROPER BYTE FROM GROM
     073C 0402
1030 073E 0580                  INC   R0
1031 0740 DB60                  MOVB  @R2LB,@GRMWA(R13)
     0742 83E5
     0744 0402
1032 0746 0202                  LI    R2,>2000
     0748 2000
1033 074A D0DD                  MOVB  *R13,R3     R3 = THE BYTE FROM THE TABLE.
1034 074C 0A03                  SLA   R3,R0       GET PROPER BIT INTO THE STATUS CARRY.
1035 074E 1801                  JOC   YUP         IF BIT IS 0, NO COINCIDENCE.
1036 0750 04C2  NOCOIN          CLR   R2                NO, WE HAVE NO COINCIDENCE
1037 0752 D802  YUP             MOVB  R2,@STATUS        YES, WE HAVE COINCIDENCE
     0754 837C
1038 0756 10BB  SOUT9           JMP   TSTRTN
```

```
1040                    *=======================================================
1041 075S               SNAIL
1042                    *=======================================================
1043                    *          COMPUTE ADDRESS FOR MOVE INSTRUCTION
1044 0758 B249  MVADDR  AB    R9,R9          IS ADDRESS IN GROM ?
1045 075A 180F          JOC   GETMAD         YES
1046 075C D0DD  INGROM  MOVB  *R13,R3        LOAD GROM ADDRESS
1047 075E 0584          INC   R4             SET GROM FLAG
1048 0760 D81D          MOVB  *R13,@R3LB
     0762 83E7
1049 0764 C30B          MOV   R11,R12
1050 0766 B249          AB    R9,R9          IS GROM ADDRESS INDEXED ?
1051 0768 1704          JNC   NOGNDX         NO
1052 076A D05D          MOVB  *R13,R1        INDEX IS ONLY ONE BYTE
1053 076C 06A0          BL    @MADDA         GET INDEX VALUE
     076E 077E'
1054 0770 A0C0          A     R0,R3          ADD INDEX TO ADDRESS
1055 0772 C043  NOGNDX  MOV   R3,R1          RETURN ADDRESS IN R1
1056 0774 0919          SRL   R9,1
1057 0776 045C          B     *R12           RETURN
1058                    *=======================================================
1059                    *          GLOBAL ADDRESS DECODE:
1060                    *               R1 = ADDRESS OF VARIABLE
1061                    *               R0 = VALUE OF VARIABLE
1062                    *               SINGLE BYTE VALUES ARE SIGN EXTENDED
1063                    *               IF ADDRESS IS IN VDP, THEN THE MSBYTE OF R
1064                    *                 IS SET TO >80
1065 0778 04C5  GETMAB  CLR   R5             BYTE FLAG
1066 077A D05D  GETMAD  MOVB  *R13,R1        GET NEXT BYTE FROM GAME ROM
1067 077C 111E          JLT   MLONG          LONG ADDRESS
1068 077E 0981  MADDA   SRL   R1,8           SINGLE BYTE ADDRESS
1069 0780 0221  MADD    AI    R1,PAD         ADD CPU RAM ADDRESS
     0782 8300
1070 0784 0281          CI    R1,PAD+>7D     CHARACTER BUFFER ?
     0786 837D
1071 0788 160F          JNE   MADC
1072                    *          LOAD CB FROM VDP MEMORY
1073 078A 04CA          CLR   R10            SET FLAG TO READ FROM VDP
1074 078C C18B          MOV   R11,R6         SAVE RETURN ADDRESS
1075 078E 06A0          BL    @GETDAD        SET VPD RAM ADDRESS
     0790 0884'
1076 0792 C2C6          MOV   R6,R11         RESTORE RETURN ADDRESS
1077 0794 D02F          MOVB  @VDPRD(R15),R0 LOAD CB FROM DISPLAY
     0796 FBFE
1078 0798 23A0          COC   @IN2,R14       IS IT MULTICOLOR MODE??????????
     079A 0072'
1079 079C 1603          JN    MADE           NO, WHO CARES???
1080 079E 1701          JNC   $+4            WHICH NYBBLE????
1081 07A0 0A40          SLA   R0,4           LOW NYBBLE
1082 07A2 0940          SRL   R0,4           HIGH NYBBLE
1083 07A4 D800  MADE    MOVB  R0,@CHRBUF     STORE CHARACTER
     07A6 837D
1084 07A8 D011  MADC    MOVB  *R1,R0         LOAD VALUE OF VARIABLE
1085 07AA D145  MADC2   MOVB  R5,R5          SINGLE OR DOUBLE INSTRUCTION ?
1086 07AC 1602          JNE   MDOUB          DOUBLE
1087 07AE 0880  MSIGN   SRA   R0,8           SIGN EXTEND BYTE VALUE
1088 07B0 045B          RT
1089 07B2 D821  MDOUB   MOVB  @1(R1),@R0LB   GET SECOND HALF OF DOUBLE
     07B4 0001
     07B6 83E1
```

```
1090 07B8 045B          RT
1091 07BA D81D  MLONG   MOVB *R13,@R1LB    GET SECOND HALF OF ADDRESS
     07BC 83E3
 ,92 07BE C281          MOV  R1,R10        SAVE CONTROL BITS
1093 07C0 0241          ANDI R1,>FFF       DELETE CONTROL BITS
     07C2 0FFF
1094 07C4 0281          CI   R1,>F00       LOOK FOR EXTENDED ADDRESS
     07C6 0F00
1095 07C8 1103          JLT  MLONG1        NO, JUST A REGULAR ADDRESS
1096 07CA 0A81          SLA  R1,8          EXTENDED ADDRESS. MAKE ROOM
1097 07CC D81D          MOVB *R13,@R1LB    GET ADDRESS
     07CE 83E3
1098 07D0 0A2A  MLONG1  SLA  R10,2         BASE ADDRESS NEEDED ?
1099 07D2 1708          JNC  MVDP          NO
1100 07D4 D19D          MOVB *R13,R6       GET ADDRESS OF BASE ADDRESS
1101 07D6 0986          SRL  R6,8
1102 07D8 D026          MOVB @PAD(R6),R0   LOAD BASE ADDRESS
     07DA 8300
1103 07DC D826          MOVB @PAD+1(R6),@R0LB
     07DE 8301
     07E0 83E1
1104 07E2 A040          A    R0,R1
1105 07E4 0A1A  MVDP    SLA  R10,1         VDP OR 9985 RAM SPACE ?
1106 07E6 1715          JNC  MCPU          9985 RAM SPACE
1107 07E8 05C4          INCT R4            SET VDP INDICATOR
1108 07EA 0A1A          SLA  R10,1         INDIRECT ADDRESS ?
1109 07EC 1706          JNC  MVDPA         NO
 110 07EE D021          MOVB @PAD(R1),R0   LOAD INDIRECT ADDRESS
     07F0 8300
1111 07F2 D821          MOVB @PAD+1(R1),@R0LB
     07F4 8301
     07F6 83E1
1112 07F8 C040          MOV  R0,R1
1113 07FA D7E0  MVDPA   MOVB @R1LB,*R15    LOAD ADDRESS INTO VDP
     07FC 83E3
1114 07FE D7C1          MOVB R1,*R15
1115 0800 0A80          SLA  R0,8          NO OP
1116 0802 D02F          MOVB @VDPRD(R15),R0 RECALL CONTENTS OF VDP RAM
     0804 FBFE
1117 0806 D145          MOVB R5,R5         SINGLE OR DOUBLE INSTRUCTION ?
1118 0 08 13D2          JEQ  MSIGN         SINGLE
1119 0 0A D82F          MOVB @VDPRD(R15),@R0LB RECALL LSBYTE FROM VDP RAM
     080C FBFE
     080E 83E1
1120 0810 045B          RT
1121 0812 0A1A  MCPU    SLA  R10,1         INDIRECT ADDRESS ?
1122 0814 17B5          JNC  MADD          NO
1123 0 16 0281          CI   R1,>7C        INDIRECT ON STATUS BYTE ?
     0 1  007C
1124 0 1A 1605          JNE  MADB          NO
1125 0 1C D060          MOVB @STKDAT,R1    GET TOP STACK ADDRESS
     081E 8372
1126 0820 780E          SB   R14,@STKDAT   R14=01XX
     0822 8372
1127 0824 10AC          JMP  MADDA
1128 0826 D061  MADB    MOVB @PAD(R1),R1   LOAD INDIRECT ADDRESS
     0828 8300
1129 082A 10A9          JMP  MADDA
1130                    *==========================================================*
```

```
1132 082C 06A0  RGBA   BL   @GETSTK       RECALL GROM BASE ADDRESS
     082E 0842'
1133 0830 C364         MOV  @PAD(R4),R13
     0832 8300
1134 0834 DB44         MOVB R4,@GRMWD(R13)      SYNCHRONIZE YOUR GROMS
     0836 0400
1135 0838 5820  RETURN SZCB @BIT2,@STATUS        RESET CONDITION BIT
     083A 011B'
     083C 837C
1136 083E 020B  RETNC  LI   R11,NEXT
     0840 0070'
1137 0842 D120  GETSTK MOVB @STKADD,R4   LOAD ADDRESS OF SUBROUTINE STACK
     0844 8373
1138 0846 0984         SRL  R4,8          MOVE TO LOW ORDER BYTE
1139 0848 0660  GTSTK  DECT @STKADD       NEW VALUE OF STACK POINTER
     084A 8373
1140 084C DB64  GTSTK1 MOVB @PAD(R4),@GRMWA(R13)  LOAD RETURN ADDRESS
     084E 8300
     0850 0402
1141 0852 DB64         MOVB @PAD+1(R4),@GRMWA(R13)
     0854 8301
     0856 0402
1142 0858 045B         RT
1143              *
1144              *              PUSH PROGRAM COUNTER IN K ONTO STACK
1145 085A D19D  CALL   MOVB *R13,R6       GET BRANCH ADDRESS FROM GROM
1146 085C 020B         LI   R11,LDKADD    SET RETURN POINTER
     085E 0060'
1147 0860 D81D         MOVB *R13,@R6LB
     0862 83ED
1148 0864 05E0  PUTSTK INCT @STKADD       NEW VALUE OF STACK POINTER
     0866 8373
1149 0868 D120         MOVB @STKADD,R4    LOAD ADDRESS OF STACK
     086A 8373
1150 086C 0984         SRL  R4,8          MOVE TO LOW ORDER BYTE
1151 086E D92D         MOVB @GRMRA(R13),@PAD(R4)    SAVE ADDRESS ON STACK
     0870 0002
     0872 8300
1152 0874 D92D         MOVB @GRMRA(R13),@PAD+1(R4)
     0876 0002
     0878 8301
1153 087A 0624         DEC  @PAD(R4)
     087C 8300
1154 087E 045B         RT
1155              *===============================================================
1156              *                LOAD XPTR AND RETURN CHRBUF
1157 0880 020A  GETDA  LI   R10,WRVDP     LOAD VDP WRITE FLAG
     0882 4000
1158 0884 D1E0  GETDAD MOVB @XPT,R7       LOAD VALUE OF XPT
     0886 837F
1159 0888 23A0         COC  @IN2,R14      TEST IF MULTICOLOR MODE
     088A 0072'
1160 088C 130E         JEQ  MCMDA
1161 088E 0A37         SLA  R7,3          MASK OUT TRUE XPT VALUE
1162 0890 0987         SRL  R7,8          MOVE TO LOW ORDER BYTE
1163 0892 D1E0         MOVB @YPT,R7       LOAD VALUE OF YPT
     0894 837E
1164 0896 0937         SRL  R7,3          MOVE TO CORRECT POSITION
1165 0898 A1CA         A    R10,R7        READ OR WRITE FLAG
1166 089A D7E0  SETVDP MOVB @R7LB,*R15    LOAD CHARACTER BUFFER
```

```
1212 0900 0300  REMOTE LIMI 0
     0902 0000
1213 0904 02E0         LWPI WORKSP          USE REGULAR REGISTERS
     0906 83E0
1214 0908 04CC         CLR  R12             LOAD CRU ADDRESS
1215 090A 23A0         COC  @H20,R14        IS TIMER FLAG ON?
     090C 0032'
1216 090E 1602         JNE  TIM1            NO, SEE WHAT IS INTERRUPTING ME
1217 0910 0460         B    @TIMER          HANDLE TIMER INTERRUPT
     0912 0000
1218 0914 1F02  TIM1   TB   2               VDP INTERRUPT??
1219 0916 1619         JNE  VDPINT          YES
1220              *
1221              *      PERIPHERAL ROM INTERRUPT ROUTINES MAY CHANGE ALL
1222              *      REGISTERS EXCEPT R0,R11-R15.
1223              *      R11 HAS THE RETRUN ADDRESS FOR THE INTERRUPT ROUTIN
1224              *      R12 IS LOADED FOR THE CRU SPACE OF THE PERIPHERAL
1225              *      R13 HAS THE CURRENT GROM SLOT ADDRESS
1226              *      R14 HAS A >01XX WHERE THE XX IS USED BY THE INTERPR
1227              *      R15 HAS THE ADDRESS OF THE VDP WRITE ADDRESS ADDRES
1228              *      INTERRUPT ROUTINE SHOULD DO A RETURN WHEN DONE
1229              *
1230 0918 020C         LI   R12,>0F00       TABLE OF CRU BITS TO SELECT ROM
     091A 0F00
1231 091C 1D01         SBO  1               TURN OFF EXTERNAL INTERRUPT
1232 091E 1E00  ILOOP  SBZ  0               TURN OFF LAST ROM
1233 0920 022C         AI   R12,>0100       NEXT ROM
     0922 0100
1234 0924 028C         CI   R12,>2000       FINISHED?
     0926 2000
1235 0928 130E         JEQ  EMERG2          END OF ROM CRU BITS
1236 092A 1D00         SBO  0               LOAD CRU BITS TO SELECT ROM
1237 092C 9820         CB   @H4000,@HAA     VALID ROM?
     092E 4000
     0930 000D'
1238 0932 16F5         JNE  ILOOP           NO
1239 0934 C0A0         MOV  @H400C,R2       GET INTERRUPT ROUTINE ADDRESS
     0936 400C
1240 0938 13F2  LOOP1  JEQ  ILOOP           NO ROUTINE, GO TO NEXT ROM
1241 093A C002         MOV  R2,R0           SAVE POINTER TO NEXT ROUTINE
1242 093C C0A2         MOV  @2(R2),R2       GET ENTRY ADDRESS
     093E 0002
1243 0940 0692         BL   *R2             JUMP TO INTERRUPT ROUTINE
1244 0942 C090         MOV  *R0,R2          NEXT ROUTINE'S ADDRESS
1245 0944 10F9         JMP  LOOP1           GO TO NEXT ROUTINE
1246 0946 0460  EMERG2 B    @EM RGE
     0948 0AB8'
1247              *=============================================================
1248              *      V D P   I N T E R R U P T   H A N D L E R
1249              *
1250 094A 1D02  VDPINT SBO  2               RESET VDP INTERRUPT ON 9901
1251 094C D060         MOVB @INTFLG,R1
     094E 83C2
1252 0950 0A11         SLA  R1,1
1253 0952 1702         JNC  TSTMOT
1254 0954 0460         B    @VSTAT
     0956 0AB4'
1255              *=============================================================
1256              *          AUTOMATIC SPRITE MOTION      (INTERRUPT ROUTINE)
1257 0958 0A11  TSTMOT SLA  R1,1
```

```
1258 095A 1846          JOC   TSTSND
1259 095C D320          MOVB  @MOTION,R12
     095E 837A
 260 0960 1343          JEQ   TSTSND
 261 0962 098C          SRL   R12,8
1262 0964 0202          LI    R2,VDPRD+VDPWA
     0966 8800
1263 0968 0203          LI    R3,VDPWD+VDPWA
     096A 8C00
1264 096C 0208          LI    R8,RSMOT
     096E 0780
1265 0970 D7E0  MLOOP   MOVB  @R8LB,*R15   LOAD ADDRESS TO READ SML
     0972 83F1
1266 0974 D7C8          MOVB  R8,*R15
1267 0976 04C4          CLR   R4
1268 0978 D112          MOVB  *R2,R4        READ DELTA Y
1269 097A 04C6          CLR   R6
1270 097C D192          MOVB  *R2,R6        READ DELTA X
1271 097E 0844          SRA R4,4            MOVE RIGHT ONE DIGIT
1272 0980 D152          MOVB  *R2,R5        READ TEMP Y
1273 0982 0845          SRA   R5,4          MOVE RIGHT ONE DIGIT
1274 0984 A144          A     R4,R5
1275 0986 D1D2          MOVB  *R2,R7
1276 0988 0846          SRA   R6,4          MOVE RIGHT ONE DIGIT
1277 098A 0847          SRA   R7,4          MOVE RIGHT ONE DIGIT
1278 098C A1C6          A     R6,R7
1279 098E 0228          AI    R8,-QSAML     CHANGE ADDRESS TO SAL
     0990 FB80
 280 0992 D7E0          MOVB  @R8LB,*R15    LOAD ADDRESS TO READ SAL
     0994 83F1
1281 0996 D7C8          MOVB  R8,*R15
1282 0998 04C4          CLR   R4
1283 099A D112          MOVB  *R2,R4        READ Y POSITION
1284 099C A105          A     R5,R4         ADD Y MOVEMENT TO POSITION
1285                    ********************************************************
1286                    ***     To fix the sprite flicker problem,
1287                    ***     add following code back (it has been taken out
1288                    ***     of the /4A shipped 03/16/81) 07/29/81.
12 9 099E-0284          CI    R4,6*VDELTA+255
     09A0 C0FF
1290 09A2 1209          JLE   ONSCRN
1291 09A4-0284          CI    R4,7*VDELTA
     09A6 E000
1292 09A8 1306          JH    ONSCRN
1293 09AA C145          MOV   R5,R5
1294 09AC 1502          JGT   $+6
1295 09AE-0224          AI    R4,6*VDELTA
     09B0 C000
1296 09B2 0224          AI    R4,VDELTA
     09B4 2000
1297                    ********************************************************
1298 09B6 04C6  ONSCRN  CLR   R6
1299 09B8 D192          MOVB  *R2,R6
 300 09BA A187          A     R7,R6
 301 09BC 0268          ORI   R8,WRVDP
     09BE 4000
1302 09C0 D7E0          MOVB  @R8LB,*R15    LOAD ADDRESS TO WRITE SAL
     09C2 83F1
1303 09C4 D7C8          MOVB  R8,*R15
1304 09C6 04C4          MOVB  R4,*R2
```

```
1305 09C8 0228          AI    R8,QSAML+2
     09CA 0482
1306 09CC D4C6          MOVB  R6,*R3
1307 09CE 06C5          SWPB  R5
1308 09D0 D7E0          MOVB  @R8LB,*R15    LOAD ADDRESS TO WRITE SML
     09D2 83F1
1309 09D4 D7C8          MOVB  R8,*R15
1310 09D6 0945          SRL   R5,4
1311 09D8 D4C5          MOVB  R5,*R3
1312 09DA 06C7          SWPB  R7
1313 09DC 0947          SRL   R7,4
1314 09DE D4C7          MOVB  R7,*R3
1315 09E0 0228          AI    R ,2-WRVDP
     09E2 C002
1316 09E4 060C          DEC   R12
1317 09E6 15C4          JGT   MLOOP
1318             *===================================================================
1319             *      AUTO-SOUND FEATURE              (INTERRUPT ROUTINE)
1320 09E8 0A11   TSTSND SLA   R1,1
1321 09EA 183D          JOC   TSTQT
1322 09EC D0A0          MOVB  @STFLGS,R2
     09EE 83CE
1323 09F0 133A          JEQ   TSTQT
1324 09F2 780E          SB    R14,@STFLGS         R14=01XX
     09F4 83CE
1325 09F6 1637          JNE   TSTQT
1326 09F8 C0E0          MOV   @SNDADD,R3
     09FA 83CC
1327 09FC C14E          MOV   R14,R5
1328 09FE 0915          SRL   R5,1
1329 0A00 180A          JOC   SNDRAM
1330 0A02 06A0          BL    @PUTSTK
     0A04 0864'
1331 0A06 0205          LI    R5,GRMWA
     0A08 0402
1332 0A0A A14D          A     R13,R5
1333 0A0C D543          MOVB  R3,*R5
1334 0A0E D560          MOVB  @R3LB,*R5
     0A10 83E7
1335 0A12 C18D          MOV   R13,R6
1336 0A14 1007          JMP   USND
1337 0A16 0205   SNDRAM LI    R5,VDPWA
     0A18 8C02
1338 0A1A D560          MOVB  @R3LB,*R5
     0A1C 83E7
1339 0A1E D543          MOVB  R3,*R5
1340 0A20 0206          LI    R6,VDPRD+VDPWA
     0A22 8 00
1341 0A24 D216   USND   MOVB  *R6,R8
1342 0A26 130F          JEQ   NEWADD
1343 0A28 9220          CB    @HFF,R8       DO I SWITCH SOURCE TYPE ?
     0A2A 0A9C'
1344 0A2C 130A          JEQ   NEW1          YES
1345 0A2E 0988          SRL   R8,8
1346 0A30 A0C8          A     R8,R3
1347 0A32 D816   SLOOP  MOVB  *R6,@SGCADR   SEND BYTE TO SOUND CHIP
     0A34 8400
1348 0A36 0608          DEC   R8
1349 0A38 16FC          JNE   SLOOP         LOOP UNTIL COUNT IS ZERO
1350 0A3A 05C3          INCT  R3
```

```
1351 0A3C D096          MOVB *R6,R2
1352 0A3E 1309          JEQ  XSOUND
1353 0A40 1009          JMP  SNDXIT
 354 0A42 2BA0  NEW1    XOR  @C1,R14        CHANGE VDP TO GROM OR GROM TO VDP
     0A44 0378'
1355 0A46 D0D6  NEWADD  MOVB *R6,R3         GET HIGH BYTE OF NEW ADDRESS
1356 0A48 0202          LI   R2,>100
     0A4A 0100
1357 0A4C D816          MOVB *R6,@R3LB      GET LOW BYTE OF NEW ADDRESS
     0A4E 83E7
1358 0A50 1001          JMP  SNDXIT
1359 0A52 7082  XSOUND  SB   R2,R2          TURN OFF SOUND PROCESSING
1360 0A54 C803  SNDXIT  MOV  R3,@SNDADD
     0A56 83CC
1361 0A58 D802          MOVB R2,@STFLGS
     0A5A 83CE
1362 0A5C 0285          CI   R5,VDPWA
     0A5E 8C02
1363 0A60 1302          JEQ  TSTQT
1364 0A62 06A0          BL   @GETSTK
     0A64 0842'
1365                    *==========================================================*
1366 0A66 0A11  TSTQT   SLA  R1,1
1367 0A68 180D          JOC  VSTAT
1368 0A6A 020C          LI   R12,>24        LOAD CONSOLE ADDRESS IN CRU BASE
     0A6C 0024
1369 0A6E 30E0          LDCR @H00,3         LOAD FOR ROW ZERO
     0A70 0012'
 370 0A72 0B7C          SRC  R12,7          KILL TIME
1371 0A74 020C          LI   R12,6          CRU ADDRESS FOR CHARACTER LINES
     0A76 0006
1372 0A78 3605          STCR R5,            RECALL COLUMN INFORMATION
1373 0A7A 2560          CZC  @FNCEQ,R5      IS IT A  "FNCT'N  ="  ?
     0A7C 004C'
1374 0A7E 1602          JN   VSTAT
1375 0A80 0420          BLWP @0             RESET VECTOR TRAP LOCATION
     0A 2 0000
1376                    *==========================================================*
1377 0A 4 D 2F  VSTAT   MOVB @VDPSTA(R15),@VDPST LOAD VDP STATUS IN STATUS BL
     0A 6 FC00
     0A 8 837B
137  0A A 02E0          LWPI INTWSP         USE INTERRUPT WORK SPACE
     0A C 83C0
1379 0A8E 05CB          INCT R11            INCREMENT SCREEN TIMER
1380 0A90 160B          JNE  TIMIN1         NO TIMEOUT
13 1 0A92 D30A  TIMIN   MOVB R10,R12
13 2 0A94 098C          SRL  R12,8
13 3 0A96 026C          ORI  R12,>8160      TURN OFF VDP
     0A9  8160
13 4      0A9C' HFF     EQU  $+2
1385 0A9A 024C          ANDI R12,>FFBF
     0A9C FFBF
1386 0A9E D820          MOVB @AR12LB,@VDPWA
     0AA0 83D9
     0AA2 8C02
1387 0AA4 D80C          MOVB R12,@VDPWA
     0AA6 8C02
1388 0AA8 02E0  TIMIN1  LWPI WORKSP
     0AAA 83E0
1389 0AAC 380E          AB   R14,@TIME      INCREMENT TIME IN STATUS BLOCK
```

```
     0AAE 8379
1390 0AB0 C320          MOV   @INTPTR,R12
     0AB2 83C4
1391 0AB4 1301          JEQ   EMERGE
1392 0AB6 069C          BL    *R12
1393 0AB8 04C8  EMERGE  CLR   R8              CLEAR REGISTER FOR BASIC
1394 0ABA 02E0          LWPI  INTWSP          RETURN TO CALLING ROUTINE
     0ABC 83C0
1395 0ABE 0380          RTWP                  FROM ALTERNATE WORKSPACE
```

```
1397                    *
1398                    *       SEARCH ROM CROM CROM FOR DSR OR LINK
1399                    *
1400 0AC0 04C1  SROM    CLR   R1              VERSION FOUND OF DSR ETC
1401 0AC2 C320          MOV   @CRULST,R12     SEARCH ROM FOR ROUTINE
     0AC4 83D0
1402 0AC6 1618          JNE   SGO             IF <> 0 THEN CONTINUE SEARCH
1403 0AC8 020C          LI    R12,>0F00       START OVER AGAIN
     0ACA 0F00
1404 0ACC C30C  NOROM   MOV   R12,R12         ANYTHING TO TURN OFF?
1405 0ACE 1301          JEQ   NOOFF           NO
1406 0AD0 1E00          SBZ   0               YES, TURN IT OFF
1407 0AD2 022C  NOOFF   AI    R12,>0100       NEXT ROM'S TURN ON
     0AD4 0100
1408 0AD6 04E0          CLR   @CRULST         CLEAR JUST IN CASE WE'RE FINISHED
     0AD8 83D0
1409 0ADA 028C          CI    R12,>2000       AT THE END
     0ADC 2000
1410 0ADE 1320          JEQ   NOSET           NO MORE ROMS TO TURN ON
1411 0AE0 C80C          MOV   R12,@CRULST     SAVE ADDRESS OF NEXT CRU
     0AE2 83D0
1412 0AE4 1D00          SBO   0               TURN ON ROM
1413 0AE6 0202          LI    R2,>4000        START AT BEGINNING
     0AE8 4000
1414 0AEA 9812  SGO1    CB    *R2,@HAA        IS IT A VALID ROM?
     0AEC 000D'
1415 0AEE 16EE          JNE   NOROM           NO
1416 0AF0 B820  SGO1A   AB    @TYPE,@R2LB     GO TO FIRST POINTER
     0AF2 836D
     0AF4  3 5
1417 0AF6 1003          JMP SGO2
1418 0AF8 C0A0  SGO     MOV   @SADDR,R2       CONTINUE WHERE WE LEFT OFF
     0AFA 83D2
1419 0AFC 1D00          SBO   0               TURN ROM BACK ON
1420 0AFE C092  SGO2    MOV   *R2,R2          IS ADDRESS A ZERO
1421 0B00 13 5          JEQ   NOROM           YES, NO PROGRAM TO LOOK AT
1422 0B02 C802          MOV   R2,@SADDR       REM MBER WHERE WE GO NEXT
     0B04  3D2
1423 0B06 05C2          INCT  R2              GO TO ENTRY POINT
1424 0B08 C272          MOV   *R2+,R9         GET ENTRY ADDRESS
1425 0B0A 06A0          BL    @NAME           SE  IF NAME MATCHES
     0B0C 0BE '
1426 0B0E 10F4          JMP   SGO             NO MATCH, TRY NEXT PROGRAM
1427 0B10 05 1          INC   R1              NEXT VERSION FOUND
1428 0B12 0699          BL    *R9             MATCH, CALL SUBROUTINE
1429 0B14 10F1          JMP   . GO            NOT RIGHT VERSION
1430 0B16 1 00           BZ   0               TURN OFF ROM
1431 0B18 1001          JMP   NOGR2           S T STATUS AND GO TO GPL
1432 0B1A 04D   NOGR1   CLR   *R              NO MOR  TO DO
1433 0B1C 06A0  NOGR2   BL     G TSTK         RESTORE GROM ADDRESS
     0B1E 0 42'
1434 0B20 0460  NOS T   B     @RESET          INDICATE NO MORE
     0B22 006A'
 435                    *=============================================================*
1436 0B24 0207  SGROM   LI    R7,SADDR
     0B26 83D2
1437 0B28 0208          LI    R8,CRULST
     0B2A 83D0
1438 0B2C 06A0          BL    @PUTSTK         SAVE GROM ADDRESS
     0B2E 0864'
```

```
1439 0B30 C057  SGROMA MOV  *R7,R1        START WHERE WE LEFT OFF
1440 0B32 C098         MOV  *R8,R2        IS IT A RESTART?
1441 0B34 1604         JNE  SGROM3        NO
1442 0B36 0202         LI   R2,GRMRD      START OF GROMS
     0B38 9800
1443 0B3A 0201  SGROM1 LI   R1,>E000      START OF GROM
     0B3C E000
1444 0B3E 2460  SGROM3 CZC  @H1FFF,R1     IS IT A NEW GROM OR A CONTINUATIOI
     0B40 0128'
1445 0B42 160E         JNE  SGROM2
1446 0B44 C602         MOV  R2,*R8        SAVE GROM BASE ADDRESS
1447 0B46 D881         MOVB R1,@GRMWA(R2) LOAD ADDRESS
     0B48 0402
1448 0B4A D8A0         MOVB @R1LB,@GRMWA(R2)
     0B4C 83E3
     0B4E 0402
1449 0B50 B820         AB   @TYPE,@R1LB   LOOK FOR PROGRAM ADDRESS
     0B52 836D
     0B54 83E3
1450 0B56 D801         MOVB R1,@SAVEG     SAVE GROM ADDRESS OF HEADER
     0B58 83CB
1451 0B5A 9812         CB   *R2,@HAA      VALID GROM?
     0B5C C00D'
1452 0B5E 1632         JN   NOGR          NO GROM HERE
1453      0B61' H81    EQU  $+1
1454 0B60 D881  SGROM2 MOVB R1,@GRMWA(R2) LOOK FOR PROGRAM
     0B62 0402
1455 0B64 D8A0         MOVB @R1LB,@GRMWA(R2)
     0B66 83E3
     0B68 0402
1456 0B6A 0A4A         SLA  R10,4         STALL
1457 0B6C D0D2         MOVB *R2,R3        READ PROGRAM ADDRESS
1458 0B6E 1000         NOP
1459 0B70 D812         MOVB *R2,@R3LB
     0B72 83E7
1460 0B74 C5C3         MOV  R3,*R7        GET NEXT HEADER'S ADDRESS
1461 0B76 1326         JEQ  NOGR          IF ZERO, GO TO NEXT GROM
1462 0B78 05C3         INCT R3            GO TO PROGRAM ENTRY ADDRESS
1463 0B7A D 83         MOVB R3,@GRMWA(R2) GO TO PROGRAM ENTRY ADDRESS
     0B7C 0402
1464 0B7E D8A0         MOVB @R3LB,@GRMWA(R2)
     0B 0  3E7
     0B82 0402
1465 0B84 1000         NOP
1466 0B86 D252         MOVB *R2,R9        ENTRY ADDRESS
1467 0B88 0A4A         SLA  R10,4         STALL
1468 0B A D 12         MOV  *R2,@R9LB
     0B C  3F3
1469 0B8E 06A0         BL   @NAME         S   IF NAME MATCHES
     0B90 0BE8'
1470 0B92 10CE         JMP  SGROMA        NO LOOK FOR NEXT PROGRAM
1471 0B94 B820         AB   @H2,@STKDAT   FOUSP NAME SO PUSH IT
     0B96 0030'
     0B98 8372
1472 0B9A B80E         AB   R14,@TEMP2    INCREASE PROGRAM COUNT
     0B9C 836C
1473 0B9E D120         MOVB @STKDAT,R4
     0BA0 8372
1474 0BA2 0984         SRL  R4,8
1475 0BA4 0643         DECT R3            POINT BACK TO START OF HEADER
```

```
1476 0BA6 9820          CB   @TYPE,@H06   IS IT A USER PROGRAM  LOOKUP?
     0BA8 836D
     0BAA 0C04'
 477 0BAC 1601          JNE  SGROM4       YES
.478 0BAE C243          MOV  R3,R9        PUSH HEADER ADDRESS FOR USER PROGRAI
1479 0BB0 D909  SGROM4  MOVB R9,@PAD(R4)  NO, PUSH ENTRY ADDRESS
     0BB2 8300
1480 0BB4 D920          MOVB @R9LB,@PAD+1(R4)
     0BB6 83F3
     0BB8 8301
1481 0BBA C342          MOV  R2,R13       GO TO THAT LIBRARY
1482 0BBC 06A0  SGSET   BL   @GETSTK      RESTORE GROM ADDRESS
     0BBE 0842'
1483 0BC0 0460          B    @SET         SET STATUS AND RETURN
     0BC2 00CE'
1484 0BC4 04C1  NOGR    CLR  R1           GET ADDRESS OF GROM HEADER
1485 0BC6 D060          MOVB @SAVEG,R1
     0BC8 83CB
1486 0BCA 0221          AI   R1,->2000    NEXT GROM DOWN
     0BCC E000
1487 0BCE C5C1          MOV  R1,*R7       SAVE ADDRESS OF WHERE WE'RE AT
1488 0BD0 0281          CI   R1,>E000     FINISHED?
     0BD2 E000
1489 0BD4 16B4          JNE  SGROM3       NO, CHECK THIS GROM
1490 0BD6 8CB2          C    *R2+,*R2+    INCREMENT GROM MAPPED ADDRESS BY FOU
1491 0BD8 C602          MOV  R2,*R8       SAVE THE NEW MAP ADDRESS
1492 0BDA 0282          CI   R2,GRMRD+>40 AT END OF LIBRARY?
     0BDC 9840
 493 0BDE 139D          JEQ  NOGR1        YES
.494 0BE0 D160          MOVB @SCLEN,R5    ARE WE LOOKING FOR A MENU?
     0BE2 8355
1495 0BE4 16AA          JNE  SGROM1       YES SO DO ONLY ONE SLOT
1496 0BE6 109A          JMP  NOGR2        NO CONTINUE SEARCH
1497 0BE8 D160  NAME    MOVB @SCLEN,R5    GET LENGTH AS COUNTER
     0BEA 8355
1498 0BEC 130D          JEQ  NAM 2A       ZERO LENGTH, DON'T DO MATCH
1499 0BEE 9485          CB   R5,*R2       DOES LENGTH MATCH?
1500 0BF0 160C          JNE  NAME3        NO
1501 0BF2 09 5          SRL  R5,          MOVE TO RIGHT PLACE
1502 0BF4 0206          LI   R6,FAC
     0BF6 834A
1503 0BF8 02 2  NAME1   CI   R2,GRMRD     IS IT GROM?
     0BFA 9 00
1504 0BFC 1401          JHE  NAME2        YES, DON'T INCREMENT ADDRESS
1505 0BF  0582          INC  R2
1506 0C00 9436  NAME2   CB   *R6+,*R2     IS NAME THE SAME?
1507 0C02 1603          JN   NAM 3        NO
150  0C04 0605  H06     DEC  R5           MOR  TO LOOK AT ?
1509 0C06 16F           JN   NAM 1        YE
1510 0C0  05CB  NAME2A  INCT R11          R TURN , NAME FOUND
1511 0C0A 045B  NAME3   RT                R TURN
1512      4020  OPBR    EQU  >4020        ADDRESS TO HANDLE OPERR IN DSR
1513 0C0C 06A0  OPERR   BL   @SETUP
     0C0E 0C28'
 514 0C10 0460          B    @OPBR
     0C12 4020
1515      401C  BPBR    EQU  >401C        ADDRESS TO HANDLE BP IN DSR
1516 0C14 06A0  BP      BL   @SETUP
     0C16 0C28'
1517 0C18 0460          B    @BPBR
```

```
      0C1A 401C
 1518 0C1C 02E0  PRCXOP LWPI DSRWSP
      0C1E 2800
 1519 0C20 06A0         BL   @SETUP
      0C22 0C28'
 1520 0C24 0460         B    @ALBRK
      0C26 4028
 1521 0C28 020C  SETUP  LI   R12,>1B00    SET UP CRU
      0C2A 1B00
 1522 0C2C 1D00         SBO  0
 1523 0C2E 045B         RT
 1524                   *===============================================
 1525 0C36              RORG >0C36
 1526                   *               BRANCH TABLES
 1527 0C36 0270' ITAB   DATA MISCLN,MOVDAT,BRESET,BSET
      0C38 061E'
      0C3A 011A'
      0C3C 010E'
 1528 0C3E 0838' MSCTAB DATA RETURN,RETNC,RANDNO,KEYSCN,BKORND,BLONG,CALL,A
      0C40 083E'
      0C42 027A'
      0C44 02AE'
      0C46 029E'
      0C48 0104'
      0C4A 085A'
      0C4C 05A2'
 1529 0C4E 04DE'         DATA FORMAT,TSTST,TSTST,ENTRY,TSTST,TSTST,PARSEG,XM
      0C50 00F4'
      0C52 00F4'
      0C54 0024'
      0C56 00F4'
      0C5  00F4'
      0C5A 0000
      0C5C 0608'
 1530 0C5E 0000         DATA CONTG,EXECG,RTNG,RGBA,OPERR,OPERR,OPERR,OPERR
      0C60 0000
      0C62 0000
      0C64 0 2C'
      0C66 0C0C'
      0C6  0C0C'
      0C6A 0C0C'
      0C6C 0C0C'
 1531 0C6E 0C0C'         DATA OPERR,OPERR,OPERR,OPERR,OPERR,OPERR,OPERR,BP
      0C70 0C0C'
      0C72 0C0C'
      0C74 0C0C'
      0C76 0C0C'
      0C7  0C0C'
      0C7A 0C0C'
      0C7C 0C14'
 1532       0BFE' ATAB   EQU  $->80
 1533 0C7E 0136'         DATA ABS,NEG,INV,CLR,FETCH,CASE,PUSH,IFZ
      0C 0 013A'
      0C82 0140'
      0C 4 013E'
      0C 6 0144'
      0C   0162'
      0C8A 016E'
      0C8C 00EA'
 1534 0C8E 0186'         DATA INC,DEC,INCT,DECT,OPERR,OPERR,OPERR,OPERR
```

```
      0C90 0188'
      0C92 0184'
      0C94 0182'
      0C96 0C0C'
      0C98 0C0C'
      0C9A 0C0C'
      0C9C 0C0C'
1535       0C4  ' BTAB   EQU  $->50
1536 0C9E 0188'         DATA ADD,MINUS,MUL,DIV,AND,OR,XOR,STORE
      0CA0 0186'
      0CA2 01CE'
      0CA4 01EA'
      0CA6 0190'
      0CA8 0196'
      0CAA 019A'
      0CAC 019E'
1537 0CAE 01A2'         DATA EXCH,HIGH,HIGHEQ,GREATR,GREQ,EQUAL,IFAND
      0CB0 00D6'
      0CB2 00DA'
      0CB4 00DE'
      0CB6 00CC'
      0CB8 00EC'
      0CBA 00E2'
1538 0CBC 01B0'         DATA SRA,SLL,SRL,SRC,COINC,CPERR,INOUT,DGBA,CPERR
      0CBE 01B4'
      0CC0 01B8'
      0CC2 01C2'
      0CC4 06D2'
      0CC6 0C0C'
      0CC8 05C8'
      0CCA 004E'
      0CCC 0C0C'
1539 0CCE 0660' MSRC   DATA CPUSRC,GRMSRC,VDPSRC
      0CD0 0672'
      0CD2 0664'
1540 0CD4 06 2' MDST   DATA CPUDST,GRMDST,VDPDST,REGDST
      0CD6 0686'
      0CD  06BA'
      0CDA 069  '
1541 0CDC 050A' FBTAB  DATA STRACR,STRDWN,RPTACR,RPTDWN
      0CD  0508'
      0CE0 0504'
      0CE2 0502'
1542 0CE4 0534'         DATA SKPACR,SKPDWN,RPTBLK,SPECL
      0CE6 0532'
      0CE8 053A'
      0CEA 056C'
1543 0C C 05D6' IOTAB  DATA SOUND,SOUND,CRUIN,CRUOUT
      0CE  05D6'
      0CF0 05E  '
      0CF2 05EA'
1544 0CF4 0000          DATA WRITE,READ,VERIFY
      0CF6 0000
      0CF8 0000
1545 0CFA 0000  XMLTAB DATA FLTTAB,XTAB,>2000,>3FC0
      0CFC 0000
      0CFE 2000
      0D00 3FC0
1546 0D02 3FE0          DATA >3FE0,>4010,>4030,>6010
      0D04 4010
```

```
      0D06 4030
      0D08 6010
 1547 0D0A 6030    .      DATA >6030,>7000,>8000,>A000
      0D0C 7000
      0D0E 8000
      0D10 A000
 1548      0D14' HC000    EQU  $+2
 1549 0D12 B000           DATA >B000,>C000,>D000,PAD
      0D14 C000
      0D16 D000
      0D18 8300
 1550           *=============================================================
 1551      0024'          END  ENTRY
NO ERRORS,    0003 WARNINGS
```

```
$          0D1A'          0148  0149  0163  0164  0168  0169  0170  0200  0203
                          0277  0282  0466  0486  0564  0594  0624  0783  0793
                          1080  1294  1384  1453  1532  1535  1548
  JPA      00BC'  0234    0223
ABOPS      0086'  0213    0206
ABS        0136'  0290    1533
ADD        0188'  0326    1536
ADDB       0222'  0394    0327
ALBRK      4028   0132    1520
ALL        05A2'  0808    1528
ALLOOP     05AC'  0811    0813
ALOCK      043E'  0647    0642
AND        0190'  0330    1536
AOPS       0080'  0229    0221
AR12LB     83D9   0108    1386
ATAB       0BFE'  1532    0232
BBJOY      16E0   0114    0502
BIASI      0592'  0799    0790
BIASV      058E'  0798    0789
BIT12      0012'  0150    0917
BIT2       011B'  0277    0199  0243  1135
BIT4       0013'  0149    0391
BKGR1      02AA'  0450    0444
BKGRND     029E'  0447    1528
BLONG      0104'  0268    152
BOPS       00C0'  0235    022
BP         0C14'  1516    1531
 BR        401C   1515    1517
 AD        0122'  0281    0274
BRESET     011A'  0278    1527
BSET       010E'  0272    1527
BSRC       065E'  0892    0934
BTAB       0C4E'  1535    0237
C1         0378'  0564    1354
C1000   D  0036'  0173    0029  0702  0709
C2      D  0072'  0201    0029  0474
CALL       085A'  1145    1528
CASE       0162'  0308    0313  1533
CHKBEX     04DC'  0710    0703
CHKBRK     04B2'  0697    0160
CHRBUF     837D   0065    1083  1191
CHTAB   R          0031
CLOCK      000C'  0142
CLR        013E'  0294    1533
CNTLP      036C'  0554    0556
COINC      06D2'  0965    153
COLBAS     0006   0113    0494  0521
CONTG   R  0C5 '  0031    1530
CPUDST     06 2'  0907    1540
CPUSRC     0660'  0894    1539
CRUIN      05   '  083     1543
CRULST     83D0   0086    1401  1408  1411  1437
 RUOUT     05EA'  0839    1543
  KB2      03DC'  0608    0606
DBNC1      83C9   0081    0586
DBNC2      83CA   0082    0587
DBNCE      83C8   0080    0579  0582  0583  0592  0597  0598  0608
DEBOUN     03AA'  0592    0510  0572
DEC        0188'  0325    1534
DECT       0180'  0321    1534
```

```
DGBA      004E'   0191  1539
DGBADD    005C'   0195  0172
DIV       01EA'   0371  1536
DIV1      01FC'   0379  0377
DIV2      0216'   0389  0386
DIV3      020A'   0384  0382
DIVC      0206'   0383  0380
DNTSHF    0710'   1009  1006
DSRWSP    2800    0127  1518
DTYPE     0646'   0883  0681
EMERG2    0946'   1246  1235
EMERGE    0AB8'   1393  1246  1391
ENDBLK    0550'   0770  0785
ENDFMT    05B4'   0815  0771
ENTRY   D 0024'   0163  0028  0138  1529  1551
EQUAL     00EC'   0256  1537
EXCH      01A2'   0339  1537
EXECG   R 0C60'   0031  1530
EXSCN     0492'   0675  0671
FAC       834A    0050  1502
FBTAB     0CDC'   1541  0724
FETCH     0144'   0297  1533
FILLST    0228'   0395  0329  0332  0334  0336
FINISH    054A'   0766  0777
FIRE      02F1'   0486  0500
FLTTAB  R 0CFA'   0030  1545
FNCEQ     004C'   0186  1373
FORMAT    04DE'   0714  1529
FUNCTN    04E6'   0717  0750  0764  0768  0780  0796  0800
GETDA     0880'   1157  0421  0716  0748  0795
GETDAD    0884'   1158  1075
GETMAB    0778'   1065  0798  0802
GETMAD    077A'   1066  0216  0234  0870  1045
GETRTN    08A2'   1169  1189
GETSTK  D 0842'   1137  0028  0669  1132  1364  1433  1482
GOTK5     0360'   0548  0539
GOTKEY    0358'   0538
GREATR    00DE'   0249  1537
GREQ      00CC'   0242  1537
GRMDST    0686'   0909  1540
GRMRA   D 0002    0041  0027  0284  1151  1152
GRMRD   D 9800    0039  0027  0040  0041  0042  0165  1442  1492  1503
GRMSRC    0672'   0901  1539
GRMWA   D 0402    0040  0027  0197  0198  0503  0504  0629  0630  0778  077
                        0901  0902  0910  0911  0981  0983  1029  1031  114
                        1141  1331  1447  1448  1454  1455  1463  1464
GRMWD   D 0400    0042  0027  0913  1134
GTSTK     0848'   1139  0300
GTSTK1    0 4C'   1140  0305
H00       0012'   014   0698  1369
H020      03B4'   0594  0647
H03       0074'   0203  0705
H06       0C04'   150   1476
H0D       0025'   0164  0658
H0F       02CA'   0466  0660
H1FFF     0128'   0282  1444
H2        0030'   0168  1471
H20     D 0032'   0170  0027  1215
H4000     4000    0125  1237
H400C     400C    0124  1239
```

```
H5F     0587'   0793  0654
H80     0470'   0665  0662  0667
H81     0B61'   1453  0674
        000D'   0143  1237  1414  1451
HC0C0   0D14'   1548
HE4     056F'   0783
HFF     0A9C'   1384  1343
HIGH    00D6'   0245  1537
HIGHEQ  00DA'   0247  1537
IFAND   00E2'   0251  1537
IFZ     00EA'   0255  1533
ILOOP   091E'   1232  1238  1240
IMMOPS  0030'   0169  0222
IN2     0072'   0200  0201  1078  1159
INC     0186'   0323  1534
INCT    0184'   0322  1534
INGROM  075C'   1046
INOUT   05C8'   0825  1538
INTFLG  83C2    0076  1251
INTPTR  83C4    0077  1390
INTROM  0000    0126
INTTIM  0008'   0140
INTWSP  83C0    0070  0139  0140  1378  1394
INV     0140'   0295  1533
IOTAB   0CEC'   1543  082
ITAB    0C36'   1527  0209
JOYX    8377    0059  0508
JOYY    8376    0058  0507
  CAN   02F4'   0492  0469  0472
JSCAN1  0310'   0501  0499
KBDFLG  83C6    0078  0478  0636
KCNTRL  1790    0118  0617
KEYBRD  8375    0057  0668
KEYSCN  02AE'   0457  1528
KEYTAB  1700    0115  0627
KFNCTN  1760    0117  0620
KILTIM  049C'   0681  0682
KL10MS  0498'   0680  0581  0596
KSCAN   02B2'   045   0147
KSCAN1  02EC'   0480  0465
KSCAN2  032E'   0514  0482
KSHIFT  1730    0116  0623
KSPLIT  17C0    0119  0614
LDKADD  0060'   0197  0271  0287  1146
LIMM    062C'   0873  0869
LINE    0405'   0624  0493
LOOP    050A'   0738  0725  0732
LOOP1   093 '   1240  1245
LOOPA   050E'   0740
LOOPB   050C'   0739  0805
LOOPV   059E'   0804  0803
MADB    0826'   112   1124
MADC    07A8'   1084  0383  1071
  DC2   07AA'   1085  0227
 ADD    0780'   1069  1122
MADDA   077E'   1068  1053  1127  1129
MADE    07A4'   1083  1079
MALPC   2838    0130
MAP4    046C'   0664  0661
MAPIT   040E'   0628  0616  0619  0622  0626
```

```
MCMDA     08AA'   1173  1160
MCPU      0812'   1121  1106
MDOUB     07B2'   1089  1086
MDST      0CD4'   1540  0890
MINUS     0186'   0324  1536
MISCLN    0270'   0426  1527
MLONG     07BA'   1091  1067
MLONG1    07D0'   1098  1095
MLOOP     0970'   1265  1317
MODIFY    03E6'   0613  0593
MOTION    837A    0062  1259
MOVDAT    061E'   0867  1527
MSCTAB    0C3E'   152   0428
MSIGN     07AE'   1087  1118
MSRC      0CCE'   1539  0888
MUL       01CE'   0359  1536
MULO      01D6'   0363  0361
MUL1      01E0'   0367  0365  0390  0392
MUL2      01E6'   0369  0344
MVADDR    0758'   1044  0878  0885
MVDP      07E4'   1105  1099
MVDPA     07FA'   1113  0381  1109
MVSRC     0634'   0876  0872
NAME      0BE8'   1497  1425  1469
NAME1     0BF8'   1503  1509
NAME2     0C00'   1506  1504
NAME2A    0C08'   1510  149
NAME3     0C0A'   1511  1500  1507
NEG       013A'   0292  1533
NEW1      0A42'   1354  1344
NEWADD    0A46'   1355  1342
NEWMOD    03E2'   0609  0600  0603
NEXT    D 0070'   0202  0026  0244  0258  0318  0400  0450  0457  0815  083
                        1136
NOBR      0116'   0275  0280
NOCOIN    0750'   1036  1010  1012  1014  1016
NOGNDX    0772'   1055  1051
NOGR      0BC4'   1484  1452  1461
NOGR1     0B1A'   1432  1493
NOGR2     0B1C'   1433  1431  1496
NOKEY     03 2'   0576  0475  0653  0655
NOKEY2    0394'   0582  0580
NOOFF     0AD2'   1407  1405
NOROM     0ACC'   1404  1415  1421
NOSET     0B20'   1434  1410
NOTONE    0354'   0531  052
NXTROW    037A'   0569  0534  0541  0549  0563
OLDCHR    0478'   066   0585  0588  0635  0657  0659  0663  0666
OLDMOD    83C7    0079  0577  0609  0613
ONSCRN    09B6'   1298  1290  1292
OPBR      4020    1512  1514
OPERR     0C0C'   1513  1530  1530  1530  1530  1531  1531  1531  1531  15
                        1531  1531  1534  1534  1534  1534  1538  1538
OR        0196'   0333  1536
PAD       8300    0049  0050  0051  0052  0053  0054  0055  0056  0057  0C
                        0059  0060  0061  0062  0063  0064  0065  0066  00'
                        0070  0071  0075  0076  0077  0078  0079  0080  008
                        0082  0083  0084  0085  0086  0087  0088  0089  00
                        0096  0097  0098  0099  0100  0101  0102  0103  010
                        0105  0106  0107  0108  0109  0193  0202  0317  041
```

ABEL	VALUE	DEFN	REFERENCES
			0763 0776 0794 0849 1069 1070 1102 1103 1110
			1111 1128 1133 1140 1141 1151 1152 1153 1479
			1480 1549
RSEG R	0C5A'	0030	1529
LAYER	8374	0056	0462 0476
RCXOP	0C1C'	1518	0179
ROCSS	007A'	0206	0156
JSH	016E'	0314	1533
JTSTK D	0864'	1148	0028 0191 0192 0298 0459 0891 0977 1330 1438
EXIT	05E4'	0836	0851 0863
BAML	0480	0122	1279 1305
)	0000		0171 0196 0219 0225 0231 0238 0251 0252 0259
			0260 0261 0263 0321 0322 0324 0328 0330 0331
			0333 0335 0337 0340 0345 0347 0352 0357 0363
			0373 0384 0467 0470 0480 0533 0578 0579 0582
			0583 0586 0587 0633 0647 0654 0658 0660 0662
			0667 0668 0688 0690 0692 0692 0825 0833 0871
			0965 0971 0972 1005 1005 1007 1008 1023 1025
			1030 1034 1054 1077 1081 1082 1083 1084 1087
			1102 1104 1110 1112 1115 1116 1173 1174 1177
			1178 1179 1183 1190 1194 1197 1200 1201 1241
			1244
)LB	83E1	0096	0394 1089 1103 1111 1119
	0001		0218 0224 0225 0226 0343 0374 0375 0378 0514
			0518 0519 0520 0527 0527 0538 0538 0551 0562
			0562 0565 0569 0604 0607 0608 0614 0617 0620
			0623 0627 0628 0629 0787 0799 0804 0826 0839
			0842 0847 0879 0894 0897 0898 0901 0903 0987
			1015 1017 1018 1020 1052 1055 1066 1068 1069
			1070 10 4 1089 1092 1093 1094 1096 1104 1110
			1111 1112 1114 1123 1125 1128 1128 1251 1252
			1257 1320 1366 1400 1427 1439 1443 1444 1447
			1450 1454 1484 1485 1486 1487 1488
.0	000A		1073 1092 1098 1105 1108 1121 1157 1165 1168
			1381 1456 1467
.1	000B		0395 0396 0400 0420 0457 0458 0675 0687 0815
			0894 0899 0904 0907 0913 0919 0920 0921 0931
			1049 1074 1076 1136 1146 1188 1379 1510
.1LB	83F7	0106	
.2	000C		0460 0492 0494 0517 0521 0601 0602 0605 0640
			0644 0680 0681 0687 0688 0697 0699 0700 0701
			0702 0704 0706 0707 0708 0709 0718 0760 0761
			0766 0773 0839 0840 0840 1049 1057 1214 1230
			1233 1234 1259 1261 1316 1368 1370 1371 1381
			1382 1383 1385 1387 1390 1392 1401 1403 1404
			1404 1407 1409 1411 1521
2LB	83F9	0107	0690
3	000D		0165 0193 0194 0195 0197 0198 0205 0224 0268
			0270 0275 0281 0284 0301 0310 0312 0437 0448
			0503 0504 0507 0508 0629 0630 0633 0717 0738
			0772 0774 0778 0779 0787 0794 0808 0854 0873
			0875 0901 0902 0904 0910 0911 0913 0971 0973
			0975 0981 0983 0985 0987 0989 0991 1029 1031
			1033 1046 1048 1052 1066 1091 1097 1100 1133
			1134 1140 1141 1145 1147 1151 1152 1332 1335
			1481
3LB	83FB	0109	
4	000E		0166 0314 0776 0832 0833 0835 0867 0915 0917
			1078 1126 1159 1215 1324 1327 1354 1389 1472
5	000F		0167 0403 0405 0408 0410 0422 0472 0474 0740

```
                                0811  0896  0897  0899  0921  0923  0927  0929  0931
                                1077  1113  1114  1116  1119  1166  1167  1185  1187
                                1190  1201  1265  1266  1280  1281  1302  1303  130
                                1309  1377
R1LB        83E3      0097      0630  0896  0902  1091  1097  1113  1448  1449  140
R2          0002              0194  0219  0238  0252  0255  0259  0290  0292  029
                              0295  0301  0302  0308  0328  0331  0333  0335  033
                              0339  0340  0345  0347  0351  0351  0352  0356  035
                              0359  0367  0369  0372  0373  0389  0408  0415  041
                              0417  0422  0515  0548  0548  0550  0571  0571  072
                              0731  0731  0751  0803  0825  0827  0827  0828  084
                              0842  0843  0844  0847  0848  0849  0879  0907  091
                              0912  0922  0924  0928  0929  0930  0967  0969  097
                              0985  1013  1022  1023  1024  1025  1026  1027  102
                              1028  1029  1032  1036  1037  1239  1241  1242  124
                              1243  1244  1262  1268  1270  1272  1275  1280  129
                              1322  1351  1356  1359  1359  1361  1413  1414  141
                              1420  1420  1422  1423  1424  1440  1442  1446  144
                              1448  1451  1454  1455  1457  1459  1463  1464  146
                              1468  1481  1490  1490  1491  1492  1499  1503  150
                              1506
R2LB        83E5      0098      0317  0356  0394  0410  0911  0915  0923  0927  103
                              1416
R3          0003              0218  0343  0374  0404  0405  0409  0411  0415  041
                              0418  0495  0509  0509  0551  0552  0553  0554  060
                              0628  0641  0641  0649  0649  0656  0715  0739  079
                              0826  0834  0966  0967  1007  1011  1013  1019  102
                              1033  1034  1046  1054  1055  1263  1304  1306  131
                              1314  1326  1333  1339  1346  1350  1355  1360  1
                              1460  1462  1463  1475  1478
R3LB        83E7      0099      0403  0500  0592  0597  0598  0608  0636  1048  133
                              1338  1357  1459  1464
R4          0004              0193  0195  0207  0208  0209  0213  0217  0256  025
                              0272  0273  0275  0278  0279  0297  0299  0303  030
                              0306  0342  0379  0379  0401  0401  0428  0429  043
                              0434  0440  0442  0496  0497  0498  0501  0502  050
                              0522  0523  0524  0529  0530  0533  0555  0726  072
                              0734  0741  0772  0778  0828  0830  0855  0858  085
                              0860  0861  0861  0862  0874  0876  0882  0883  088
                              0887  0887  0888  1047  1107  1133  1134  1137  113
                              1140  1141  1149  1150  1151  1152  1153  1267  126
                              1271  1274  1282  1283  1284  1289  1291  1295  129
                              1304  1473  1474  1479  1480
R4LB        83E9      0100      0504
R5          0005              0209  0210  0214  0215  0262  0263  0310  0312  032
                              0326  0349  0349  0354  0354  0360  0360  0364  036
                              0371  0376  0376  0385  0385  0406  0406  0413  041
                              0435  0436  0441  0462  0463  0464  0479  0493  050
                              0540  0540  0579  0583  0584  0584  0592  0598  059
                              0599  0615  0615  0634  0634  0719  0723  0724  072
                              0727  0774  0779  0808  0811  0867  0973  0974  097
                              0976  0981  0982  0983  0984  1027  1065  1085  108
                              1117  1117  1272  1273  1274  1284  1293  1293  130
                              1310  1311  1327  1328  1331  1332  1333  1334  1
                              1338  1339  1362  1372  1373  1494  1497  1499  1
                              1508
R5LB        83EB      0103      0443
R6          0006              0196  0197  0268  0284  0285  0286  0297  0306  03
                              0316  0317  0420  0423  0437  0438  0439  0442  04
```

			0577	0595	0670	0738	0739	0740	0761	0762	0763
			0773	0775	0776	0804	0888	0892	0989	1011	1074
			1076	1100	1101	1102	1103	1145	1202	1269	1270
			1276	1278	1298	1299	1300	1306	1335	1340	1341
			1347	1351	1355	1357	1502	1506			
R6LB		83ED	0101	0198	0270	1147					
R7		0007		0447	0516	0529	0609	0613	0618	0621	0625	0741
				0742	0744	0746	0755	0810	0812	0816	0817	0818
				0819	0820	0883	0889	0889	0890	0890	0895	0900
				0905	0991	1009	1158	1161	1162	1163	1164	1165
				1167	1168	1170	1180	1181	1182	1183	1184	1187
				1198	1275	1277	1278	1300	1312	1313	1314	1436
				1439	1460	1487						
R7LB		83EF	0104	0448	1166	1185						
R8		0008		0229	0230	0232	0232	0233	0235	0236	0237	0237
				0239	0359	0362	0362	0363	0367	0372	0388	0389
				0717	0719	0720	0721	0722	0749	0753	0755	0756
				0784	0788	0792	0794	0871	0873	0933	0965	0966
				0968	0970	1008	1009	1015	1021	1022	1174	1175
				1176	1179	1180	1186	1191	1192	1195	1200	1264
				1266	1279	1281	1301	1303	1305	1309	1315	1341
				1343	1345	1346	1348	1393	1432	1437	1440	1446
				1491								
R8LB		83F1	0102	0366	0387	0763	0875	1265	1280	1302	1308	
R9		0009		0205	0207	0214	0220	0222	0229	0235	0259	0283
				0286	0339	0366	0369	0384	0387	0426	0427	0428
				0714	0759	0767	0770	0770	0829	0838	0844	0845
				0846	0850	0854	0855	0856	0857	0860	0868	0877
				0880	0880	1044	1044	1050	1050	1056	1424	1428
				1466	1478	1479						
R9LB		83F3	0105	0281	1468	1480						
RAND16		83C0	0075	0434	0436							
RANDNO		027A'	0433	1528								
RANDOM		8378	0060	0443								
READ	R	0CF6'	0030	1544								
REG100		06A8'	0920	0918								
REGADJ		06AC'	0921	0916								
REGDST		0698'	0915	1540								
REMOTE		0900'	1212	0139								
RESET	D	006A'	0199	0028	0242	0246	0248	0250	0254	0265	0276	0309
				1434								
RESTOR		05B8'	0816	0747	0791							
RETNC		083E'	1136	0935	1528							
RETURN		0838'	1135	1528								
RGBA		082C'	1132	1530								
RNGCHK		04A2'	0687	0637	0651	0664						
RNGRT		04B0'	0693	0689	0691							
ROWBAS		0024	0112	0492	0517							
ROWLP		0336'	0517	0570								
RPTACR		0504'	0731	1541								
RPTBLK		053A'	0759	1542								
RPTDWN		0502'	0729	1541								
RSAVE		83D8	0090	0458	0675							
RST		0780	0121	1264								
RSTP5		0442'	0648	0646								
RTN		026A'	0422	1188								
RTNG	R	0C62'	0031	1530								
SADDR		83D2	0087	1418	1422	1436						
SAVEG		83CB	0083	1450	1485							

```
LABEL     VALUE   DEFN  REFERENCES
SCLEN     8355    0051  1494  1497
SCNKEY    '000E'  0147
SET   D   '00CE'  0243  0028  0245  0247  0249  0253  0264  1483
SETUP     '0C28'  1521  1513  1516  1519
SETV      '08A4'  1170  0809
SETVDP    '089A'  1166  0449  1171  1199
SGCADR    8400    0043  1347
SGO       '0AF8'  1418  1402  1426  1429
SGO1      '0AEA'  1414
SGO1A     '0AF0'  1416
SGO2      '0AFE'  1420  1417
SGROM  D  '0B24'  1436  0029
SGROM1    '0B3A'  1443  1495
SGROM2    '0B60'  1454  1445
SGROM3    '0B3E'  1444  1441  1489
SGROM4    '0BB0'  1479  1477
SGROMA    '0B30'  1439  1470
SGSET     '0BBC'  1482
SHIFTQ    2200    0123
SKPACR    '0534'  0755  1542
SKPDWN    '0532'  0753  1542
SLL       '01B4'  0347  1538
SLOOP     '0A32'  1347  1349
SNAIL     '075B'  1041
SNDADD    83CC    0084  0834  1326  1360
SNDRAM    '0A16'  1337  1329
SNDXIT    '0A54'  1360  1353  1358
SOUND     '05D6'  0832  1543  1543
SOUT9     '0756'  1038
SPECL     '056C'  0784  1542
SPKCMD    9400    0044
SPKSTA    9000    0045
SRA       '01B0'  0345  1538
SRC       '01C2'  0354  1538
SRC1      '01CA'  0357  0355
SRL       '01B8'  0349  153
SRL1      '01BE'  0352  0350
SROM   D  '0AC0'  1400  0029
SSFLG     2836    0129
SSTEP     4024    0131
STATUS    837C    0064  0199  0243  0257  0262  0272  0278  0371  0391  039
                        0670  1037  1135
STCPU     '0254'  0413  0402
STFLGS    83CE    0085  0835  1322  1324  1351
STKADD    8373    0055  0718  1137  1139  1148  1149
STKDAT    8372    0054  0314  0315  1125  1126  1471  1473
STORE     '019E'  0337  1536
STRACR    '050A'  0736  1541
STRDWN    '0508'  0734  1541
TEMP2     836C    0052  1472
TESTLP    '06CA'  0933  0908  0914  0925
TEXIT     '0252'  0412  0419
TIM1      '0914'  1218  1216
TIME      8379    0061  1389
TIMER  R  '0912'  0030  1217
TIMIN     '0A92'  1381  0140
TIMIN1    '0AA8'  1388  1380
TIMOUT    83D6    0089  0673
TRAP      '022E'  0400  0291  0293  0296  0307  0338  0346  0348  0353  035
                        0370
```

LABEL		VALUE	DEFN	REFERENCES							
TRAP1		024A'	0410	0407							
TRAP2		025A'	0416	0414							
TRAPA		0232'	0401	0341	0368						
TR[illegible]GN		0078'	0205	0158							
T[illegible]JT		0958'	1257	1253							
TSTQT		0A66'	1366	1321	1323	1325	1363				
TSTRTN		06CE'	0935	1038							
TSTSND		09E8'	1320	1258	1260						
TSTST		00F4'	0259	1529	1529	1529	1529				
TYPE		836D	0053	1416	1449	1476					
UPXY		051A'	0744	0757							
USND		0A24'	1341	1336							
USTRCB		08EC'	1197	1193							
VARBLE		0596'	0802	0786							
VDELTA		2000	0120	1289	1291	1295	1296				
VDPDST		06BA'	0926	1540							
VDPINT		094A'	1250	1219							
VDPRD	D	FBFE	0038	0026	0899	1077	1116	1119	1190	1262	1340
VDPREG		8000	0033	0447							
VDPSRC		0664'	0896	1539							
VDPST		837B	0063	1377							
VDPSTA		FC00	0036	1377							
VDPWA	D	8C02	0035	0026	0036	0037	0038	0167	1262	1263	1337	1340
				1362	1386	1387						
VDPWD	D	FFFE	0037	0026	0408	0410	0422	0740	0811	0931	1201	1263
VERIFY	R	0CF8'	0031	1544							
VSTAT		0A84'	1377	1254	1367	1374					
VSTRCB		08F0'	1198	1196							
WP[illegible]2		0456'	0656	0650							
W[illegible]XB		0444'	0649	0639							
WORKSP		83E0	0071	0138	1213	1388					
WRITE	R	0CF4'	0030	1544							
WRVDP	D	4000	0034	0026	0404	0928	1157	1170	1198	1301	1315
XML		0608'	0854	1529							
XMLTAB		0CFA'	1545	0860							
XOPWSP		280A	0128	0177	0179						
XOR		019A'	0335	1536							
XPT		837F	0067	0820	1158						
XSOUND		0A52'	1359	1352							
XTAB	R	0CFC'	0030	1545							
YPT		837E	0066	0817	1163	1173					
YUP		0752'	1037	1035							
ZEND		0524'	0747	0743	0745						

```
SOURCE ACCESS NAME=      BASC1.TI994A.V080581.SRC.FLTPT
OBJECT ACCESS NAME=      PCD2.AEM.OBJ4A.FLTPT
LISTING ACCESS NAME=     PCD2.AEM.LST4A.FLTPT
ERROR ACCESS NAME=       PCD2.AEM.SRC.ERROR
 TIONS=                  XREF
ACRO LIBRARY PATHNAME=


LINE      KEY   NAME

 0005      A    FPEQ
                     =>BASC1.TI994A.V0 0581.SRC.FPEQ
```

```
                        IDT  'FLTPT'
                        ********************************************************
                        *
                        COPY FPEQ
A0001 0000              DATA1  CSEG
A0002                   ********************************************************
A0003                   *  LOCATIONS USED BY THE INTERPRETER IN HOME COMPUTER
A0004        8300       PAD    EQU  >8300
A0005        0004       I      EQU  4
A0006        0005       J      EQU  5
A0007        0006       K      EQU  6
A000         0007       L      EQU  7
A0009        000        N      EQU  8
A0010                   ********************************************************
A0011                   *      PROCESSOR RAM FIXED LOCATIONS
A0012                   *
A0013        8800       VDPRD  EQU  >8800        ADDRESS TO READ VDP
A0014        8C00       VDPWD  EQU  >8C00        ADDRESS TO WRITE VDP
A0015        834A       FAC    EQU  PAD+>4A      FLOATING ACCUMULATOR
A0016        8354       FDVSR  EQU  FAC+10       DIVISOR STORAGE DURING DIVISION
A0017        8354       FLTNDX EQU  FAC+10       INSTRUCTION INDEX SAVE
A0018        835C       ARG    EQU  FAC+18       FLOATING ARGUMENT
A0019        836C       FLTERR EQU  PAD+>6C      ERROR ADDRESS FOR MATH ROUTINES
A0020        836E       VSPTR  EQU  PAD+>6E      VALUE STACK POINTER
A0021        8376       JOYY   EQU  PAD+>76
A0022        8310       PROA   EQU  PAD+>10      PROCESSOR ROLLOUT AREA
A0023        03C0       VROA   EQU  >3C0         VIDEO RAM ROLLOUT AREA
A0024        0008       CC     EQU  R
A0025        8373       STKADD EQU  PAD+>73
A0026        8375       SIGN   EQU  PAD+>75      TEMPORARY SIGN STORAGE
A0027        8377       JOYX   EQU  PAD+>77
A002         8376       EXP    EQU  PAD+>76      TEMPORARY EXPONENT STORAGE
A0029        837C       STATUS EQU  PAD+>7C      STATUS REGISTER
A0030        83E0       WS     EQU  PAD+>E0      WORKSPACE MNEMONIC
A0031        83EF       LLB    EQU  PAD+>EF      LOWER HALF OF L
A0032                   ********************************************************
A0033                   *      SUBROUTINE TO POP VALUE STACK
A0034                   *
A0035                          R F  GRMWA,GRMRA,WRVDP
A0036                   *
A0037 0000 0205  POPSTK LI     J,-              COUNT R FOR LOOP
      0002 FFF
A003  0004 0206         LI     K,ARG            ADDRESS TO STORE OPERAND
      0006 835C
A0039 0008 D7E0         MOVB   @VSPTR+1,*R15 LOAD VDP ADDRESS
      000A 836F
A0040 000C 0207         LI     L,VDPRD          ADDR SS TO READ VDP
      000    00
A0041 0010 D7 0         MOV     VSPTR,*R15      MSB OF ADDRESS
      0012 836E
A0042 0014 A805         A      J, VSPTR         DECREMENT STACK COUNTER
      0016 836E
A0043 0018 DD97  STKMOV MOVB   *L,*K+           RECALL BYTE FROM VDP
A0044 001A 0585         INC    J                INCR MENT LOOP COUNTER
A0045 001C 16FD         JNE    STKMOV           8 BYTES?
A0046 001E 045B         RT
A0047                   *
A0048 0020 D7E0  GETCH  MOVB   @R6LB,*R15          LOAD VDP ADDRESS
      0022 83ED
A0049 0024 1000         NOP
```

```
A0050 0026 D7C6          MOVB R6,*R15            THE ZERO DISP IS FOR TIMING
A0051 0028 0586          INC  R6                 NEXT ADDRESS
A0052 002A D220          MOVB @VDPRD,CC
      002C 8800
 0053 002E 0988  GETCH1  SRL  CC,8          SHIFT TO LOWER BYTE
A0054 0030 045B          RT                      RETURN
A0055                 *
A0056 0032 DB46  GETCHG MOVB R6,@GRMWA(R13)
      0034 0000
A0057 0036 DB60          MOVB @R6LB,@GRMWA(R13)
      003  3ED
      003A 0034+
A0058 003C 05 6          INC  R6
A0059 003E D21D          MOVB *R13,CC
A0060 0040 10F6          JMP  GETCH1
A0061                 *
A0062 0042               CEND
 0006                 *
 0007                    DEF  FLTTAB,ERRXI1,ROUND,ROUNU,PACKUP,FADD
 0008                    DEF  SADD,FSUB,SSUB,FDIV,SDIV,FMULT,SMULT,FCOMP
 0009                    DEF  SCOMP,SCOMPB,ROUN1
 0010                    DEF  ROUNUP,BIGFLT,OVEXP1,FZERO
 0011                 *
 0012                 ********************************************************************
 0013                 *
 0014       83E5  R2LB   EQU  PAD+>E5
 0015       83ED  R6LB   EQU  PAD+>ED
 0016       0200  DZERR  EQU  >200          DIVIDE BY ZERO ERROR CODE
 0017       0100  OFERR  EQU  >100          OVERFLOW ERROR CODE
  01        0100  HIBYTE EQU  256           MULTIPLIER FOR HIGH BYTE
 0019             *                        CONSTANTS
```

```
0022                *       ENTRY TO FLOATING POINT ROUTINES
0023 0000 0000      FLTTAB  DATA 0,ROUND,ROUNU,STEXIT,OVEXP,OV        GROUP 1
     0002 023A'
     0004 0298'
     0006 028A'
     0008 02A8'
     000A 02B2'
0024 000C 0066'             DATA FADD,FSUB,FMULT,FDIV,FCOMP                GROUP
     000E 0062'
     0010 016E'
     0012 02DA'
     0014 0020'
0025 0016 006A'             DATA SADD,SSUB,SMULT,SDIV,SCOMP                GROUP
     0018 005A'
     001A 0172'
     001C 02DE'
     001E 002C'
```

```
0028                    *       FLOATING COMPARE
0029                    *
0030                    *       COMPARE ARG WITH FAC
0031                    *
 032                    *       CALL:     FLTPT   COMP (SCOMP)
0033 0020 C28B  FCOMP   MOV   R11,R10
0034 0022 0203          LI    R3,STEX01     EXIT FOR GPL COMPARE
     0024 0290'
0035 0026 1007          JMP   FCOMP1
0036                    *
0037                    *        TACK COMPARE ENTRY FOR BASIC
003                     *
0039 0028 C0CB  SCOMP   MOV   R11,R3        DON'T USE STEX01 FOR BASIC
0040 002A 1003          JMP   SCOMP1
0041                    *
0042                    *       STACK COMPARE ENTRY FOR GPL
0043                    *
0044 002C 0203  SCOMP   LI    R3,STEX01     EXIT FOR GPL COMPARE
     002E 0290'
0045 0030 C28B          MOV   R11,R10
0046 0032 06A0  SCOMP1  BL    @POPSTK       STACK COMPARE
     0034 0000+
0047 0036 0207  FCOMP1  LI    R7,ARG
     003  835C
0048 003A 0205          LI    R5,FAC
     003C 834A
0049 003E 8D57          C     *R7,*R5+      COMPARE THE FIRST WORDS
0050 0040 160B          JNE   FCOMRT        DONE COMPARING IF NOT EQUAL
0051 0042 C1B7          MOV   *R7+,R6       SIGN OF NUMBERS
 052 0044 1309          JEQ   FCOMRT        NUMBERS ARE ZERO AND EQUAL
0053 0046 1503          JGT   FCOMO1        BOTH NEGATIVE
0054 0048 C185          MOV   R5,R6
0055 004A C147          MOV   R7,R5
0056 004C C1C6          MOV   R6,R7
0057 004E 8D77  FCOMO1  C     *R7+,*R5+     BOTH POSITIVE,
0058 0050 1603          JN    FCOMRT        CONTINUE COMPARING UNTIL UNEQUAL
0059 0052 8D77          C     *R7+,*R5+     OR  ND OF NUMBER
0060 0054 1601          JN    FCOMRT
0061 0056 8557          C     *R7,*R5       THE LAST ENVELOPE!
0062 0058 0453  FCOMRT  B     *R3            XIT AS SPECIFIED
```

```
0065                    *
0066                    *       FLOATING ADDITION AND SUBTRACTION
0067                    *
0068                    *       THE TOP TWO STACK ELEMENTS ARE POPPED,
0069                    *       ADDED (OR SUBTRACTED), AND THE RESULT
0070                    *       PUSHED ON THE STACK.
0071                    *
0072                    *       CALL:    B      FADD     (SADD)
0073                    *                B      FSUB     (SSUB)
0074                    *
0075                    *       WARNINGS:       WRNOV  OVERFLOW
0076                    *
0077 005A C28B  SSUB    MOV   R11,R10         SAVE RETURN ADDRESS
0078 005C 06A0          BL    @POPSTK         STACK SUBTRACTION
     005E 0000+
0079 0060 C2CA          MOV   R10,R11
0080 0062 0520  FSUB    NEG   @FAC            NEGATE THE RH ARGUMENT
     0064 834A
0081 0066 C28B  FADD    MOV   R11,R10         SAVE RETURN ADDRESS
0082 0068 1003          JMP   FADD1           GO TO PROCESS
0083                    *
0084 006A C28B  SADD    MOV   R11,R10         SAVE RETURN ADDRESS
0085 006C 06A0          BL    @POPSTK
     006E 0000+
0086 0070 C1E0  FADD1   MOV   @ARG,R7         IS ARGUMENT ZERO?
     0072 835C
0087 0074 130A          JEQ   FADD02          YES, NO CHANGE TO FAC
008  0076 C220          MOV   @FAC,R8         IS FAC ZERO?
     007   34A
0089 007A 1609          JN    FADD03          NO, GO ADD FAC, ARG
0090 007C 0201          LI    R1,-8           YES, MOVE ARG TO FAC
     007E FFF8
0091 0080 C 61  FADD01  MOV   @ARG+8(R1),@FAC+8(R1)
     00 2  364
     00 4  352
0092 0086 05C1          INCT  R1
0093 0088 11FB          JLT   FADD01
0094 00 A 0460  FADD02  B     @STEX           XIT TO GLI WITH STATUS
     00 C 028C
0095 008E 29C8  FADD03  XOR   R ,R7           SIGN DIFFERENCE
0096 0090 0760          AB    @FAC            TAKE ABSOLUTE VALUES OF FAC
     0092  34A
0097 0094 0760          AB    @ARG            AND ARG
     0096 835C
009  009  0203          LI    R3,-            NSURE THAT LARGEST NUMBER
     009A FFF
0099                    *                       I  IN FAC
0100 009C 8  3  FADD20  C     @FAC+ (R3),@ARG+8(R3)
     009  8352
     00A0  364
0101 00A2 150E          JGT   FADD05          TRUE INITIALLY
0102 00A4 1103          JLT   FADD21          NEED TO SWAP THIS WORD AND FOLLOW
0103 00A6 05C3          INCT  R3
0104 00A8 16F9          JNE   FADD20          COMPARE ALL 4 WORDS
0105 00AA 100A          JMP   FADD05          FAC = ARG
0106 00AC C023  FADD21  MOV   @ARG+8(R3),R0
     00A  8364
0107 00B0 C8E3          MOV   @FAC+8(R3),ARG+8(R3)
     00B2 8352
     00B4 8364
```

```
 0108 00B6 C8C0          MOV  R0,@FAC+8(R3)
      00B8 8352
 0109 00BA 05C3          INCT R3
 0110 00BC 16F7          JNE  FADD21        CONTINUE THE SWAP
  111 00BE 2A07          XOR  R7,R8
```

```
0113 00C0 04C5  FADD05 CLR  R5             HANDY ZERO
0114 00C2 04E0         CLR  @FAC+8         CLEAR GUARD DIGITS FOR FAC
     00C4 8352
0115 00C6 04E0         CLR  @ARG+8         AND ARG
     00C8 8364
0116 00CA D808         MOVB R8,@SIGN       SAVE RESULT SIGN
     00CC 8375
0117 00CE 04C6         CLR  R6             CLEAR HIGH BYTE OF EXP DIFF
0118 00D0 D820         MOVB @FAC,@WS+13    FAC EXP TO R8(R6)
     00D2 834A
     00D4 83ED
0119 00D6 C806         MOV  R6,@EXP        USE FAC EXP AS RESULT EXP
     00D8 8376
0120 00DA D805         MOVB R5,@FAC        CLEAR HIGH BYTE OF FAC TO CHECK
     00DC 834A
0121               *                       FOR OVERFLOW
0122 00DE 7820         SB   @ARG,@WS+13    SUBTRACT SMALLER EXPONENT
     00E0 835C
     00E2 83ED
0123 00E4 0286         CI   R6,7           SMALLER NUMBER TOO SMALL TO
     00E6 0007
0124               *                       AFFECT THE SUM?
0125 00E8 1540         JGT  FADD15         YES, RETURN WITH LARGER NUMBER
0126               *                       IN FAC.
```

```
0128 00EA C006          MOV   R6,R0         EXPONENT DIFFERENCE
0129 00EC 0208          LI    R8,1*HIBYTE   1 FOR BYTE OPERATIONS
     00EE 0100
0130 00F0 0209          LI    R9,100*HIBYTE 100 FOR BYTE OPERATIONS
     00F2 6400
0131 00F4 0205          LI    R5,FAC+9      POINTER TO LOW BYTE OF BIG NUM
     00F6 8353
0132 00F8 0206          LI    R6,ARG+9      AND LOW BYTE OF SMALL NUMBER
     00FA 8365
0133 00FC 6180          S     R0,R6         ADJ ARG POINTER TO ALIGN RADIX
0134 00FE C100          MOV   R0,R4         ADD/SUBTRACT LOOP COUNT IS
0135 0100 0224          AI    R4,-9         BYTES LEFT IN SMALL NUMBER
     0102 FFF7
0136 0104 C047          MOV   R7,R1         TWO NUMBERS HAVE SAME SIGN?
0137 0106 1120          JLT   FADD11        NO, SUBTRACT THEM
0138              *                         YES, ADD THEM
```

```
0140 0108 B556  FADD06 AB   *R6,*R5       ADD A BYTE OF SMALL TO LARGER
0141 010A 9255         CB   *R5,R9        IF SUM LT RADIX
0142 010C 1A03         JL   FADD07          THEN CONTINUE TO NEXT BYTE
0143 010E 7549         SB   R9,*R5        SUBTRACT RADIX FROM THIS BYTE
0144 0110 B948         AB   R8,@-1(R5)      AND ADD CARRY TO NEXT BYTE
     0112 FFFF         .
0145 0114 0605  FADD07 DEC  R5            TO NEXT HIGHER BIG NUMBER BYTE
0146 0116 0606         DEC  R6             AND NEXT HIGHER SMALL NUMBER
0147 0118 0584         INC  R4            IF NOT ALL SIGNIF BYTES OF SMALL
0148 011A 11F6         JLT  FADD06         ADDED, THEN CONTINUE
0149 011C 1002         JMP  FADD09        ELSE PROPAGATE CARRY
0150 011E 0605  FADD08 DEC  R5            WAS LARGER, POINT TO NEXT BYTE
0151 0120 B548         AB   R ,*R5        ADD CARRY TO NEXT BYTE
0152 0122 7549  FADD09 SB   R9,*R5        SUBTRACT RADIX FROM NEXT BYTE
0153 0124 15FC         JGT  FADD08        DONE IF REACHED ONE BYTE
0154            *                         SMALLER THAN RADIX
0155 0126 13FB         JEQ  FADD08        CONTINUE IF RESULT = RADIX
0156 0128 B549         AB   R9,*R5        RADIX SUBTRACTED ONCE TOO OFTEN
0157 012A D060         MOVB @FAC,R1       CARRY OUT OF HIGH ORDER RESULT?
     012C 834A
0158 012E 1308         JEQ  FADD10        NO, ROUND RESULT
0159 0130 05A0         INC  @EXP          YES, INCREMENT EXPONENT
     0132 8376
0160 0134 0201         LI   R1,FAC+8
     0136 8352
0161 0138 0202         LI   R2,9
     013A 0009
0162      013F' LB1    EQU  $+3
0163 013C D851  FADD30 MOVB *R1,@1(R1)    SHIFT FAC RIGHT ONE BYTE
     013E 0001
0164 0140 0601         DEC  R1
0165 0142 0602         DEC  R2
0166 0144 16FB         JN   FADD30
0167 0146 107A  FADD10 JMP  ROUN1
```

```
0169 0148 7556  FADD11 SB   *R6,*R5      SUBT A BYTE OF SMALL FROM BIG
0170 014A 1504         JGT  FADD12
0171 014C 1303         JEQ  FADD12
0172 014E B549         AB   R9,*R5
0173 0150 7948         SB   R8,@-1(R5)
     0152 FFFF
0174 0154 0605  FADD12 DEC  R5
0175 0156 0606         DEC  R6
0176 0158 0584         INC  R4
0177 015A 11F6         JLT  FADD11
0178 015C 1003         JMP  FADD14
0179 015E B549  FADD13 AB   R9,*R5
01 0 0160 0605         D C  R5
01 1 0162 7548         SB   R ,*R5
0182 0164 D115  FADD14 MOVB *R5,R4
0183 0166 11FB         JLT  FADD13
0184 0168 104C         JMP  NORMAL
0185 016A 0460  FADD15 B    @PACKUP
     016C 026C'
```

```
0188                    *       FLOATING MULTIPLICATION
0189                    *
0190                    *       FAC  := ARG * FAC
0191                    *
0192                    *       CALL:      B       @FMULT
0193                    *
0194                    *       WARNINGS:          WRNOV  OVERFLOW
0195                    *
0196 016E C28B  FMULT   MOV   R11,R10      SAVE RETURN ADDRESS
0197 0170 1003          JMP   FMULT1       GO TO PROCESS
0198 0172 C28B  SMULT   MOV   R11,R10      STACK MULTIPLICATION
0199 0174 06A0          BL    @POPSTK
     0176 0000+
0200 0178 0203  FMULT1  LI    R3,FAC       IF FAC IS ZERO
     017A 834A
0201 017C 0205          LI    R5,ARG
     017E 835C
0202 0180 C213          MOV   *R3,R8       IF FAC IS ZERO
0203 0182 1346          JEQ   FMULZR       THEN RESULT IS ZERO
0204 0184 2A15          XOR   *R5,R8       COMPUTE RESULT SIGN
0205 0186 0755          ABS   *R5          IF ARG IS ZERO
0206 0188 1343          JEQ   FMULZR         THEN ZERO FAC AND RETURN
0207 018A 0753          ABS   *R3          TAKE ABS VALUE OF FAC
020  018C 04C9          CLR   R9             TO ZERO LOW BYTE OF RESULT EXP
0209 018E D253          MOVB  *R3,R9      RESULT EXP = FAC EXP
0210 0190 B255          AB    *R5,R9        + ARG EXP
0211 0192 06C9          SWPB  R9
0212 0194 0229          AI    R9,-63         - BIAS
     0196 FFC1
0213 0198 C809          MOV   R9,@EXP
     019A  376
0214 019C D808          MOVB  R8,@SIGN     SAVE TIL NORMAL, ROUND
     019E 8375
0215 01A0 0205          LI    R5,FAC+8     LOW ORDER DIGITS
     01A2 8352
0216 01A4 04F5  FMCLR   CLR   *R5+          WILL BE
0217 01A6 0285          CI    R5,FAC+16       FORMED
     01A8  35A
021  01AA 16FC          JNE   FMCLR            HER .
```

```
0220                    *
0221                    *       R0-R1                   WORK REGISTERS FOR MPY,DIV
0222                    *       R2                      CURRENT RESULT DIGIT
0223                    *       R3                      CURRENT FAC DIGIT
0224                    *       R4                      REGISTER NUMBER LOOP COUNT
0225                    *       R5                      FAC LOOP COUNT
0226                    *       R6                      POINTER TO RESULT IN FAC
0227                    *       R7                      NUMBER OF SIGNIFICANT BYTES IN
0228                    *                               ARG FRACTION
0229                    *       R8                      RB(R0) POINTER
0230                    *       R9                      RADIX 100 VALUE
0231                    *
0232 01AC 0205                  LI    R5,FAC+8          BYTES IN FAC+1
     01AE 8352
0233 01B0 0605  FMUL02  DEC   R5                        CHANGE SIGNIFICANT BYTE COUNT
0234                    *                               FOR LAST ZERO BYTE
0235 01B2 D015          MOVB  *R5,R0                    IF NEXT FAC BYTE IS ZERO
0236 01B4 13FD          JEQ   FMUL02                      THEN DECREMENT COUNT FOR IT
0237 01B6 0207          LI    R7,8                      COUNT SIGNIFICANT BYTES IN ARG
     01B8 0008
0238 01BA 0607  FMUL03  DEC   R7                        DECREMENT FOR ZERO BYTE
0239 01BC D027          MOVB  @ARG(R7),R0               IF THIS BYTE OF ARG IS ZERO
     01BE 835C
0240 01C0 13FC          JEQ   FMUL03                      THEN DECREMENT COUNT.
0241 01C2 04C0          CLR   R0                        MPY, DIV WORK REG
0242 01C4 3880          MPY   R0,R2                     CURRENT RESULT HIGH BYTE
0243 01C6 C185          MOV   R5,R6
0244 01C8 0208          LI    R8,WS+1                   RB(R0)
     01CA 83E1
0245 01CC 0209          LI    R9,100                    RADIX
     01CE 0064
0246                    *
0247 01D0 C107  FMUL04  MOV   R7,R4                     INNER LOOP CTR = BYTES IN ARG
0248 01D2 A187          A     R7,R6                     RESULT PTR TO END OF NEXT PARTIAL
0249                    *                               PRODUCT
0250 01D4 D815          MOVB  *R5,@WS+7                   RB(R3) IS NEXT DIGIT OF FAC
     01D6 83E7
0251 01D8 D543          MOVB  R3,*R5                    CLEAR FAC DIGIT FOR NEXT PARTIAL
0252 01DA D624  FMUL05  MOVB  @ARG(R4),*R8              GET NEXT DIGIT OF ARG
     01DC 835C
0253 01DE 3803          MPY   R3,R0                       AND MPY IT
0254 01E0 D816          MOVB  *R6,@WS+5                 TO CORRESPONDING PARTIAL PRODUCT
     01E2 83E5
0255                    *                               DIGIT IN RB(R2)
0256 01E4 A042          A     R2,R1                     ADD IN NEXT PARTIAL PROD DIGIT
0257 01E6 3C09          DIV   R9,R0                     CONVERT PRODUCT TO RADIX DIGIT
0258                    *                               AND CARRY.
0259 01E8 D5A0          MOVB  @WS+3,*R6                 STORE NEW RESULT DIGIT IN FAC
     01EA 83E3
0260 01EC 0606          DEC   R6                        POINT TO NEXT HIGHER BYTE OF RESULT
0261 01EE B59           AB    *R ,*R6                   ADD IN CARRY TO NEXT HIGHER BYTE
0262                    *                               OF RESULT
0263 01F0 0604          DEC   R4                        IF ALL ARG DIGITS NOT DONE
0264 01F2 15F3          JGT   FMUL05                      THEN CONTINUE
0265 01F4 0606          DEC   R6                        POINT TO START OF NEXT PARTIAL PROD
0266 01F6 0605          DEC   R5                        IF FAC DIGITS REMAIN,
0267 01F8 0285          CI    R5,FAC
     01FA 834A
0268 01FC 15E9          JGT   FMUL04                      THEN CONTINUE
0269 01FE 04E0  FMEND   CLR   @FAC+10                   CLEAR ERROR INDICATOR
```

0200 8354

```
0272                    *
0273                    *       CALL:   SIGN        SIGN
0274                    *               EXP         EXPONENT WITHOUT SIGN
0275                    *               JMP         NORMAL
0276                    *
0277                    *       DESTROYS:           R0 - R2,
0278                    *
0279                    *       WARNINGS:           WRNOV    OVERFLOW
0280 0202 0201  NORMAL LI    R1,-9          NUMBER OF BYTES IN FAC INCLUDING
     0204 FFF7
0281                    *                             GUARD BYTES.
0282 0206 D0A1  NORM01 MOVB  @FAC+10(R1),R2  IS NEXT BYTE OF FAC NON-ZERO?
     0208 8354
0283 020A 1607          JNE   NORM02        YES, SHIFT REST LEFT.
0284 020C 0581          INC   R1            NO.  ALL BYTES ZERO?
0285 020E 11FB          JLT   NORM01              YES, LOOK AT NEXT BYTE.
0286                    *
0287                    *       ZERO FAC
0288                    *
0289                    *       SETS FAC := 0
0290                    *
0291                    *       CALL:     BL        FZERO
0292                    *
0293                    *       DESTROYS:           NONE
0294 0210               FMULZR
0295 0210 04E0  FZERO  CLR   @FAC          INSTALL FLOATING ZERO
     0212 834A
0296 0214 04E0          CLR   @FAC+2        CLEAR POSSIBLE BASIC TYPE CODE
     0216 834C
0297 0218 1039          JMP   STEX          AND EXIT WITH STATUS
0298 021A C001  NORM02 MOV   R1,R0          NUMBER OF NON-ZERO BYTES
0299 021C 0220          AI    R0,9          FIRST BYTE NON-ZERO?
     021E 0009
0300 0220 130D          JEQ   ROUN1         YES, FINISH
0301 0222 6800          S     R0,@EXP       NO, ADJUST EXPONENT FOR SHIFT
     0224 8376
0302 0226 0202          LI    R2,FAC+1      POINT TO FIRST BYTE OF FAC
     0228 8343
0303 022A DCA1  NORM03 MOVB  @FAC+10(R1),*R2+  MOVE NON-ZERO BYTE
     022C 8354
0304                    *                               TO FAC FIRST DIGIT.
0305 022E 0581          INC   R1            IF NON-ZERO BYTES REMAIN
0306 0230 11FC          JLT   NORM03          THEN MOVE ANOTHER BYTE
0307                    *                   ZERO LOW-ORDER BYTES REMAINING
0308 0232 DC81  NORM04 MOVB  R1,*R2+        MOVE A ZERO
0309 0234 0600          DEC   R0            LAST BYTE DONE?
0310 0236 15FD          JGT   NORM04        NO, CONTINUE.
0311                    *                   Y S, ROUND THE NUMBER IN FAC
0312                    *                   AND FINISH UP
```

```
0315                    *       ROUND THE NUMBER IN THE FAC USING THE GUARD DIGITS
0316                    *
0317                    *       CALL:   SIGN        SIGN
0318                    *               EXP         EXPONENT WITHOUT SIGN
0319                    *               JMP         ROUND
0320                    *
0321                    *       DESTROYS:           R0 - R2
0322                    *
0323                    *       WARNINGS:           WRNOV  OVERFLOW
0324                    *
0325 0238 1001                  JMP   ROUN1         SKIP SAVE IF INTERNAL CALL
0326 023A C28B  ROUND   MOV   R11,R10       SAVE RETURN ADDRESS
0327 023C 0200  ROUN1   LI    R0,50*HIBYTE
     023E 3200
0328 0240 8020          C     @FAC+8,R0     IS ROUNDING NECESSARY?
     0242 8352
0329 0244 1113          JLT   PACKUP        NO, PUT EXPONENT BACK
0330 0246 0201          LI    R1,7          ROUND UP, GET NUMBER OF FAC BYTES
     0248 0007
0331 024A 0202  ROUNUP  LI    R2,1*HIBYTE   1 (FOR BYTE INSTR)
     024C 0100
0332 024E 0200          LI    R0,100*HIBYTE  100 (SAME)
     0250 6400
0333 0252 B842  ROUN02  AB    R2,@FAC(R1)   ADD ONE TO A BYTE OF FAC
     0254 834A
0334 0256 9021          CB    @FAC(R1),R0   IF BYTE NOT GREATER THAN RADIX
     0258 834A
0335 025A 1A08          JL    PACKUP        THEN PUT EXPONENT IN FAC
0336 025C 7840          SB    R0,@FAC(R1)   BRING DIGIT BACK IN RANGE
     025E 834A
0337 0260 0601          DEC   R1            IF CARRY PAST HIGH BYTE OF FAC
0338 0262 15F7          JGT   ROUN02        THEN CARRY TO NEXT HIGHER BYTE
0339 0264 05A0          INC   @EXP          FRACTION HAS OVERFLOWED
     0266 8376
0340                    *                   (MEANS NUMBER WAS ALL 9'S)
0341                    *                   SHIFT NUMBER BY ADDING 1 TO EXP
0342 0268 D802          MOVB  R2,@FAC+1     MAKE THE HIGH BYTE A 1
     026A 834B
```

```
0345                    *       PUT SIGN AND EXPONENT IN FAC
0346                    *
0347                    *       CALL:   SIGN        SIGN IN HIGH ORDER BIT
0348                    *               EXP         EXPONENT WITHOUT SIGN
0349                    *               JMP         PACKUP
0350                    *
0351 026C C0E0  PACKUP  MOV   @EXP,R3
     026E 8376
0352 0270 0283          CI    R3,128
     0272 0080
0353 0274 141A          JHE   OVEXP1
0354 0276 D820          MOVB  @WS+7,@FAC     PUT EXPONENT IN FAC
     0278 83E7
     027A 834A
0355 027C D0A0          MOVB  @SIGN,R2
     027E 8375
0356 0280 0542          INV   R2             IF SIGN IS NEGATIVE,
0357 0282 1102          JLT   PACK01
0358 0284 0520          NEG   @FAC             THEN INVERT 1ST WORD
     0286 834A
0359      0288' PACK01  EQU   $
0360 0288 1001          JMP   STEX           SKIP SAVE IF INTERNAL CALL
0361 028A C28B  STEXIT  MOV   R11,R10        SAVE RETURN ADDRESS
0362 028C C060  STEX    MOV   @FAC,R1        SET STATUS ON FAC
     028E  34A
0363 0290 02C2  STEX01  STST  R2             AND PUT IT
0364 0292 D802          MOVB  R2,@STATUS     IN THE STATUS REGISTER
     0294 837C
0365 0296 045A          B     *R10           THEN RETURN TO GLI
0366                    *
0367                    *       ROUND FAC BEGINNING AT DIGIT SPECIFIED IN ARG
0368 0298 C28B  ROUNU   MOV   R11,R10        SAVE RETURN ADDRESS
0369 029A D060          MOVB  @FAC+10,R1       PICK UP OFFSET
     029C 8354
0370 029E 0981          SRL   R1,8
0371 02A0 10D4          JMP   ROUNUP         AND DO IT.
```

```
0374                    *       ERROR ROUTINE EXITS
0375                    *
0376                    *       DIVIDE BY ZERO ERROR
0377 02A2               FDIV01
0378 02A2 0209  DIVZER  LI    R9,DZERR      DIVIDE BY ZERO CODE FOR USER
     02A4 0200
0379 02A6 1008          JMP   BIGFLT        LARGEST MAGNITUDE WITH SIGN
0380                    *
0381                    *       OVER/UNDERFLOW
0382 02A8 C28B  OVEXP   MOV   R11,R10       SAVE RETURN ADDRESS
0383 02AA D0A0  OVEXP1  MOVB  @EXP,R2       IS EXPONENT NEGATIVE?
     02AC  376
0384 02AE 11B0          JLT   FZERO         YES, RETURN ZERO.
0385 02B0 1001          JMP   OV1           SKIP SAVE IF INTERNAL CALL
0386 02B2 C28B  OV      MOV   R11,R10       SAVE RETURN ADDRESS
0387 02B4 0209  OV1     LI    R9,OFERR
     02B6 0100
0388                    *
0389                    *       SUPPLY THE LARGEST MAGNITUDE VALUE WITH PROPER SIGN
0390 02B8 0200  BIGFLT  LI    R0,->7F63     HIGH WORD OF LARGEST POS VALUE
     02BA 809D
0391 02BC D0A0          MOVB  @SIGN,R2      IS FAC NEGATIVE?
     02BE 8375
0392 02C0 1101          JLT   BIGF01        YES, PUT HIGH WORD IN FAC
0393 02C2 0500          NEG   R0
0394 02C4 0202  BIGF01  LI    R2,FAC        GET PTR TO FAC
     02C6 834A
0395 02C8 CC80          MOV   R0,*R2+       PUT APPROPRIATE HIGH WORD IN FAC
0396 02CA 0200          LI    R0,>6363      GET 99'S
     02CC 6363
0397 02CE CC80          MOV   R0,*R2+       PUT IN FAC TO GIVE LARGEST POS
0398 02D0 CC80          MOV   R0,*R2+       OR MOST NEGATIVE NUMBER
0399 02D2 C480          MOV   R0,*R2
0400 02D4 D809  ERRXI1  MOVB  R9,@FAC+10    PLACE ERROR CODE IN RAM
     02D6 8354
0401 02D8 10D9          JMP   STEX          NO ROUTINE SPECIFIED, RETURN
0402                    *
```

```
0405                    *       FLOATING DIVISION
0406                    *
0407                    *       FAC := ARG / FAC
0408                    *
0409                    *       CALL:    B     @FDIV
0410                    *
0411                    *       WARNINGS:      WRNOV     OVERFLOW
0412                    *                      WRNOVO    DIVISION BY ZERO
0413                    *
0414 02DA C28B          FDIV    MOV  R11,R10        SAVE RETURN ADDRESS
0415 02DC 1003                  JMP FDIV1           GO TO PROCESS
0416 02DE C28B          SDIV    MOV  R11,R10        SAVE RETURN ADDRESS
0417 02E0 06A0                  BL   @POPSTK        STACK DIVISION
     02E2 0000+
0418 02E4 0203          FDIV1   LI   R3,FAC         POINTER TO FAC
     02E6 834A
0419 02E8 C213                  MOV  *R3,R8         GET DIVISOR FIRST WORD
0420 02EA 0200                  LI   R0,ARG         POINTER TO ARG
     02EC 835C
0421 02EE 2A10                  XOR  *R0,R8         NO, COMPUTE SIGN OF QUOTIENT
0422 02F0 D808                  MOVB R8,@SIGN       SAVE SAME
     02F2 8375
0423 02F4 0753                  ABS  *R3            ABS OF DIVISOR
0424 02F6 13D5                  JEQ  FDIV01         CAN'T BE ZERO
0425 02F8 0750                  ABS  *R0            IS DIVIDEND ZERO?
0426 02FA 138A                  JEQ  FMULZR         YES, RESULT IS ZERO.
0427 02FC D250                  MOVB *R0,R9         GET DIVIDEND EXPONENT
042  02FE 7253                  SB   *R3,R9         SUBTRACT EXPONENTS TO GET
0429                    *                           QUOTIENT EXPONENT
0430 0300 0889                  SRA  R9,8           GET DIFFERENCE IN LOW BYTE
0431 0302 0229                  AI   R9,64          ADD BIAS TO EXPONENT
     0304 0040
0432 0306 C809                  MOV  R9,@EXP          AND SAVE FOR RESULT
     030  8376
0433                    *                          MOVE FAC TO DIVISOR STORAGE
0434 030A 0204                  LI   R4,4
     030C 0004
0435 030E 0205                  LI   R5,ARG+8
     0310  364
0436 0312 C8F3          FDV01   MOV  *R3+,@10-2(R3)
     0314 000
0437 0316 04F5                  CLR  *R5+
0438 0318 0604                  DEC  R4
0439 031A 15FB                  JGT  FDV01          LOOP TIL FOUR BYTES MOVED
0440                    *
0441 031C D 04                  MOVB R4,@ARG        CLEAR EXTRA HIGH BYTE OF
     031   35C
0442                    *                           DIVIDEND
```

```
0444                    *
0445                    *       REFERENCE FOR DIVISION ALGORITHM:
0446                    *       DONALD E. KNUTH, THE ART OF COMPUTER PROGRAMMING,
0447                    *       VOLUME 2, SEMINUMERICAL ALGORITHMS, ADDISON-WESLEY,
0448                    *       1969, P. 235 FF.
0449                    *
0450                    *       THE DIVIDEND IS THE SERIES OF RADIX DIGITS:
0451                    *            U0,U1,U2, ... ,U7      (IN ARG)
0452                    *       THE DIVISOR IS THE SERIES OF RADIX DIGITS:
0453                    *               V1,V2, ... ,V7      (IN FAC+8, OR FDVSR)
0454                    *       (U0 IS THE EXTRA HIGH BYTE OF THE DIVIDEND)
0455                    *
0456                    *       NORMALIZE DIVISOR AND DIVIDEND SO V1 GT 50.
0457                    *       IF V1 LT 50, MULTIPLY DIVISOR AND DIVIDEND BY
0458                    *       INT(100/(V1+1))
0459                    *
0460                    *       R0-R1  MPY,DIV WORK REGISTERS
0461                    *       R2     CARRY
0462                    *       R3     MULTIPLIER
0463                    *       R4     LOOP COUNT
0464                    *       R5     PTR TO RB(R0)
0465                    *       R6     PTR TO RB(R1)
0466                    *       R7     100
0467                    *
0468 0320 0205          FDIV    LI   R5,WS+1        GET POINTERS INTO MULTIPLY
     0322 83E1
0469                    *                           WORK AREA
0470 0324 0206                  LI   R6,WS+3
     0326 83E3
0471      032A' LW100   LB100   EQU  $+2
0472      032B' LB100           EQU  $+3
0473 0328 0207                  LI   R7,100         RADIX
     032A 0064
0474 032C 04C2                  CLR  R2             CLEAR HIGH BYTE OF WHERE V1 WILL B
0475 032E D820                  MOVB @FDVSR+1,@WS+5  GET V1 IN RB(R2)
     0330 8355
     0332 83E5
0476 0334 0282                  CI   R2,49          IS V1 ALREADY NORMALIZED?
     0336 0031
0477 0338 151E                  JGT  FDIV06         YES, PROCEED WITH DIVISION
0478 033A 0582                  INC  R2             NO, COMPUTE V1+1
0479 033C 04C3                  CLR  R3             GET RADIX IN 2 REGS FOR DIV
0480 033E C107                  MOV  R7,R4          GET RADIX
0481 0340 3CC2                  DIV  R2,R3          COMPUTE MULTIPLIER =
0482                    *                             INT(100/(V1+1))
```

```
04 4 0342 0209          LI    R9,FDVSR+8
     0344  35C
04 5 0346 0204  FDVLP   LI    R4,           GET NUMBER OF BYTES IN DIVID ND+1
     034. 000
04 6 034A 0604  FDIV04  DEC   R4            IGNOR  ZERO BYT S AT END
0487 034C 0609          D C   R9
04 8 034E D019          MOV   *R9,R0        IS NEXT HIGHER ORDER BYTE ZERO?
0489 0350 13FC          J Q   FDIV04        YES, KEEP LOCKING FOR NON-ZERO
0490 0352 04C0          CLR   R0            NO, CLEAR CARRY INTO LOW ORDER BYTE
0491 0354 C0 0  FDIV05  MOV   R0,R2         SAVE CARRY FROM LAST BYTE
0492 0356 D559          MOVB  *R9,*R5       GET N XT BYTE OF DIVIDEND
0493 0    3 03          MPY    3, 0         MULTIPLY THIS BYTE BY MULTIPLIER
0494 03 A A042          A     R , 1         ADD IN CARRY FROM PREVIOUS BYTE
0495 035C 3C07          DIV   R7, 0         CNVRT TO A RADIX DIGIT AND A CARRY
0496 035E D656          MOV   *R6,*R9       PUT RESULT BYTE IN DIVIDEND
0497 0360 0609          D C   R9
049. 0362 0604          DEC   R4            LOOP UNTIL ALL DIVIDEND BYT S
0499 0364 15F7          JGT   FDIV05        NO, CONTINUE MULTIPLYING
0500 0366 0289          CI    R9,FDVSR
     036  8354
0501 036A 1603          JN    FDVLPA
0502 036C 0209          LI    R9,ARG+
     036E  364
0503 0370 10EA          JMP   FDVLP
0504 0372 D 15  FDVLPA  M VB  *R ,@ARG      Y S, PUT CARRY OUT OF HIGH ORD R
     0374 835C
0505            *                           IN HIGHEST BYTE.
```

```
0551 03A0 04C2  FDIV08 CLR  R2             CLEAR HIGH BYTE OF WHERE U(J) WILL
0552                   *                   BE
0553 03A2 D81B         MOVB *R11,@WS+5      R3(R2) IS U(J)
     03A4 83E5
0554 03A6 38A0         MPY  @LW100,R2       COMPUTE 100*U(J)
     03A8 032A'
0555 03AA 04C0         CLR  R0              WHERE U(J+1) WILL BE
0556 03AC D923         MOVB @1(R11),@WS+1 GET U(J+1)
     03AE 0001
     03B0 83E1
0557 03B2 A0C0         A    R0,R3           100*U(J)+U(J+1)
0558 03B4 3C87         DIV  R7,R2           GET Q AND REMAINDER
0559 03B6 38E0         MPY  @LW100,R3       100*REMAINDER
     03B8 032A'
0560 03BA D823         MOVB @2(R11),@WS+1 U(J+2)
     03BC 0002
     03BE 83E1
0561 03C0 A100         A    R0,R4           100*REM + U(J+2)
0562 03C2 C002         MOV  R2,R0           GET Q FOR THE TEST
0563 03C4 3808         MPY  R8,R0           COMPUTE V2*Q
0564 03C6 8802         C    R2,@LW100       DOES Q = 100?
     03C8 032A'
0565                   *                    I.E., V1 = U(J)?
0566 03CA 1302         JEQ  FDIV09          YES, MAKE Q = 99
0567 03CC 6044         S    R4,R1           NO, COMPUTE V2*Q-(100*REM+U(J+2))
0568 03CE 1003         JMP  FDIV11          GO CHECK IF IT IS IN RANGE
0569 03D0 6044  FDIV09 S    R4,R1           COMPUTE V2*Q-(100*REM+U(J+2))
0570 03D2 0602  FDIV10 DEC  R2              DECREMENT Q
0571 03D4 6049         S    R9,R1           COMPUTE ABOVE FOR NEW Q
0572 03D6 15FD  FDIV11 JGT  FDIV10          IF Q TOO BIG MAKE IT SMALLER
0573 03D8 C082         MOV  R2,R2           IS Q ZERO?
0574 03DA 1329         JEQ  FDIV16          YES, DO NO SUBTRACTING.
0575                   *                    NO, SUBTRACT Q*V FROM U
```

```
0577                    *       R0-R1   MPY,DIV WORK REGISTERS
0578                    *       R2      QUOTIENT DIGIT
0579                    *       R3      CARRY
0580                    *       R4      LOOP COUNT
0581                    *       R5      QUOTIENT BYTE LOOP COUNT
0582                    *       R6      NUMBER SIGNIF BYTES IN DIVISOR
0583                    *       R7      V1
0584                    *       R8      V2
0585                    *       R9      100 * V1 + V2
0586                    *       R11     POINTER INTO DIVIDEND
0587                    *
0588 03DC 04C3                  CLR   R3              CLEAR CARRY INTO FIRST BYTE
0589 03DE C106                  MOV   R6,R4           GET DIVISOR LOOP COUNT
0590 03E0 A2C6                  A     R6,R11          TO LOW ORDER BYTE OF DIVIDEND
0591                    *                             OF INTEREST
0592 03E2 C0C0  FDIV12  MOV   R0,R3           SAVE CARRY FROM PREV BYTE
0593 03E4 D824                  MOVB  @FDVSR(R4),@WS+1  GET NEXT BYTE OF DIVISOR
     03E6 8354
     03E8 83E1
0594 03EA 3802                  MPY   R2,R0           MPY BYTE OF DIVISOR BY QUOTIENT
0595 03EC A043                  A     R3,R1           ADD IN CARRY FROM LAST DIVISOR BYTI
0596 03EE 3C20                  DIV   @LW100,R0       CONVERT RESULT TO A RADIX 100 DIGI
     03F0 032A'
0597                    *                             AND CARRY.
0598 03F2 76E0                  SB    @WS+3,*R11      SUBTRACT PRODUCT BYTE FROM DIVIDEN
     03F4 83E3
0599 03F6 1504                  JGT   FDIV13          IS RESULT POSITIVE?
0600 03F8 1303                  JEQ   FDIV13          OR ZERO?
0601 03FA B6E0                  AB    @LB100,*R11     NO, ADD RADIX BACK
     03FC 032B'
0602 03FE 0580                  INC   R0              INCREMENT PRODUCT CARRY TO BORROW
0603                    *                             FROM NEXT BYTE
0604 0400 060B  FDIV13  DEC   R11             POINT TO NEXT HIGHER BYTE OF DVDND
0605 0402 0604                  DEC   R4              SUBTRACTED ALL BYTES OF DIVISOR?
0606 0404 15EE                  JGT   FDIV12          NO, CONTINUE SUBTRACTING.
0607 0406 76E0                  SB    @WS+1,*R11      YES, SUB CARRY FROM DIVISOR PRODUC
     0408 83E1
0608 040A 1511                  JGT   FDIV16          HIGH ORDER FROM HIGHEST ORDER
0609 040C 1310                  JEQ   FDIV16          DIVIDEND BYTE. NEGATIVE RESULT?
0610                    *                             YES, ADD DIVIDEND BACK IN, Q WAS
0611                    *                             ONE TOO BIG.
```

```
 0613 040E 0602          DEC   R2              DEC Q, WAS ONE TOO BIG.
 0614 0410 C106          MOV   R6,R4           GET ADD-BACK LOOP COUNT
 0615 0412 A2C6          A     R6,R11          POINT TO LOW ORDER BYTE OF DIVIDEND
 0616                    *                     OF INTEREST
 0617 0414 B6E4  FDIV14  AB    @FDVSR(R4),*R11  ADD BYTE OF DIVISOR TO DIVIDEND
      0416 8354
 0618 0418 981B          CB    *R11,@LB100     RESULT LARGER THAN RADIX?
      041A 032B'
 0619 041C 1A05          JL    FDIV15          NO, RESULT IS CORRECT
 0620 041E 76E0          SB    @LB100,*R11     YES, SUBTRACT RADIX
      0420 032B'
 0621 0422 BAE0          AB    @LB1,@-1(R11)   ADD 1 FOR CARRY TO HIGHER BYTE
      0424 013F'
      0426 FFFF
 0622 0428 060B  FDIV15  DEC   R11             TO NEXT HIGHER BYTE OF DIVIDEND
 0623 042A 0604          DEC   R4              DONE ADDING IN ALL BYTES OF DIVIDND
 0624 042C 15F3          JGT   FDIV14          NO, ADD IN THE NEXT ONE
 0625 042E D960  FDIV16  MOVB  @WS+5,@FAC+10(R5)  PUT AWAY NEXT QUOTIENT BYTE
      0430 83E5
      0432 8354
 0626 0434 058B          INC   R11             HIGH ORDER OF NEXT SIGNIF DVDND
 0627 0436 0585          INC   R5              COMPUTED ALL NECESSARY BYTES OF QUO
 0628 0438 1133          JLT   FDIV08          NO, CONTINUE
 0629 043A 0460          B     @FMEND           YES, NORMALIZE AND FINISH UP.
      043C 01FE'
 0630                    END
NO ERRORS,      NO WARNINGS
```

```
LABEL     VALUE   DEFN  REFERENCES

                        0326  0361  0368  0382  0386  0414  0416  0549  0553
                        0556  0560  0590  0598  0601  0604  0607  0615  0617
                        0618  0620  0621  0622  0626
R13       000D          A0056 A0057 A0059
R15       000F          A0039 A0041 A0048 A0050
R2        0002          0161  0165  0242  0256  0292  0302  0303  0308  0331
                        0333  0342  0355  0356  0363  0364  0383  0391  0394
                        0395  0397  0398  0399  0474  0476  0478  0481  0491
                        0494  0551  0554  0558  0562  0564  0570  0573  0573
                        0594  0613
R2LB      83E5    0014
R3        0003          0034  0039  0044  0062  0098  0100  0100  0103  0106
                        0107  0107  0108  0109  0200  0202  0207  0209  0251
                        0253  0351  0352  0418  0419  0423  0428  0436  0436
                        0479  0481  0493  0557  0559  0588  0592  0595
R4        0004          0134  0135  0147  0176  0182  0247  0252  0263  0434
                        0438  0441  0480  0485  0486  0498  0561  0567  0569
                        0589  0593  0605  0614  0617  0623
R5        0005          0048  0049  0054  0055  0057  0059  0061  0113  0120
                        0131  0140  0141  0143  0144  0145  0150  0151  0152
                        0156  0169  0172  0173  0174  0179  0180  0181  0182
                        0201  0204  0205  0210  0215  0216  0217  0232  0233
                        0235  0243  0250  0251  0266  0267  0435  0437  0468
                        0492  0504  0548  0625  0627
R6        0006          A0050 A0051 A0056 A0058 0051  0054  0056  0117  0119
                        0123  0128  0132  0133  0140  0146  0169  0175  0242
                        024   0254  0259  0260  0261  0263  0470  0496  0526
                        0537  0539  0589  0590  0614  0615
R6LB      83ED    0015  A0048 A0057
R7        0007          0047  0049  0051  0055  0056  0057  0059  0061  0C
                        0095  0111  0136  0237  0238  0239  0247  0248  04,
                        0480  0495  0542  0544  0558
R8        0008          A0024 0088  0095  0111  0116  0129  0144  0151  0173
                        0181  0202  0204  0214  0244  0252  0261  0419  0421
                        0422  0544  0545  0547  0563
R9        0009          0130  0141  0143  0152  0156  0172  0179  0208  0209
                        0210  0211  0212  0213  0245  0257  0378  0387  040C
                        0427  0428  0430  0431  0432  0484  0487  0488  0492
                        0496  0497  0500  0502  0547  0571
ROUND2        0252'   0333  033
ROUN1   D     023C'   0327  0009  0167  0300  0325
ROUND   D     023A'   0326  0007  0023
ROUNU   D     0298'   0368  0007  0023
ROUNUP  D     024A'   0331  0010  0371
SADD    D     006A'   0084  000   0025
SCOMP   D     002C'   0044  0009  0025
SCOMP1        0032'   0046  0040
SCOMPB  D     002 '   0039  0009
SDIV    D     02D '   0416  000   0025
SIGN          8375    A0026 0116  0214  0355  0391  0422
SMULT   D     0172'   019   0008  0025
SSUB    D     005A'   0077  0008  0025
STATUS        837C    A0029 0364
STEX          028C'   0362  0094  0297  0360  0401
STEX01        0290'   0363  0034  0044
STEXIT        028A'   0361  0023
STKADD        8373    A0025
STKMOV        0018+   A0043 A0045
VDPRD         8800    A0013 A0040 A0052
VDPWD         8C00    A0014
```

```
0001                    IDT   'CSN'
```

```
 0003                    REF   OVEXP1,ROUNUP,ROUN1,FZERO
 0004                    REF   FADD
 0005                    REF   ASSGNV,FBSYMB              BASIC SUPPORT ROUTINES
 0006                    REF   SYMB,SMBB            MORE BASIC SUPPORT ROUTINES
 0007                    REF   VPUSHG,VPOP    EVEN MORE BASIC SUPPORT ROUTINES
 0008                    REF   PGMCH           EVEN MORE BASIC SUPPORT ROUTINES
 0009                    REF   SROM,SGROM
 0010                    DEF   CSN,CSNGR,XTAB
 0011                    DEF   CFI
 0012                    COPY  FPEQ
A0001 0000         DATA1  CSEG
A0002                   ********************************************************************
A0003                   *  LOCATIONS USED BY THE INTERPRETER IN HOME COMPUTER              *
A0004        8300  PAD    EQU   >8300                                                      *
A0005        0004  I      EQU   4
A0006        0005  J      EQU   5
A0007        0006  K      EQU   6
A0008        0007  L      EQU   7
A0009        0008  N      EQU   8
A0010                   ********************************************************************
A0011                   *      PROCESSOR RAM FIXED LOCATIONS                               *
A0012                   *                                                                  *
A0013        8800  VDPRD  EQU   >8800         ADDRESS TO READ VDP
A0014        8C00  VDPWD  EQU   >8C00         ADDRESS TO WRITE VDP
A0015        834A  FAC    EQU   PAD+>4A       FLOATING ACCUMULATOR                        *
A0016        8354  FDVSR  EQU   FAC+10        DIVISOR STORAGE DURING DIVISION             *
A0017        8354  FLTNDX EQU   FAC+10        INSTRUCTION INDEX SAVE                      *
A0018        835C  ARG    EQU   FAC+18        FLOATING ARGUMENT                           *
A0019        836C  FLTERR EQU   PAD+>6C       ERROR ADDRESS FOR MATH ROUTINES             *
A0020        836E  VSPTR  EQU   PAD+>6E       VALUE STACK POINTER                         *
A0021        8376  JOYY   EQU   PAD+>76
A0022        8310  PROA   EQU   PAD+>10       PROCESSOR ROLLOUT AREA
A0023        03C0  VROA   EQU   >3C0          VIDEO RAM ROLLOUT AREA
A0024        000   CC     EQU   R8
A0025        8373  STKADD EQU   PAD+>73
A0026        8375  SIGN   EQU   PAD+>75       TEMPORARY SIGN STORAGE                      *
A0027        8377  JOYX   EQU   PAD+>77
A0028        8376  EXP    EQU   PAD+>76       TEMPORARY EXPONENT STORAGE                  *
A0029        837C  STATUS EQU   PAD+>7C       STATUS REGISTER                             *
A0030        83E0  WS     EQU   PAD+>E0       WORKSPACE MNEMONIC                          *
A0031        83EF  LLB    EQU   PAD+>EF       LOWER HALF OF L                             *
A0032                   ********************************************************************
A0033                   *      SUBROUTINE TO POP VALUE STACK
A0034                   *
A0035                   REF   GRMWA,GRMRA,WRVDP
A0036                   *
A0037 0000 0205  POPSTK LI    J,-8           COUNTER FOR LOOP
      0002 FFF8
A003  0004 0206         LI    K,ARG          ADDRESS TO STORE OPERAND
      0006  35C
A0039 0008 D7E0         MOVB  @VSPTR+1,*R15 LOAD VDP ADDRESS
      000A  36F
A0040 000C 0207         LI    L,VDPRD        ADDRESS TO READ VDP
      000E 8800
A0041 0010 D7E0         MOVB  @VSPTR,*R15    MSB OF ADDRESS
      0012 836E
A0042 0014 A805         A     J,@VSPTR       DECREMENT STACK COUNTER
      0016 836E
A0043 0018 DD97  STKMOV MOVB  *L,*K+         RECALL BYTE FROM VDP
A0044 001A 0585         INC   J              INCREMENT LOOP COUNTER
```

```
A0045 001C 16FD          JNE  STKMOV        8 BYTES?
A0046 001E 045B          RT
A0047                *
A0048 0020 D7E0  GETCH   MOVB @R6LB,*R15         LOAD VDP ADDRESS
      0022 83ED
A0049 0024 1000          NOP
A0050 0026 D7C6          MOVB R6,*R15            THE ZERO DISP IS FOR TIMING
A0051 0028 0586          INC  R6                 NEXT ADDRESS
A0052 002A D220          MOVB @VDPRD,CC
      002C 8800
A0053 002E 0988  GETCH1  SRL  CC,8           SHIFT TO LOWER BYTE
A0054 0030 045B          RT                      RETURN
A0055                *
A0056 0032 DB46  GETCHG  MOVB R6,@GRMWA(R13)
      0034 0000
A0057 0036 DB60          MOVB @R6LB,@GRMWA(R13)
      0038 83ED
      003A 0034+
A0058 003C 0586          INC  R6
A0059 003E D21D          MOVB *R13,CC
A0060 0040 10F6          JMP  GETCH1
A0061                *
A0062 0042               CEND
 0013       83E1  R0LB   EQU  PAD+>E1
 0014       83E3  R1LB   EQU  PAD+>E3
 0015       83E5  R2LB   EQU  PAD+>E5
 0016       83E7  R3LB   EQU  PAD+>E7
 0017       83E9  R4LB   EQU  PAD+>E9
 0018       83EB  R5LB   EQU  PAD+>EB
 0019       83ED  R6LB   EQU  PAD+>ED
 0020       83EF  R7LB   EQU  PAD+>EF
 0021       83F1  R8LB   EQU  PAD+>F1
 0022       83F3  R9LB   EQU  PAD+>F3
 0023       83F5  R10LB  EQU  PAD+>F5
 0024       83F7  R11LB  EQU  PAD+>F7
 0025 0000    32  LW50H  BYTE 50
 0026 0001    03  ERRI   BYTE ERRIOV
 0027 0002               EVEN
 0028       0030  X3000  EQU  >30
 0029       002B  XPLUS  EQU  >2B
 0030       002D  XMINUS EQU  >2D
 0031       0039  X3900  EQU  >39
 0032       002E  XPOINT EQU  >2E
 0033       0100  A2     EQU  >100
 0034             * STRING  TO NUMBER CONVERSIONS
 0035             *
 0036             *
 0037             *       REGISTER USAGE:
 003              *       R0 - POINTER
 0039             *       R1 - EXPONENT SIGN
 0040             *       R2 - TP1 (SAVED TEXT POINTER)
 0041             *       R3 - ADDRESS OF GETCH (FOR BL)
 0042             *       R4 - ACCUMULATOR FOR CSINT
 0043             *       R5 - ACCUMULATOR FOR CSINT
 0044             *       R6 - TEXT POINTER
 0045             *       R7 - RELATIVE POSITION COUNTER
 0046             *       R8 - CURRENT CHARACTER (LSBYTE)
 0047             *       R9 - UNUSED
 0048             *       R10 - SAVED LINK
 0049             *       R11 - LINK
```

```
                *       R12 - NON-ZERO CHARACTER POINTER
                *
                *
                *LOCATIONS ASSIGNMENTS
0006            TP      EQU  R6
0002            TP1     EQU  R2
000C            TPNZ    EQU  R12
8389            GROMFL  EQU  PAD+>89
                *
                * CALLED WITH FAC=ADDRESS OF STRING
                *         BL    CSN
                * RETURNS WITH CONTAINING NUMBER
                *
                * WARNINGS:  WRNOV - OVERFLOW.
                * ERROR:     ERRSNN - SYNTAX ERROR IN NUMBER
                *
                *         RPOS-RELATIVE POSITION OF FIRST NONZERO DIGIT
                *              WRT ONE'S DIGIT
                *             EG:  10,+1;  1,+0;  .1,-1;  .01,-2
                *
                *         TPNZ  POINTER TO FIRST NONZERO CHARACTER
                *         TP    TEXT POINTER
                *         SGN   SIGN
                *         TP1   POINTER TO CHARACTER AFTER REQUIRED POSITION
                *               OF FIRST DIGIT
                *
                *         CC    CURRENT CHARACTER
                *
                ********************************************************************
                *
                * CONVERT STRING TO INTEGER
                *
                *       THE VALUE IS RETURNED IN R4
                *
0002 04C4       CSINT   CLR   R4
0004 04C0               CLR   R0              CHARACTER COUNTER FOR EXPONENT
0006 C248               MOV   R11,R9
0008 1008               JMP   CSI02
000A'           CSI01   EQU   $                  MULTIPLY PREVIOUS BY TEN
000A 3920               MPY   @C10,R4
000C 0022'
000E C104               MOV   R4,R4             TEST FOR OVERFLOW
0010 160D               JNE   CSI05             YES OVERFLOW
0012 0580               INC   R0              COUNT THE CHARACTER
0014 A148               A     R8,R5
0016 C105               MOV   R5,R4             IS INTEGER > 32767
0018 1109               JLT   CSI05             YES, TOO BIG
001A'           CSI02   EQU   $                 GET NEXT DIGIT
001A 0693               BL    *R3
001C 0228               AI    CC,-'0'           ASCII TO BINARY
001E FFD0
0022'           C10     EQU   $+2             <***CONSTANT 10
0020 0288               CI    CC,>A             COMPARE TO TEN
0022 000A
0024 1AF2               JL    CSI01             CHARACTER OK
0026 C000               MOV   R0,R0           ANY CHARACTERS ?
0028 1306               JEQ   CSNZ10          NO - ERROR
002A 0459               B     *R9               RETURN GOOD
002C 0209       CSI05   LI    R9,CSNOFL         SET RETURN ADDRESS TO ERROR
002E 0038'
```

```
0154                    *
0155 008A 0587  CSN05   INC   R7              INCREMENT RELATIVE POSITION
0156 008C 0693          BL    *R3
0157      008E' CSN06   EQU   $               GET A ZERO
0158 008E 0288          CI    CC,X3000        LESS THAN '0'
     0090 0030
0159 0092 1A03          JL    CSN07           YES
0160 0094 0288          CI    CC,X3900        LESS THAN '9'
     0096 0039
0161 0098 12F8          JLE   CSN05           YES
0162      009A' CSN07   EQU   $               '.'
0163 009A 0288          CI    CC,XPOINT       END OF INTEGER PART?
     009C 002E
0164 009E 1614          JNE    CSNG            LOOK FOR EXPONENT OR END
0165                    *
0166                    *
0167                    *  CONVERT A FLOATING POINT NUMBER
0168                    *
0169                    *  THE NUMBER HAS A DECIMAL POINT
0170 00A0 0582  CSNF01  INC   TP1             MOVE DIGIT POINTER PAST
0171                    *                       DECIMAL POINT
0172 00A2 C1C7          MOV   R7,R7
0173 00A4 1102          JLT   CSNF03          NO SIGNIF DIGIT TO LEFT OF .
0174 00A6 1007          JMP   CSNF04          LOOK FOR LAST DIGIT
0175 00A8 0607  CSNF02  DEC   R7
0176 00AA       CSNF03
0177 00AA 0693          BL    *R3              GET NEXT CHARACTER
0178 00AC 0288          CI    CC,X3000        IGNORE LEADING ZEROES
     00AE 0030
0179 00B0 13FB          JEQ   CSNF02          YES
0180 00B2 0606          DEC   TP
0181                    *                          THIS IS FIRST SIGNIFICANT
0182                    *                          DIGIT OF THE END OF THE # .
0183                    *                          POINT BACK TO FIRST NONZERO
0184 00B4 C306          MOV   TP,TPNZ       CHARACTER
0185 00B6       CSNF04
0186 00B6 0693          BL    *R3
0187 00B8 0288          CI    CC,X3000
     00BA 0030
0188 00BC 1A03          JL    CSNF05          TOO SMALL FOR DIGIT
0189 00BE 0288          CI    CC,X3900
     00C0 0039
0190 00C2 12F9          JLE   CSNF04          IN RANGE, KEEP LOOKING
0191 00C4 8086  CSNF05  C     TP,TP1
0192 00C6 13B5          JEQ   CSN10           NUMBER HAS NO DIGITS *BAD*
0193                    *
0194                    * LOOK FOR EXPONENT OR END OF NUMBER
0195                    *
0196 00C8 C086  CSNG    MOV   TP,TP1        POINTER TO LAST CHAR.+2
0197 00CA 04C4          CLR   R4                    .
0198 00CC 0602          DEC   TP1
0199 00CE 04C1          CLR   R1
0200 00D0 0288          CI    CC,>45          IS NEXT CHARACTER A 'E'
     00D2 0045
0201 00D4 160F          JNE   CSNH            NO, EXPONENT DEFAULT: EXP=0
0202 00D6 0693          BL    *R3
0203                    *                          GET A '+'
0204 00D8 0288          CI    CC,XPLUS
     00DA 002B
0205 00DC 1306          JEQ   CSNG03          IGNORE + SIGN
```

```
0206                    *                               GET A '-'
0207 00DE 0288          CI    CC,XMINUS
     00E0 002D
0208 00E2 1602          JNE   CSNG02            NO MINUS SIGN
0209 00E4 0601          DEC   R1                 CHANGE SIGN TO NEGATIVE
0210 00E6 1001          JMP   CSNG03
0211                    *
0212 00E8 0606  CSNG02  DEC   TP                BACK UP
0213 00EA 06A0  CSNG03  BL    CSINT             GET INTEGER VALUE
     00EC 0002'
0214 00EE D041          MOVB  R1,R1
0215 00F0 1301          JEQ   CSNH
0216 00F2 0504          NEG   R4
0217                    *
0218                    * PACK FRACTION INTO FAC
0219                    *
0220 00F4 0606  CSNH    DEC   TP                POINT TO FIRST CHARACTER AFTER
0221 00F6 C806          MOV   TP,@FAC+12        STORE FIRST NON CONVERTED POSITI
     00F8 8356
0222 00FA 808C          C     TPNZ,TP1        WAS THE FRACTION ZERO?
0223 00FC 139A          JEQ   CSNZER            YES, NUMBER IS ZERO
0224 00FE 0224          AI    R4,128
     0100 0080
0225 0102 04C1          CLR   R1                CLEAR WORK REGISTER
0226 0104 A107          A     R7,R4             COPY EXPONENT FOR PLACE FLAG
0227 0106 C1C4          MOV   R4,R7
0228 0108 0814          SRA   R4,1              BASE 100
0229 010A C804          MOV   R4,@EXP
     010C 8376
0230 010E 0B17          SRC   R7,1
0231 0110 0205          LI    R5,8              INITIALIZE LOOP
     0112 0008
0232 0114 0200          LI    R0,FAC+1
     0116 834B
0233 0118 C18C          MOV   TPNZ,TP
0234 011A 8086  CSNH01  C     TP,TP1            END OF FRACTION?
0235 011C 130F          JEQ   CSNH03            YES ZERO UNUSED FRACTION
0236 011E 0693          BL    *R3             GET NEXT CHARACTER
0237                    *                             '.'
0238 0120 0288          CI    CC,XPOINT
     0122 002E
0239 0124 13FA          JEQ   CSNH01            IGNORE THE DECIMAL POINT
0240                    *                             '0'
0241 0126 0228          AI    CC,-'0'           ASCII TO BINARY
     0128 FFD0
0242 012A 0547          INV   R7
0243 012C 1105          JLT   CSNH02            1'S DIGIT
0244 012E 3A20          MPY   @C10,R8
     0130 0022'
0245 0132 D060          MOVB  @R9LB,R1          SAVE
     0134 83F3
0246 0136 10F1          JMP   CSNH01            GET ONE'S DIGIT
0247 0138 B060  CSNH02  AB    @R8LB,R1          ADD ONES AND TENS DIGIT
     013A 83F1
0248 013C DC01  CSNH03  MOVB  R1,*R0+           STORE DIGIT
0249 013E 04C1          CLR   R1              IN CASE NUMBER ENDS
0250 0140 0605          DEC   R5                MORE?
0251 0142 16EB          JNE   CSNH01            YES
0252 0144 0460  CSNH04  B     ROUN1             RETURN
     0146 0000
```

```
0253                * ROUTINE ADDRESSES  AND THEIR FLTPT NUMBERS
0254                *
0255                *                16  17   18  19   20    21    22
0256 0148 0056'     XTAB    DATA    CSN,CSNGR,CFI,SYMB,SMSB,ASSGNV,FBSYMB
     014A 004A'
     014C 0160'
     014E 0000
     0150 0000
     0152 0000
     0154 0000
0257                *
025                 *                 23   24    25   26    27
0259 0156 0000              DATA    VPUSHG,VPOP,SROM,SGROM,PGMCH
     0158 0000
     015A 0000
     015C 0000
     015E 0000
```

```
0261                    *
0262                    *
0263                    *       FLOATING TO INTEGER CONVERSION
0264                    *
0265                    *       CALL BL CFI         FAC HAS FLOATING NUMBER
0266                    *                           RETURNS WITH INTEGER IN FAC'S FIRS
0267                    *                           INTEGER IS 16 BIT TWO'S COMPLEMENT
0268                    *
0269                    *       USES R0-R6
0270                    *
0271          0160' CFI     EQU $
0272 0160 C120              MOV  @FAC,R4        IS FAC ZERO ?
     0162 834A
0273 0164 1342              JEQ  CFISI1         YES- ALL DONE
0274 0166 04C0              CLR  R0             ZERO RESULT IN CASE FAC=0
0275 0168 0202              LI   R2,FAC+1       GET POINTER RO HIGH ORDER BYTE OF
     016A 8343
0276 016C 04C3              CLR  R3             CLEAR HI BYTE OF CURRENT FRACTION
0277 016E 0760              ABS  @FAC            MAKE SURE FIRST DIGIT IS POSITIVE
     0170 834A
0278 0172 04C5              CLR  R5             CLR LOW BYTE OF WHERE EXPONENT WIL
0279 0174 D160              MOVB @FAC,R5          GET EXPONENT
     0176 834A
0280 0178 0285              CI   R5,>3F00       IS NUMBER LESS THAN ONE
     017A 3F00
0281 017C 1134              JLT  CFIRI          YES RESULT IS < .01, RESULT = 0
0282 017E 1318              JEQ  CFI03          .01<NUMBER<1 , RESULT = 1
0283 0180 0285              CI   R5,>4100       IS NUMBER LESS THAN 100000?
     0182 4100
0284 0184 1112              JLT  CFI02          IT IS BETWEEN 1 AND 100
0285 0186 1308              JEQ  CFI01          IT IS BETWEEN 100 AND 10000
0286 0188 0285              CI   R5,>4200       IS NUMBER TOO BIG TO CONVERT
     018A 4200
0287 018C 1325              JH   CFI08          TOO BIG, ERROR
0288 018E D832              MOVB *R2+,@R0LB      GET DIGIT
     0190 83E1
0289 0192 3820              MPY  @LW100,R0       MULTIPLY BY RADIX TO CONVERT TO B
     0194 01C8'
0290 0196 C001              MOV  R1,R0          GET RESULT OF MULTIPLY FOR NEXT DI
0291 0198 D832  CFI01       MOVB *R2+,@R3LB      GET NEXT DIGIT
     019A 83E7
0292 019C A003              A    R3,R0          ADD TO PREVIOUS RESULT
0293 019E 3820              MPY  @LW100,R0       MULTIPLY BY RADIX
     01A0 01C8'
0294 01A2 C000              MOV  R0,R0          TEST FOR OVERFLOW
0295 01A4 1619              JNE  CFI08          YES- ERROR
0296 01A6 C001              MOV  R1,R0          NO- GET RESULT FOR LAST DIGIT
0297 01A8 1117              JLT  CFI08          OVERFLOW IF HI BIT SET
029  01AA D832  CFI02       MOVB *R2+,@R3LB      GET LAST RADIX DIGIT TO LEFT OF F
     01AC 83E7
0299 01AE A003              A    R3,R0          ADD IT TO RESULT
0300 01B0 9832  CFI03       CB   *R2+,@LW50H     IS ROUNDING NECESSARY?
     01B2 0000'
0301 01B4 110B              JLT  CFI06          NO PUT ON PROPER SIGN
0302 01B6 1509              JGT  CFI05          YES ADD A 1 TO IT
0303 01B8 C104              MOV  R4,R4          MAYBE--?????
0304 01BA 1507              JGT  CFI05          NUMBER IS POSITIVE  ROUND UP
0305 01BC D0F2  CFI04       MOVB *R2+,R3        GET NEXT RADIX DIGIT
0306 01BE 1605              JNE  CFI05          NONZERO ROUND UP
0307 01C0 0282              CI   R2,FAC+8       LOOK AT REST OF DIGITS
```

```
      01C2 8352
0308  01C4 1AFB          JL    CFI04          NO LOOK AT NEXT ONE
0309  01C6 1002          JMP   CFI06          ROUND DOWN
0310       8000  SGNBIT  EQU   >8000
0311  01C8 0064  LW100   DATA  100
0312  01CA 0580  CFI05   INC   R0             ROUND UP
0313  01CC 0280  CFI06   CI    R0,SGNBIT      IS RESULT 32769 ??
      01CE 8000
0314  01D0 1A07          JL    CFIRSI         NO PUT ON THE PROPER SIGN
0315  01D2 1B02          JH    CFI08          NO IT IS GREATER--OVERFLOW
0316  01D4 C104          MOV   R4,R4          IS NUMBER NEGATIVE ?
0317  01D6 1106          JLT   CFIRS2         YES, PUT ON CORRECT SIGN
0318       01D8' CFI08   EQU   $              OVERFLOW RETURN TO ERROR LOCATION
0319  01D8 D820          MOVB  @ERRI,@FAC+10  ERROR CODE
      01DA 0001'
      01DC 8354
0320  01DE 045B          RT                   RETURN
0321               *
0322       0003  ERRIOV  EQU   >03             INTEGER OVERFLOW ERROR CODE
0323               *
0324  01E0 0544  CFIRSI  INV   R4             IS NUMBER NEGATIVE?
0325  01E2 1101          JLT   CFIRI          NO RETURN POSITIVE NUMBER
0326  01E4 0500  CFIRS2  NEG   R0             RETURN NEGATIVE NUMBER
0327  01E6 C800  CFIRI   MOV   R0,FAC         RETURN NUMBER IN FAC
      01E8 834A
0328       01EA' CFISI1  EQU   $
0329  01EA 045B          RT
0330                     END
NO ERRORS,      NO WARNINGS
```

```
FLTNDX      8354    A0017
FZERO   R   0034'    0003  0109
GETCH       0020+   A0048  0132
GETCH1      002E+   A0053 A0060
GETCHG      0032+   A0056  0129
GRMRA   R           A0035
GRMWA   R   003A+   A0035 A0056 A0057
GROMFL      8389     0057  0127
I           0004    A0005
J           0005    A0006 A0037 A0042 A0044
JOYX        8377    A0027
JOYY        8376    A0021
K           0006    A0007 A0038 A0043
L           0007    A0008 A0040 A0043
LLB         83EF    A0031
LW100       01C8'    0311  0289  0293
LW50H       0000'    0025  0300
N           0008    A0009
OVEXP1  R   0048'    0003  0120
PAD         8300    A0004 A0015 A0019 A0020 A0021 A0022 A0025 A0026 A0027 A0028
                          A0029 A0030 A0031  0013  0014  0015  0016  0017  0018
                           0019  0020  0021  0022  0023  0024  0057
PGMCH   R   015E'    0008  0259
POPSTK      0000+   A0037
PROA        8310    A0022
R0          0000           0085  0092  0102  0102  0232  0248  0274  0289  0290
                           0292  0293  0294  0294  0296  0299  0312  0313  0326
                           0327
R0LB        83E1    0013   0288
R1          0001           0119  0199  0209  0214  0214  0225  0245  0247  0248
                           0249  0290  0296
R10         000A           0111  0133
R10LB       83F5    0023
R11         000B           0086  0133
R11LB       83F7    0024
R12         000C           0056
R13         000D          A0056 A0057 A0059
R15         000F          A0039 A0041 A0048 A0050
R1LB        83E3    0014
R2          0002           0055  0275  0288  0291  0298  0300  0305  0307
R2LB        83E5    0015
R3          0003           0097  0127  0129  0132  0135  0145  0156  0177  0186
                           0202  0236  0276  0292  0299  0305
R3LB        83E7    0016   0291  0298
R4          0004           0084  0089  0090  0090  0094  0197  0216  0224  0226
                           0227  0228  0229  0272  0303  0303  0316  0316  0324
R4LB        83E9    0017
R5          0005           0093  0094  0231  0250  0278  0279  0280  0283  0286
R5LB        83EB    0018
R6          0006          A0050 A0051 A0056 A0058  0054
R6LB        83ED    0019 A0048 A0057
R7          0007           0136  0142  0149  0152  0155  0172  0172  0175  0226
                           0227  0230  0242
R7LB        83EF    0020
R8          0008          A0024  0093  0244
R8LB        83F1    0021   0247
R9          0009           0086  0104  0105
R9LB        83F3    0022   0245
ROUN1   R   0146'    0003  0252
ROUNUP  R           0003
```

```
SGNBIT      8000    0310  0313
SGROM   R   015C'   0009  0259
SIGN        8375   A0026  0149
SMBB    R   0150'   0006  0256
SROM    R   015A'   0009  0259
STATUS      837C   A0029
STKADD      8373   A0025
STKMOV      0018+  A0043 A0045
SYMB    R   014E'   0006  0256
TP          0006    0054  0115  0116  0134  0137  0150  0180  0184  0191  0196
                          0212  0220  0221  0233  0234
TP1         0002    0055  0117  0137  0143  0170  0191  0196  0198  0222  0234
TPNZ        000C    0056  0117  0150  0151  0184  0222  0233
VDPRD       8800   A0013 A0040 A0052
VDPWD       8C00   A0014
VPOP    R   0158'   0007  0259
VPUSHQ  R   0156'   0007  0259
VROA        03C0   A0023
VSPTR       826E   A0020 A0039 A0041 A0042
WRVDP   R          A0035
WS          83E0   A0030
X3000       0030    0028  0147  0158  0178  0187
X3900       0039    0031  0160  0189
XMINUS      002D    0030  0140  0207
XPLUS       002B    0029  0138  0204
XPOINT      002E    0032  0163  0238
XTAB    D   0148'   0256  0010
```

```
SOURCE ACCESS NAME=       BASC1.TI994A.V080581.SRC.PARSE
OBJECT ACCESS NAME=       PCD2.AEM.OBJ4A.PARSE
LISTING ACCESS NAME=      PCD2.AEM.LST4A.PARSE
ERROR ACCESS NAME=        PCD2.AEM.SRC.ERROR
OPTIONS=                  XREF
MACRO LIBRARY PATHNAME=   PCD2.AEM.OBJ4A.PMACS1


LINE      KEY   NAME

 0001      LI   PCD2.AEM.OBJ4A.PMACS1
                    =>PCD2.AEM.OBJ4A.PMACS1
 0021       A   FPEQ
                    =>BASC1.TI994A.V080581.SRC.FPEQ
```

```
      0006 835C
 0039 0008 D7E0          MOVB @VSPTR+1,*R15 LOAD VDP ADDRESS
      000A 836F
A0040 000C 0207          LI   L,VDPRD       ADDRESS TO READ VDP
      000E 8800
A0041 0010 D7E0          MOVB @VSPTR,*R15   MSB OF ADDRESS
      0012 836E
A0042 0014 A805          A    J,@VSPTR      DECREMENT STACK COUNTER
      0016 836E
A0043 0018 DD97  STKMOV  MOVB *L,*K+        RECALL BYTE FROM VDP
A0044 001A 0585          INC  J             INCREMENT LOOP COUNTER
A0045 001C 16FD          JNE  STKMOV        8 BYTES?
A0046 001E 045B          RT
A0047              *
A0048 0020 D7E0  GETCH   MOVB @R6LB,*R15          LOAD VDP ADDRESS
      0022 83ED
A0049 0024 1000          NOP
A0050 0026 D7C6          MOVB R6,*R15             THE ZERO DISP IS FOR TIMING
A0051 0028 0586          INC  R6                  NEXT ADDRESS
A0052 002A D220          MOVB @VDPRD,CC
      002C 8800
A0053 002E 0988  GETCH1  SRL  CC,8           SHIFT TO LOWER BYTE
A0054 0030 045B          RT                       RETURN
A0055              *
A0056 0032 DB46  GETCHG  MOVB R6,@GRMWA(R13)
      0034 0000
A0057 0036 DB60          MOVB @R6LB,@GRMWA(R13)
      0038 83ED
      003A 0034+
A0058 003C 0586          INC  R6
A0059 003E D21D          MOVB *R13,CC
A0060 0040 10F6          JMP  GETCH1
A0061              *
A0062 0042               CEND
 0022              *
 0023                    DEF  PARSEG,CONTG,EXECG,RTNG
 0024                    DEF  VPUSH,VPOP,PSHPRS
 0025                    DEF  SETREG,SAVREG,PGMCHR,ERR1
 0026                    DEF  EXEC50,PARSE,BASE
 0027                    DEF  ERRSYN,ERR,CALGPL
 0028              *
 0029                    REF  CSNGR,CFI,NEXT
 0030                    REF  SCOMPB,SADD,SSUB,SMULT,SDIV
 0031                    REF  GROMA,STVDP3
 0032                    REF  SYM,SMB,GETV,GETV1,MOVFAC,ASSGNV
 0033                    REF  RESET,ASSG,GETSTK,PUTSTK
 0034                    REF  ERRT
 0035                    REF  ENTRY
 0036              *
 0037       0020 BRKKEY  EQU  >20                 BREAK KEY
 003        83E1 R0LB    EQU  PAD+>E1
 0039       83E3 R1LB    EQU  PAD+>E3
 0040       83ED R6LB    EQU  PAD+>ED
 0041       83EF R7LB    EQU  PAD+>EF
 0042       83F3 R9LB    EQU  PAD+>F3
 0043       83F5 R10LB   EQU  PAD+>F5
 0044       83F7 R11LB   EQU  PAD+>F7
 0045              *
 0046       831A STREND  EQU  PAD+>1A            END OF STRING SPACE
 0047       831C SREF    EQU  PAD+>1C            TEMPORARY STRING PTR
```

```
0048       8322  ERRCOD EQU  PAD+>22          RETURN CODE FROM EXEC.
0049       8324  STVSPT EQU  PAD+>24          BASE OF VALUE STACK (-8)
0050       8326  RTNADD EQU  PAD+>26          ADDRESS TO RETURN TO IN GPL
0051       8328  NUDTAB EQU  PAD+>28          POINTER TO NUD TABLE
0052       832C  PGMPTR EQU  PAD+>2C          BASIC PROGRAM TEXT POINTER
0053       832E  EXTRAM EQU  PAD+>2E          LINE BUFFER POINTER
0054       8330  STLN   EQU  PAD+>30          LAST LINE POINTER IN L.N. BUFF
0055       8332  ENLN   EQU  PAD+>32          FIRST LINE POINTER IN L.N. BUF
0056       8340  SYMPTR EQU  PAD+>40          POINTER TO HIGHEST FREE BYTE (
0057       8342  CHAT   EQU  PAD+>42          ONE CHARACTER BUFFER
0058       8343  BASE   EQU  PAD+>43          OPTION BASE
0059       8344  BUFFY  EQU  PAD+>44          IMPERATIVE INDICATOR
0060       8388  FLAG   EQU  PAD+>88          EXECUTION FLAGS
0061       8389  GROMFL EQU  PAD+>89          GROM/VDPRAM(0) FLAG
0062       83BA  STKEND EQU  PAD+>BA          END OF SUBR. STACK (2 SPARE E
0063             *
0064       00C9  LN$    EQU  >C9              GROM LINE NUMBER TOKEN
0065       0085  GO$    EQU  >85              GO TOKEN
0066       0086  GOTO$  EQU  >86              GOTO TOKEN
0067       0087  GOSUB$ EQU  >87              GOSUB TOKEN
0068       008D  LET$   EQU  >8D              LET TOKEN VALUE
0069       0081  ELSE$  EQU  >81              ELSE TOKEN
0070       00B1  TO$    EQU  >B1              TO TOKEN
0071       00B0  THEN$  EQU  >B0              THEN TOKEN
0072       00C8  NUM$   EQU  >C8              NUMERIC CONSTANT
0073       00A1  SUB$   EQU  >A1              SUB TOKEN
0074       00B3  COMMA$ EQU  >B3              COMMA TOKEN
0075       00B8  CONC$  EQU  >B8              CONCATENATE (&)
0076       00BE  EQ$    EQU  >BE              EQUAL TOKEN
0077       00C0  GT$    EQU  >C0              GREATER THAN TOKEN
0078       00C1  PLUS$  EQU  >C1              PLUS TOKEN
0079       00C2  MINUS$ EQU  >C2              MINUS TOKEN
0080       00C3  MULT$  EQU  >C3              MULTIPLY TOKEN
0081       00C4  DIVI$  EQU  >C4              DIVIDE TOKEN
0082       00C5  EXPON$ EQU  >C5              EXPONENTIATION TOKEN
0083             *
0084             *       DATA CONSTANTS
0085             *
0086             *       OTHERS ARE LOCATED THROUGHOUT THE CODE
0087             *
0088 0000 0104' EXRTNA DATA EXRTN         RETURN FOR EXEC
```

```
 0090                   *
 0091                   *       GRAPHICS LANGUAGE ENTRY TO PARSE
 0092                   *
 0093 0002 06A0  PARSEG BL    @SETREG
      0004 05B4'
 0094 0006              MAC1    ^^^^^^^^^^^^^^^^
*0001 0006 D2ED         MOVB @GRMRA(R13),R11
      0008 0000
*0002 000A D82D         MOVB @GRMRA(R13),@R11LB
      000C 0008'
      000E 83F7
*0003 0010 022B         AI    R11,>7FFF       ADD MSBY - 1
      0012 7FFF
 0095                   *
 0096                   *       9985 ENTRY TO PARSE
 0097                   *
 0098 0014 05C9  PARSE  INCT R9                 PUT RETURN ADDRESS ON STACK
 0099 0016 0289         CI    R9,STKEND        STACK FULL ?
      0018 83BA
 0100 001A 1B1D         JH    P20
 0101 001C C64B         MOV   R11,*R9
 0102 001E D1C8  P05    MOVB  R8,R7            TEST CURRENT CHARACTER
 0103 0020 1102         JLT   P10
 0104 0022 0460         B     @PSYM            IF NOT TOKEN
      0024 02CE'
 0105                   *
 0106 0026 06A0  P10    BL    @PGMCHR          GET NEXT CHARACTER
      0028 06B8'
 0107      002A' CB9    EQU   $          ***** 1-BYTE CONSTANT   '9'
 0108 002A 0977         SRL   R7,7             CHANGE LAST CHAR. TO OFFSET
 0109 002C 0227         AI    R7,->B7*2
      002E FE92
 0110 0030 0287         CI    R7,NTABLN
      0032 004C
 0111 0034 1328         JH    CONT15
 0112 0036 C1E7         MOV   @NTAB(R7),R7     GET NUD ADDRESS
      0038 041C'
 0113 003A 152B         JGT   B9985            IF 9985 CODE
 0114 003C 0247  P17    ANDI  R7,>7FFF         IF GPL CODE, GET RID OF MSB
      003E 7FFF
 0115 0040 A1E0         A     @NUDTAB,R7       ADD IN TABLE ADDRESS
      0042 8328
 0116 0044              NUDGO5
 0117 0044 06A0         BL    @SAVREG          RESTORE GPL POINTERS
      0046 05C6'
 0118 0048              MAC2    ^^^^^^^^^^^^^^^^
*0001 0048 DB47         MOVB R7,@GRMWA(R13)
      004A 003A+
*0002 004C DB60         MOVB @R7LB,@GRMWA(R13)
      004E  3EF
      0050 004A'
 0119 0052 0460         B     @RESET           GO BACK TO GPL
      0054 0000
 0120                   *
 0121 0056 0460  P20    B     @VPSH23
      0058 065C'
 0122                   *
```

```
 0124                    *
 0125                    *       CONTINUE ROUTINE FOR PARSE
 0126                    *
 0127 005A 06A0  CONTC   BL    @SETREG
      005C 05B4'
 0128 005E       CONT
 0129 005E C199          MOV   *R9,R6              GET LAST ADDRES FROM STACK
 0130 0060 1508          JGT   CONT10              990 CODE IF NOT NEG.
 0131 0062 0246          ANDI  R6,>7FFF            GET GROM ADDRESS
      0064 7FFF
 0132 0066               MAC3    ^^^^^^^^^^^^^^^^
*0001 0066 DB46          MOVB  R6,@GRMWA(R13)
      0068 0050'
*0002 006A DB60          MOVB  @R6LB,@GRMWA(R13)
      006C 83ED
      006E 0068'
*0003 0070 C18D          MOV   R13,R6
 0133                    *
 0134 0072 9216  CONT10  CB    *R6,R8              TEST PRECEDENCE
 0135 0074 1455          JHE   NUDEND
 0136 0076-0288          CI    R8,CONC$*256        CONCATENATE(&)?
      0078 B800
 0137 007A 130C          JEQ   CONT20              YES-CHEAT ON TABLES
 0138 007C 0978          SRL   R8,7                TABLE OFFSET
 0139 007E 0228          AI    R8,->BE*2           MIN TOKEN FOR LED (*2)
      0080 FE84
 0140 0082 0288          CI    R8,LTBLEN           MAX TOKEN FOR LED (*2)
      0084 0010
 0141 0086 1B6F  CONT15  JH    NOLED
 0142 0088 C1E8          MOV   @LTAB(R8),R7
      008A 0468'
 0143 008C 04C8          CLR   R8
 0144 008E 06A0          BL    @PGMCHR             GET NEXT CHARACTER
      0090 06B8'
 0145 0092 0457  B9985   B     *R7                 GO TO ROUTINE
 0146 0094 0200  CONT20  LI    R0,CONCAT           GO TO GRAPHICS
      0096 0008
 0147 0098 1068          JMP   EXIT                LANGUAGE
 0148                    *
 0149                    *
 0150 009A 0649  NUDE10  DECT  R9                  BACK UP STACK.
 0151 009C 0227          AI    R7,>8001            GET RID OF GROM FLAG AND
      009E 8001
 0152                    *                         SKIP OVER PRECEDENCE
 0153 00A0 10D1          JMP   NUDG05              MERGE WITH OTHER CODE TO RETU
 0154                    *
 0155                    *
```

```
0190 0100 119D          JLT  P17                IF GROM CODE
0191 0102 0457          B    *R7                IF 9985 CODE
0192                *
0193 0104    00  EXRTN   BYTE 0
0194 0105    65  CBH65   BYTE >65               UNUSED BYTE FOR CONSTANT
0195 0106 C020          MOV  @BUFFY,R0          IMPERATIVE MODE ?
     0108 8344
0196 010A 1331          JEQ  EXEC50             YES
0197 010C 6820          S    @C4,@EXTRAM        NO, GO TO NEXT LINE
     010E 0488'
     0110 832E
0198 0112 8820          C    @EXTRAM,@STLN      END OF PROGRAM ?
     0114 832E
     0116 8330
0199 0118 14CB          JHE  EXEC10             NO, LOOP FOR NEXT LINE
0200 011A 1029          JMP  EXEC50             YES, QUIT PROGRAM
0201                *
```

```
0203                    *     TEST FOR REQUIRED END-OF-LINE AFTER A STATEMENT.
0204 011C D208          EOL     MOVB R8,R8            EOL TOKEN?
0205 011E 1623                  JNE  ERR1
0206                    *
0207                    *       RETURN FROM CALL TO PARSE
0208                    *            (ENTERED FROM CONT)
0209                    *
0210 0120 C1D9          NUDEND MOV   *R9,R7             GET RETURN ADDRESS.
0211 0122 11BB                 JLT   NUDE10             RETURN TO GPL.
0212 0124 0649                 DECT  R9                 BACK UP STACK.
0213 0126 0467                 B     @2(R7)
     0128 0002
0214                    *
0215                    *       RETURN FROM "CALL" TO GPL
0216                    *
0217 012A               RTNG
0218 012A 06A0                 BL    @SETREG
     012C 05B4'
0219 012E 10F8                 JMP   NUDEND
0220                    *
```

```
0253                    *
0254                    *       BREAKPOINT
0255                    *
0256 015A               BRKPNT
0257 015A D020                  MOVB @GROMFL,R0         GROM PROGRAM ?
     015C 8389
0258 015E 16BB                  JNE  EXEC14             YES, IGNORE BREAKPOINT
0259 0160 0200          BRKPN1  LI   R0,BRKFL           BREAKPOINT RETURN VECTOR
     0162 0001
0260 0164 1002                  JMP  EXIT
0261                    *
0262                    *       ERRORS
0263                    *
0264 0166               ERRSYN
0265 0166               ERR1
0266 0166               NONUD
0267 0166               NOLED
0268 0166 0200                  LI   R0,ERRSN           SYNTAX ERROR CODE
     0168 0003
0269 016A               EXIT
0270 016A C800          ERR     MOV  R0,@ERRCOD
     016C 8322
0271                    *
0272                    *       GENERAL RETURN TO BASIC
0273                    *
0274 016E C1E0          EXEC50  MOV  @RTNADD,R7         RETURN ADDRESS
     0170 8326
0275 0172 0460                  B    @NUDGO5            USE COMMON CODE TO LINK BACK
     0174 0044'
0276                    *
0277 0176               STOP
0278 0176 0649          END     DECT R9                 POP LAST CALL TO PARSE
0279 0178 10FA                  JMP  EXEC50
0280                    *
0281                    *       ERROR CODES
0282                    *
0283      0003          ERRSN   EQU  >0003              ERROR SYNTAX
0284      0103          ERROM   EQU  >0103              ERROR OUT OF MEMORY
0285      0203          ERRIOR  EQU  >0203              ERROR INDEX OUT OF RANGE
0286      0303          ERRLNF  EQU  >0303              ERROR LINE NOT FOUND
0287      0403          ERREX   EQU  >0403              ERROR - EXECUTION
0288      0503          ERRBS   EQU  >0503              ERROR - BAD SUBSCRIPT
0289      0603          ERRTM   EQU  >0603              ERROR - STRING/NUMBER MISMATCH
0290                    *                           >0004 WARNING NUMERIC OVERFLOW
0291      0005          LEXPON  EQU  >0005                LED FOR EXPONENTIATION (IN GPL)
0292      0001          BRKFL   EQU  >0001                BREAKPOINT RETURN VECTOR
0293                            REF  C2               >0002 TRACE MODE RETURN VECTOR
0294      0006          NUDD2   EQU  >0006                FUNCTION REFERENCE
0295      0007          GETSTR  EQU  >0007                SYSTEM GET STRING
0296      000           CONCAT  EQU  >0008                CONCATENATE (&) STRINGS
0297      002A'         COMPCT  EQU  CB9                  GARBAGE COLLECTION
0298                    *
0299                    *
0300                    *       WARNING ROUTINE (ONLY OVERFLOW)
0301                    *
0302 017A C820          WARN$$  MOV  @C4,@ERRCOD        ERROR CODE FOR GPL
     017C 0488'
     017E 8322
0303 0180 0208                  LI   R11,CONT-2         TO OPTIMIZE
     0182 005C'
```

```
0304                    *
0305                    *       RETURN TO GPL AS CALL
0306                    *
0307 0184 05C9  CALGPL  INCT R9
0308 0186 C643          MOV   R11,*R9           STACK RETURN
0309 0188 10F2          JMP   EXEC50
0310                    *
0311                    *
0312                    *       TRACE A LINE (CALL GPL ROUTINE)
0313                    *
0314 018A C820  TRACE   MOV   @C2,@ERRCOD       RETURN VECTOR
     018C 0000
     018E 8322
0315 0190 020B          LI    R11,EXEC11-2      RETURN ENTRY IN 9985
     0192 00C8'
0316 0194 10F7          JMP   CALGPL
0317                    *
```

```
0338                    *
0339                    *       NUD ROUTINE FOR "GO"
0340                    *
0341 01C8 04C3  GO      CLR   R3                DUMMY "ON" INDEX
0342 01CA 1017          JMP   ON30              MERGE WITH "ON" CODE
0343                    *
0344                    *
0345                    *       NUD ROUTINE FOR "ON"
0346                    *
0347 01CC       ON
0348 01CC 06A0          BL    @PARSE            PARSE EXPRESSION FOR VALUE
     01CE 0014'
0349 01D0   B3          BYTE  COMMA$
0350 01D1   66  CBH66   BYTE  >66               UNUSED BYTE FOR CONSTANT
0351 01D2 06A0          BL    @NUMCHK           ENSURE IT'S A NUMBER
     01D4 05AA'
0352 01D6 04E0          CLR   @FAC+10           CLEAR ERROR BYTE FOR CFI
     01D8 8354
0353 01DA 06A0          BL    @CFI              CONVERT TO INTEGER
     01DC 0000
0354 01DE D020          MOVB  @FAC+10,R0        TEST ERROR CODE
     01E0 8354
0355 01E2 1603          JNE   GOTO90            IF OVERFLOW
0356 01E4 C0E0          MOV   @FAC,R3           GET VALUE
     01E6 834A
0357 01E8 1503          JGT   ON20              MUST BE POSITIVE
0358 01EA 0200  GOTO90  LI    R0,ERRIOR         NOT NEGATIVE
     01EC 0203
0359 01EE 10BD  GOTO95  JMP   ERR
0360                    *
0361 01F0       ON20
0362 01F0-0288          CI    R8,GO$*256        BARE "GO" ?
     01F2 8500
0363 01F4 1608          JNE   ON40              NO, CHECK OTHER POSSIBILITIES
0364 01F6 06A0          BL    @PGMCHR           YES, GET NEXT CHARACTER
     01F8 06B8'
0365 01FA-0288  ON30    CI    R8,TO$*256        "GO TO" ?
     01FC B100
0366 01FE 1353          JEQ   GOTO50            YES
0367 0200-0288          CI    R8,SUB$*256       "GO SUB" ?
     0202 A100
0368 0204 1005          JMP   ON50              MERGE CODE
0369                    *
0370 0206-0288  ON40    CI    R8,GOTO$*256      "GOTO" ?
     0208 8600
0371 020A 134D          JEQ   GOTO50            YES
0372 020C-0288          CI    R ,GOSUB$*256     "GOSUB" ?
     020E 8700
0373 0210 16AA  ON50    JNE   ERR1              NO, SYNTAX ERROR
0374 0212 06A0          BL    @PGMCHR
     0214 06B8'
0375 0216 1002          JMP   GOSUB2
0376                    *
0377 0218 10A6  ERR1B   JMP   ERR1              SYNTAX ERROR
```

```
 0379                    *
 0380                    *       NUD ROUTINE FOR "GOSUB"
 0381                    *
 0382 021A 04C3  GOSUB   CLR   R3                    DUMMY COUNTER FOR "ON" CODE
 0383                    *
 0384                    *       COMMON GOSUB CODE
 0385                    *
 0386 021C C820  GOSUB2  MOV   @EXTRAM,@FAC          SAVE CURRENT PROGRAM ADDRESS
      021E 832E
      0220 834A
 0387 0222 D820          MOVB  @CBH66,@FAC+2         INDICATE GOSUB ENTRY
      0224 01D1'
      0226 834C
 0388 0228 C803          MOV   R3,@FAC+6             SAVE "ON" COUNT
      022A 8350
 0389                    *                           IN CASE OF GARBAGE COLLECTION
 0390 022C 06A0          BL    @VPUSH                PUSH RETURN
      022E 05E4'
 0391 0230 C0E0          MOV   @FAC+6,R3             RESTORE COUNT
      0232 8350
 0392 0234 1001          JMP   GOTO20
 0393                    *
 0394                    *
 0395                    *       NUD ROUTINE FOR "GOTO"
 0396                    *
 0397 0236 04C3  GOTO    CLR   R3                    DUMMY INDEX FOR "ON" CODE
 0398                    *
 0399                    *       COMMON (ON) GOTO/GOSUB THEN/ELSE CODE
 0400                    *
 0401 0238       GOTO20
 0402                    *       GET LINE NUMBER FROM PROGRAM
 0403                    *
 0404 0238-0288          CI    R8,LN$*256            SPECIAL LINE NUMBER TOKEN ?
      023A C900
 0405 023C 16ED          JNE   ERR1B                 ELSE SYNTAX ERROR
 0406                    *
 0407 023E 06A0  GETL10  BL    @PGMCHR               GET MSByte OF LINE NUMBER
      0240 06B8'
 0408 0242 D008          MOVB  R8,R0                 SAVE IT
 0409 0244 06A0          BL    @PGMNXT               GET LSByte OF LINE NUMBER
      0246 06DA'
 0410                    *
 0411 0248       GETL30
 0412 0248 0603          DEC   R3                    COUNT FOR ON
 0413 024A 1528          JGT   GOTO40                LOOP IF NOT THERE
 0414                    *
 0415                    *       FIND PROGRAM LINE
 0416                    *
 0417 024C               MAC5  ^^^^^^^^^^^^^^^
*0001 024C C060          MOV   @STLN,R1              GET INTO LINE # BUFFER
      024E 8330
*0002 0250 D0A0          MOVB  @GROMFL,R2            WHERE DO WE GET THE ARGS FROM?
      0252 8389
*0003 0254 1307          JEQ   GOTO31                FROM RAM - ACT NATURALLY
*0004 0256 DB41          MOVB  R1,@GRMWA(R13)        WRITE OUT LOWER ADDRESS
      0258 006E'
*0005 025A C08D          MOV   R13,R2                GET GROM READ ADDR IN R2
*0006 025C DB60          MOVB  @R1LB,@GRMWA(R13)     PLUS WRITE OUT HIGH ADDRESS
      025E 83E3
      0260 0258'
```

```
*0007 0262 1005          JMP   GOTO32              CONTINUE IN COMMON CODE
*0008 0264 D7E0  GOTO31  MOVB  @R1LB,*R15          GET IT FROM THE VDP
      0266 83E3
*0009 0268 0202          LI    R2,VDPRD
      026A 8800
*0010 026C D7C1          MOVB  R1,*R15
*0011 026E       GOTO32
*0012 026E 8801          C     R1,@ENLN            FINISHED W/# BUFFER?
      0270 8332
*0013 0272 140B          JHE   GOTO34
*0014 0274 9012          CB    *R2,R0              COMPARE 1ST BYTE OF #-MATCH?
*0015 0276 1607          JNE   GOTO35              NOT A MATCH-MOVE ON
*0016 0278 9212          CB    *R2,R8              2ND BYTE MATCH?
*0017 027A 130A          JEQ   GOTO36              YES-LINE IS FOUND!
*0018 027C D0D2  GOTO33  MOVB  *R2,R3              SKIP 1ST BYTE OF LINE PTR
*0019 027E 0221          AI    R1,4                ADVANCE TO NEXT # IN BUFFER
      0280 0004
*0020 0282 D0D2          MOVB  *R2,R3              SKIP 2ND BYTE OF LINE PTR
*0021 0284 10F4          JMP   GOTO32
*0022 0286 D0D2  GOTO35  MOVB  *R2,R3              SKIP 2ND BYTE OF #
*0023 0288 10F9          JMP   GOTO33
 0418 028A 0200  GOTO34  LI    R0,ERRLNF           LINE NOT FOUND
      028C 0303
 0419 028E 10AF          JMP   GOTO95              ERROR EXIT
 0420             *
 0421 0290 05C1  GOTO36  INCT  R1                  ADJUST TO LINE PTR.
 0422 0292 C801          MOV   R1,@EXTRAM          SAVE FOR EXECUTE
      0294 832E
 0423 0296 0649          DECT  R9                  POP SAVED LINK TO GOTO
 0424 0298 0460          B     @EXEC10             REENTER EXEC CODE DIRECTLY
      029A 00B0'
 0425             *                          MUST BE CHANGED IF IMPERATIVE GOTO
 0426             *                          ETC. ARE ALLOWED.............
 0427             *
 0428 029C 06A0  GOTO40  BL    @PGMCHR
      029E 06B8'
 0429 02A0-0288          CI    R8,COMMA$*256       COMMA NEXT ?
      02A2 8300
 0430 02A4 16A2          JNE   GOTO90              NO, ERROR
 0431 02A6 06A0  GOTO50  BL    @PGMCHR             YES, GET NEXT CHARACTER
      02A8 06B8'
 0432 02AA 10C6          JMP   GOTO20              AND LOOP
 0433             *
 0434 02AC 10B5  ERR1C   JMP   ERR1B
```

```
                *
                *       NUD ENTRY FOR "RETURN"
                *
!AE             RETURN
!AE 06A0        RETU10  BL    @VPOP                 POP ENTRY
!B0 0668'
!B2 9820                CB    @CBH66,@FAC+2         FLAG FOR GOSUB ENTRY
!B4 01D1'
!B6 834C
!B8 16FA                JNE   RETU10                LOOP TILL FIND ONE
!BA D208                MOVB  R8,R8                 END OF LINE ?
!BC 16F7                JNE   ERR1C                 NO, ERROR
!BE C820                MOV   @FAC,@EXTRAM          GET RETURN TEXT POINTER
!C0 834A
!C2 832E
!C4 0460                B     @NUDEND               GO ADJUST IT
!C6 0120'
                *
```

```
0449 02C8 0200  SYMB20 LI   R0,NUDD2
     02CA 0006
0450 02CC 1090         JMP  GOTO95
0451              *
0452              ********************************************************
0453              *      SUBROUTINE FOR A SYMBOL(VARIABLE)
0454              ********************************************************
0455              *
0456              *
0457 02CE 06A0  PSYM   BL   @SYM              GET SYMBOL TABLE ENTRY
     02D0 0000
0458 02D2 06A0         BL   @GETV
     02D4 0134'
0459 02D6 834A         DATA FAC
0460 02D8 0A11         SLA  R1,1              FUNCTION REFERENCE?
0461 02DA 11F6         JLT  SYMB20            YES-SPECIAL CODE
0462 02DC 06A0         BL   @SMB              GET VALUE SPACE POINTER
     02DE 0000
0463 02E0 9820         CB   @FAC+2,@CBH65     STRING REFERENCE?
     02E2 834C
     02E4 0105'
0464 02E6 1302         JEQ  SYMB10            YES-SPECIAL CODE
0465 02E8 06A0         BL   @MOVFAC           GET VALUE SPACE INTO FAC
     02EA 0000
0466 02EC 0460  SYMB10 B    @CONT             CONTINUE PARSE
     02EE 005E'
0467              *
```

```
0469                    *
0470                    *       NUD ENTRY FOR IF STATEMENT
0471                    *
0472 02F0 06A0  IF      BL    @PARSE              EVALUATE EXPRESSION
     02F2 0014'
0473 02F4    B3         BYTE  COMMA$
0474 02F5    67  CBH67  BYTE  >67                 UNUSED BYTE FOR CONSTANT
0475 02F6 06A0          BL    @NUMCHK             ENSURE IT'S A NUMBER
     02F8 05AA'
0476 02FA 04C3          CLR   R3                  DUMMY ON INDEX
0477 02FC-0288          CI    R8,THEN$*256
     02FE B000
0478 0300 16D5          JNE   ERR1C               ERROR IF THEN NOT NEXT TOKEN
0479 0302 0520          NEG   @FAC                TEST IF TRUE I.E. NOT ZERO
     0304 834A
0480 0306 16CF          JNE   GOTO50              AND BRANCH TO LINE NUMBER
0481 0308 06A0          BL    @PGMCHR             ADVANCE TO LINE NUMBER TOKEN
     030A 06B8'
0482 030C-0288          CI    R8,LN$*256          SPECIAL LINE NUMBER TOKEN?
     030E C900
0483 0310 16CD          JNE   ERR1C               ERROR IF NOT LINE NUMBER
0484 0312 05E0          INCT  @PGMPTR             SKIP LINE NUMBER
     0314 832C
0485 0316 06A0          BL    @PGMCHR             GET NEXT CHAR. FROM TOKEN
     0318 06B8'
0486 031A-0288          CI    R8,ELSE$*256        TEST IF LUCKY TOKEN IS ELSE
     031C 8100
0487 031E 13C3          JEQ   GOTO50              IF SO BRANCH TO LINE NUMBER
0488 0320 0460          B     @EOL                MUST BE END OF LINE
     0322 011C'
0489                    *
```

```
0491                    *
0492                    *
0493                    ***************************************************
0494                    *       SUBROUTINE FOR 'LET'
0495                    ***************************************************
0496                    *
0497 0324 06A0  NLET    BL    @SYM              GET SYMBOL TABLE ADDRESS
     0326 02D0'
0498 0328 06A0          BL    @SMB              GET VALUE SPACE POINTER
     032A 02DE'
0499 032C-0288          CI    R8,EQ$*256        IS TOKEN AN '='?
     032E BE00
0500 0330 163D          JNE   ERR1C             NO-ERROR
0501 0332 06A0          BL    @PGMCHR           GET NEXT TOKEN
     0334 06B8'
0502 0336 06A0          BL    @PSHPRS           PUSH AND PARSE
     0338 05D6'
0503 033A   8D          BYTE  LET$
0504 033B   30  HHUSC   BYTE  >30               UNUSED BYTE FOR CONSTANT
0505 033C 06A0          BL    @ASSG             ASSIGN THE VARIABLE
     033E 0000
0506 0340 0460          B     @CONT             CONTINUE PARSE
     0342 005E'
0507 0344               BSS   10                BECAUSE OF THE 10 BYTES FUNC.
0508                    *                       BIT CHECKING WERE TAKEN OUT O
0509                    *                       LET HANDLER (ADDED IN SMB) AN
0510                    *                       THE ABSOLUTE ADDRESS SCHEME
0511                    *                       USED IN EXTENDED BASIC, THE
0512                    *                       ADDRESSES IN PARSE3S MUST B
0513                    *                       MAINTAINED.
```

```
0515                    *
0516                    *
0517                    ***************************************************************
0518                    *       SUBROUTINE FOR 'NEXT'
0519                    ***************************************************************
0520                    *
0521 034E 06A0  NNEXT   BL    @SYM                    GET S.T.  I.D.
     0350 0326'
0522 0352 C120          MOV   @FAC,R4
     0354 834A
0523 0356       NEXT2
0524 0356 06A0          BL    @VPOP                   GET 'FOR' ENTRY OFF STACK
     0358 0668'
0525 035A 9820          CB    @FAC+2,@CBH67           CHECK FOR 'FOR' ENTRY
     035C 834C
     035E 02F5'
0526 0360 1302          JEQ   NEXT3                   IS A 'FOR' ENTRY-ERROR
0527 0362 0460          B     @VPOP20                 NOT - ERROR
     0364 06B2'
0528                    *
0529 0366 8804  NEXT3   C     R4,@FAC                 CHECK IF MATCHING 'FOR' ENTRY
     0368 834A
0530 036A 1304          JEQ   NEXT4                   IS A MATCH
0531 036C 6820          S     @C16,@VSPTR             LOOP VARIABLES DON'T MATCH
     036E 03A2'
     0370 836E
0532 0372 10F1          JMP   NEXT2
0533 0374 06A0  NEXT4   BL    @MOVFAC                 GET INDEX VALUE
     0376 02EA'
0534 0378 06A0          BL    @SAVREG
     037A 05C6'
0535 037C 06A0          BL    @SADD                   ADD IN THE INCREMENT
     037E 0000
0536 0380 06A0          BL    @SETREG
     03 2 05B4'
0537 03 4 A820          A     @C16,@VSPTR
     03 6 03A2'
     03 8 836E
0538 038A 06A0          BL    @ASSG                   SAVE NEW INDEX VALUE
     038C 033E'
0539 038E 6820          S     @C8,@VSPTR              POINT TO THE LIMIT
     0390 05E6'
     0392  36E
0540 0394 06A0          BL    @SCOMPB                 TEST W/IN LIMIT
     0396 0000
0541 0398 02C4          STST  R4                      SAVE RESULT OF COMPARE
0542 039A 130A          JEQ   NEXT5                   IF = DO LAST LOOP
0543 039C C0E0          MOV   @VSPTR,R3               CHECK FOR A DECREMENT
     039  836E
0544       03A2' C16    EQU   $+2                     CONSTANT 16
0545 03A0 0223          AI    R3,16
     03A2 0010
0546 03A4 06A0          BL    @GETV1
     03A6 0000
0547 03A8 D041          MOVB  R1,R1                   CHECK IF A DECREMENT
0548 03AA 1112          JLT   NEXT6                   DECREMENT
0549 03AC 0A14          SLA   R4,1                    CHECK OUT OF LIMIT
0550 03AE 150E          JGT   NEXT8                   OUT OF LIMIT
0551 03B0 A820  NEXT5   A     @C24,@VSPTR             POINT TO I.D.
     03B2 0648'
```

```
      03B4 836E
 0552 03B6 C0E0          MOV   @VSPTR,R3          GOTO TOP OF 'FOR' LOOP
      03B8 836E
 0553 03BA 0223          AI    R3,6
      03BC 0006
 0554 03BE 06A0          BL    @GETV1
      03C0 03A6'
 0555 03C2               MACB
*0001 03C2 D820          MOVB  @VDPRD,@EXTRAM+1
      03C4 8800
      03C6 832F
 0556 03C8 D801          MOVB  R1,@EXTRAM
      03CA 832E
 0557 03CC 0460  NEXT8   B     @CONT              CONTINUE PARSE
      03CE 005E'
 0558                    *
 0559                    *       TEST LIMIT FOR DECREMENT
 0560                    *
 0561 03D0 0A14  NEXT6   SLA   R4,1               CHECK OUT OF LIMIT
 0562 03D2 15EE          JGT   NEXT5              W/IN LIMIT
 0563 03D4 10F3          JMP   NEXT8              CONTINUE PARSE
 0564                    *
```

```
0566                    *
0567                    *       STATMENT TABLE
0568                    *
0569 03D6 0166'         DATA NONUD              (SPARE) (80)
0570 03D8 0166'         DATA NONUD              ELSE
0571 03DA 0166'         DATA NONUD              (RESERVED FOR SR-62)
0572 03DC 0166'         DATA NONUD              (RESERVED FOR SR-62)
0573 03DE 02F0'         DATA IF                 IF  (84)
0574 03E0 01C8'         DATA GO                 GO
0575 03E2 0236'         DATA GOTO               GOTO
0576 03E4 021A'         DATA GOSUB              GOSUB
0577 03E6 02AE'         DATA RETURN             RETURN
0578 03E8 0120'         DATA NUDEND             DEF
0579 03EA 0120'         DATA NUDEND             DIM
0580 03EC 0176'         DATA END                END
0581 03EE 8000          DATA >8000+0            FOR
0582 03F0 0324'         DATA NLET               LET
0583 03F2 8002          DATA >8000+2            BREAK
0584 03F4 8004          DATA >8000+4            UNBREAK
0585 03F6 8006          DATA >8000+6            TRACE (90)
0586 03F8 8008          DATA >8000+8            UNTRACE
0587 03FA 8016          DATA >8000+22           INPUT
0588 03FC 0120'         DATA NUDEND             DATA (93)
0589 03FE 8012          DATA >8000+18           RESTORE
0590 0400 8014          DATA >8000+20           RANDOMIZE
0591 0402 034E'         DATA NNEXT              NEXT
0592 0404 800A          DATA >8000+10           READ
0593 0406 0176'         DATA STOP               STOP (98)
0594 0408 803E          DATA >8000+62           DELETE
0595 040A 0120'         DATA NUDEND             REM
0596 040C 01CC'         DATA ON                 ON
0597 040E 800C          DATA >8000+12           PRINT
0598 0410 800E          DATA >8000+14           CALL
0599 0412 0120'         DATA NUDEND             OPTION
0600 0414 8018          DATA >8000+24           OPEN
0601 0416 801A          DATA >8000+26           CLOSE (A0)
0602 0418 0166'         DATA NONUD              SUB
0603 041A 803C  STMTTB  DATA >8000+60           DISPLAY
0604                    *
0605                    *
0606                    *
0607 041C 801C  NTAB    DATA >8000+28           ( (87)
060  041E 0166'         DATA NONUD              RESERVED FOR SR-62
0609 0420 0166'         DATA NONUD              SPARE
0610 0422 0166'         DATA NONUD              RESERVED FOR SR-62
0611 0424 0166'         DATA NONUD              RESERVED FOR SR-62
0612 0426 0166'         DATA NONUD              SPARE
0613 0428 0166'         DATA NONUD              RESERVED FOR SR-62
0614 042A 0166'         DATA NONUD              =
0615 042C 0166'         DATA NONUD              <
0616 042E 0166'         DATA NONUD              > (C0)
0617 0430 801E          DATA >8000+30           +
0618 0432 8020          DATA >8000+32           -
0619 0434 0166'         DATA NONUD              *
0620 0436 0166'         DATA NONUD              /
0621 0438 0166'         DATA NONUD              ^
0622 043A 0166'         DATA NONUD              SPARE
0623 043C 8010          DATA >8000+16           QUOTED STRING
0624 043E 0196'         DATA NUMCON             UNQUOTED STRING (NUMERIC)(C8)
0625 0440 0166'         DATA NONUD              LINE NUMBER
```

```
0626 0442 804A          DATA >8000+74           EOF
0627 0444 8022          DATA >8000+34           ABS
0628 0446 8024          DATA >8000+36           ATN
0629 0448 8026          DATA >8000+38           COS
0630 044A 8028          DATA >8000+40           EXP
0631 044C 802A          DATA >8000+42           INT
0632 044E 802C          DATA >8000+44           LOG   (DO)
0633 0450 802E          DATA >8000+46           SGN
0634 0452 8030          DATA >8000+48           SIN
0635 0454 8032          DATA >8000+50           SQR
0636 0456 8034          DATA >8000+52           TAN
0637 0458 8036          DATA >8000+54           LEN
0638 045A 8038          DATA >8000+56           CHR$
0639 045C 803A          DATA >8000+58           RND
0640 045E 8040          DATA >8000+64           SEG$
0641 0460 8046          DATA >8000+70           POS
0642 0462 8044          DATA >8000+68           VAL
0643 0464 8042          DATA >8000+66           STR$
0644 0466 8048          DATA >8000+72           ASC
0645      004C  NTABLN  EQU  $-NTAB
0646            *
0647            *       LED TABLE
0648            *
0649 0468       LTAB
0650 0468 0496'         DATA EQUALS             = (BE)
0651 046A 0478'         DATA LESS               <
0652 046C 0486'         DATA GREATR             > (CO)
0653 046E 0526'         DATA PLUS               +
0654 0470 0552'         DATA MINUS              -
0655 0472 055E'         DATA TIMES              *
0656 0474 056A'         DATA DIVIDE             /
0657 0476 0576'         DATA EXPON              ^ (CS)
0658      0010  LTBLEN  EQU  $-LTAB
0659            *
```

```
0661                    *
0662                    *       LED ROUTINES
0663                    *
0664                    *       LOGICAL COMPARISONS ENCODE TYPE OF COMPARISON AND USE
0665                    *       COMMON CODE TO PARSE EXPRESSION AND SET UP STATUS BITS
0666                    *
0667                    *       TYPES ARE EQUAL(0), NOT EQUAL(1), LESS THAN(2),
0668                    *       LESS OR EQUAL(3), GREATER THAN(4), AND GREATER OR EQUA
0669                    *       THIS CODE IS SAVED ON THE SUBROUTINE STACK.
0670                    *
0671 0478              LESS
0672 0478 0202                  LI    R2,2              LESS THAN CODE FOR COMMON RTN
     047A 0002
0673 047C-0288                  CI    R8,GT$*256        TEST FOR > TOKENAL
     047E C000
0674 0480 1604                  JNE   LT10              JUMP IF NO MATCH
0675 0482 0642                  DECT  R2                NOT EQUAL CODE FOR COMMON RTN
0676 0484 1005                  JMP   LT15
0677                    *
0678 0486              GREATR
0679      0488' C4      EQU   $+2                       CONSTANT 4
0680 0486 0202                  LI    R2,4              GREATER THAN CODE FOR COMMON RT
     0488 0004
0681 048A-0288         LT10     CI    R8,EQ$*256        TEST FOR EQUAL TOKEN
     048C BE00
0682 048E 1605                  JNE   LTST01            NOT GREATER OR NOT LESS BRANCH
0683 0490 06A0         LT15     BL    @PGMCHR           MUST BE PLAIN OLD > OR < TOKEN
     0492 06B8'
0684 0494 1001                  JMP   LEDLE
0685                    *
0686 0496              EQUALS
0687 0496 0702                  SETO  R2                EQUAL BIT FOR COMMON ROUTINE
0688 0498 0582         LEDLE    INC   R2                SETS TO ZERO
0689 049A              LTST01
0690 049A 05C9                  INCT  R9
0691 049C C642                  MOV   R2,*R9            SAVE STATUS MATCHING CODE
0692 049E 06A0                  BL    @PSHPRS           PICK UP ARGUMENTS
     04A0 05D6'
0693 04A2   C0                  BYTE  GT$,0
     04A3   00
0694 04A4 C119                  MOV   *R9,R4            RECALL TYPE CODE
0695 04A6 0649                  DECT  R9                RESET STACK POINTER
0696 04A8 D324                  MOVB  @LTSTAB(R4),R12   GET ADDRESS BIAS TO BRANCH TO
     04AA 04E2'
0697 04AC 088C                  SRA   R12,8             RIGHT JUSTIFY
0698 04AE 06A0                  BL    @ARGTST           TEST FOR MATCHING ARGUMENTS
     04B0 0584'
0699 04B2 131A                  JEQ   LTST20            IF BOTH ARE STRING
0700 04B4 06A0                  BL    @SCOMPB           FLOATING POINT COMPARISON
     04B6 0396'
0701 04B8 046C         LTST15   B     @LTSTXX(R12)      USE APPROPRIATE ROUTINE
     04BA 04BC'
0702 04BC              LTSTXX
0703 04BC 1504         LTSTGE   JGT   LTRUE             TEST IF GREATER OR EQUAL
0704 04BE 1303         LTSTEQ   JEQ   LTRUE             TEST IF EQUAL
0705 04C0 04C4         LFALSE   CLR   R4                FALSE IS ZERO
0706 04C2 1003                  JMP   LTST90
0707 04C4 13FD         LTSTNE   JEQ   LFALSE            TEST IF NOT EQUAL
0708 04C6 0204         LTRUE    LI    R4,>BFFF          TRUE IS MINUS ONE
     04C8 BFFF
```

```
0709 04CA 0203  LTST90 LI   R3,FAC              STORE RESULT IN FAC
     04CC 834A
0710 04CE CCC4         MOV  R4,*R3+             STORE EXP AND FIRST BYTE OF .
0711 04D0 04F3         CLR  *R3+                ZERO REMAINING DIGITS
0712 04D2 04F3         CLR  *R3+                ZERO REMAINING DIGITS
0713 04D4 04F3         CLR  *R3+                ZERO REMAINING DIGITS
0714 04D6 1039         JMP  LEDEND              END OF LED ROUTINE
0715 04D8 13F6  LTSTLE JEQ  LTRUE               TEST LESS THAN OR EQUAL
0716 04DA 11F5  LTSTLT JLT  LTRUE               TEST LESS THAN
0717 04DC 10F1         JMP  LFALSE
0718 04DE 15F3  LTSTGT JGT  LTRUE               TEST GREATER THAN
0719 04E0 10EF         JMP  LFALSE
0720                   *
0721                   *      DATA TABLE FOR OFFSETS FOR TYPES
0722                   *
0723 04E2   02  LTSTAB BYTE LTSTEQ-LTSTXX        EQUAL (0)
0724 04E3   08         BYTE LTSTNE-LTSTXX       NOT EQUAL (1)
0725 04E4   1E         BYTE LTSTLT-LTSTXX       LESS THAN (2)
0726 04E5   1C         BYTE LTSTLE-LTSTXX       LESS OR EQUAL (3)
0727 04E6   22         BYTE LTSTGT-LTSTXX       GREATER THAN (4)
0728 04E7   00         BYTE LTSTGE-LTSTXX       GREATER OR EQUAL (5)
0729                   *
0730                   *      STRING COMPARISON
0731                   *
0732 04E8       LTST20
0733 04E8 C2A0         MOV  @FAC+4,R10          POINTER TO STRING 1
     04EA 834E
0734 04EC D1E0         MOVB @FAC+7,R7           R7 =  RH STRING LENGTH
     04EE 8351
0735 04F0 06A0         BL   @VPOP               GET LH ARG BACK
     04F2 0668'
0736 04F4 C120         MOV  @FAC+4,R4           POINTER TO STRING 2
     04F6 834E
0737 04F8 D1A0         MOVB @FAC+7,R6           R6 =  LH STRING LENGTH
     04FA 8351
073  04FC D146         MOVB R6,R5               R5 WILL CONTAIN SHORTER LENG
0739 04FE 91C6         CB   R6,R7
0740 0500 1101         JLT  CSTR05              JUMP IF LENGTH 2 < LENGTH 1
0741 0502 D147         MOVB R7,R5
0742 0504 0985  CSTR05 SRL  R5,8                SHIFT FOR SPEED AND TEST ZEF
0743 0506 130D         JEQ  CSTR20              IF ZERO SET STATUS WITH LENG
0744 0508 C0CA  CSTR10 MOV  R10,R3              CURRENT CHARACTER LOCATION
0745 050A 058A         INC  R10                 INCREMENT POINTER
0746 050C 06A0         BL   @GETV1              GET FROM VDP
     050E 03C0'
0747 0510 D001         MOVB R1,R0               AND SAVE FOR COMPARISON
0748 0512 C0C4         MOV  R4,R3               CURRENT CHAR LOCATION IN ARG
0749 0514 0584         INC  R4                  INCREMENT POINTER
0750 0516 06A0         BL   @GETV1              GET FROM VDP
     0518 050E'
0751 051A 9001         CB   R1,R0               COMPARE CHARACTERS
0752 051C 16CD         JNE  LTST15              RETURN WITH STATUS IF NOT E
0753 051E 0605         DEC  R5                  OTHERWISE DECREMENT COUNTER
0754 0520 15F3         JGT  CSTR10              AND LOOP FOR EACH CHARACTER
0755 0522 91C6  CSTR20 CB   R6,R7               STATUS SET BY LENGTH COMPAR
0756 0524 10C9         JMP  LTST15              RETURN TO DO TEST OF STATUS
```

```
0758                    *
0759                    *       ARITHMETIC FUNCTIONS
0760                    *
0761 0526               PLUS
0762 0526 06A0                  BL    @PSHPRS           PARSE FOR VALUES
     0528 05D6'
0763 052A   C2                  BYTE  MINUS$,0
     052B   00
0764 052C 0202                  LI    R2,SADD           ADDRESS OF ROUTINE
     052E 037E'
0765 0530 04E0  LEDEX           CLR   @FAC+10           CLEAR ERROR CODE
     0532 8354
0766 0534 06A0                  BL    @ARGTST           TEST ARGUMENTS
     0536 05B4'
0767 0538 1336                  JEQ   ARGT05            IF STRING
0768 053A 06A0                  BL    @SAVREG           SAVE REGISTERS IN RAM
     053C 05C6'
0769 053E 0692                  BL    *R2               GO TO ROUTINE
0770 0540 06A0                  BL    @SETREG           RESTORE REGISTERS
     0542 05B4'
0771 0544 D0A0                  MOVB  @FAC+10,R2        TEST FOR OVERFLOW
     0546 8354
0772 0548 1602                  JNE   LEDERR
0773 054A 0460  LEDEND          B     @CONT
     054C 005E'
0774                    *
0775 054E 0460  LEDERR          B     @WARN$$
     0550 017A'
0776                    *
```

```
 0807                   *
 0808                   *       TEST ARGUMENTS ON BOTH STACK AND IN FAC
 0809                   *            BOTH MUST BE OF SAME TYPE
 0810                   *       CALL;
 0811                   *            BL   @ARGTST
 0812                   *            JEQ            IF STRING
 0813                   *            JNE            IF NUMERIC
 0814                   *
 0815 0584              ARGTST
 0816 0584 C1A0                 MOV  @VSPTR,R6          GET STACK POINTER
      0586 836E
 0817 0588 05C6                 INCT R6
 0818 058A                      MAC6  ^^^^^^^^^^^^^^^^
*0001 058A D7E0                 MOVB @R6LB,*R15
      058C 83ED
*0002 058E 1000                 NOP
*0003 0590 D7C6                 MOVB R6,*R15
*0004 0592 1000                 NOP                     THE NOP WASTES SOME MORE TIME
*0005 0594 9820                 CB   @VDPRD,@CBH65
      0596 8800
      0598 0105'
 0819 059A 1A07                 JL   ARGT10             NO, NUMERIC
 0820 059C 1B04                 JH   ARGT05             NO, OTHER
 0821 059E 9820                 CB   @FAC+2,@CBH65      YES, IS OTHER THE SAME ?
      05A0 834C
      05A2 0105'
 0822 05A4 1306                 JEQ  ARGT20             YES, DO STRING COMPARISON
 0823 05A6 0460  ARGT05 B      @ERRT                    DATA TYPES DON'T MATCH
      05A8 0000
 0824                   *
 0825                   *
 0826 05AA              NUMCHK
 0827 05AA 9820  ARGT10 CB     @FAC+2,@CBH65            SECOND OP. CAN'T BE STRING
      05AC 834C
      05AE 0105'
 0828 05B0 14FA                 JHE  ARGT05             IF SO, ERROR
 0829 05B2 045B  ARGT20 RT                              NO ERROR, RETURN W/ STATUS
```

```
 0831                   *
 0832                   *
 0833                   *       SUBROUTINE TO SET UP REGISTERS ON ENTRY FROM GPL
 0834                   *
 0835 05B4              SETREG
 0836 05B4 04C8                 CLR   R8
 0837 05B6 D220                 MOVB  @CHAT,R8          GET CURRENT CHARACTER
      05B8 8342
 0838 05BA D260                 MOVB  @STKADD,R9        GET STACK ADDRESS
      05BC 8373
 0839 05BE 0989                 SRL   R9,8
 0840 05C0 0229                 AI    R9,PAD            ADD IN BASE
      05C2 8300
 0841 05C4 045B                 RT
 0842                   *
 0843                   *
 0844                   *       SUBROUTINE TO RESTORE GPL MEMORY LOCATIONS
 0845                   *            ALSO USED TO SAVE R8 AND R9 FOR CALLS
 0846                   *              TO FLOATING POINT
 0847                   *
 0848 05C6              SAVREG
 0849 05C6 D808                 MOVB  R8,@CHAT          PUT CURRENT CHAR. IN FOR GPL
      05C8 8342
 0850 05CA                      MACC
*0001 05CA 0229         SAVRE2 AI    R9,-PAD           CALCULATE CURRENT STACK ADDRE
      05CC 7D00
*0002 05CE D820                 MOVB  @R9LB,@STKADD
      05D0 83F3
      05D2 8373
*0003 05D4 045B                 RT
 0851                   *
```

```
0853                    *
0854                    *       PUSH FOLLOWED BY PARSE
0855                    *
0856 05D6 05C9  PSHPRS  INCT R9
0857 05D8 0289          CI    R9,STKEND         STACK FULL ?
     05DA 83BA
0858 05DC 133F          JH    VPSH23            YES, ERROR
0859 05DE C648          MOV   R11,*R9           SAVE RETURN ON STACK
0860 05E0 020B          LI    R11,P05           OPTIMIZE
     05E2 001E'
0861                    *
0862                    *       STACK PUSH ROUTINE
0863                    *
0864 05E4               VPUSH
0865      05E6' C8      EQU   $+2               CONSTANT 8
0866 05E4 0200          LI    R0,8              NUMBER TO PUSH
     05E6 0008
0867 05E8 A800          A     R0,@VSPTR         BUMP POINTER
     05EA 836E
0868 05EC C060          MOV   @VSPTR,R1         GET STACK POINTER
     05EE 836E
0869 05F0               MAC7  ^^^^^^^^^^^^^^^^
*0001 05F0 D7E0         MOVB  @R1LB,*R15
     05F2 83E3
*0002 05F4 0261         ORI   R1,WRVDP
     05F6 0000
*0003 05F8 D7C1         MOVB  R1,*R15
*0004 05FA 0201         LI    R1,FAC
     05FC 834A
*0005 05FE D831  VPSH15 MOVB  *R1+,@VDPWD
     0600 8C00
0870 0602 0600          DEC   R0                COUNT
0871 0604 15FC          JGT   VPSH15
0872 0606 C00B          MOV   R11,R0            SAVE RETURN ADDRESS
0873 0608 9820          CB    @FAC+2,@CBH65     PUSHING A STRING?
     060A 834C
     060C 0105'
0874 060E 160E          JNE   VPSH20            NO
0875 0610 C1A0          MOV   @VSPTR,R6         ENTRY ON STACK
     0612 836E
0876 0614 0226          AI    R6,4              PTR TO STRING
     0616 0004
0877 0618 C060          MOV   @FAC,R1
     061A 834A
0878 061C 0281          CI    R1,SREF-PAD       IS IT A TEMPORARY STRING?
     061E 001C
0879 0620 1605          JNE   VPSH20            NO-OK
0880 0622 C060  VPSH19  MOV   @FAC+4,R1         ADDR OF STRING
     0624 834E
0881 0626 1302          JEQ   VPSH20            IF NULL STRING
0882 0628 06A0          BL    @ TVDP3           SET THE BACKPOINTER
     062A 0000
0883                    *
0884 062C C060  VPSH20  MOV   @VSPTR,R1         GET STACK POINTER
     062E 836E
0885 0630 0221          AI    R1,16             CORRECT BY 8
     0632 0010
0886 0634 8801          C     R1,@STREND        AT LEAST 8 LEFT?
     0636 831A
0887 0638 123B          JLE   VPOP18
```

```
 0888 063A 05C9          INCT R9
 0889 063C C640          MOV  R0,*R9
 0890 063E D820          MOVB @COMPCT,@ERRCOD+1 DO A GARBAGE COLLECTION
      0640 002A'
      0642 8323
 0891 0644 06A0          BL   @CALGPL
      0646 0184'
 0892                *                          TWO BYTES WASTED FOR COMMON CODE
 0893 0648 0018  C24     DATA 24                 UNUSED WORD FOR CONSTANT
 0894 064A C019          MOV  *R9,R0
 0895 064C 0649          DECT R9
 0896 064E C060          MOV  @VSPTR,R1
      0650 836E
 0897 0652 0221          AI   R1,16
      0654 0010
 0898 0656 8801          C    R1,@STREND         OUT OF MEMORY?
      0658 831A
 0899 065A 122A          JLE  VPOP18             NO - OK
 0900 065C 0200  VPSH23  LI   R0,ERROM           OUT OF MEMORY ERROR
      065E 0103
 0901 0660 06A0  VPSH25  BL   @SETREG            IN CASE OF GPL CALL
      0662 05B4'
 0902 0664 0460          B    @ERR
      0666 016A'
 0903                *
 0904                *
 0905                *       STACK POP ROUTINE
 0906                *
 0907 0668 0202  VPOP    LI   R2,FAC
      066A 834A
 0908 066C C060          MOV  @VSPTR,R1          GET STACK POINTER
      066E 836E
 0909 0670 8801          C    R1,@STVSPT         CHECK FOR STACK UNDERFLOW
      0672 8324
 0910 0674 121E          JLE  VPOP20             YES !, ERROR
 0911 0676               MAC8    ^^^^^^^^^^^^^^^
*0001 0676 D7E0          MOVB @R1LB,*R15
      0678 83E3
*0002 067A 0200          LI   R0,8
      067C 0008
*0003 067E D7C1          MOVB R1,*R15
*0004 0680 6800          S    R0,@VSPTR
      0682 836E
*0005 0684 DCA0  VPOP10  MOVB @VDPRD,*R2+
      0686 8800
 0912 0688 0600          DEC  R0                 COUNT
 0913 068A 15FC          JGT  VPOP10
 0914 068C C00B          MOV  R11,R0
 0915 068E 9820          CB   @FAC+2,@CBH65      POP A STRING?
      0690 834C
      0692 0105'
 0916 0694 160D          JNE  VPOP18             NO - OK
 0917 0696 04C6          CLR  R6                 FOR BACK POINTER CLEAR
 0918 0698 C0E0          MOV  @FAC,R3
      069A 834A
 0919 069C 0283          CI   R3,SREF-PAD        POP A TEMP
      069E 001C
 0920 06A0 13C0          JEQ  VPSH19             YES - FREE IT
 0921 06A2 06A0          BL   @GETV1             GET NEW PTR FROM S.T.
      06A4 0518'
```

```
922 06A6                BMAC6                       BOTH BYTES
001 06A6 D820           MOVB @VDPRD,@R1LB
    06A8 8800
    06AA 83E3
923 06AC C801           MOV  R1,@FAC+4              SET NEW PTR TO STRING
    06AE 834E
924 06B0 0450  VPOP1B   B    *R0                    RETURN
925            *
926 06B2 0200  VPOP20   LI   R0,ERREX               EXECUTION ERROR
    06B4 0403
927 06B6 10D4           JMP  VPSH25
928            *
929            *
```

```
 0931                   *
 0932                   *       GET NEXT CHARACTER FROM BASIC PROGRAM
 0933                   *            THE RETURNED STATUS REFLECTS THE CHARACTER
 0934                   *
 0935 06B8              PGMCHR
 0936 06B8 D220                 MOVB @GROMFL,R8         TEST GROM FLAG
      06BA 8389
 0937 06BC 1607                 JNE  PGMC10             YES, DO GROM INPUT
 0938 06BE                      MAC4   ^^^^^^^^^^^^^^^^^^^^^^^^
*0001 06BE D7E0                 MOVB @PGMPTR+1,*R15
      06C0 832D
*0002 06C2 020A                 LI   R10,VDPRD
      06C4 8800
*0003 06C6 D7E0                 MOVB @PGMPTR,*R15
      06C8 832C
*0004 06CA 1007                 JMP  PGMNXT
*0005 06CC DB60  PGMC10         MOVB @PGMPTR,@GRMWA(R13)
      06CE 832C
      06D0 0260'
*0006 06D2 DB60                 MOVB @PGMPTR+1,@GRMWA(R13)
      06D4 832D
      06D6 06D0'
*0007 06D8 C28D                 MOV  R13,R10        GET GROM READ DATA PORT ADDR IN
*0008 06DA 05A0  PGMNXT         INC  @PGMPTR
      06DC 832C
*0009 06DE D21A                 MOVB *R10,R8
*0010 06E0 045B                 RT
 0939                   *
 0940                   *
 0941                   *
 0942                           END
NO ERRORS,    0014 WARNINGS
```

```
$          06E2'        0107  0168  0169  0169  0544  0645  0658  0679  0865
ARG        835C   A0018 A0038
ARGT05     05A6'  0823  0767  0820  0828
ARGT10     05AA'  0827  0819
ARGT20     05B2'  0829  0822
ARGTST     0584'  0815  0698  0766
ASSG    R  038C'  0033  0505  0538
ASSGNV  R        0032
B9985      0092'  0145  0113
BASE    D  8343   0058  0026
BMAC6   M        0922
BRKFL      0001   0292  0259
BRKKEY     0020   0037  0169
BRKPN1     0160'  0259  0169
BRKPNT     015A'  0256  0175
BUFFY      8344   0059  0163  0195
C0         00B6'  0169
C1000   R        0169
C16        03A2'  0544  0531  0537
C2      R  018C'  0293  0314
C24        0648'  0893  0551
C4         0488'  0679  0197  0302
C8         05E6'  0865  0539
CALGPL  D  0184'  0307  0027  0316  0891
CB3        00B3'  0169
CB9        002A'  0107  0297
CBH65      0105'  0194  0463  0818  0821  0827  0973  0915
CBH66      01D1'  0350  0387  0441
CBH67      02F5'  0474  0525
CC         0008   A0024 A0052 A0053 A0059
CFI     R  01DC'  0029  0353
CHAT       8342   0057  0837  0849
COMMA$     00B3   0074  0349  0429  0473
COMPCT     002A'  0297  0890
CONC$      00B8   0075  0136
CONCAT     0008   0296  0146
CONT       005E'  0128  0303  0335  0466  0506  0557  0773
CONT10     0072'  0134  0130
CONT15     0086'  0141  0111
CONT20     0094'  0146  0137
CONTG   D  005A'  0127  0023
CSNGR   R  01AC'  0029  0328
CSTR05     0504'  0742  0740
CSTR10     0508'  0744  0754
CSTR20     0522'  0755  0743
DATA1      0000+  A0001
DIVI$      00C4   0081  0788  0795
DIVIDE     056A'  0793  0656
ELSE$      0081   0069  0486
 ND        0176'  027   0580
 NLN       8332   0055  0417
ENTRY   R        0035
EOL        011C'  0204  0488
EQ$        00BE   0076  0499  0681
EQUALS     0496'  0686  0650
ERR     D  016A'  0270  0027  0359  0902
ERR1    D  0166'  0265  0025  0188  0205  0245  0331  0373  0377
ERR1B      0218'  0377  0405  0434
ERR1C      02AC'  0434  0444  0478  0483  0500
ERRBS      0503   0288
```

```
ERRCOD      8322    0048  0162  0270  0302  0314  0590
ERREX       0403    0287  0926
ERRIOR      0203    0285  0358
ERRLNF      0303    0286  0418
ERROM       0103    0284  0900
ERRSN       0003    0283  0268
ERRSYN  D   0166'   0264  0027
ERRT    R   05A8'   0034  0823
ERRTM       0603    0289
EXEC10      00B0'   0168  0199  0424
EXEC11      00CA'   0173  0315
EXEC12      00C2'   0170
EXEC14      00D6'   0176  0258
EXEC15      00DE'   0178  0164
EXEC20      00EE'   0184  0181
EXEC50  D   016E'   0274  0026  0196  0200  0279  0309
EXECG   D   00A2'   0160  0023
EXIT        016A'   0269  0147  0260  0804
EXP         8376   A0028
EXPON       0576'   0800  0657
EXPON$      00C5    0082  0802
EXRTN       0104'   0193  0088
EXRTNA      0000'   0088  0179
EXTRAM      832E    0053  0173  0197  0198  0386  0422  0445  0555  0556
FAC         834A   A0015 A0016 A0017 A0018  0242  0323  0326  0330  0333  0352
                           0354  0356  0386  0387  0388  0391  0441  0445  0459
                           0463  0479  0522  0525  0529  0709  0733  0734  0736
                           0737  0765  0771  0821  0827  0869  0873  0877  08
                           0907  0915  0918  0923
FDVSR       8354   A0016
FLAG        8388    0060  0170
FLTERR      836C   A0019
FLTNDX      8354   A0017
GETCH       0020+  A0048
GETCH1      002E+  A0053 A0060
GETCHG      0032+  A0056
GETL10      023E'   0407
GETL30      0248'   0411
GETSTK  R           0033
GETSTR      0007    0295
GETV    R   02D4'   0032  0240  0458
GETV1   R   06A4'   0032  0546  0554  0746  0750  0921
GO          01C8'   0341  0574
GO$         0085    0065  0362
GOSUB       021A'   0382  0576
GOSUB$      0087    0067  0372
GOSUB2      021C'   0386  0375
GOTO        0236'   0397  0575
GOTO$       0086    0066  0370
GOTO20      0238'   0401  0392  0432
GOTO31      0264'   0417  0417
GOTO32      026E'   0417  0417  0417
GOTO33      027C'   0417  0417
GOTO34      028A'   0418  0417
GOTO35      0286'   0417  0417
GOTO36      0290'   0421  0417
GOTO40      029C'   0428  0413
GOTO50      02A6'   0431  0366  0371  0480  0487
GOTO90      01EA'   0358  0355  0430
GOTO95      01EE'   0359  0419  0450
```

```
GREATR      0486'   0678  0652
?RMRA   R   000C'   A0035  0094  0094
GRMWA   R   06D6'   A0035  A0056 A0057  0118  0118  0132  0132  0417  0417  0938
                           0938
GROMA   R           0031
GROMFL      8389    0061  0257  0417  0936
GT$         00C0    0077  0673  0693
HHUSC       033B'   0504
I           0004    A0005
IF          02F0'   0472  0573
J           0005    A0006 A0037 A0042 A0044
JOYX        8377    A0027
JOYY        8376    A0021
K           0006    A0007 A0038 A0043
L           0007    A0008 A0040 A0043
LEDEND      054A'   0773  0714
LEDERR      054E'   0775  0772
LEDEX       0530'   0765  0783  0790  0797
LEDLE       0498'   0688  0684
LESS        0478'   0671  0651
LET$        008D    0068  0503
LEXPON      0005    0291  0803
LFALSE      04C0'   0705  0707  0717  0719
LLB         83EF    A0031
LN$         00C9    0064  0404  0482
LT10        048A'   0681  0674
LT15        0490'   0683  0676
LTAB        0468'   0649  0142  0658
LTBLEN      0010    0658  0140
LTRUE       04C6'   0708  0703  0704  0715  0716  0718
LTST01      049A'   0689  0682
LTST15      04B8'   0701  0752  0756
LTST20      04E8'   0732  0699
LTST90      04CA'   0709  0706
LTSTAB      04E2'   0723  0696
LTSTEQ      04BE'   0704  0723
LTSTGE      04BC'   0703  072
LTSTGT      04DE'   0718  0727
LTSTLE      04D8'   0715  0726
LTSTLT      04DA'   0716  0725
LTSTNE      04C4'   0707  0724
LTSTXX      04BC'   0702  0701  0723  0724  0725  0726  0727  0728
MAC1    M           0094
MAC2    M           011
MAC3    M           013
MAC4    M           093
MAC5    M           0417
MAC6    M           0 1
MAC7    M           0 69
MAC     M           0911
MAC8    M           0555
MACC    M           0 50
MACD    M           0169
MINUS       0552'   0779  0654
1INUS$      00C2    0079  0763  0781
MOVFAC  R   0376'   0032  0465  0533
MULT$       00C3    0080
N           0008    A0009
NEXT    R           0029
NEXT2       0356'   0523  0532
```

```
NEXT3       0366'   0529  0526
NEXT4       0374'   0533  0530
NEXT5       03B0'   0551  0542  0562
NEXT6       03D0'   0561  0548
NEXT8       03CC'   0557  0550  0563
NLET        0324'   0497  0182  0582
NNEXT       034E'   0521  0591
NOLED       0166'   0267  0141
NONUD       0166'   0266  0569  0570  0571  0572  0602  0608  0609  0610  0611
                          0612  0613  0614  0615  0616  0619  0620  0621  0622
                          0625
NTAB        041C'   0607  0112  0645
NTABLN      004C    0645  0110
NUDD2       0006    0294  0449
NUDE10      009A'   0150  0211
NUDEND      0120'   0210  0135  0219  0446  0578  0579  0588  0595  0599
NUDGO5      0044'   0116  0153  0275
NUDTAB      8328    0051  0115
NUM$        00C8    0072
NUMC50      01C4'   0335
NUMCHK      05AA'   0826  0351  0475
NUMCON      0196'   0322  0624
ON          01CC'   0347  0596
ON20        01F0'   0361  0357
ON30        01FA'   0365  0342
ON40        0206'   0370  0363
ON50        0210'   0373  0368
P05         001E'   0102  0860
P10         0026'   0106  0103
P17         003C'   0114  0190
P20         0056'   0121  0100
PAD         8300    A0004 A0015 A0019 A0020 A0021 A0022 A0025 A0026 A0027 A0028
                          A0029 A0030 A0031  0038  0039  0040  0041  0042  0043
                           0044  0046  0047  0048  0049  0050  0051  0052  0053
                           0054  0055  0056  0057  0058  0059  0060  0061  0062
                           0840  0850  0878  0919
PARSE   D   0014'   0098  0026  0348  0472
PARSEG  D   0002'   0093  0023
PATCH1  D   0130'   0239  0236
PGMC10      06CC'   0938  0937
PGMCHR  D   06B8'   0935  0025  0106  0144  0174  0180  0332  0364  0374  0407
                          042   0431  0481  0485  0501  0683
PGMNXT      06DA'   0938  0185  0409  0938
PGMPTR      832C    0052  0173  0176  0177  0323  0325  0330  0484  0938  093
                          093   0938  0938
PLUS        0526'   0761  0653
PLUS$       00C1    0078
POPSTK      0000+   A0037
PROA        8310    A0022
PSHPRS  D   05D6'   0856  0024  0502  0692  0762  0780  0787  0794  0801
PSYM        02CE'   0457  0104
PUTSTK  R           0033
R0          0000          0146  0163  0170  0171  0195  0257  0259  0268  027
                          0333  0354  0358  0408  0417  0418  0449  0747  07
                          0803  0866  0867  0870  0872  0889  0894  0900  09
                          0911  0912  0914  0924  0926
ROLB        83E1    0038
R1          0001          0243  0244  0246  0417  0417  0417  0417  0417  042
                          0422  0460  0547  0547  0556  0747  0751  0868  086
                          0869  0869  0869  0877  0878  0880  0884  0895  088
```

```
                              0896  0897  0898  0908  0909  0911  0923
R10           000A            0177  0733  0744  0745  0938  0938  0938
R10LB         83F5    0043
R11           000B            0094  0094  0101  0239  0303  0308  0315  0859  0860
                              0872  0914
R11LB         83F7    0044    0094
R12           000C            0696  0697  0701
R13           000D            A0056 A0057 A0059 0094  0094  0118  0118  0132  0132
                              0132  0417  0417  0417  0938  0938  0938
R15           000F            A0039 A0041 A0048 A0050 0417  0417  0818  0818  0869
                              0869  0911  0911  0938  0938
R1LB          83E3    0039    0417  0417  0869  0911  0922
R2            0002            0239  0248  0417  0417  0417  0417  0417  0417  0417
                              0417  0672  0675  0680  0687  0688  0691  0764  0769
                              0771  0782  0789  0796  0907  0911
R3            0003            0341  0356  0382  0388  0391  0397  0412  0417  0417
                              0417  0476  0543  0545  0552  0553  0709  0710  0711
                              0712  0713  0744  0748  0918  0919
R4            0004            0243  0246  0522  0529  0541  0549  0561  0694  0696
                              0705  0708  0710  0736  0748  0749
R5            0005            0738  0741  0742  0753
R6            0006            A0050 A0051 A0056 A0058 0129  0131  0132  0132  0134
                              0737  0738  0739  0755  0816  0817  0818  0875  0876
                              0917
R6LB          83ED    0040    A0048 A0057 0132  0818
R7            0007            0102  0108  0109  0110  0112  0112  0114  0115  0118
                              0142  0145  0151  0184  0186  0187  0189  0189  0191
                              0210  0213  0274  0734  0739  0741  0755
R7LB          83EF    0041    0118
R8            0008            A0024 0102  0134  0136  0138  0139  0140  0142  0143
                              0176  0184  0204  0204  0324  0325  0362  0365  0367
                              0370  0372  0404  0408  0417  0429  0443  0443  0477
                              0482  0486  0499  0673  0681  0836  0837  0849  0936
                              0938
R9            0009            0098  0099  0101  0129  0150  0178  0179  0210  0212
                              027   0307  030   0423  0690  0691  0694  0695  0838
                              0839  0840  0850  0856  0857  0859  0888  0889  0894
                              0895
R9LB          83F3    0042    0850
RESET   R     0054'   0033    0119
RETU10        02AE'   0440    0442
RETURN        02AE'   0439    0577
RTNADD        8326    0050    0274
RTNG    D     012A'   0217    0023
SADD    R     052E'   0030    0535  0764
SAVRE2        05CA'   0850    0327
SAVREG  D     05C6'   084     0025  0117  0534  0768
SCOMPB  R     04B6'   0030    0540  0700
SDIV    R     0572'   0030    0796
STRG    D     05B4'   0035    0025  0093  0127  0161  0218  0329  0536  0770  0901
SIGN          8375    A0026
SMB     R     032A'   0032    0462  049
SMULT   R     0566'   0030    0789
SREF          831C    0047    0878  0919
SSUB    R     055A'   0030    0782
STATUS        837C    A0029
STKADD        8373    A0025   0838  0850
STKEND        83BA    0062    0099  0857
STKMOV        0018+   A0043 A0045
STLN          8330    0054    0198  0417
```

```
LABEL     VALUE   DEFN  REFERENCES

STMTTB    041A'   0603  0189
STOP      0176'   0277  0593
STREND    831A    0046  0886  0898
STVDP3 R  062A'   0031  0882
STVSPT    8324    0049  0909
SUB$      00A1    0073  0367
SYM     R 0350'   0032  0457  0497  0521
SYMB10    02EC'   0466  0464
SYMB20    02C8'   0449  0461
SYMPTR    8340    0056
THEN$     00B0    0071  0477
TIMES     055E'   0786  0655
TO$       00B1    0070  0365
TRACE     018A'   0314  0172
VDPRD     8800    A0013 A0040 A0052  0417  0555  0819  0911  0922  0938
VDPWD     8C00    A0014 0869
VPOP    D 0668'   0907  0024  0440  0524  0735
VPOP10    0684'   0911  0913
VPOP18    06B0'   0924  0887  0899  0916
VPOP20    06B2'   0926  0527  0910
VPSH15    05FE'   0869  0871
VPSH19    0622'   0880  0920
VPSH20    062C'   0884  0874  0879  0881
VPSH23    065C'   0900  0121  0858
VPSH25    0660'   0901  0927
VPUSH   D 05E4'   0864  0024  0390
VROA      03C0    A0023
VSPTR     836E    A0020 A0039 A0041 A0042  0531  0537  0539  0543  0551  05
                        0816  0867  0868  0875  0884  0896  0908  0911

WARN$$    017A'   0302  0334  0775
WRVDP   R 05F6'   A0035 0869
WS        83E0    A0030
```

```
                    1         TITLE MONITOR
                    2         GROM 0
                    3  ********************************************************
2828                4  M$PSCN EQU  >2828              BASIC PSCAN PORTION
2010                5  M$EDIT EQU  >2010              BTAB IN BASIC EDIT PORTION
4D00                6  M$EXEC EQU  >4D00              BASIC EXEC PORTION
1310                7  M$TAPE EQU  >1310              TAPEDSR
                    8  ********************************************************
                    9  *****     DIRECT REFERENCE TO PGMCTR IN "EXEC"        ******
                   10  ********************************************************
37B4               11  PGMCTR EQU  >37B4              BASIC GPL SUBPROGRAM SETUP
1320               12  M$EUR  EQU  M$TAPE+>10         TRANSLATE TO EUROPEAN
                   13  ********************************************************
                   14  *
                   15  *       SYSTEM MONITOR AND POWERUP ROUTINES
                   16  *
                   17  *       CHANGES MADE 9/6/79 TO LINK$$ AND BRPRG.  TOM FERRIO
                   18  *
                   19  ********************************************************
                   20  *       CONSTANT DECLARATIONS
                   21  *
2844               22  MEMCHK EQU  M$PSCN+>1C         BASIC MEMORY ACQUISITION
284C               23  WARN$$ EQU  M$PSCN+>24         BASIC WARNING ROUTINE
284E               24  ERR$$  EQU  M$PSCN+>26         BASIC ERROR ROUTINE
4D12               25  GETSTR EQU  M$EXEC+>12         BASIC GET STRING ROUTINE
2010               26  EXEC$$ EQU  M$EDIT             GROM BASIC EXECUTION
1310               27  DSR    EQU  M$TAPE             DEVICE SERVICE ROUTINE LINKS
0019               28  SROM   EQU  25                 CALL ROM SEARCH XML
001A               29  SGROM  EQU  26                 CALL GROM SEARCH XML
0004               30  H04    EQU  >04
7D00               31  CPOFST EQU  ->8300             CPU OFFSET FOR GPL INTERPRETER
DD00               32  RMNGRM EQU  >6000+CPOFST       BEGINNING OF ROM IN GROM
                   33  ********************************************************
                   34  *       VARIABLE ASSIGNMENTS
                   35  *
0000               36  ENTADD EQU  0              ENTRY ADDRESS FOR ROM IN GROM
0000               37  CLEARU EQU  0              USED TO INITIALIZE CRU BITS
0002               38  CRLIST EQU  2              USED TO INITIALIZE CRU BITS
004A               39  FAC    EQU  >4A            FLOATING ACCUMULATOR
0052               40  INDEXM EQU  FAC+8          USED FOR INDEXING GROM LISTS
0054               41  SCTEMP EQU  FAC+10         TEMPORARY FOR SCAN IN LINK$$
0055               42  SCLEN  EQU  FAC+11         TEMPORARY LENGTH OF LINK NAME
0056               43  SCNAME EQU  FAC+12         TEMPORARY LINK NAME ADDRESS
0058               44  TEMP1  EQU  FAC+14         TEMPORARY COUNTER IN GROM
0059               45  TEMP   EQU  FAC+15         TEMPORARY COUNTER IN GROM
005A               46  TEMLEN EQU  FAC+16         TEMPORARY LINK TO PGM DESC BLKS
005C               47  ARG    EQU  FAC+18         SECOND ARGUMENT
0064               48  SCTMP2 EQU  ARG+8          SECOND TEMPORARY IN LINK$$
006A               49  DTEMP3 EQU  >6A
006C               50  TEMP2  EQU  >6C            COUNTER OF PROGRAMS FOUND
006D               51  TYPE   EQU  >6D            TYPE OF PROGRAM TO SEARCH FOR
0070               52  MEMLEN EQU  >70            LENGTH OF VDP MEMORY
0072               53  STKBAS EQU  >72            DATA STACK POINTER
0073               54  SUBSTK EQU  >73            SUBROUTINE STACK POINTER
0074               55  KEYBRD EQU  >74            KEYBOARD TO SCAN
```

```
0075    56  KEY     EQU  >75          KEY FOUND
0078    57  RANDOM  EQU  >78
00C2    58  SCRTCH  EQU  >C2
00D0    59  CRULST  EQU  >D0          DOUBLE ZERO IF START SEARCH
        60  *                         SET BY ROM CODE IF CONTINUE SEARC
```

```
                    62  ********************************************************************
                    63  *       GROM HEADER BLOCK
                    64  *
0000 AA             65  HAA     DATA >AA            VALID GROM IDENTIFICATION
0001 02             66          DATA 2              VERSION NUMBER
0002 00             67          DATA 0              NUMBER OF PROGRAMS
0003 00             68          DATA 0              RESERVED
0004 0000           69          DATA #0
0006 0000           70          DATA #0             USER PROGRAM ADDRESS
0008 1310           71          DATA #DSR           DSR ROUTINE
000A 1320           72          DATA #M$EUR         LINK PROGRAM ADDRESS
000C 000000         73          DATA #0,#0          RESERVED FOR EXPANSION
000F 00
```

```
                    75  ********************************************************
                    76  *       TRANSFER VECTORS
                    77  *
0010 43DC           78          BR    LINK$$          LINK PROGRAMS
0012 443C           79          BR    RETN$$          RETURN FROM LINK OR DSR
0014 49A9           80          BR    CNS$$           NUMBER TO STRING
0016 4396           81          BR    CHR1$$          LOAD 6 BY 8 UPPERCASE CHARACTER S
0018 439E           82          BR    CHR2$$          LOAD 5 BY 7 UPPERCASE CHARACTER S
001A 4446           83          BR    VWARN$          BASIC WARNING MESSAGE
001C 4449           84          BR    VERR$           BASIC ERROR MESSAGE
001E 444C           85          BR    VEXEC$          BASIC IN GROM STARTUP
0020 4052           86          BR    PWRUP$          PROGRAM EXIT TRAP
0022 51FE           87          BR    INT$$           GREATEST INTEGER FUNCTION
0024 4C82           88          BR    PWR$$           EXPONENTIATION ROUTINE
0026 4D59           89          BR    SQR$$           SQUARE ROOT FUNCTION
0028 4DB4           90          BR    EXP$$           EXPONENTIAL FUNCTION
002A 4E64           91          BR    LOG$$           NATURAL LOG FUNCTION
002C 4EF9           92          BR    COS$$           COSINE FUNCTION
002E 4F01           93          BR    SIN$$           SINE FUNCTION
0030 4F5F           94          BR    TAN$$           TANGENT FUNCTION
0032 4F80           95          BR    ATN$$           ARCTANGENT FUNCTION
0034 43CE           96          BR    TON1$$          ACCEPT TONE
0036 43D6           97          BR    TON2$$          BAD RESPONSE TONE
0038 054D12         98          B     GETSTR          GET STRING SPACE ROUTINE FOR BASI
003B 525E           99          BR    BITRVR          BIT REVERSAL ROUTINE
003D 4417          100          BR    LNK$LQ          SPECIAL ENTRY TO LINK FOR CASSETT
003F 052844        101          B     MEMCHK          BASIC MEMORY ACQUISITION
0042 0537B4        102          B     PGMCTR          BASIC GPL SUBPROGRAM SETUP
0045 60            103          DATA  >60             BASIC SCREEN CHARACTER OFFSET
0046 0D00          104          DATA  #>0D00          ADDRESS FOR SPEECH RELATIVE TO CP
0048 1100          105          DATA  #>1100          SECOND SPEECH ADDRESS
004A 43C2          106          BR    CHR3$$          LOAD 5 BY 7 LOWERCASE CHARACTER S
                   107  *  Following 3 lines has been changed to double address 8/
004C 04B4          108          DATA  #CHAR1$          ADDRESS OF 6X8 UPPERCASE DEFS
004E 06B4          109          DATA  #CHAR2$          ADDRESS OF 5X7 UPPERCASE DEFS
0050 0874          110          DATA  #CHAR3$          ADDRESS OF 5X7 LOWERCASE DEFS
```

```
                    112  ********************************************************
                    113  *       SYSTEM POWERUP ROUTINE
                    114  *       THIS ROUTINE INITIALIZES THE SYSTEM.  THE SOUND
                    115  *       AND VDP CIRCUITS ARE CLEARED, THE DEFAULT VALUES
                    116  *       FOR THE VDP REGISTERS, CHARACTER SET, COLOR TABLE,
                    117  *       AND STATUS BLOCK ARE LOADED.  THE VIDEO RAM SIZE
                    118  *       IS D TERMINED AND STORED AT LOCATION >70
                    119  *
                    120  ********************************************************
                    121  PWRUP$
0052 8780CE         122          DCLR @>CE            CLEAR SOUND LIST INDICATOR
0055 BE8F11         123          ST   >70,@>1100      TURN OFF SPEECH
0058 0070
005A BE8100         124          ST   >9F,@>100       TURN OFF SOUND GENS
005D 9F
005E BE8100         125          ST   >BF,@>100       TURN OFF SOUND GENS
0061 BF
0062 BE8100         126          ST   >DF,@>100       TURN OFF SOUND GENS
0065 DF
0066 BE8100         127          ST   >FF,@>100       TURN OFF SOUND GENS
0069 FF
006A BF72FF         128          DST  >FF7E,@STKBAS INITIALIZE THE SUBROUTINE STACK
006D 7E
                    129  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    130  *  To fix the problem with bit map graphics programs leaving
                    131  *  VDP messed up, VDP register 0 is reset at powerup reset
                    132  *  8/07/81.  The following line is changed
                    133  *       MOVE 7 FROM RCM(#VDPRG$+1) TO VDP(1)
                    134  *  to
006E 390008         135          MOVE 8 FROM RCM(#VDPRG$+1) TO VDP(0)
0071 000451
                    136  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    137  *
                    138  *       LAST POWER UP ROUTINE DOES SCREENS AND MENUS
                    139  *
                    140  PWRT$$
0074 8600           141          CLR  @0              CLEAR CPU RAM
0076 350071         142          MOVE >71 FROM @0 TO @1
0079 0100
007B 35003E         143          MOVE >3E FROM @>00 TO @>82
007E 808200
0081 35000B         144          MOVE >B FROM @>00 TO @>74
0084 7400
0086 350008         145          MOVE 8 FROM @0 TO @SCRTCH
0089 80C200
                    146  ****** ST   1,@CRLIST+2
                    147  ****** ST   16,@CRLIST+1             CLEAR HHU REQUEST
008C BF0303         148          DST  >0308,@CRLIST+1         DISABLE KBD INTERRUPTS
008F 08
0090 F60203         149          I/O  @CRLIST,3
0093 BF0310         150          DST  >1001,@CRLIST+1         CHANGE 6/7/79
0096 01
0097 F60203         151          I/O  @CRLIST,3
009A BE0318         152          ST   24,@CRLIST+1            TURN ON AUDIO GATE
009D F60203         153          I/O  @CRLIST,3
```

```
00A0 8400      154          INV  @CLEARU                    WRITE A ONE TO CRU
00A2 BE0302    155          ST   2,@CRLIST+1                ENABLE VDP INTERRUPT
00A5 F60203    156          I/O  @CRLIST,3
00A8 BE0301    157          ST   1,@CRLIST+1                ENABLE EXTERNAL INTERR
00AB F60203    158          I/O  @CRLIST,3
               159  ****** ST   22,@CRLIST+1
               160  ****** ST   2,@CRLIST+2                 TURN ON MOTORS FOR CS1
00AE BF0316    161          DST  >1602,@CRLIST+1            CHANGE 6/7/79
00B1 02
00B2 F60203    162          I/O  @CRLIST,3
00B5 0603CE    163          CALL TON1$$                     BEEP AT PERSON
00B8 86A000    164          CLR  RAM(0)
00BB BE7010    165          ST   >10,@MEMLEN
00BE BEB070    166  BRP10   ST   >A0,RAM(@MEMLEN)          DETERMINE MEMORY LENGT
00C1 A0
00C2 8EA000    167          $IF RAM(0) .NE. 0 GOTO BRP20
00C5 40DC
00C7 390001    168          MOVE 1 FROM ROM(#H80) TO VDP(1)
00CA 01044F
00CD 86B070    169          CLR  RAM(@MEMLEN)
00D0 A07070    170          A    @MEMLEN,@MEMLEN
00D3 D67040    171          $IF  @MEMLEN .NE. >40 GOTO BRP10
00D6 40BE
00D8 BE80FD    172          ST   8,@>FD                     SAVE IN R14LB!
00DB 08
00DC 9370      173  BRP20   DDEC @MEMLEN
00DE 390001    174          MOVE 1 FROM ROM(#H20) TO VDP(1)
00E1 010244
00E4 86A000    175          CLR RAM(0)                      CLEAR MEMORY IN THE VDPRAM
00E7 350FFF    176          MOVE >0FFF FROM RAM(0) TO RAM(1)
00EA A001A0
00ED 00
00EE 310020    177          MOVE 32 FROM ROM(#TABLE$) TO TABLE(0)
00F1 A38004
00F4 59
00F5 310200    178          MOVE 512 FROM ROM(#CHAR1$) TO CHAR(>20)
00F8 A90004
00FB B4
00FC 310050    179          MOVE 80 FROM ROM(#TIBUG$) TO CHAR(1)
00FF A80809
0102 50
0103 0720      180          ALL  : :
0105 BE7E05    181          ST 5,YPT
               182  RSCAN
0108 BC747E    183          ST YPT,@KEYBRD                  RESET ALL KEYBOARDS FOR SCAN
010B 03        184          SCAN
010C 927E      185          DEC  YPT
010E 4108      186          BR   RSCAN
0110 877E      187          DCLR YPT
               188          $REPEAT
0112 BE7560    189            ST   >60,@KEY
               190            $REPEAT
0115 08C1E0    191              FMT  2('@KEY'),>11^,30<,2('@KEY')
0118 75FB01
011B 17B09D
```

```
 11E C1E075
0121 FB011F
0124 FB
0125 A67E12    192              SUB  >12,YPT                                  2
0128 A27508    193              ADD  >8,@KEY                                 2
012B D675E0    194            $UNTIL @KEY .EQ. >E0                           1
012E 4115
0130 D67E03    195          $UNTIL YPT .EQ. 3
0133 4112
0135 877E      196          HOME
               197          FMT  5^,15<,'1,2,3',29<,'4,5,6',29<,'7,8,9';
0137 08A48E    198               8^,16<,':READY-PRESS ANY KEY TO BEGIN:'
013A 020102
013D 039C02
0140 040506
0143 9C0207
0146 0809A7
0149 8F1B52
014C 454144
014F 592D50
0152 524553
0155 532041
0158 4E5920
015B 4B4559
015E 20544F
0161 204245
0164 47494E
0167 FB
0168 310011    199          MOVE 17 FROM ROM(#TI) TO RAM(296)
016B A12804
016E 96
016F 310018    200          MOVE 24 FROM ROM(#TI1979) TO RAM(708)
0172 A2C404
0175 8F
0176 31000D    201          MOVE 13 FROM ROM(#HC) TO RAM(362)
0179 A16A04
017C A7
               202  *
               203  *       CHECK FOR EUROPEAN GROM AND GO TO THEIR
               204  *       FOREIGN LANGUAGE CODE
               205  *
017D BE4310    206          ST   >10,@>43
0180 06037C    207          CALL EUR?
               208  *
               209  *       SCAN FOR OTHER POWER UP ROUTINES
               210  *
0183 8780D0    211          DCLR @CRULST       START SEARCH FROM BEGINNING
0186 8655      212          CLR @SCLEN         NO NAME NEEDED
0188 BE6D04    213          ST   4,@TYPE       LOOK FOR POWER UPS
018B 0F19      214  XROM    XML  SROM          LOOK FOR ROM CODE
018D 6183      215          BS   XROM          LOOK FOR ANOTHER ROUTINE IN ROM
018F BF7200    216  XGROM1  DST  >0080,@STKBAS
0192 80
0193 BF9072    217          DST  #PWR207,+STKBAS PUSH LAST POWER UP ROUTINE ADDRE
0196 01A2
```

```
0198 0F1A     218          XML  SGROM         SEARCH GROM
019A BD9073   219          DST  *STKBAS,*SUBSTK GO TO POWER  UP  ROUTINE
019D 9072
019F 9672     220          DECT @STKBAS
01A1 00       221          RTN                CALL ROUTINE
01A2 8F80D0   222  PWR207  DCZ  @CRULST       FINISHED LOOKING?
01A5 418F     223          BR   XGROM1        NO
01A7 390001   224          MOVE 1 FROM RGM(#VDPRG$) TO VDP(1)
01AA 010450
              225  *
              226  *       WAIT FOR KEY DOWN
              227  *
01AD 8674     228  WAIT    CLR  @KEYBRD
01AF 02FF     229  WAIT10  RAND                    INITIALIZE RANDOM NUMBER
01B1 03       230          SCAN                    TEST CONSOLE KEYBOARD
01B2 41AF     231          BR    WAIT10
              232  *
              233  *       BUILD TABLE OF AVAILABLE PROGRAMS
              234  *
01B4 0603CE   235  BUILD   CALL TON1$$
01B7 0720     236          ALL  : :                        CLEAR SCREEN
01B9 BE72FE   237  BUIL3   ST   >FE,@STKBAS  INITIALIZE STACK
01BC BE6D06   238          ST   6,@TYPE       LOOK FOR USER PROGRAMS
01BF 866C     239          CLR  @TEMP2        INITIALIZE PROGRAM COUNT
01C1 8680FB   240          CLR  @>FB
01C4 31001E   241          MOVE 30 FROM ROM(>6000) TO RAM(>400)
01C7 A40060
01CA 00
01CB BE80FB   242          ST   4,@>FB
01CE 04
01CF 31001E   243          MOVE 30 FROM ROM(>6000) TO RAM(>420)
01D2 A42060
01D5 00
01D6 8658     244          CLR @TEMP1
01D8 865905   245          $FOR @TEMP = 0 TO 29
01DB 01DF90
01DE 59CE59
01E1 1D61F0
01E4 D4E400   246          $IF RAM(>400(TEMP1)) .NE. RAM(>420(TEMP1)) GOTO  BU
01E7 58E420
01EA 5841F2
01ED 0501DD   247          $END
01F0 4200     248          BR   BUIL5
01F2 9472     249  BUIL1   INCT @STKBAS
01F4 879072   250          DCLR *STKBAS
01F7 9472     251          INCT @STKBAS
01F9 BF9072   252          DST  #LIBH,*STKBAS
01FC 12A5
01FE 906C     253          INC  @TEMP2
0200 D68FDD   254  BUIL5   $IF  @RMNGRM .EQ. >AA THEN
0203 00AA42
0206 27
0207 BD588F   255            DST @RMNGRM+6,@TEMP1
020A DD06
020C 8F5862   256  BUIL6     $IF @TEMP1 .DNE. 0 THEN
```

```
 20F 27
0210 9472      257              INCT @STKBAS                                  2
0212 BF9072    258              DST  -1,*STKBAS                               2
0215 FFFF
0217 9472      259              INCT @STKBAS                                  2
0219 BD9072    260              DST  @TEMP1,*STKBAS                           2
021C 58
021D 906C      261              INC  @TEMP2                                   2
021F BD58CF    262              DST  @CPOFST(TEMP1),@TEMP1                    2
0222 7D0058
0225 420C      263              BR   BUIL6                                    2
               264            $END IF
               265          $END IF
0227 9472      266  BUIL    INCT @STKBAS
0229 879072    267          DCLR *STKBAS
022C 0F1A      268          XML  SGROM
022E 6227      269          BS   BUIL
0230 9672      270          DECT @STKBAS
0232 D70212    271          $IF @2 .DNE. #LIBH GOTO BUIL2
0235 A54243
0238 310001    272          MOVE 1 FROM ROM(>6000) TO @TEMP
023B 596000
023E D659AA    273          $IF @TEMP .NE. >AA GOTO BUIL3
0241 41B9
     0243      274  BUIL2   EQU  $
     0244      275  H20     EQU  $+1
0243 390001    276          MOVE 1 FROM ROM(#VDPRG$+1) TO VDP(1)
0246 010451
               277          FMT  1^,2<,'1,2,3',29<;
               278               '4,5,6',29<,'7,8,9';
0249 08A081    279               1^,31<,':PRESS:'
024C 020102
024F 039C02
0252 040506
0255 9C0207
0258 0809A0
025B 9E0450
025E 524553
0261 53FB
0263 310011    280          MOVE 17 FROM ROM(#TI) TO RAM(40)
0266 A02804
0269 96
026A 31000D    281          MOVE 13 FROM ROM(#HC) TO RAM(104)
026D A06804
0270 A7
0271 BF5200    282          DST  >E4,@INDEXM  START OF MENU
0274 E4
0275 BE5830    283          ST   >30,@TEMP1
0278 8E6C42    284          $IF @TEMP2 .EQ. 0 THEN                            1
027B 93
027C 08FF02    285            FMT XPT=2,':INSERT CARTRIDGE:'                  1
027F 0F494E
 282 534552
0285 542043
0288 415254
```

```
02 B 524944
028E 4745FB
0291 42EF      286           BR    BUIL4
               287         $END IF
               2 8         $REPEAT
0293 9058      2 9           INC   @TEMP1
0295 BCB052    290           ST    @TEMP1,RAM  INDEXM)
029  5
0299 9552      291           DINCT @IND XM
029B 310003    292           MOV   3 FROM ROM #FOR$) TO RAM(@INDEXM)
029E B05209
02A1 4D
02A2 A35200    293           DADD  4,@IND XM
02A5 04
02A6 BD6A90    294           DST   *STKBAS,@D EMP3
02A9 72
02AA 9672      295           DECT  @STKBAS
02AC BD5C90    296           DST   *STKBAS,@A G
02AF 72
02B0 9672      297           D CT  @STKBAS
02B2 A36A00    298           DADD  4,@DTEMP3
02B5 04
02B6 865       299           CLR   @ARG+2
02B8 8E5C62    300           $IF @ARG .NE. 0 THEN
02BB D0
02BC 350001    301             MOVE 1 FROM @ POFST(DTEMP3) TO @ARG+3
02BF 5FCF7D
02C2 006A
02C4 916A      302             DINC @DTEMP3
02C6 345EB0    303             MOVE @ARG+2 F OM @CPOFST(DTEMP3) TO RAM(@INDEXM
02C9 52CF7D
02CC 006A
02CE 42 0      304           $SELSE IF
02D0 330001    305             MOVE 1 FROM R M(@DTEMP3) TO @ARG+3
02D3 5F0000
02D6 6A
02D7 916A      306             DINC @DTEMP3
02D9 325EB0    307             MOVE @ARG+2 F OM ROM(@DTEMP3) TO RAM(@INDEXM)
02DC 520000
02DF 6A
               308           $END IF
02E0 A35200    309           DADD  58,@INDEXM
02E3 3A
02E4 D27200    310         $UNTIL @STKBAS .L '. 0
02E7 6293
               311  *
               312  *      SEE IF EUROPEAN G OM IS PLUGGED IN AND
               313  *      IF SO THEN GO TO 'HEIR TRANSLATION ROUTINE
               314  *
02E9 BE4313    315          ST    >13,@>43
02EC 06037C    316          CALL  EUR?
02EF 390001    317  BUIL4   MOVE 1 FROM ROM(# /DPRG$) TO VDP(1)
02F2 010450
               318  *
               319  *       PROMPT USER SELEC 'ION
```

```
                320  *
02F5 8674       321  PRIME  CLR  @KEYBRD      WAIT FOR KEY TO GO DOWN
02F7 02FF       322  RNDIT  RAND
02F9 03         323         SCAN
02FA 42F7       324         BR   RNDIT
02FC A67531     325         S  >31,@KEY
02FF C8756C     326         $IF @KEY .L. @T MP2 GOTO BRPRG
0302 430A
0304 0603D6     327  BADKEY CALL TON2$$
0307 0502F7     328         B    RNDIT
                329  *
                330  *      BRANCH TO STARTING ADDRESS OF PROGRAM
                331  *
030A 0603CE     332  BRPRG  CALL TON1$$
030D A46C75     333         SUB @KEY,@TEMP2
0310 926C       334         DEC @TEMP2
0312 E26C02     335         SLL @TEMP2,2          GENERATE STARTING ADDRESS
0315 BC7890     336         ST  *TEMP2,@RANDOM
0318 6C
0319 946C       337         INCT @TEMP2
031B BD5C90     338         DST *TEMP2,@ARG       GET HEADER ADDRESS
031E 6C
031F 955C       339         DINCT @ARG            ENTRY POINT
0321 BF729E     340         DST  >9E80,@STKBAS
0324 80
0325 8E7863     341         $IF @RANDOM .NE. 0 THEN
0328 32
0329 BD8080     342           DST @CPOFST(ARG),@>80
032C CF7D00
032F 5C
0330 433A       343         $SELSE IF
0332 330002     344           MOVE 2 FROM ROM(@ARG) TO @>80
0335 808000
0338 005C
                345         $END IF
033A 0720       346         ALL  : :
                347         $REPEAT
033C 8F80CE     348         $UNTIL @>CE .EQ. 0
033F 433C
0341 CF7010     349         $IF @MEMLEN .DGT. >1000 THEN
0344 004356
0347 BD0070     350           DST  @MEMLEN,@0
034A A7000F     351           DSUB >0FFF,@0
034D FF
034E 3400AF     352           MOVE @0 FROM RAM(>0FFF) TO RAM(>1000)
0351 1000AF
0354 0FFF
                353         $END IF
0356 8600       354         CLR @0
0358 35006F     355         MOVE >6F FROM @0 TO @1
035B 0100
035D 35003C     356         MOVE >3C FROM @0 TO @>84
0360 808400
0363 8674       357         CLR  @KEYBRD
0365 35001F     358         MOVE 31 FROM TABLE(0) TO TABLE(1)
```

```
0368 A381A3
036B 80
036C 878082    359          DCLR @>82          FINAL CLEAR
036F 8E7863    360          $IF @RANDOM .NE. 0 THEN
0372 7B
0373 9673      361            DECT @SUBSTK
0375 BD0080    362            DST @>80,@ENTADD
0378 80
0379 0FF0      363            XML >F0
               364          $END IF
037B 00        365          RTN                BRANCH TO USER PROGRAM
               366  EUR?
037C BE4260    367    ST    >60,@>42
037F 310002    368    MOVE 2 FROM ROM(>6000) TO @40
0382 286000
0385 D628AA    369    $IF @40 .EQ. >AA THEN
0388 4395
038A D22900    370      $IF @41 .LT. >00 THEN
038D 6395
038F 9473      371        INCT @SUBSTK
0391 BD9073    372        DST  @>42,*SUBSTK
0394 42
               373      $END IF
               374    $END IF
0395 00        375  NOEUR  RTN
```

```
                    377  ************************************************************
                    378  *       CHARACTER SET COPY AND SPECIAL TONE GENERATION
                    379  ************************************************************
                    380  CHR1$$
0396 310200         381          MOVE 512 FROM ROM(#CHAR1$) TO RAM(@FAC)
0399 B04A04
039C B4
039D 00             382          RTN
                    3 3  CHR2$$
039E BF80D0         384          DST  #CHAR2$,@CRULST   TEMPORARY POINTER
03A1 06B4
03A3 BE80D2         385          ST   64,@CRULST+2
03A6 40
03A7 86B04A         386  CHR2$B  CLR  RAM(@FAC)
03AA 330007         387          MOVE 7 FROM ROM(@CRULST) TO RAM(1(FAC))
03AD E0014A
03B0 0000D0
03B3 A34A00         388          DADD 8,@FAC
03B6 08
03B7 A380D0         389          DADD 7,@CRULST
03BA 0007
03BC 9280D2         390          DEC  @CRULST+2
03BF 43A7           391          BR   CHR2$B
03C1 00             392          RTN
                    393  CHR3$$
03C2 BF80D0         394          DST  #CHAR3$,@CRULST
03C5 0874
03C7 BE80D2         395          ST   31,@CRULST+2
03CA 1F
03CB 0503A7         396          B    CHR2$B
                    397  TON1$$
03CE BF5804         398          DST  #BEEP$,@TEMP1
03D1 79
03D2 F65800         399  TONEX   I/O  @TEMP1,0
03D5 00             400          RTN
                    401  TON2$$
03D6 BF5804         402          DST  #GONG$,@TEMP1
03D9 84
03DA 43D2           403          BR   TONEX              ALWAYS BRANCH
```

```
                    405  ********************************************************
                    406  *       LINKS BETW  N PROGRAMS AND DSR'S
                    407  *       REWRITTEN 9/6/79 TO DETECT FILE NAME WITH A "."
                    408  *            AS THE FIRST CHARACTER.
                    409  *
03DC 8 6D           410  LINK$$ FETCH @TYPE          GET PROGRAM TYPE TO LOOK FOR
03DE 8654           411         CLR  @SCLEN-1
03 0 BC55B0         412         ST   RAM(@SCNAME),@SCLEN    LINK NAME LENGTH
03 3 56
03 4 8658           413         CLR  @TEMP1             CHECK FOR EMBEDDED PERIODS
03E6 BD5256         414         DST  @SCNAME,@INDEXM
03E9 9152           415  LNK$LP DINC @INDEXM            LOOP COUNTER
03EB D45855         416         $IF  @TEMP1 .EQ. @SCLEN GOTO LNK$LN   IF END OF NAM
03EE 63FA
03F0 D6B052         417         $IF  RAM(@INDEXM) .EQ. '.' GOTO LNK$LN
03F3 2E63FA
03F6 9058           418         INC  @TEMP1
03F8 43E9           419         BR   LNK$LP
                    420  *
03FA 8E5864         421  LNK$LN $IF  @TEMP1 .EQ. 0 GOTO SC$121
03FD 38
03FE BC5558         422         ST   @TEMP1,@SCLEN      SET LENGTH TO USE
0401 D25508         423         $IF @SCLEN .LT. 8 THEN
0404 6438
0406 8654           424         CLR  @SCTEMP        LOAD LINK NAME INTO @FAC
0408 8780D0         425         DCLR @CRULST
040B 9156           426         DINC @SCNAME
040D 34544A         427         MOV  @SCTEMP FROM RAM(@SCNAME) TO @FAC
0410 B056
0412 A15654         428         DADD @SCLEN-1,@SCNAME
0415 0F19           429         XML  SROM
0417 9473           430  LNK$LQ INCT @SUBSTK        SAVE RETURN GROM MEMORY MAP ADDRE
0419 BD9073         431         DST  @>FA,*SUBSTK
041C 80FA
041E 0F1A           432  LNK$LO XML  SGROM
0420 442C           433         BR   SC$120         NONE FOUND
0422 9473           434         INCT @SUBSTK        PUSH ADDRESS
0424 BD9073         435         DST  *STKBAS,*SUBSTK
0427 9072
0429 9672           436         DECT @STKBAS        ADJUST POINTER
042B 00             437         RTN                 GO TO ROUTINE
042C 8F80D0         438  SC$120 DCZ  @CRULST        FINISHED LOOKING?
042F 441E           439         BR   LNK$LO         NO, KEEP LOOKING
0431 BD80FA         440         DST  *SUBSTK,@>FA OLD GROM MAP ADDRESS
0434 9073
0436 9673           441         DECT  @SUBSTK       RETURN ADDRESS
                    442         $END IF             LENGTH TOO LONG
0438 D40000         443  SC$121 CEQ  @0,@0          SET STATUS BIT NO PROGRAM FOUND
043B 01             444         RTNC
                    445  RETN$$
043C 9673           446         DECT @SUBSTK        RETURN FROM LINK PROGRAM
043E BD80FA         447         DST  *SUBSTK,@>FA
0441 9073
0443 9673           448         DECT @SUBSTK
0445 00             449         RTN
```

```
0446 05284C   450  VWARN$ B    WARN$$        IN DIFFERENT GROM
0449 052 4E   451  VERR$  B    ERR$$         IN DIFFERENT GROM
044C 052010   452  VEXEC$ B    EXEC$$        IN DIFFERENT GROM
```

```
 59C 000000    510          DATA #>0000,#>007C,#>007C,#>0000    =
05A4 004020    511          DATA #>0040,#>2010,#>0810,#>2040    >
05AC 384404    512          DATA #>3844,#>0408,#>1010,#>0010    ?
05B4 007884    513          DATA #>007 ,#> 49C,#>A498,#>807C    @
05BC 788484    514          DATA #>7884,#>8484,#>FC84,#>8484    A
05C4 F84444    515          DATA #>F844,#>4478,#>4444,#>44F8    B
05CC 788480    516          DATA #>7884,#> 080,#>8080,#>8478    C
05D4 F84444    517          DATA #>F844,#>4444,#>4444,#>44F8    D
05DC FC8080    518          DATA  >FC80,#> 0F0,#>8080,#>80FC    E
05E4 FC8080    519          DATA #>FC80,#> 0F0,#>8080,#>8080    F
05EC 788480    520          DATA #>7884,#>8080,#>9C84,#>8478    G
05F4 848484    521          DATA #>8484,#>84FC,#>8484,#>8484    H
05FC 7C1010    522          DATA #>7C10,#>1010,#>1010,#>107C    I
0604 040404    523          DATA #>0404,#>0404,#>0484,#>8478    J
060C 8890A0    524          DATA #>8 90,#>A0C0,#>A090,#>8884    K
0614 404040    525          DATA #>4040,#>4040,#>4040,#>407C    L
061C 84CCB4    526          DATA #>84CC,#>B484,#>8484,#>8484    M
0624 84C4A4    527          DATA #>84C4,#>A494,#>8C84,#>8484    N
062C FC8484    528          DATA #>FC84,#>8484,#>8484,#>84FC    O
0634 F88484    529          DATA #>F884,#>8484,#>F880,#>8080    P
063C 788484    530          DATA #>7884,#>84 4,#>8494,#>8874    Q
0644 F88484    531          DATA #>F884,#>8484,#>F890,#>8884    R
064C 788480    532          DATA #>7884,#>807 ,#>0404,#>8478    S
0654 7C1010    533          DATA #>7C10,#>1010,#>1010,#>1010    T
065C 848484    534          DATA #>8484,#>8484,#>8484,#>8478    U
0664 444444    535          DATA #>4444,#>4444,#>2828,#>1010    V
066C 848484    536          DATA #>8484,#>8484,#>84B4,#>CC84    W
0674 84 448    537          DATA #>8484,#>4830,#>3048,#>8484    X
067C 444444    538          DATA #>4444,#>4428,#>1010,#>1010    Y
0684 FC0408    539          DATA #>FC04,#>0810,#>2040,#>80FC    Z
068C 382020    540          DATA #>3820,#>2020,#>2020,#>2038    [
0694 008040    541          DATA #>0080,#>4020,#>1008,#>0400    \
069C 701010    542          DATA #>7010,#>1010,#>1010,#>1070    ]
06A4 102844    543          DATA #>102 ,#>4482,#>0000,#>C000    ^
06AC 000000    544          DATA #>0000,#>0000,#>0000,#>00FC    _
               545 CHAR2$
06B4 000000    546          DATA 000,000,000,000,000,000,000      space
06BB 101010    547          DATA 010,010,010,010,010,000,010      !
06C2 282828    548          DATA 028,028,028,000,000,000,000      "
06C9 28287C    549          DATA 028,028,07C,028,07C,028,028      #
06D0 385450    550          DATA 038,054,050,038,014,054,038      $
06D7 606408    551          DATA 060,064,008,010,020,04C,00C      %
06DE 205050    552          DATA 020,050,050,020,054,048,034      &
06E5 080810    553          DATA 008,008,010,000,000,000,000      '
06EC 081020    554          DATA 008,010,020,020,020,010,008      (
06F3 201008    555          DATA 020,010,008,008,008,010,020      )
06FA 002810    556          DATA 000,028,010,07C,010,028,000      *
0701 001010    557          DATA 000,010,010,07C,010,010,000      +
0708 000000    558          DATA 000,000,000,000,030,010,020      ,
070F 000000    559          DATA 000,000,000,07C,000,000,000      -
0716 000000    560          DATA 000,000,000,000,000,030,030      .
071D 000408    561          DATA 000,004,008,010,020,040,000      /
0724 384444    562          DATA 038,044,044,044,044,044,038      0
072B 103010    563          DATA 010,030,010,010,010,010,038      1
0732 384404    564          DATA 038,044,004,008,010,020,07C      2
```

```
0739 384404    565          DATA 038,044,004,018,004,044,038      3
0740 081828    566          DATA 008,018,028,048,07C,008,008      4
0747 7C4078    567          DATA 07C,040,078,004,004,044,038      5
074E 182040    568          DATA 018,020,040,078,044,044,038      6
0755 7C0408    569          DATA 07C,004,008,010,020,020,020      7
075C 384444    570          DATA 038,044,044,038,044,044,038      8
0763 384444    571          DATA 038,044,044,03C,004,008,030      9
076A 003030    572          DATA 000,030,030,000,030,030,000      :
0771 003030    573          DATA 000,030,030,000,030,010,020      ;
0778 081020    574          DATA 008,010,020,040,020,010,008      <
077F 00007C    575          DATA 000,000,07C,000,07C,000,000      =
0786 201008    576          DATA 020,010,008,004,008,010,020      >
078D 384404    577          DATA 038,044,004,008,010,000,010      ?
0794 38445C    578          DATA 038,044,05C,054,05C,040,038      @
079B 384444    579          DATA 038,044,044,07C,044,044,044      A
07A2 782424    580          DATA 078,024,024,038,024,024,078      B
07A9 384440    581          DATA 038,044,040,040,040,044,038      C
07B0 782424    582          DATA 078,024,024,024,024,024,078      D
07B7 7C4040    583          DATA 07C,040,040,078,040,040,07C      E
07BE 7C4040    584          DATA 07C,040,040,078,040,040,040      F
07C5 3C4040    585          DATA 03C,040,040,05C,044,044,038      G
07CC 444444    586          DATA 044,044,044,07C,044,044,044      H
07D3 381010    587          DATA 038,010,010,010,010,010,038      I
07DA 040404    588          DATA 004,004,004,004,004,044,038      J
07E1 444850    589          DATA 044,048,050,060,050,048,044      K
07E8 404040    590          DATA 040,040,040,040,040,040,07C      L
07EF 446C54    591          DATA 044,06C,054,054,044,044,044      M
07F6 446464    592          DATA 044,064,064,054,04C,04C,044      N
07FD 7C4444    593          DATA 07C,044,044,044,044,044,07C      O
0804 784444    594          DATA 078,044,044,078,040,040,040      P
080B 384444    595          DATA 038,044,044,044,054,048,034      Q
0812 784444    596          DATA 078,044,044,078,050,048,044      R
0819 384440    597          DATA 038,044,040,038,004,044,038      S
0820 7C1010    598          DATA 07C,010,010,010,010,010,010      T
0827 444444    599          DATA 044,044,044,044,044,044,038      U
082E 444444    600          DATA 044,044,044,028,028,010,010      V
0835 444444    601          DATA 044,044,044,054,054,054,028      W
083C 444428    602          DATA 044,044,028,010,028,044,044      X
0843 444428    603          DATA 044,044,028,010,010,010,010      Y
084A 7C0408    604          DATA 07C,004,008,010,020,040,07C      Z
0851 382020    605          DATA 038,020,020,020,020,020,038      [
0858 004020    606          DATA 000,040,020,010,008,004,000      \
085F 380808    607          DATA 038,008,008,008,008,008,038      ]
0866 001028    608          DATA 000,010,028,044,000,000,000      ^
086D 000000    609          DATA 000,000,000,000,000,000,07C      _
               610 CHAR3$
0874 002010    611          DATA 000,020,010,008,000,000,000      `
087B 000038    612          DATA 000,000,038,044,07C,044,044      a
0882 000078    613          DATA 000,000,078,024,038,024,078      b
0889 00003C    614          DATA 000,000,03C,040,040,040,03C      c
0890 000078    615          DATA 000,000,078,024,024,024,078      d
0897 00007C    616          DATA 000,000,07C,040,078,040,07C      e
089E 00007C    617          DATA 000,000,07C,040,078,040,040      f
08A5 00003C    618          DATA 000,000,03C,040,05C,044,038      g
08AC 000044    619          DATA 000,000,044,044,07C,044,044      h
```

```
                  657 ********************************************************
                  658 *
                  659 *              HC Mathematical package
                  660 *
                  661 ********************************************************
                  662 *
                  663 *       FLOATING POINT ROUTINES (GRAPHICS LANGUAGE)
                  664 *
                  665 *       ENTRY POINTS:
                  666 *            CNS$$     CONVERT NUMBER TO STRING
                  667 *            PUSH$$    ON FLOATING POINT STACK
                  668 *            PWR$      EXPONENTIATION
                  669 *            SQR$      SQUARE ROOT
                  670 *            EXPON$    EXPONENTIAL
                  671 *            LOG$      NATURAL LOG
                  672 *            COS$      COSINE
                  673 *            SIN$      SINE
                  674 *            TAN$      TANGENT
                  675 *            ATN$      ARCTANGENT
                  676 *            INT$      GREATEST INTEGER
                  677 ********************************************************
                  678 *
                  679 *
                  680 ********************************************************
                  681 *
                  682 *       Processor and video RAM roll-out area and
                  683 *       floating point scratch area general definitions:
                  684 *         (see individual routine listings for specifics)
                  685 *
0010              686 PROA$   EQU  >10          Processor roll-out area
03C0              687 VROA$   EQU  >3C0         Video RAM roll-out area
006E              688 VSPTR   EQU  >6E          Floating stack pointera
                  689 *
                  690 *       Following locations in status block are used by
                  691 *       floating point routines :
                  692 *
0075              693 SGN$    EQU  >75          1-byte sign (also for keycode)
0076              694 EXP$    EQU  >76          2-byte exponent
                  695 *
                  696 ********************************************************
                  697 *
                  698 *       Floating point instruction indices:
                  699 *
0001              700 ROUND$ EQU  1             Round via guard digits
0002              701 ROUNU$ EQU  2             Round up - CARRY propagation
0003              702 FACSTS EQU  3             Set status on FAC
0004              703 OVUN   EQU  4             Overflow/underflow
0005              704 OV     EQU  5             Overflow
0006              705 ADD    EQU  6             Add
0007              706 SUB    EQU  7             Subtract
0008              707 MULT   EQU  8             Multiply
0009              708 DIV    EQU  9             Divide
000A              709 COMP   EQU 10             Floating point compare
000B              710 SADD   EQU 11             Stack add
000C              711 SSUB   EQU 12             Stack subtract
```

```
000D      712  SMULT  EQU 13          Stack multiply
000E      713  SDIV   EQU 14          Stack divide
000F      714  SCOMP  EQU 15          Stack compare
0010      715  CSN$$  EQU 16          Convert string to number
0012      716  CFI$$  EQU 18          Convert floating to integer (16)
          717  *
          718  *      Error codes
          719  *
0001      720  WRNOV  EQU  1          Overflow error
0001      721  DIVZER EQU  1          Same as division by zero
0002      722  ERRSNN EQU  2          Syntax error
0003      723  ERRIOV EQU  3          Integer overflow on conversion
0004      724  ERRSQR EQU  4          Negative arg for square root
0005      725  ERRNIP EQU  5          Negative num to non-integral power
0006      726  ERRLOG EQU  6          LOG of neg number or zero
0007      727  TRIGER EQU  7          Invalid arg in trig function
```

```
                    754  ROLOUT
09A0 35001A         755    MOVE 26 FROM @PROA$ TO RAM(VROA$)  Roll out to video RAM
09A3 A3C010
09A6 8752           756    DCLR @FAC+8              Clear up FAC+8
09A8 00             757    RTN
                    758  *
                    759  CNS$$
09A9 0609A0         760    CALL ROLOUT
09AC 35000A         761    MOVE 10 FROM @FAC TO @CNSIT Copy contents of FAC
09AF 1A4A
09B1 BF5820         762    DST  :  :,@FAC+14        Assume positive number
09B4 20
09B5 D24A00         763    $IF @FAC .LT. 0 THEN     Number was negative
09B8 69BF
09BA 834A           764      DNEG @FAC              Change FAC to positive
09BC BE592D         765      ST   :-:,@FAC+15       And indicate - sign
                    766    $END IF
09BF BE165A         767    ST   FAC+16,@SPTR$       Indicate start of number (= digits
09C2 8F4A49         768    $IF @FAC .DEQ. 0 THEN    Zero gets a special treatment
09C5 E2
                    769  CNSZ
09C6 BF9016         770      DST  :0 :-: :,*SPTR$ Place one "0" plus end of string
09C9 3000
09CB 9016           771      INC  @SPTR$            Update pointer in string
09CD CE5500         772      $IF @CW$ .LE. 0 GOTO CNSLEA  We're not in calculator mo
09D0 4A9F
09D2 BE1001         773      ST   1,@VPTR$          ASSUME NO FIX MODE
09D5 D25700         774      $IF @FIX$ .GE. 0 THEN Fix mode selected
09D8 49DD
09DA A01057         775        ADD  @FIX$,@VPTR$  Compute # of 0s plus one
                    776      $END IF
09DD 060C46         777      CALL CNSPER            Make that a ".00 --- 0"
09E0 4A9F           778      BR   CNSLEA            Suppress leading zeroes and return
                    779    $END IF
                    780  *
                    781  *       Free format floating point
                    782  *
09E2 060BD3         783    CALL CNSTEN              Get base 10 exponent
09E5 8E554A         784    $IF @CW$ .NE. 0 GOTO CNSC Calculator mode = special
09E8 EF
09E9 CA770A         785    $IF @EXP$+1 .L. 10 THEN Might be printable as integer
09EC 6A04
09EE BC104A         786      ST   @FAC,@VPTR$       Might even be an integer - check
09F1 A210CC         787      ADD  FAC+2-64,@VPTR$   exponent through first frac. byt
                    788      $REPEAT
09F4 8E9010         789        $IF *VPTR$ .NE. 0 GOTO CNSF02 Print fixed format
09F7 4A04
09F9 9010           790          INC  @VPTR$        Next fractional byte please
09FB D21052         791      $UNTIL @VPTR$ .GE. FAC+8 Reached end of number
09FE 49F4
0A00 8618           792      CLR  @TYP$             Indicate that number is integer
0A02 4A24           793    $SELSE
                    794  CNSF02
0A04 BE1205         795      ST   5,@MSD$CH         Assume E-notation rounding
0A07 CE7709         796      $IF @EXP$+1 .LE. 9 THEN Number not too big for fixed
```

```
                    842  *
                    843  CNSG
0A64 BE1208         844    ST   8,@MSD$CH          Get maximum number of digits to prt
                    845  CNSG02
0A67 BE1503         846    ST   3,@DP$             Figure out where "." goes
0A6A A41514         847    SUB  @DE$,@DP$          Take a leading zero into account
0A6D 060C05         848    CALL CNSDIG            Convert back to decimal digits
0A70 060C59         849    CALL CNSUTR            Suppress trailing zeroes
                    850  *
                    851  *       Put exponent into the buffer
                    852  *
                    853  *       DATA:    SPTR$   Text pointer into buffer
                    854  *                EXP$    Exponent
                    855  *                CALL CNSEXP
                    856  *                SPTR$   Updated to point after exponent
                    857  *
0A73 BF9016         858    DST  :E+:,*SPTR$        Assume positive exponent
0A76 452B
0A78 8E766A         859    $IF @EXP$ .NE. 0 THEN  Assumption incorrect....
0A7B 83
0A7C BF9016         860      DST  :E-:,*SPTR$     Make that an "E-" then
0A7F 452D
0A81 8176           861      DABS @EXP$           Always think positive here !!!!
                    862    $END IF
0A83 9416           863    INCT @SPTR$            Skip the "E+" or "E-" field
0A85 BF9016         864    DST  :**:,*SPTR$       Be prepared for failure
0A88 2A2A
0A8A CA7764         865    $IF @EXP$+1 .L. 100 THEN Exponent within range
0A8D 6A9A
0A8F AE760A         866      DIV  10,@EXP$        Get result in EXP$,EXP$+1
0A92 A37630         867      DADD :00:,@EXP$      Combine ASCII in EXP$
0A95 30
0A96 BD9016         868      DST  @EXP$,*SPTR$    Store the result in *SPTR,*SPTR+1
0A99 76
                    869    $END IF
0A9A 9416           870    INCT @SPTR$            Skip entire xx in E+xx result
0A9C 869016         871    CLR  *SPTR$            Mark end of result string
                    872  *
                    873  *       Suppress leading zeroes and float the sign
                    874  *
                    875  *       CALL:  CALL   CNSLEA
                    876  *              MSD$CH ASCII sign
                    877  *              FAC+12 Pointer to start of number
                    878  *
                    879  CNSLEA
0A9F BE5659         880    ST   FAC+15,@FAC+12    Get pointer to sign
0AA2 BC1259         881    ST   @FAC+15,@MSD$CH   Get actual sign
                    882    $REPEAT
0AA5 BE9056         883      ST   : :,*FAC+12     Replace 0 or sign by space
0AA8 20
0AA9 9056           884      INC  @FAC+12         And try the next one
0AAB D69056         885    $UNTIL *FAC+12 .NE. :0: Suppress all 0s
0AAE 306AA5
0AB1 8E9056         886    $IF *FAC+12 .NE. 0 THEN Not end of number.....
0AB4 6ACA
```

```
0AB6 D69056   887      $IF *FAC+12 .NE. :E: THEN Also no reason to stop
0AB9 456ACA
0ABC D79056   888        $IF *FAC+12 .DNE. :. :-: : THEN Not yet end of numbe
0ABF 2E006A
0AC2 CA
0AC3 D79056   889          $IF *FAC+12 .DNE. :.E: GOTO CNSL03 No decimal ".E"
0AC6 2E454A
0AC9 D3
              890        $END IF
              891      $END IF
              892    $END IF
0ACA 9656     893    DECT @FAC+12            Point back to position of sign
0ACC BF9056   894    DST  : 0:,*FAC+12       Make that a zero without sign
0ACF 2030
0AD1 4AD9     895    BR   CNSSTR               e.i.  blank the sign.......
              896  CNSL03
0AD3 9256     897    DEC  @FAC+12            Point back to where the sign goes
0AD5 BC9056   898    ST   @MSD$CH,*FAC+12    Put original sign back in buffer
0AD8 12
              899  *
              900  *  Finish up - compute the length of the string and return
              901  *
              902  *       CALL:     FAC+12  Pointer to first character in stri
              903  *                         The string ends with a zero byte.
              904  *
              905  CNSSTR
0AD9 BC5556   906    ST   @FAC+12,@FAC+11    Save beginning of string
              907    $REPEAT                 First character has to be sign !!
0ADC 9056     908      INC  @FAC+12          Look for end of string
0ADE 8E9056   909    $UNTIL *FAC+12 .EQ. 0   Found it
0AE1 4ADC
0AE3 A45655   910    SUB  @FAC+11,@FAC+12    Compute length of string
              911  *
              912  *       Restore processor roll-out area ends
              913  *
              914  ROLIN
0AE6 35001A   915    MOVE 26 FROM RAM(VROA$) TO @PROA$
0AE9 10A3C0
0AEC 8752     916    DCLR @FAC+8
0AEE 01       917    RTNC
```

```
                 *
                 *       Calculator mode output
                 *
                 CNSC
0AEF D07755        $IF @EXP$+1 .GE. @CW$ GOTO CNSCOV # too large for width
0AF2 6B42
0AF4 060B33        CALL CNSCSD             Compute number of sign. digits
0AF7 D212FF        $IF @MSD$CH .LT. -1 GOTO CNSCUF Number too small
0AFA 4B58
0AFC D01255        $IF @MSD$CH .GE. @CW$ THEN Limit MSD$CH
0AFF 4B06
0B01 BC1255          ST   @CW$,@MSD$CH                      to width
0B04 9212            DEC  @MSD$CH                               - 1
                   $END IF
0B06 060B73        CALL CNSRND             Round according to MSD$CH
                 *
                 *       NOTE: Rounding may change the legality of a number
                 *       with respect to its size.
                 *
                 *<><><><><><><><><><><><><><><><><><><><>*
                 * TEST CHANGED FROM .GT. TO .GE.         *
                 * ON 3/13/81 BY ALLEN ACREE              *
                 * TO CORRECT BUG                         *
                 *<><><><><><><><><><><><><><><><><><><><>*
0B09 D07755        $IF @EXP$+1 .GE. @CW$ GOTO CNSCOV Got too big !!!!
0B0C 6B42
0B0E 060B33        CALL CNSCSD             Compute number of sign. digits
0B11 A21203        ADD  3,@MSD$CH          Add in overhead for display
0B14 D21203        $IF @MSD$CH .LT. 3 GOTO CNSCUF Totally insignificant
0B17 4B58
0B19 BC1055        ST   @CW$,@VPTR$
0B1C 9410          INCT @VPTR$             Calculate width + 2
0B1E CC1210        $IF @MSD$CH .GT. @VPTR$ THEN
0B21 4B26
0B23 BC1210          ST   @VPTR$,@MSD$CH  Don't exceed width + 2
                   $END
0B26 A41214        SUB  @0E$,@MSD$CH       Take leading 0s into account
0B29 060BEC        CALL CNSD$1             Convert digits into human rep.
0B2C D25700        $IF @FIX$ .LT. 0 GOTO CNSF10
0B2F 4A5F
0B31 4A9F          BR   CNSLEA             Finish up and return
                 *
                 *       Compute number of significant digits
                 *
                 CNSCSD
0B33 BC1255        ST   @CW$,@MSD$CH       MSD$CH = EXP + WIDTH (Non-fix)
0B36 D25700        $IF @FIX$ .GE. 0 THEN
0B39 4B3E
0B3B BC1257          ST   @FIX$,@MSD$CH   MSD$CH = EXP + FIX (Fix mode)
                   $END
0B3E A01277        ADD  @EXP$+1,@MSD$CH   Update both results
0B41 00            RTN
                 *
                 *       Calculator OVERFLOW
                 *
```

```
                    966  CNSCOV
 0B42 8E564B        967    $IF @CE$ .EQ. 0 THEN   No E-format allowed
 0B45 5C
                    968  CNSCV1
 0B46 BE9016        969      ST   :E:,*SPTR$            field with
 0B49 45
 0B4A 9016          970      INC  @SPTR$
 0B4C 9255          971      DEC  @CW$                      EEEEEEs
 0B4E 4B46          972      BR   CNSCV1
 0B50 869016        973      CLR  *SPTR$            Mark end of number
 0B53 BE5659        974      ST   FAC+15,@FAC+12    Point to sign
 0B56 4A09          975      BR   CNSSTR              and finish up
                    976  *
                    977  *       Calculator UNDERFLOW
                    978  *
                    979  CNSCUF
 0B58 8E5669        980      $IF @CE$ .EQ. 0 GOTO CNSZ  No E-format -> display 0
 0B5B C6
                    981    $END IF
                    982  *
                    983  *       Calculator mode E-format
                    984  *
 0B5C 35000A        985    MOVE 10 FROM @CNSIT TO @FAC Restore original FAC
 0B5F 4A1A
 0B61 BC1256        986    ST   @CE$,@MSD$CH       MSD$CH=CE$-1
 0B64 9212          987    DEC  @MSD$CH
 0B66 060B73        988    CALL CNSRND             Perform some rounding
 0B69 BC1256        989    ST   @CE$,@MSD$CH       MSD$CH=CE$+2-OE$
 0B6C 9412          990    INCT @MSD$CH
 0B6E A41214        991    SUB  @OE$,@MSD$CH
 0B71 4A67          992    BR   CNSG02
```

```
                    994  *
                    995  *       Round the number in FAC
                    996  *
                    997  *       CALL:    MSD$CH Number of decimal digits to right of
                    998  *                       most sign. digit to round
                    999  *                EXP$   Base ten exponent
                   1000  *                OE$    0 = EXP$ is even, 1 = EXP$ is odd
                   1001  *                CALL  CNSRND
                   1002  *
                   1003  *                EXP$   Base ten exponent of rounded result
                   1004  *                OE$    0 = EXP$ is even, 1 = EXP$ is odd
                   1005  *
                   1006  CNSRND
0B73 A47712        1007    SUB  @MSD$CH,@EXP$+1   Compute base 10 exp of rounding pos
0B76 A41214        1008    SUB  @OE$,@MSD$CH      Don't forget first digit
0B79 E61201        1009    SRL  @MSD$CH,1         Compute actual FAC address
0B7C 9412          1010    INCT @MSD$CH               for rounding check
0B7E B2127F        1011    AND  >7F,@MSD$CH       (SINCE SRL DOESN'T WORK FOR NEG NUM
0B81 CE1207        1012    $IF @MSD$CH .LE. 7 THEN Not everything is sign.
0B84 6BD3
0B86 BE1031        1013      ST   49,@VPTR$       Assume rounding of highest nibble
0B89 DA7701        1014      $IF .BITO @EXP$+1 .EQ. 1 THEN Rounded on odd ten's pos.
0B8C 6B91
0B8E BE1004        1015        ST   4,@VPTR$      So we have to round the low nibble
                   1016      $END IF
0B91 BE144A        1017      ST   FAC,@OE$        Get pointer into FAC
0B94 8775          1018      DCLR @SGN$           Clear both SGN$ and EXP$
0B96 BC774A        1019      ST   @FAC,@EXP$+1    Save it to see if it will change
0B99 BC184A        1020      ST   @FAC,@TYP$      Get exponent in the low byte
0B9C A01412        1021      ADD  @MSD$CH,@OE$    Compute address of rounding byte
0B9F A09014        1022      ADD  @VPTR$,*OE$     Add to round - 1 into correct byte
0BA2 10
0BA3 BC5412        1023      ST   @MSD$CH,@FAC+10 Install round start point
0BA6 0F02          1024      FLTPT ROUNU$         Propagate CARRY upwards in FAC
0BA8 8612          1025      CLR  @MSD$CH
0BAA D6144B        1026      $IF @OE$ .EQ. FAC+1 THEN rounding at first byte of FAC
0BAD 4BB4
0BAF D4184A        1027        $IF @TYP$ .NE. @FAC GOTO CNSR03 FAC is correct as is
0BB2 4BC7
                   1028      $END IF
0BB4 D61004        1029      $IF @VPTR$ .EQ. 4 THEN No rounding on byte boundary
0BB7 4BC9
0BB9 BC1390        1030        ST   *OE$,@MSD$CH+1 Retrieve rounded byte
0BBC 14
0BBD AE120A        1031        DIV  10,@MSD$CH    Divide and multiply to get
0BC0 AA120A        1032        MUL  10,@MSD$CH      rid of the remainder
0BC3 BC9014        1033        ST   @MSD$CH+1,*OE$ Put byte back into FAC
0BC6 13
                   1034  CNSR03
0BC7 9014          1035        INC  @OE$          Avoid rounding byte from
                   1036      $END IF                showing up in result
                   1037      $REPEAT
0BC9 869014        1038        CLR  *OE$          Zero next byte of FAC
0BCC 9014          1039        INC  @OE$          Take next byte of FAC
0BCE D21452        1040      $UNTIL @OE$ .GE. FAC+8 Until rest of FAC is zeroed
```

```
= 0BD1 4BC9
                1041     $END IF
                1042   * B     CNSTEN              Get new base ten exponent of FAC
                1043   *
                1044   *       Get _10 exponent of the number in FAC
                1045   *
                1046   *       CALL:    CALL  CNSTEN
                1047   *                EXP$  base 10 exponent
                104    *                OE$    XP$ odd/even - 0 = even, 1 = odd
                1049   *
                1050   CNSTEN
  0BD3 BC764A   1051     ST    @FAC,@EXP$          Get base 100 exponent
  0BD6 A67640   1052     SUB   64,@EXP$            Remove bias
  0BD9 E27601   1053     SLL   @EXP$,1             Multiply by 2 (shift left once)
  0BDC DF7600   1054     DSRA  @EXP$,8             Fix high order byte (copy sign)
  0BDF 08
  0BE0 8614     1055     CLR   @OE$                Assume even exponent
  0BE2 D24B0A   1056     $IF @FAC+1 .GE. 10 THEN First digit of FAC = 2 dec. digi
  0BE5 4BEB
  0BE7 9077     1057       INC   @EXP$+1           Add this into EXP$+1
  0BE9 9014     105        INC   @OE$              And indicate odd base 10 exp
                1059     $END IF
  0BEB 00       1060     RTN
```

```
                1062  CNSD$1
0BEC A47714     1063    SUB   @DE$,@EXP$+1
0BEF BC1577     1064    ST    @EXP$+1,@DP$        Would like DP$ = EXP + 3
0BF2 A21503     1065    ADD   3,@DP$             DP$ is not used by CNSPER !!
0BF5 D27700     1066    $IF @EXP$+1 .LT. 0 THEN Only something behind "."
0BF8 6C05
0BFA BE10FF     1067      ST    -1,@VPTR$       VPTR$=-1-EXP
0BFD A41077     1068      SUB   @EXP$+1,@VPTR$
0C00 060C46     1069      CALL  CNSPER          Create '.00---0'-string
0C03 8615       1070      CLR   @DP$             and prevent second "."
                1071    $END IF
                1072  *
                1073  *  Convert fraction of floating number in FAC into ASCII
                1074  *
                1075  *      CALL:  MSD$CH Number of decimal digits + 1 to cnvrt
                1076  *             DP$    Number of the digit the point is to
                1077  *                    the left of.
                1078  *             SPTR$  Text pointer for result
                1079  *
                1080  *             CALL   CNSDIG
                1081  *             MSD$CH 0
                1082  *             SPTR$  Updated to point to end of digits
                1083  *             SPSV$  Pointer to decimal point
                1084  *
                1085  CNSDIG
0C05 8752       1086    DCLR  @FAC+8            Don't print any guard digits
0C07 BE134B     1087    ST    FAC+1,@FP$        Get pointer to first byte of FAC
0C0A 060C33     1088    CALL  CNSD03            Check for leading decimal point
                1089  CNSD01
0C0D 8610       1090    CLR   @VPTR$            Clear high byte for divide
0C0F BC1190     1091    ST    *FP$,@VPTR$+1       get next byte of FAC
0C12 13
0C13 9013       1092    INC   @FP$
0C15 AE100A     1093    DIV   10,@VPTR$         Separate the decimal digits
0C18 A31030     1094    DADD  :00:,@VPTR$       Create ASCII result
0C1B 30
0C1C 060C2D     1095    CALL  CNSD02            Put first one in the buffer
0C1F BC1011     1096    ST    @VPTR$+1,@VPTR$   Swab the high byte down....
0C22 060C2D     1097    CALL  CNSD02            And store that one too
0C25 4C0D       1098    BR    CNSD01            Continue with this loop
                1099  *
                1100  CNSD04
0C27 BE102E     1101    ST    :.:,@VPTR$        Time to store a decimal point
0C2A BC1716     1102    ST    @SPTR$,@SPSV$     Remember where the "." is located
                1103  CNSD02
0C2D BC9016     1104    ST    @VPTR$,*SPTR$     Store digit in buffer
0C30 10
0C31 9016       1105    INC   @SPTR$            Update bufferpointer
                1106  CNSD03
0C33 9215       1107    DEC   @DP$              Time for decimal yet ????
0C35 6C27       1108    BS    CNSD04            Yep...insert one "." in buffer
0C37 D61669     1109    $IF @SPTR$ .NE. FAC+31 THEN Still no field overflow
 0C3A 6C40
0C3C 9212       1110      DEC   @MSD$CH        Completed all sign. digits ???
0C3E 4C45       1111    $SELSE                  Fall through if they have been
```

```
0C40 9673     1112       DECT @SUBSTK          Restore return address
              1113  CNSIN2
0C42 869016   1114       CLR  *SPTR$           Put 0 at end of number
              1115     $SEND IF
0C45 00       1116     RTN
```

```
                    1118  *
                    1119  *  Put some zeroes in the buffer and maybe a decimal point
                    1120  *
                    1121  *       CALL VPTR$     Number of zeroes + 1
                    1122  *            SPTR$     Text pointer into buffer
                    1123  *            CALL   CNSPER      ( Put a decimal first)
                    1124  *            CALL   CNSZ R      ( Just put in zeroes)
                    1125  *            SPTR$     Updated to point behind the zeroes
                    1126  *
                    1127  CNSPER
0C46 BE9016         1128    ST    :.:,*SPTR$          Put in one decimal point
0C49 2E
                    1129  CNSZ02
0C4A 9016           1130    INC   @SPTR$              >>>>> CHECK FOR VPTR CONTENTS <<<<<
0C4C 9210           1131    DEC   @VPTR$             More zeroes to insert ?????
0C4E CE1000         1132    $IF @VPTR$ .LE. 0 GOTO CNSIN2 Finish up if done
0C51 4C42
0C53 BE9016         1133    ST    :0:,*SPTR$          Put one zero in the buffer
0C56 30
0C57 4C4A           1134    BR    CNSZ02                and continue
                    1135  *
                    1136  *       Remove trailing zeroes
                    1137  *
                    1138  *     CALL:   SPTR$  Pointer one past end of number
                    1139  *             SPSV$  Pointer to decimal point
                    1140  *             TYP$   Zero if an integer is being printed
                    1141  *             CALL   CNSUTR
                    1142  *             SPTR$  Pointer to new end of number
                    1143  *
                    1144  CNSUTR
                    1145    $REPEAT
0C59 9216           1146      DEC   @SPTR$            Point back to next digit in number
0C5B D69016         1147    $UNTIL *SPTR$ .NE. :0:  Found a non-zero
0C5E 306C59
0C61 9016           1148    INC   @SPTR$
0C63 8E184C         1149    $IF @TYP$ .NE. 0 GOTO CNSIN2 Integer being printed
0C66 42
0C67 BC1617         1150    ST    @SPSV$,@SPTR$     End is at decimal point
0C6A 4C42           1151    BR    CNSIN2            Put a zero at the end of the string
```

```
                1153  ********************************************************
                1154  *       FLOATING POINT STACK PUSH, POP ROUTINES
                1155  ********************************************************
                1156  PUSH$$
0C6C A36E00     1157          DADD 8,@VSPTR
0C6F 08
0C70 350008     1158          MOVE 8 FROM @FAC TO RAM(@VSPTR)
0C73 B06E4A
0C76 00         1159          RTN
                1160  POP$$
0C77 350008     1161          MOVE 8 FROM RAM(@VSPTR) TO @FAC
0C7A 4AB06E
0C7D A36EFF     1162          DADD -8,@VSPTR
0C80 F8
0C81 00         1163          RTN
```

```
                1165  ************************************************************
                1166  *       FLOATING POINT MATH FUNCTIONS                      *
                1167  ************************************************************
     03DA       1168  FPSAVE EQU  >3DA          FLOATING SAVE LOCATION
     03DC       1169  FPSIGN EQU  >3DC          FLOATING SIGN SAVE
     0012       1170  P$     EQU  PROA$+2       POINTER TO POLYNOMIAL COEFF.
     0016       1171  Q$     EQU  P$+4          WORK REGISTER (2-BYTE)
     0018       1172  A$     EQU  Q$+2          8-BYTE FLOATING POINT SAVE REG
     001A       1173  C$     EQU  A$+2          8-BYTE TEMP STORE FOR ONE COEFF.
     8000       1174  SGNBIT EQU  > 000         END-OF-COEFF MARKER
                1175  ************************************************************
                1176  *       EXPONENTIATION                                     *
                1177  *                                                          *
                1178  *       FAC := B ^ E                                       *
                1179  *                                                          *
                1180  *       CALL:    EXTERNAL RAM STACK = B                    *
                1181  *                FAC = E                                   *
                1182  *                CALL  PWR($$)                             *
                1183  *                FAC RETURNS B^E OF APPROPRIATE TYPE       *
                1184  *       ERRORS:  ERRNIP   NEGATIVE NUMBER RAISED TO NON-   *
                1185  *                         INTEGRAL POWER.                  *
                1186  *                XXXXX    DIVISION BY ZERO                 *
                1187  *                XXXXX    OVERFLOW                         *
                1188  *                                                          *
                1189  *       STACK LEVELS USED:                                 *
                1190  *                                                          *
                1191  *       IF E = 0           THEN RESULT := 1                *
                1192  *       IF B = 0                                           *
                1193  *                THEN IF E LT 0     THEN DIVZER (WITH      *
                1194  *                                     (POSITIVE RESULT)    *
                1195  *                                     ELSE RESULT := 0     *
                1196  *                         RESULT = IF SIGN = NEG THEN -R ELSE R  *
                1197  ************************************************************
                1198  PWR$$
0C82 0611F6     1199          CALL POPARG       POP B TO ARG
0C85 8F4A       1200          DCZ   @FAC        IF E IS ZERO
0C87 6D41       1201          BS    PWRG01      THEN RESULT = 1
0C89 8F5C       1202          DCZ   @ARG        IF B IS ZERO
0C8B 6D39       1203          BS    PWRG02        THEN RETURN ZERO OR WARNING
0C8D A36E00     1204          DADD 8,@VSPTR     USE B ON STACK
0C90 08
0C91 8675       1205          CLR  @SGN$        ASSUME SIGN IS POSITIVE
0C93 060C6C     1206          CALL PUSH$$       IS E A FLOATING INTEGER
0C96 0611FE     1207          CALL  INT$$
0C99 8675       1208          CLR  @SGN$
0C9B 0611E0     1209          CALL XTFAC$       FAC=E STACK=INT(E)
0C9E 0F0F       1210          FLTPT SCOMP
0CAC 4D43       1211          BR    PWR$$3      NO, TRY FLOATING CODE
                1212  *
                1213  *       COMPUTE INTEGER POWER  B ^ E
                1214  *
                1215  *            E IN FAC          B ON STACK
0CA2 060C6C     1216          CALL PUSH$$          PUT E ON TOP OF B ON STACK
0CA5 8654       1217          CLR  @FAC+10
0CA7 0F12       1218          FLTPT CFI$$          TRY TO CONVERT "E" TO INTEGER
```

```
0CA9 814A    1219          DABS  @FAC              ABS OF EXPONENT
0CAB BDA3DB  1220          DST   @FAC,RAM(FPSAVE+1), SAVE THE INTEGER EXPONENT
0CAE 4A
0CAF 060C77  1221          CALL  POP$$             RETURN E TO FAC; B ON STACK
0CB2 8E544D  1222          $IF  @FAC+10 .NE. 0 GOTO PWR$$1
0CB5 02
             1223  *                              RETURN TO F.P. CODE IF E>32767
             1224  *
0CB6 0611E0  1225          CALL XTFAC$$           GET B AS ACCUMULATOR
             1226  *                              PUT E ON STACK FOR LATER SIGN CH
0CB9 060C6C  1227          CALL PUSH$$            PUT COPY OF B ON STACK FOR
             1228  *                                MULTIPLIER
             1229  *                          B,E ON STACK;  ACCUMULATOR (ACC) IN F
0CBC 93A3DB  1230          DDEC RAM(FPSAVE+1)  REDUCE EXPONENT BY ONE SINCE
             1231  *                                SINCE ACC STARTS WITH "B"
0CBF 6CE4    1232          BS    PWRJ40           IF ZERO THEN DONE ALREADY
             1233  *
0CC1 DAA3DC  1234  PWRJ30 $IF .BITO RAM(FPSAVE+2) .EQ. 0 GOTO PWRJ10
0CC4 016CCD
             1235  *                              IF THIS BIT IS SET
0CC7 0F0D    1236          FLTPT SMULT            MULTIPLY IN THIS POWER
0CC9 A36E00  1237          DADD 8,@VSPTR          RESTORE STACK
0CCC 08
0CCD E7A3DB  1238  PWRJ10 DSRL RAM(FPSAVE+1),1  GET NEXT BIT
0CD0 0001
0CD2 8FA3DB  1239          $IF RAM(FPSAVE+1) .DEQ. 0 GOTO PWRJ40  IF DONE
0CD5 6CE4
0CD7 0611E0  1240          CALL XTFAC$            EXCHANGE: B IN FAC; ACC ON STACK
0CDA 060C6C  1241          CALL PUSH$$            COPY B ONTO STACK
0CDD 0F0D    1242          FLTPT SMULT            SQUARE IT FOR NEW B
0CDF 0611E0  1243          CALL XTFAC$            RESTORE ORDER; B ON STACK; ACC IN
0CE2 4CC1    1244          BR    PWRJ30           LOOP FOR NEXT BIT
             1245  *
             1246  *
0CE4 A36EFF  1247  PWRJ40 DADD -16,@VSPTR     DONE; CLEAN UP STACK
0CE7 F0
0CE8 D2E008  1248          $IF RAM(8(VSPTR)) .LT. 0 THEN   TEST EXPONENT SIGN
0CEB 6E006D
0CEE 01
0CEF 8E544C  1249            $IF @FAC+10 .EQ. 0 THEN
0CF2 FD
0CF3 310008  1250              MOVE 8 FROM ROM(#EXPG) TO @ARG    YES,
0CF6 5C1071
0CF9 0F09    1251              FLTPT DIV             COMPUTE INVERSE
0CFB 4D01    1252            $SELSE
0CFD 874A    1253              DCLR @FAC       IF OVERFLOW THEN RESULT=0
0CFF 8654    1254              CLR  @FAC+10    NO ERROR
             1255            $END IF
             1256          $END IF
0D01 00      1257          RTN
             1258  *
             1259  *       INTEGER EXPONENT OUT OF INTEGER RANGE
             1260  *
             1261  *            E IN FAC, B ON STACK
0D02 D2B06E  1262  PWR$$1 $IF  RAM(@VSPTR) .LT. 0 THEN   TEST B
```

```
                1296  *       SQUARE ROOT FUNCTION
                1297  *
                1298  *       REFERENCE FOR SCIENTIFIC FUNCTION APPROXIMATIONS:
                1299  *       JOHN F. HART, ET AL, COMPUTER APPROXIMATIONS,
                1300  *         JOHN WILEY & SONS, 1968.
                1301  *
                1302  *       FAC := SQR(FAC)
                1303  *
                1304  *       CALL:   CALL  SQR($$)
                1305  *
                1306  *       ERRORS:    ERRSQR     SQUARE ROOT OF NEGATIVE NUMBER
                1307  *                             ATTEMPTED.
                1308  *
                1309  *       STACK LEVELS USED:
                1310  *
                1311  *       IF FAC = 0          THEN SQR := 0
                1312  *       IF FAC LT 0         THEN ERRSQR
                1313  *       FAC = A * 100^N,   .01 LE A LT 1
                1314  *       SQR := 10^N * SQR(A)
                1315  *       NEWTON'S METHOD WITH A FIXED NUMBER OF ITERATIONS
                1316  *       IS USED TO APPROXIMATE SQR(A):
                1317  *       A RATIONAL FUNCTION APPROXIMATION IS USED FOR Y(0)
                1318  *                                          (HART SQRT 0231)
                1319  *           Y(N+1) = (Y(N) + A/Y(N))/2
                1320  *
                1321  SQR$$
 0D59 0609A0    1322          CALL ROLOUT
 0D5C 0F03      1323          FLTPT FACSTS
 0D5E 6AE6      1324          BS   ROLIN          FAC IS ZERO, RETURN ZERO
 0D60 0A        1325          GT
 0D61 4DAF      1326          BR   SQR02          FAC LT ZERO, ERROR
 0D63 BC144A    1327          ST   @FAC,@0E$
 0D66 BE4A3F    1328          ST   >3F,@FAC       CREATE A IN RANGE .01 LE A LT 1
 0D69 A214C1    1329          A    -63,@0E$       REMOVE BIAS FROM EXPONENT
 0D6C DF1400    1330          DSRA @0E$,8         SIGN EXTEND
 0D6F 08
 0D70 E31400    1331          DSLL @0E$,1         SAVE 2 * N
 0D73 01
 0D74 060C6C    1332          CALL PUSH$$         SAVE A
 0D77 060C6C    1333          CALL PUSH$$         AND SAVE A AGAIN
 0D7A BF1210    1334          DST  #SQRP,@P$
 0D7D 2B
 0D7E 0611AD    1335          CALL POLY           COMPUTE P(A)
 0D81 0611E0    1336          CALL XTFAC$         STACK = P(A), FAC = A
 0D84 BF1210    1337          DST  #SQRQ,@P$
 0D87 45
 0D88 0611AD    1338          CALL POLY           COMPUTE Q(A)
 0D8B 0F0E      1339          FLTPT SDIV          COMPUTE P(A) / Q(A)
                1340  *       AT THIS POINT, FAC = Y(0), STACK = A
 0D8D BE1203    1341          ST   3,@P$
 0D90 350008    1342  SQR01   MOVE 8 FROM RAM(@VSPTR) TO @ARG
 0D93 5CB06E
 0D96 060C6C    1343          CALL PUSH$$         PUSH Y(N)
 0D99 0F09      1344          FLTPT DIV           COMPUTE A/Y(N)
 0D9B 0F0B      1345          FLTPT SADD          COMPUTE A/Y(N) + Y(N)
```

```
 09D 310008   1346   *      MOVE 8 FROM ROM(#FHALF) TO @ARG
0DA0 5C0FE3
0DA3 0F08    1347           FLTPT MULT          COMPUTE .5 * (A/Y(N)+Y(N))
0DA5 9212    1348           DEC   @P$           DECREMENT LOOP COUNTER
0DA7 4D90    1349           BR    SQR01         LOOP 3 TIMES
0DA9 A36EFF  1350           DADD -8,@VSPTR      POP A OFF STACK
0DAC F8
0DAD 4E3D    1351           BR    EXPSQT        TO FINISH UP. . .
             1352   *      Jump always
0DAF BE5404  1353   SQR02  ST    ERRSQR,@FAC+10
0DB2 4AE6    1354           BR    ROLIN         TO EXIT
             1355   *      Jump always
```

```
                1357  *       EXPONENTIAL FUNCTION
                1358  *
                1359  *       FAC := EXP(FAC)
                1360  *
                1361  *       CALL:   CALL    XP($$)
                1362  *
                1363  *       WARNINGS:   WRNOV   OVERFLOW
                1364  *
                1365  *       STACK LEVELS US D:
                1366  *
                1367  *       X := FAC * LOG10(E)
                1368  *       SO EXP(FAC) = 10^X
                1369  *       MAKE SURE X IS IN RANGE   LOG100(X) = LOG10(X)/2
                1370  *       N := INT(X)
                1371  *       R := X-N,   0 LE R LT 1
                1372  *       IF R LT .5 THEN S := R
                1373  *                  ELSE S := R - .5
                1374  *       A RATIONAL FUNCTION APPROXIMATION IS USED FOR 10^S
                1375  *       (HART EXPD 1444)
                1376  *       EXP := IF R LT .5  THEN 10^N * 10^S
                1377  *                          ELSE 10^N * 10^.5 * 10^S
                1378  *
                1379  EXP$$
 ODB4 0609A0    1380          CALL ROLOUT
 ODB7 310008    1381          MOVE 8 FROM ROM(#LOG10E) TO @ARG
 ODBA 5C0FF3
 ODBD 0F08      1382          FLTPT MULT          X = FAC * LOG10(E)
 ODBF 060C6C    1383          CALL PUSH$$         SAVE X
 ODC2 0611FE    1384          CALL  INT$$          COMPUTE N = INT(X)
 ODC5 310008    1385          MOVE 8 FROM ROM(#EXC127) TO @ARG
 ODC8 5C0FDB
 ODCB 0F0A      1386          FLTPT   COMP        IS N GT 127?
 ODCD 6DE7      1387          BS    EXP03
 ODCF 0A        1388          GT
 ODD0 4DD9      1389          BR    EXP01         YES, OUT OF RANGE
 ODD2 835C      1390          DNEG @ARG
 ODD4 0F0A      1391          FLTPT COMP          IS -127 GT N?
 ODD6 0A        1392          GT
 ODD7 4DE7      1393          BR    EXP03         YES, N IS IN RANGE
                1394  *       N IS OUT OF RANGE.
 ODD9 A36EFF    1395  EXP01   DADD -8,@VSPTR      POP X OFF STACK
 ODDC F8
 ODDD BD764A    1396          DST   @FAC,@EXP$    RECALL EXPONENT SIGN
 ODE0 8675      1397          CLR   @SGN$         RESULT IS POSITIVE
 ODE2 0F04      1398          FLTPT OVUN          TAKE OVER OR UNDERFLOW ACTION
 ODE4 050AE6    1399          B     ROLIN         TO EXIT
 ODE7 060C6C    1400  EXP03   CALL  PUSH$$       SAVE VALUE ON STACK
 ODEA 0F12      1401          FLTPT CFI$$         CONVERT TO INTEGER EXPONENT
 ODEC BD144A    1402          DST   @FAC,@DE$     AND SAVE IN TEMPORARY
 ODEF 060C77    1403          CALL POP$$          RESTORE VALUE
 ODF2 E31400    1404          DSLL @DE$,1         COMPUTE 2 * N IN DE$
 ODF5 01
 ODF6 0F0C      1405          FLTPT SSUB          COMPUTE R := X - N
 ODF8 310008    1406          MOVE 8 FROM ROM(#FHALF) TO @ARG
 ODFB 5C0FE3
```

```
)DFE 0F0A    1407          FLTPT COMP          IS .5 GT R?
0E00 0A      1408          GT
0E01 6E09    1409          BS    EXP04         YES, S := R
0E03 835C    1410          DNEG @ARG           -.5
0E05 0F06    1411          FLTPT ADD           COMPUTE S := R - .5
0E07 9114    1412          DINC @0E$           REMEMBER R GE .5, 2*N + 1
0E09 060C6C  1413  EXP04   CALL PUSH$$         SAVE A COPY OF S
0E0C BF1210  1414          DST  # XPP,@P$
0E0F 57
0E10 06119D  1415          CALL POLYW          COMPUTE S * P(S^2)
0E13 0611E0  1416          CALL XTFAC$         FAC = S, STACK = S * P(S^2)
0E16 BF1210  1417          DST  #EXPQ,@P$
0E19 71
0E1A 0611A6  1418          CALL POLYX          COMPUTE Q(S^2)
0E1D 0611F6  1419          CALL POPARG         S * P(S^2) -> ARG
0E20 A36E00  1420          DADD 8,@VSPTR
0E23 08
0E24 060C6C  1421          CALL PUSH$$         SAVE COPY OF Q(S^2)
0E27 0F06    1422          FLTPT ADD           Q(S^2) + S * P(S^2)
0E29 350008  1423          MOVE 8 FROM @FAC TO @C$
0E2C 1A4A
0E2E 060C77  1424          CALL POP$$          FAC = Q(S^2), STACK = S * P(S^2)
0E31 0611E0  1425          CALL XTFAC$         REVERSE SAME
0E34 0F0C    1426          FLTPT SSUB          COMPUTE Q(S^2) - S * P(S^2)
0E36 350008  1427          MOVE 8 FROM @C$ TO @ARG
)E39 5C1A
0E3B 0F09    1428          FLTPT DIV           COMPUTE Q + P / Q - P
             1429  *
0E3D DA1501  1430  EXPSQT  $IF .BITO @0E$+1 .EQ. 1 THEN
0E40 6E4A
0E42 310008  1431          MOVE 8 FROM ROM(#SQRTEN) TO @ARG
0E45 5C0FEB
0E48 0F08    1432          FLTPT MULT          BY SQR(10) IF M ODD
             1433          $END
0E4A 310008  1434          MOVE 8 FROM ROM(#FPOS1) TO @ARG
0E4D 5C1045
0E50 DA1502  1435          $IF .BIT1 @0E$+1 .EQ. 1 THEN
0E53 6E58
0E55 BE5D0A  1436          ST   >0A,@ARG+1     ODD POWER OF TEN
             1437          $END
0E58 DF1400  1438          DSRA @0E$,2         DIVIDE TO GET EXPONENT ADJUST
0E5B 02
0E5C A05C15  1439          A    @0E$+1,@ARG      ADD IN POWER OF 100 TO EXP
0E5F 0F08    1440          FLTPT MULT
0E61 050AE6  1441          B    ROLIN          EXIT NORMALLY
```

```
                    1443  *       NATURAL LOGARITHM FUNCTION
                    1444  *
                    1445  *       FAC := LOG(FAC)
                    1446  *
                    1447  *       CALL:    CALL  LOG
                    1448  *
                    1449  *       ERRORS: ERRLOG   LOG OF NEGATIVE NUMBER OR ZERO
                    1450  *                        ATTEMPTED.
                    1451  *
                    1452  *       STACK LEVELS USED:
                    1453  *
                    1454  *       IF FAC LE 0  THEN ERRLOG
                    1455  *       LOG(FAC) = LN(FAC) = LOG10(FAC) * LN(10)
                    1456  *       FAC = A * 10^N,   .1 LE A LT 1
                    1457  *       S := A * SQR(10), 1/SQR(10) LE S LT SQR(10)
                    1458  *       LOG10(A) = LOG10(S/SQR(10))
                    1459  *                = LOG10(S) - LOG10(SQR(10))
                    1460  *                = LOG10(S) - .5
                    1461  *       LOG := (N - .5 + LOG10(S)) * LN(10)
                    1462  *            = (N - .5) * LN(10) + LN(S)
                    1463  *       A RATIONAL FUNCTION APPROXIMATION IS USED FOR
                    1464  *       LN(S)    (HART LOGE 2687)
                    1465  *
                    1466  LOG$$
0E64 0F03           1467          FLTPT FACSTS
0E66 6EF5           1468          BS   LOG02          FAC = 0, ERROR
0E68 0A             1469          GT
0E69 4EF5           1470          BR   LOG02          FAC LT 0, ERROR
0E6B 0609A0         1471          CALL ROLOUT
0E6E 060BD3         1472          CALL CNSTEN         GET BASE 10 EXPONENT
0E71 8E144E         1473          $IF @0E$ .EQ. 0 THEN
0E74 85
0E75 310008         1474          MOVE 8 FROM ROM(#FPOS1) TO @ARG
0E78 5C1045
0E7B BE5D0A         1475          ST   10,@ARG+1
0E7E 0F08           1476          FLTPT MULT          MULTIPLY FAC BY 10
0E80 060BD3         1477          CALL CNSTEN         GET NEW EXPONENT OF 10
0E83 9376           1478          DDEC @EXP$          COMPENSATE FOR MULT
                    1479          $END
0E85 9176           1480          DINC @EXP$          COMPENSATE FOR WHERE RADIX POINT
0E87 BE4A3F         1481          ST   >3F,@FAC       PUT  A  IN PROPER RANGE
0E8A BDA3DA         1482          DST  @EXP$,RAM(FPSAVE)
0E8D 76
0E8E 310008         1483          MOVE 8 FROM ROM(#SQRTEN) TO @ARG
0E91 5C0FEB
0E94 0F08           1484          FLTPT MULT          S = A * SQR(10)
0E96 06117B         1485          CALL FORMA          Z` = (S - 1) / (S + 1)
0E99 060C6C         1486          CALL PUSH$$         PUSH Z
0E9C BF1210         1487          DST  #LOGP,@P$
0E9F 93
0EA0 06119D         1488          CALL POLYW          COMPUTE Z * P(Z^2)
0EA3 0611E0         1489          CALL XTFAC$
0EA6 BF1210         1490          DST  #LOGQ,@P$
0EA9 BD
0EAA 0611A6         1491          CALL POLYX          COMPUTE Q(Z^2)
```

```
 EAD 0F0E    1492          FLTPT SDIV          COMPUTE Z * P(Z^2) / Q(Z^2)
0EAF 060C6C  1493          CALL PUSH$$
0EB2 BD5CA3  1494          DST  RAM(FPSAVE),@ARG
0EB5 DA
0EB6 310006  1495          MOVE 6 FROM ROM(#ZER3) TO @ARG+2
0EB9 5E0FE5
0EBC 8F5C    1496          DCZ  @ARG           ZERO EXP?
0EBE 6ED2    1497          BS   LOG01
0EC0 815C    1498          DABS  ARG
0EC2 CF5C00  1499          DCGT 99,@ARG        TOO LARGE?
0EC5 63
0EC6 6EE7    1500          BS   LOG03
0EC8 BE5C40  1501          ST   >40,@ARG
0ECB 8EA3DA  1502  LOG04   CZ   RAM(FPSAVE)
0ECE 6ED2    1503          BS   LOG01
0ED0 835C    1504          DNEG @ARG
0ED2 310008  1505  LOG01   MOVE 8 FROM ROM(#FHALF) TO @FAC
0ED5 4A0FE3
0ED8 0F07    1506          FLTPT SUB            COMPUTER N - .5
0EDA 310008  1507          MOVE 8 FROM ROM(#LN10) TO @ARG
0EDD 5C0FFB
0EE0 0F08    1508          FLTPT MULT          COMPUTER (N-.5) * LN(10)
0EE2 0F0B    1509          FLTPT SADD          ADD TO LN(S)
0EE4 050AE6  1510          B    ROLIN          TO EXIT @ARG
0EE7 A75C00  1511  LOG03   DSUB 100,@ARG       SUBTRACT FIRST 100
 EEA 64
0EEB BC5E5D  1512          ST   @ARG+1,@ARG+2
0EEE BF5C41  1513          DST  >4101,@ARG     LOAD EXPONENT AND LEADING DIGIT
0EF1 01
0EF2 050ECB  1514          B    LOG04
             1515  *
0EF5 BE5406  1516  LOG02   ST   ERRLOG,@FAC+10     INDICATE WHICH ERROR
0EF8 00      1517          RTN
```

```
                1519  *       COSINE FUNCTION
                1520  *
                1521  *       FAC := COS(FAC)
                1522  *
                1523  *       CALL:   CALL  COS$$
                1524  *
                1525  *       COS(FAC) := SIN(FAC + PI/2)
                1526  *
                1527  COS$$
 0EF9 310008    152             MOVE 8 FROM ROM(#PI2) TO @ARG
 0EFC 5C1003
 0EFF 0F06      1529            FLTPT ADD               COMPUTE FAC + PI/2
                1530  *       B     SIN00$            FINISH BY COMPUTING THE SIN
                1531  *       SINE FUNCTION
                1532  *       FAC := SIN(FAC)
                1533  *       CALL:    CALL  SIN$$
                1534  *
                1535  *       IF FAC LT 0             THEN SIN(FAC) = -SIN(-FAC)
                1536  *       X := 2/PI * FAC
                1537  *       K := INT(X)
                1538  *       R := X - K,  0 LE R LT 1
                1539  *       Q := K MOD 4
                1540  *       SO K = 4 * N + Q
                1541  *       FAC = PI/2 * K + PI/2 * R = 2*PI*N + PI/2 * Q
                1542  *                                              +PI/2 * R
                1543  *       SIN(FAC) = SIN(PI/2 * Q + PI/2 * R)
                1544  *
                1545  *       QUADRANT  Q   IDENTITY
                1546  *       I         0   SIN(FAC) = SIN(PI/2 * R)
                1547  *       II        1   SIN(FAC) = SIN(PI/2 + PI/2*R)
                1548  *                       = SIN(PI - (PI/2 + PI/2*R)
                1549  *                       = SIN(PI/2 * (1-R))
                1550  *       III       2   SIN(FAC) = SIN(PI+PI/2*R)
                1551  *                       = SIN(PI - (PI+PI/2*R))
                1552  *                       = SIN(PI/2 * (-R))
                1553  *       IV        3   SIN(FAC) = SIN(3*PI/2 + PI/2*R)
                1554  *                       = SIN(3*PI/2+PI/2*R-2*PI)
                1555  *                       = SIN(PI/2 * (R-1))
                1556  *
                1557  *       QUADRANT  Q   ARGUMENT TO APPROXIMATION POLYNOMIAL:
                1558  *       I         0   R         = R,      0 LE R LT 1
                1559  *       II        1   1-R       = 1-R,    0 LT 1-R LE 1
                1560  *       III       2   -R        = -R,     -1 LT -R LE 0
                1561  *       IV        3   R-1       = -(1-R), -1 LE R-1 LT 0
                1562  *
                1563  *       A POLYNOMIAL APPROXIMATION IS USED FOR SIN(PI/2 * F
                1564  *       -1 LE R LE 1          (HART SIN 3344)
                1565  SIN$$
 0F01 0609A0    1566            CALL ROLOUT
 0F04 310008    1567            MOVE 8 FROM ROM(#RPI2) TO @ARG
 0F07 5C100B
 0F0A 0F08      1568            FLTPT MULT              X = 2/PI * FAC
 0F0C BCA3DA    1569            ST    @FAC,RAM(FPSAVE)
 0F0F 4A
 0F10 814A      1570            DABS @FAC               CONSIDER POSITIVE NUMBERS
```

```
0F12 CE4A44  1571          $IF  @FAC .GT. >44 THEN
0F15 4F1C
0F17 BE5407  1572          ST   TRIGER,@FAC+10    ERROR IN RANGE OF ARGUMENT
0F1A 4AE6    1573          BR   ROLIN
             1574          $END IF
0F1C 060C6C  1575          CALL PUSH$$          SAVE X
0F1F 0611FE  1576          CALL  INT$$           K = INT(X)
0F22 8616    1577          CLR  @Q$             ASSUME Q IS ZERO
0F24 8F4A    157           DCZ  @FAC            IS FAC ZERO?
0F26 6F3D    1579          BS   SIN02           YES, Q IS ZERO
0F28 BC774A  1580          ST   @FAC,@EXP$+1
0F2B A67746  1581          S    >46,@EXP$+1
0F2E CE7700  1582          CGT  0,@EXP$+1       IS K TOO BIG FOR (K MOD 4) TO
             1583  *                            HAVE SIGNIFICANCE?
0F31 6F3A    1584          BS   SIN01           YES, DEFAULT Q TO ZERO.
0F33 A27751  1585          A    FAC+7,@EXP$+1
0F36 BC1690  1586          ST   *EXP$+1,@Q$     NO, GET 10'S AND 1'S PLACE OF K
0F39 77
0F3A B21603  1587  SIN01   AND  3,@Q$           Q = (K MOD 4)
0F3D 0F0C    1588  SIN02   FLTPT SSUB           SSUB COMPUTE R = X - K
0F3F DA1601  1589          TBR  @Q$,0           IS Q EVEN?
0F42 6F4C    1590          BS   SIN03           YES
0F44 310008  1591          MOVE 8 FROM ROM(#FPOS1) TO @ARG
0F47 5C1045
0F4A 0F07    1592          FLTPT SUB            COMPUTE 1-R
0F4C E61601  1593  SIN03   SRL  @Q$,1
0F4F 8E16    1594          CZ   @Q$             QUADRANT III OR IV?
0F51 6F56    1595          BS   SIN04           NO
0F53 84A3DA  1596          INV  RAM(FPSAVE)     YES, CHANGE SIGN OF RESULT
0F56 BF1210  1597  SIN04   DST  #SINP,@P$      GET POLYNOMIAL P'S COEFFICIENTS
0F59 E7
0F5A 06119D  1598          CALL POLYW
0F5D 4FD1    1599          BR   ATNSGN
             1600  *       TANGENT FUNCTION
             1601  *
             1602  *       FAC := TAN(FAC)
             1603  *       CALL:   CALL    TAN$$
             1604  *
             1605  *       TAN(FAC) = SIN(FAC) / COS (FAC)
             1606  *
             1607  TAN$$
0F5F 060C6C  1608          CALL PUSH$$          SAVE FAC ON STACK
0F62 060F01  1609          CALL SIN$$           COMPUTE SIN
0F65 0611E0  1610          CALL XTFAC$
0F68 060EF9  1611          CALL COS$$           COMPUTE COS
0F6B 0611F6  1612          CALL POPARG          POP STACK INTO ARG
0F6E D65407  1613          $IF @FAC+10 .EQ. TRIGER GOTO TANRTN
0F71 6F79
0F73 0F03    1614          FLTPT FACSTS         IS COS EQUAL ZERO?
0F75 6F7A    1615          BS   TAN01           YES, ISSUE WARNING
0F77 0F09    1616          FLTPT DIV            NO. TAN := SIN(ARG)/COS(ARG)
0F79 00      1617  TANRTN  RTN                  CLEAN UP AND EXIT
0F7A BC755C  1618  TAN01   ST   @ARG,@SGN$
0F7D 0F05    1619          FLTPT OV             ISSUE OVERFLOW MESSAGE
0F7F 00      1620          RTN
```

```
                1622  *       INVERSE TANGENT FUNCTION
                1623  *
                1624  *       FAC := ATN(FAC)
                1625  *
                1626  *       CALL:   CALL ATN$$
                1627  *
                1628  *       ATN(FAC) = ARCTAN(FAC)
                1629  *       IF FAC LT 0        THEN ARCTAN(FAC) = -ARCTAN(-FAC)
                1630  *       IF 0 LE FAC L  TAN(PI/8)
                1631  *            TH N T = FAC, ARCTAN(FAC) = ARCTAN (T)
                1632  *       IF TAN(PI/8) LT FAC LT TAN(3*PI/8)
                1633  *            THEN T = (FAC-1) / (FAC+1), ARCTAN (FAC) =
                1634  *                                          PI/4 + ARCTAN(T)
                1635  *       IF TAN(3*PI/8) LE FAC
                1636  *            THEN T = -1/FAC, ARCTAN(FAC) = PI/2 + ARCTAN(T
                1637  *
                1638  *       A POLYNOMIAL APPROXIMATION IS USED FOR ARCTAN(T),
                1639  *       -TAN(PI/8) LE T LE TAN(PI/8)  (HART ARCTN 4967)
                1640  *
                1641  ATN$$
0F80 0609A0     1642          CALL ROLOUT
0F83 BCA3DA     1643          ST   @FAC,RAM(FPSAVE)
0F86 4A
0F87 814A       1644          DABS @FAC             USE ABS VALUE OF FAC
0F89 8716       1645          DCLR @Q$              ASSUME ARGUMENT IS IN RANGE
0F8B 310008     1646          MOVE 8 FROM ROM(#TANPI8) TO @ARG
0F8E 5C101B
0F91 0F0A       1647          FLTPT COMP            IS TAN(PI/8) GE ARGUMENT?
0F93 6FBD       1648          BS   ATN02            YES, EQUAL
0F95 09         1649          H
0F96 6FBD       1650          BS   ATN02            YES, GT
0F98 310008     1651          MOVE 8 FROM ROM(#TAN3P8) TO @ARG
0F9B 5C1023
0F9E 0F0A       1652          FLTPT COMP            IS TAN(3*PI/8) GT ARGUMENT?
0FA0 09         1653          H
0FA1 6FB6       1654          BS   ATN01            YES, USE CASE 2
0FA3 310008     1655          MOVE 8 FROM ROM(#FPOS1) TO @ARG
0FA6 5C1045
0FA9 BF5CBF     1656          DST  ->4001,@ARG      NO, CASE 3, COMPUTE
0FAC FF
0FAD 0F09       1657          FLTPT DIV             T = -1 / ARGUMENT
0FAF BF1610     1658          DST  #PI2,@Q$         GET POINTER TO PI/2
0FB2 03
0FB3 050FBD     1659          B    ATN02            ADD IT IN AT THE END
0FB6 06117B     1660  ATN01   CALL FORMA            CASE 2:  COMPUTE T=(ARG-1)/(ARG+1
0FB9 BF1610     1661          DST  #PI4,@Q$         GET POINTER TO PI/4 FOR ADDING
0FBC 13
0FBD BF1211     1662  ATN02   DST  #ATNP,@P$        PTR TO COEFFICIENTS
0FC0 29
0FC1 06119D     1663          CALL POLYW            ARCTAN(T) = T * P(T^^2)
0FC4 8F16       1664          DCZ  @Q$              CASE 1?  YES, DONT ADD ANYTHING I
0FC6 6FD1       1665          BS   ATNSGN
0FC8 330008     1666          MOVE 8 FROM ROM(@Q$) TO @ARG
0FCB 5C0000
0FCE 16
```

```
 FCF 0F06     1667          FLTPT ADD          ADD IN THE CONSTANT
0FD1 D2A3DA   1668  ATNSGN  CGE   0,RAM(FPSAVE)
0FD4 00
0FD5 6AE6     1669          BS    ROLIN        NO, RESULT IS CORRECT AS IS
0FD7 834A     1670          DNEG  @FAC         YES, NEGATE RESULT
0FD9 4AE6     1671          BR    ROLIN
```

```
                    1673  *       MISCELLANEOUS CONSTANTS:
OFDB 41011B         1674  EXC127  DATA >41,1,27,0,0,0,0,0          127
OFDE 000000
OFE1 0000
OFE3 3F32           1675  FHALF   DATA >3F,50                       .5
OFE5 000000         1676  ZER3    DATA 0,0,0,0,0,0
OFE8 000000
OFEB 400310         1677  SQRT N  DATA >40,3,16,22,77,66,01,69      SQR(10)
OFEE 164D42
OFF1 0145
OFF3 3F2B2A         1678  LOG10E  DATA >3F,43,42,94,48,19,03,25     LOG10(E)
OFF6 5E3013
OFF9 0319
OFFB 40021E         1679  LN10    DATA >40,2,30,25,85,09,29,94      LN(10)
OFFE 195509
1001 1D5E
1003 400139         1680  PI2     DATA >40,1,57,7,96,32,67,95       PI/2
1006 076020
1009 435F
100B 3F3F42         1681  RPI2    DATA >3F,63,66,19,77,23,67,58     2/PI
100E 134D17
1011 433A
1013 3F4E35         1682  PI4     DATA >3F,78,53,98,16,33,97,45     PI/4
1016 621021
1019 612D
1018 3F292A         1683  TANPI8  DATA >3F,41,42,13,56,23,73,10     TAN(PI/8)=SQR(2)-.
101E 0D3817
1021 490A
1023 400229         1684  TAN3P8  DATA >40,2,41,42,13,56,23,73      TAN(3*PI/8)=SQR(2)+
1026 2A0D38
1029 1749
                    1685  *       SQR POLYNOMIALS  (HART SQRT 0231)
1028 3F3A51         1686  SQRP    DATA >3F,58,81,22,90,00,00,00   P02=.58812 29E+00
102E 165A00
1031 0000
1033 3F3443         1687          DATA >3F,52,67,87,50,00,00,00   P01=.52678 75E+00
1036 573200
1039 0000
103B 3E3A51         1688          DATA >3E,58,81,20,00,00,00,00   P00=.58812 E-02
103E 140000
1041 0000
1043 8000           1689          DATA #SGNBIT
                    1690  FPOS1
1045 400100         1691  SQRQ    DATA >40,01,00,00,00,00,00,00   Q01=.1 E+01
1048 000000
104B 0000
104D 3F0963         1692          DATA >3F,09,99,99,80,00,00,00   Q00=.99999 8 E-01
1050 635000
1053 0000
1055 8000           1693          DATA #SGNBIT
                    1694  *       EXP POLYNOMIALS   (HART EXPD 1441)
                    1695  *       P02 = .18312 36015 92753 84761 54 E+02
1057 40121F         1696  EXPP    DATA >40,18,31,23,60,15,92,75
105A 173C0F
105D 5C4B
```

```
                    1697  *       P01 = .83140 67212 93711 03487 3446 E+03
105F 41081F         1698          DATA >41,08,31,40,67,21,29,37
1062 284315
1065 1D25
                    1699  *       P00 = .51780 91991 51615 35743 91297 E+04
1067 41334E         1700          DATA >41,51,78,09,19,91,51,62
106A 09135B
106D 333E
106F 8000           1701          DATA #SGNBIT
                    1702  *       Q03 = .1 E+01
1071 400100         1703  EXPQ    DATA >40,1,0,0,0,0,0,0
1074 000000
1077 0000
                    1704  *       Q02 = .15937 41523 60306 52437 552 E+03
1079 41013B         1705          DATA >41,01,59,37,41,52,36,03
107C 252934
107F 2403
                    1706  *       Q01 = .27093 16940 85158 99126 11636 E+04
1081 411B09         1707          DATA >41,27,09,31,69,40,85,16
1084 1F4528
1087 5510
                    1708  *       Q00 = .44976 33557 40578 41762 54723 E+04
1089 412C61         1709          DATA >41,44,97,63,35,57,40,58
108C 3F2339
108F 283A
1091 8000           1710          DATA #SGNBIT
                    1711  *       LOG POLYNOMIALS  (HART LOGE 2687)
                    1712  *       P04 = .35670 51030 88437 69 E+00
1093 3F2343         1713  LOGP    DATA >3F,35,67,05,10,30,98,44
1096 050A1E
1099 582C
                    1714  *       P03 = -.11983 03331 36876 1464 E+02
109B BFF562         1715          DATA #>BFF5,98,30,33,31,36,88
109E 1E211F
10A1 2458
                    1716  *       P02 = .63775 48228 86166 05782 E+02
10A3 403F4D         1717          DATA >40,63,77,54,82,28,86,17
10A6 36521C
10A9 5611
                    1718  *       P01 = -.10883 71223 55838 3228 E+03
10AB BEFF08         1719          DATA #>BEFF,08,83,71,22,35,58
10AE 534716
10B1 233A
                    1720  *       P00 = .57947 38138 44442 78265 7 E+02
10B3 40395E         1721          DATA >40,57,94,73,81,38,44,44
10B6 495126
10B9 2C2C
10BB 8000           1722          DATA #SGNBIT
                    1723  LOGQ
                    1724  *       Q04 = .1 E+01
10BD 400100         1725          DATA >40,01,0,0,0,0,0,0
10C0 000000
10C3 0000
                    1726  *       Q03 = -.13132 59772 88464 0339 E+02
10C5 BFF30D         1727          DATA #>BFF3,13,25,97,72,88,46
```

```
10C8 196148
10CB 582E
                1728  *       Q02 = .47451 82236 02606 00365 E+02
10CD 402F2D     1729          DATA >40,47,45,18,22,36,02,61
10D0 121624
10D3 023D
                1730  *       Q01 = -.64076 45 07 52556 00596 E+02
10D5 BFC007     1731          DATA #>BFC0,07,64,58,07,52,56
10D8 403A07
10DB 3438
                1732  *       Q00 = .28973 69069 22217 71601 9 E+02
10DD 401C61     1733          DATA >40,28,97,36,90,69,22,22
10E0 245A45
10E3 1616
10E5 8000       1734          DATA #SGNBIT
                1735  *       SIN POLYNOMIAL  (HART SIN 3344)
                1736  SINP
                1737  *       P07 = -.64462 13674 9 E-09
                1738  *       UPDATE 6/7/79 TO FIX VALUES OF SIN, COS >1.
                1739  *       P07 = -.64473 16000 0 E-09
10E7 C4FA2C     1740          DATA #>C4FA,44,73,16,00,00,00
10EA 491000
10ED 0000
                1741  *       P06 = .56882 03332 688 E-07
10__ 3C0544     1742          DATA >3C,05,68,82,03,33,26,88
10F2 520321
10F5 1A58
                1743  *       P05 = -.35988 09117 03133 E-05
10F7 C2FD3B     1744          DATA #>C2FD,59,88,09,11,70,31
10FA 58090B
10FD 461F
                1745  *       P04 = .16044 11684 69828 31 E-03
10FF 3E013C     1746          DATA >3E,01,60,44,11,68,46,98
1102 2C0B44
1105 2E62
                1747  *       P03 = -.46817 54131 06023 168 E-02
1107 C1D251     1748          DATA #>C1D2,81,75,41,31,06,02
110A 4B291F
110D 0602
                1749  *       P02 = .79692 62624 56180 0806 E-01
110F 3F0760     1750          DATA >3F,07,96,92,62,62,45,62
1112 5C3E3E
1115 2D3E
                1751  *       P01 = -.64596 40975 06219 07082 E+00
1117 C0C03B     1752          DATA #>C0C0,59,64,09,75,06,22
111A 40094B
111D 0616
                1753  *       P00 = .15707 96323 79489 63959 E+01
111F 400139     1754          DATA >40,01,57,07,96,32,67,95
1122 076020
1125 435F
1127 8000       1755          DATA #SGNBIT
                1756  *       ATN POLYNOMIAL  (HART ARCTN 4967)
                1757  ATNP
                1758  *       P09 = -.25357 18798 82 E-01
```

```
1129 C0FE35   1759          DATA #>C0FE,53,57,18,79,88,20
112C 39124F
112F 5814
              1760  *.      P08 = .50279 13843 885 E-01
1131 3F0502   1761          DATA >3F,05,02,79,13,84,38,85
1134 4F0D54
1137 2655
              1762  *       P07 = -.65069 99940 1396 E-01
1139 C0FA32   1763          DATA #>C0FA,50,69,99,94,01,40
113C 45635E
113F 0128
              1764  *       P06 = .76737 12439 1641 E-01
1141 3F0743   1765          DATA >3F,07,67,37,12,43,91,64
1144 250C2B
1147 5B40
              1766  *       P05 = -.90895 47919 67196 E-01
1149 C0F708   1767          DATA #>C0F7,08,95,47,91,96,72
114C 5F2F5B
114F 6048
              1768  *       P04 = .11111 04992 50526 62 E+00
1151 3F0B0B   1769          DATA >3F,11,11,10,49,92,50,53
1154 0A315C
1157 3235
              1770  *       P03 = -.14285 71269 75961 157 E+00
1159 C0F21C   1771          DATA #>C0F2,28,57,12,69,75,96
115C 390C45
115F 4B60
              1772  *       P02 = .19999 99997 89961 5228 E+00
1161 3F1363   1773          DATA >3F,19,99,99,99,97,89,96
1164 636361
1167 5960
              1774  *       P01 = -.33333 33333 32253 4275 E+00
1169 C0DF21   1775          DATA #>C0DF,33,33,33,33,32,25
116C 212121
116F 2019
              1776  *       P00 = .99999 99999 99999 08253 E+00
1171 400100   1777          DATA >40,01,0,0,0,0,0,0
1174 000000
1177 0000
1179 8000     1778          DATA #SGNBIT
```

```
                1780  *       CALCULATE (X - 1) / (X + 1)
                1781  *       CALL:     FAC   X
                1782  *                 CALL  FORMA
                1783  *
117B 060C6C     1784  FORMA   CALL PUSH$$          SAVE X ON STACK
117E 061192     1785          CALL FORMA2
1181 061192     1786          CALL FORMA2
1184 0611E0     17 7          CALL XTFAC$         SWAP (X-1) AND X
11 7 310008     17 8          MOV    FROM ROM(#FP0 1) TO @ARG
118A 5C1045
118D 0F06       1789          FLTPT ADD            X + 1
118F 0F0E       1790          FLTPT SDIV           (X - 1) / (X + 1)
1191 00         1791          RTN
                1792
1192 310008     1793  FORMA2  MOVE   FROM ROM(#FHALF) TO @ARG
1195 5C0FE3
1198 835C       1794          DNEG @ARG            MAKE INTO -.5
119A 0F06       1795          FLTPT ADD            X - .5
119C 00         1796          RTN
                1797
                1798  *       EVALUATE X * P(X^^2)
                1799  *       CALL:     P$   POINTER TO POLYNOMIAL COEFFICIENTS
                1800  *                 FAC  CONTAINS X
                1801  *                 CALL POLYW
                1802  *                 FAC  R TURNS X * P(X^^2)
119D 060C6C     1803  POLYW   CALL PUSH$$          COPY X ON STACK
11A0 0611A6     1804          CALL POLYX           COMPUTE P(X^^2)
11A3 0F0D       1805          FLTPT SMULT          MULTIPLY BY X
11A5 00         1806          RTN
                1807  *       EVALUATE P(X^^2)X
                1808  *
                1809  *       CALL:     P$    POINTER TO POLYNOMIAL COEFFICIENTS
                1810  *                 FAC   CONTAINS X
                1811  *                 CALL  POLYX
                1812  *                 FAC   RETURNS P(X^^2)
                1813  *
11A6 350008     1814  POLYX   MOVE 8 FROM @FAC TO @ARG
11A9 5C4A
11AB 0F08       1815          FLTPT MULT           SQUARE X
                1816  *
                1817  *       EVALUATE P(X)
                1818  *       CALL:     P$    POINTER TO POLYNOMIAL COEFFICIENTS
                1819  *                       THEY ARE STORED CONTIGUOUSLY IN
                1820  *                       THE ORDER:
                1821  *                       C(N), C(N-1), ... C3, C2, C1, C0
                1822  *                       THE LIST IS TERMINATED BY A WORD
                1823  *                       CONTAINING SGNBIT
                1824  *                 FAC   CONTAINS X
                1825  *                 CALL  POLY
                1826  *                 FAC   RETURNS P(X) = C(N)*X^^N
                1827  *                                    +C(N-1)*X^^(N-1)+ ...
                1828  *                                    +C2*X^^2+C1*X+C0
                1829  POLY
11AD 060C6C     1830          CALL PUSH$$
11B0 330008     1831          MOVE 8 FROM ROM(@P$) TO @FAC  PUSH FIRST CONSTANT
```

```
11B3 4A0000
 1B6 12
11B7 51CA    1832          BR    POLY03
             1833  *
11B9 350008  1834  POLY02 MOVE 8 FROM RAM(@VSPTR) TO @ARG        GET  X
11BC 5CB06E
11BF 0F08    1835          FLTPT MULT       MULTIPLY PREVIOUS RESULT BY X
11C1 330008  1836          MOVE 8 FROM ROM(@P$) TO @ARG      ADD IN THIS
11C4 5C0000
11C7 12
11C8 0F06    1837          FLTPT ADD                    COEFFICIENT.
             1838  *
11CA A31200  1839  POLY03 DADD 8,@P$          POINT TO NEXT COEFF
11CD 08
11CE 330002  1840          MOVE 2 FROM ROM(@P$) TO @ARG    AND TEST IT
11D1 5C0000
11D4 12
11D5 D75C80  1841          DCEQ  SGNBIT,@ARG END OF THE LIST?
11D8 00
11D9 51B9    1842          BR    POLY02       NO, CONTINUE COMPUTING POLYNOMIAL
11DB A36EFF  1843  POLY04 DADD -8,@VSPTR     POP X OFF STACK
11DE F8
11DF 00      1844          RTN                WITH POLYNOMIAL IN FAC
             1845  *       EXCHANGE THE FAC WITH NEXT STACK ENTRY
             1846  *       CALL:   CALL    XTFAC
11E0 060C6C  1847  XTFAC$ CALL PUSH$$        PUSH FAC ONTO STACK
11E3 A36EFF  1848          DADD -8,@VSPTR     DECREMENT STACK POINTER
11E6 F8
11E7 350008  1849          MOVE 8 FROM RAM(@VSPTR) TO @FAC
11EA 4AB06E
11ED 350008  1850          MOVE 8 FROM RAM(8(VSPTR)) TO RAM(@VSPTR)
11F0 B06EE0
11F3 086E
11F5 00      1851          RTN                RETURN TO CALLING ROUTINE
             1852  *       POP THE EXTERNAL RAM STACK ENTRY INTO ARG
11F6 350008  1853  POPARG MOVE 8 FROM RAM(@VSPTR) TO @ARG
11F9 5CB06E
11FC 51DB    1854          BR    POLY04
```

```
                    1856  ************************************************************
                    1857  *       GREATEST INTEGER FUNCTION
                    1858  *       REPLACE FAC WITH INT(FAC)
                    1859  ************************************************************
                    1860  INT$$
11FE BC754A         1861  GRINT   ST    @FAC,@SGN$          SAVE SIGN OF RESULT
1201 814A           1862          DABS  @FAC                ABSOLUTE VALUE
1203 BC774A         1863          ST    @FAC,   XP$+1        EXPONENT VALUE
1206 8676           1864          CLR   @EXP$               Z RO MSD
1208 D27740         1865          $IF   @ XP$+1 .LT. >40 GOTO BITINT
1208 524B
120D CE7745         1866          $IF   @EXP$+1 .GT. >45 GOTO INT02
1210 7241
1212 BC5577         1867          ST    @ XP$+1,@FAC+11
1215 A65546         1868          SUB   >46,@FAC+11         LOCATE POSITION
1218 BC5455         1869          ST    @FAC+11,@FAC+10     SAVE FOR ROUNDUP
121B 8656           1870          CLR   @FAC+12
121D BE5752         1871          ST    FAC+8,@FAC+13
1220 A05755         1872          ADD   @FAC+11,@FAC+13     POINT TO FIRST FRACTIONAL DI
1223 B45690         1873  INT01   OR    *FAC+13,@FAC+12     REMEMBER IF NONZERO
1226 57
1227 869057         1874          CLR   *FAC+13             CLEAR THE DIGIT
122A 9057           1875          INC   @FAC+13             GET NEXT DIGIT
122C 9055           1876          INC   @FAC+11             TEST FOR END OF LOOP
122E 5223           1877          BR    INT01
1230 D27500         1878          $IF   @SGN$ .LT. 0 THEN
1233 7241
1235 8E5672         1879          $IF   @FAC+12 .EQ. 0 GOTO INT02
1238 41
1239 A25407         1880          ADD   7,@FAC+10           WHERE TO ROUND UP
123C 0F02           1881          FLTPT ROUNU$
123E 051248         1882          B     INT03
                    1883          $END IF
1241 D27500         1884  INT02   $IF @SGN$ .LT. 0 THEN
1244 7248
1246 834A           1885          DNEG  @FAC                NEGATIVE NUMBER
                    1886          $END IF
1248 8654           1887  INT03   CLR   @FAC+10             CLEAR ERROR BYTE
124A 00             1888          RTN                       RETURN TO CALLING ROUTINE
```

```
124B 874A    1890 BITINT DCLR @FAC                ZERO OR MINUS ONE
124D D27500  1891        $IF @SGN$ .LT. 0 THEN
1250 7256
1252 BF4ABF  1892        DST  >BFFF,@FAC
1255 FF
             1893        $END IF
1256 874C    1894        DCLR @FAC+2
1258 874E    1895        DCLR @FAC+4
125A 8750    1896        DCLR @FAC+6
125C 5248    1897        BR   INTO3
125E 310040  1898 BITRVR MOVE >40 FROM ROM(#BITR) TO @>00
1261 001267
1264 0FF0    1899        XML >F0
1266 00      1900        RTN
1267 8302C0  1901 BITR   DATA #>8302,#>C060,#>834A,#>C0A0,#>834C,#>D7E0
126A 60834A
126D C0A083
1270 4CD7E0
1273 83E3D7  1902        DATA #>83E3,#>D7C1,#>0261,#>4000,#>D0EF,#>FBFE
1276 C10261
1279 4000D0
127C EFFBFE
127F 020500  1903        DATA #>0205,#>0008,#>0914,#>0A13,#>1702,#>0224
1282 080914
1285 0A1317
1288 020224
128B 800006  1904        DATA #>8000,#>0605,#>16F9,#>D7E0,#>83E3,#>D7C1
128E 0516F9
1291 D7E083
1294 E3D7C1
1297 02413F  1905        DATA #>0241,#>3FFF,#>DBC4,#>FFFE,#>0581,#>0602
129A FFDBC4
129D FFFE05
12A0 810602
12A3 16E604  1906        DATA #>16E6,#>045B
12A6 5B
             1907 *
     12A5    1908 LIBH   EQU  $-2
12A7 013415  1909        DATA #BUILD,21,:REVIEW MODULE LIBRARY:
12AA 524556
12AD 494557
12B0 204D4F
12B3 44554C
12B6 452040
12B9 494252
12BC 415259
             1910        END

ERRORS=     0

LENGTH= 4799 (>12BF)

268 SYMBOLS USED
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
A$	0018	1172	1173
ADD	0006	705	1411 1422 1529 1667 1789 1795 1837
ARG	005C	47	48 296 299 300 301 303 305 307 338 339 342 344 1202 1250 1342 1346 1381 1385 1390 1406 1410 1427 1431 1434 1436 1439 1474 1475 1483 1494 1495 1496 1498 1499 1501 1504 1507 1511 1512 1512 1513 152 1567 1591 1618 1646 1651 1655 1656 1666 178 1793 1794 1 14 1834 1836 1840 1841 1853
ATN$$	0F80	1641	95
ATN01	0FB6	1660	1654
ATN02	0FBD	1662	1648 1650 1659
ATNP	1129	1757	1662
ATNSGN	0FD1	1668	1599 1665
BADKEY	0304	327	
BEEP$	0479	473	398
BITINT	124B	1890	1865
BITR	1267	1901	1898
BITRVR	125E	1898	99
BRP10	00BE	166	171
BRP20	00DC	173	167
BRPRG	030A	332	326
BUIL	0227	266	269
BUIL1	01F2	249	246
BUIL2	0243	274	271
BUIL3	01B9	237	273
BUIL4	02EF	317	286
BUIL5	0200	254	248
BUIL6	020C	256	263
BUILD	01B4	235	1909
C$	001A	1173	1423 1427
CB	0008	1	
CE$	0056	751	967 980 986 989
CFI$$	0012	716	1218 1401
CHAR	0004	1	178 179
CHAR1$	04B4	480	108 178 381
CHAR2$	06B4	545	109 384
CHAR3$	0874	610	110 394
CHR1$$	0396	380	81
CHR2$$	039E	383	82
CHR2$B	03A7	386	391 396
CHR3$$	03C2	393	106
CLEARU	0000	37	154
CNS$$	09A9	759	80
CNSC	0AEF	922	784
CNSCOV	0B42	966	923 940
CNSCSD	0B33	956	924 941
CNSCUF	0B58	979	925 943
CNSCV1	0B46	968	972
CNSD$1	0BEC	1062	836 950
CNSD01	0C0D	1089	1098
CNSD02	0C2D	1103	1095 1097
CNSD03	0C33	1106	1088
CNSD04	0C27	1100	1108
CNSDIG	0C05	1085	848
CNSF02	0A04	794	789

SYMBOL	VALUE	DEF	REFERENCE TABLE
CNSF10	0A5F	837	951
CNSG	0A64	843	811 813 830
CNSG02	0A67	845	992
CNSIN2	0C42	1113	1132 1149 1151
CNSIT	001A	744	761 9 5
CNSL03	0AD3	896	9
CNSLEA	0A9F	879	772 77 39 952
CNSPER	0C46	1127	777 1069
CNSR03	0BC7	1034	1027
CNSRND	0B73	1006	805 930 988
CNSSTR	0AD9	905	895 975
CNSTEN	0BD3	1050	783 1472 1477
CNSUTR	0C59	1144	83 849
CNSZ	09C6	769	980
CNSZ02	0C4A	1129	1134
COMP	000A	709	13 6 1391 1407 1647 1652
COS$$	0EF9	1527	92 1611
CPOFST	7D00	31	32 262 301 303 342
CRLIST	0002	38	148 149 150 151 152 153 155 156 157 158 161 162
CRULST	00D0	59	211 222 384 385 387 389 390 394 395 425 43
CSN$$	0010	715	
CW$	0055	750	772 784 923 926 927 940 944 957 971
DISPLA	0009	1	
DIV	0009	708	1251 1344 1428 1616 1657
DIVZER	0001	721	
DP$	0015	740	846 847 1064 1065 1070 1107
DSR	1310	27	71
DTEMP3	006A	49	294 29 301 302 303 305 306 307
ENTADD	0000	36	362
ERR$$	284E	24	451
ERRIOV	0003	723	
ERRLOG	0006	726	1516
ERRNIP	0005	725	1293
ERRSNN	0002	722	
ERRSQR	0004	724	1353
EUR?	037C	366	207 316
EXC127	0FDB	1674	1385
EXEC$$	2010	26	452
EXP$	0076	694	785 796 797 799 801 811 812 813 830 859 861 865 866 867 868 923 940 961 1007 1014 1019 1051 1052 1053 1054 1057 1063 1064 1066 1068 1396 1478 1480 1482 1580 1581 1582 1585 1586 1863 1864 1865 1866 1867
EXP$$	0DB4	1379	90 1277
EXP01	0DD9	1395	1389
EXP03	0DE7	1400	1387 1393
EXP04	0E09	1413	1409
EXPP	1057	1696	1414
EXPQ	1071	1703	1250 1417
EXPSQT	0E3D	1430	1351
FAC	004A	39	40 41 42 43 44 45 46 47 381 386 387 388 427 750 751 752 756 761 762 763 764 765 767 768 786 787 791 817 880 880

SYMBOL	VALUE	DEF	REFERENCE TABLE
			881 8 3 884 8 5 886 887 888 889 893 894
			897 898 906 906 908 909 910 910 916 974
			974 985 1017 1019 1020 1023 1026 1027 1040 1051
			1056 1086 1087 1109 1158 1161 1200 1217 1219 1220
			1222 1249 1253 1254 1263 1263 1264 1265 1266 1274
			1279 12 2 1284 1286 1293 1327 1328 1353 1396 1402
			1423 14 1 1505 1516 1569 1570 1571 1572 1578 1580
			15 5 1613 1643 1644 1670 1814 1831 1849 1861 1862
			1863 1 67 186 1869 1869 1870 1871 1871 1872 1872
			1873 1873 1874 1875 1876 1879 1880 1885 1887 1890
			1892 1894 1895 1896
FACSTS	0003	702	1323 1467 1614
FHALF	0FE3	1675	1346 1406 1505 1793
FIGURE	0002	1	
FIX$	0057	752	774 775 951 95 959
FLG$	0011	736	
FOR$	094D	642	292
FORMA	117B	1784	1485 1660
FORMA2	1192	1793	1785 1786
FP$	0013	738	1087 1091 1092
FPOS1	1045	1690	1286 1434 1474 1591 1655 1788
FPSAVE	03DA	1168	1220 1230 1234 1238 1239 1482 1494 1502 1569 1596
			1643 1668
FPSIGN	03DC	1169	1272 1278
GETSTR	4D12	25	98
GONG$	0484	475	402
GRINT	11FE	1861	
H04	0004	30	
H20	0244	275	174
H80	044F	458	168
HAA	0000	65	
HC	04A7	479	201 281
INDEXM	0052	40	282 290 291 292 293 303 307 309 414 415
			417
INT$$	11FE	1860	87 1207 1384 1576
INT01	1223	1873	1877
INT02	1241	1884	1866 1879
INT03	1248	1887	1882 1897
KEY	0075	56	189 191 191 193 194 325 326 333
KEYBRD	0074	55	183 228 321 357
LIBH	12A5	1908	252 271
LINK$$	03DC	410	78
LN10	0FFB	1679	1507
LNK$LN	03FA	421	416 417
LNK$LO	041E	432	439
LNK$LP	03E9	415	419
LNK$LQ	0417	430	100
LOG$$	0E64	1466	91 1275
LOG01	0ED2	1505	1497 1503
LOG02	0EF5	1516	1468 1470
LOG03	0EE7	1511	1500
LOG04	0EC8	1502	1514
LOG10E	0FF3	1678	1381
LOGP	1093	1713	1487
LOGQ	10BD	1723	1490

SYMBOL	VALUE	DEF	REFERENCE TABLE
M$EDIT	2010	5	26
M$EUR	1320	12	72
M$EXEC	4D00	6	25
M$PSCN	2828	4	22 23 24
M$TAPE	1310	7	12 27
M MCHK	2844	22	101
M MLEN	0070	52	165 166 169 170 170 171 173 349 350
MSD$CH	0012	737	795 79 800 01 818 819 821 828 830 835
			844 8 1 898 925 926 927 928 942 943 946
			947 949 957 959 961 986 987 989 990 991
			1007 1008 1009 1010 1011 1012 1021 1023 1025 1030
			1031 1032 1033 1110
MULT	0008	707	1347 13 2 1432 1440 1476 1484 1508 1568 1815 1835
NOEUR	0395	375	
OE$	0014	739	818 847 949 991 1008 1017 1021 1022 1026 1030
			1033 1035 103 1039 1040 1055 1058 1063 1327 1329
			1330 1331 1402 1404 1412 1430 1435 1438 1439 1473
OV	0005	704	12 1619
OVUN	0004	703	139
P$	0012	1170	1171 1334 1337 1341 1348 1414 1417 1487 1490 1597
			1662 1 31 1836 1839 1840
PGMCTR	37B4	11	102
PI2	1003	1680	1528 1658
PI4	1013	1682	1661
POLY	11AD	1829	1335 1338
POLY02	11B9	1834	1842
POLY03	11CA	1839	1832
POLY04	11DB	1843	1854
POLYW	119D	1803	1415 1488 1598 1663
POLYX	11A6	1814	1418 1491 1804
POP	0005	1	
POP$$	0C77	1160	1221 1403 1424
POPARG	11F6	1853	1199 1419 1612
PRIME	02F5	321	
PROA$	0010	686	735 736 737 738 739 740 741 742 743 744
			755 915 1170
PUSH$$	0C6C	1156	1206 1216 1227 1241 1332 1333 1343 1383 1400 1413
			1421 1486 1493 1575 1608 1784 1803 1830 1847
PWR$$	0C82	1198	88
PWR$$1	0D02	1262	1222
PWR$$2	0D1F	1272	1291
PWR$$3	0D4B	1291	1211
PWR207	01A2	222	217
PWRG01	0D41	1286	1201
PWRG02	0D39	1282	1203
PWRG05	0D48	1288	1283
PWRJ10	0CCD	1238	1234
PWRJ30	0CC1	1234	1244
PWRJ40	0CE4	1247	1232 1239
PWRT$$	0074	140	
PWRUP$	0052	121	86
Q$	0016	1171	1172 1577 1586 1587 1589 1593 1594 1645 1658 1661
			1664 1666
RAM	0001	1	164 166 167 169 175 176 176 199 200 201
			241 243 246 246 280 281 290 292 303 307

```
                1           TITLE  CASSETTE
                2           GROM 0
                3           ORG  >1310
                4   ********************************************************
                5   *       DEVICE SERVICE ROUTINE FOR AUDIO CASSETTE
                6   ********************************************************
                7   *       FILE MANAGEMENT ROUTINES OR USER PROGRAMS SET UP PAI
                8   *       CONTAINING CONTROL DATA FOR A CASSETTE OPERATION.
                9   *
               10   *       TH  PAB IS IN VDP RAM IMMEDIATELY PRECEDING THE DEV
               11   *       NAM  USED TO LINK TO THE DS . THROUGH THE MONITOR.
               12   *       PAB+0  -  I/O OPCODE
               13   *       PAB+1  -  FLAGS/STATUS
               14   *       PAB+2  -  DATA BUFFER ADDRESS
               15   *       PAB+4  -  LOGICAL RECORD LENGTH
               16   *       PAB+5  -  CHARACTER COUNT
               17   *       PAB+6  -  RECORD NUMBER (UNUSED)
               18   *       PAB+8  -  BIAS FOR ASCII CHARACTERS
               19   *       PAB+9  -  LENGTH OF DEVICE NAME
               20   *       PAB+10 -  FIRST CHARACTER OF DEVICE NAME
               21   ********************************************************
003D           22   LNK$LQ EQU  >3D              ADDRESS OF MIDDLE OF LINK RO
0034           23   TONE1  EQU  >34              ACCEPT TONE
0036           24   TONE2  EQU  >36              WRONG INPUT TONE
0073           25   SUBSTK EQU  >73              SUBROUTINE STACK POINTER
0074           26   PLYNUM EQU  >74              PLAYER NUMBER FOR SCAN
0075           27   KEYNUM EQU  >75              KEY CODE RETURNED
0079           2    TIMER  EQU  >79              TIMER BYTE IN STATUS
007C           29   STATUS EQU  >7C              STATUS BYTE
004A           30   FAC    EQU  >4A              BUFFER FOR PAB
0056           31   SCNAME EQU  FAC+12           ADDRESS OF PAB
005            32   TEMP1  EQU  FAC+14           TEMPORARY
0054           33   SCTEMP EQU  FAC+10           TEMPORARY
006            34   MOTOR  EQU  FAC+30           I/O CALL BLOCK FOR MOTOR
006C           35   ONOFF  EQU  FAC+34           MOTOR STATUS FLAG
005A           36   SAVSUB EQU  FAC+16           SAVE ORIGINAL SUBROUTINE POI
005C           37   CONTRL EQU  FAC+18           I/O CALL BLOCK FOR CASSETTE
004C           3    BUFADD EQU  FAC+2            BUFFER ADDRESS
0050           39   LENGTH EQU  FAC+6            RECORD NUMBER
0052           40   BIAS   EQU  FAC+8            BIAS FOR CHARACTER XFERS
0062           41   TEMP   EQU  FAC+24           TEMPORARY
0064           42   ADDR   EQU  FAC+26           TEMPORARY
0066           43   UNIT   EQU  FAC+28           TAPE NUMBER
000D           44   BAUDRW EQU  >D               WRITE/READ BAUD RATE
001            45   RETN$$ EQU  >012             RETURN ADDRESS
000            46    NTK Y EQU  >0D               NTER KEY
0045           47   SH     EQU  :E:                KEY
0043           48   SHC    EQU  :C:             C KEY
0052           49   SHR    EQU  :R:             R KEY
004            50   NOKEY  EQU  :N:             N KEY
0059           51   YESKEY EQU  :Y:             Y KEY
```

```
13 0 86EFFF     96  EOF     CLR   RAM(-4(SCNAME))
13 3 FC56
   15 5363      97          BR    EXIT
```

```
                    186  ********************************************************
                    187  *       VERIFY CASSETTE ROUTINE
                    188  ********************************************************
                    189  *
                    190  *       THE RECORD NUMBER CONTAINS THE LENGTH OF THE
                    191  *       BUFFER TO BE VERIFIED.
                    192  ********************************************************
1444 061503         193  TCHECK CALL BEGIN                  POSITION TAPE MESSAGE, MOT
1447 06149F         194         CALL MPLAY                  CASSETTE PLAY MESSAGE
144A 08             195         DATA                        "CHECKING"
                    196         $REPEAT
144B 8E80CE         197         $UNTIL @>CE .EQ. 0          WAIT FOR SOUND TO COMPLETE
144E 5448
1450 F65C06         198         I/O  @CONTRL,6              VERIFY CASSETTE I/O CALL
1453 5408           199         BR   TOKAY                  CONDITION SET IF ERROR
1455 DA7C01         200  ERROR  $IF .BITO @STATUS .EQ. 1 THEN
1458 7463
145A 06155E         201         CALL MTROFF                 TURN OFF MOTOR
145D 0614A9         202         CALL MESS                   DISPLAY MESSAGE
1460 02             203         DATA 2                      "ERROR DETECTED IN DATA -"
1461 546A           204         $SELSE
1463 06155E         205         CALL MTROFF                 TURN OFF MOTOR
1466 0614A9         206         CALL MESS                   DISPLAY MESSAGE
1469 1A             207         DATA 26                     "ERROR - NO DATA FOUND"
                    208         $END IF
146A D6EFFF         209         $IF RAM(-10(SCNAME)) .NE. 6 THEN
146D F65606
1470 7478
1472 0614FE         210         CALL SMESS                  "PRESS R TO READ"
1475 14             211         DATA 20
1476 547C           212         $SELSE
1478 0614FE         213         CALL SMESS                  "PRESS R TO RECORD"
147B 1E             214         DATA 30
                    215         $ ND IF
147C 0614FE         216         CALL SMESS                  "PRESS C TO CHECK"
147F 1C             217         DATA 28
1480 0614FE         21   ER1    CALL SMESS                  "PRESS A TO ABORT"
14 3 16             219         DATA 22
14 4 061528         220         CALL ENTER                  WAIT FOR R OR C OR E KEY
14 7 533C           221         BR   RETRY                  REATTEMPT I/O OPERATION
```

```
                    237  ********************************************************
                    238  *       SUBROUTINES
                    239  ********************************************************
1499 0614A9         240  MREC    CALL MESS                DISPLAY MESSAGE
149C 0C             241          DATA 12                  "PRESS CASSETTE RECORD"
149D 54A3           242          BR   MTAPE
149F 0614A9         243  MPLAY   CALL MESS                DISPLAY MESSAGE
14A2 0E             244          DATA 14                  "PRESS CASSETTE PLAY"
14A3 061528         245  MTAPE   CALL ENTER               WAIT FOR ENTER KEY
14A6 061549         246          CALL MTRON               TURN ON MOTOR
14A9 3502C0         247  MESS    MOV  >2C0 FROM RAM(>40) TO RAM(0)     SCROLL UP T
14AC A000A0
14AF 40
14B0 08FE16         248          FMT  YPT=22,XPT=0,BIAS=@BIAS,32':  :',':   *:',29':
14B3 FF00FD
14B6 525F20
14B9 022020
14BC 2A5C20
14BF FB
14C0 8662           249  QMESS   CLR  @TEMP
14C2 8863           250          FETCH @TEMP+1
14C4 BF6402         251  QMESS1  DST  >2E4,@ADDR          STARTING SCREEN ADDRESS
14C7 E4
14C8 BE4A03         252  LMESS   ST   >03,@FAC           LOOK FOR SUBROUTINE 03
14CB BF5400         253          DST  1,@SCTEMP           LENGTH=1
14CE 01
14CF BE6D0A         254          ST   10,@>6D             PROGRAM TYPE
14D2 8780D0         255          DCLR @>D0                START LOOKING FROM BEGIN
14D5 06003D         256          CALL LNK$LQ              FIND ROUTINE
14D  8E7575         257          $IF @KEYNUM .EQ. 0 GOTO MRTN
14DB 13
14DC BF7E17         258  PMESS   DST >171B,YPT                    POINTER TO SCREEN
14DF 1B
14E0 08FD52         259          FMT  BIAS=@BIAS,':CS1:'          OUTPUT DEVICE NAM
14 3 024353
14 6 31FB
14   D66716         260          $IF  UNIT+1 .EQ. 22 GOTO PMESS1
14 B 74F0
14 D 90A2FD         261          INC  RAM(>2FD)
14F0 D675FE         262  PMESS1  $IF @KEYNUM .EQ. -2 GOTO MRTN
14F3 7513
14F5 BF6200         263          DST  #18,@TEMP                   THEN PRESS ENTER
14F  12
14F9 061516         264          CALL SCROLL
14FC 54C4           265          BR   QMESS1
14FE 061516         266  SMESS   CALL SCROLL
1501 54C0           267          BR   QMESS
                    26   *
1503 061549         269  BEGIN   CALL MTRON               TURN ON MOTOR
1506 0614A9         270          CALL MESS                DISPLAY MESSAGE
1509 0A             271          DATA 10                  "REWIND CASSETTE TAPE"
150A 061528         272          CALL ENTER               WAIT FOR ENTER KEY
150D 06155E         273          CALL MTROFF              TURN MOTOR OFF
1510 BD5C50         274          DST  @LENGTH,@CONTRL     SET LENGTH LIMITS
                    275  MRTN
```

```
1513 050034   276          B    TONE1                GO OUT AND BEEP
1516 3502E0   277  SCROLL  MOVE >2E0 FROM RAM(>20) TO RAM(>0)
   9 A000A0
151C 20
151D 08FE17   278          FMT YPT=23,XPT=0,BIAS=@BIAS,32'  '
1520 FF00FD
1523 525F20
1526 FB
1527 00       279          RTN
              280  *
1528 03       281  ENTER   SCAN                        WAIT FOR KEY DOWN
1529 5528     282          BR   ENTER
152B BC5873   283          ST   @SUBSTK,@TEMP1         FIX UP STACK
152E BC735A   284          ST   @SAVSUB,@SUBSTK
1531 D67545   285          $IF @KEYNUM .EQ. SHE GOTO DEVERR
1534 7371
1536 D67543   286          $IF @KEYNUM .EQ. SHC GOTO TCHECK
1539 7444
153B D67552   287          $IF @KEYNUM .EQ. SHR GOTO RETRY
153E 733C
1540 BC7358   288          ST   @TEMP1,@SUBSTK
1543 D6750D   289          $IF @KEYNUM .NE. ENTKEY GOTO ENTER
1546 5528
1548 00       290  ENT1    RTN
              291  *
1549 BE6CFF   292  MTRON   ST   >FF,@ONOFF             TURN ON MOTOR
154C BD6866   293  MTRING  DST  @UNIT,@MOTOR          SET UP MOTOR CALLS
  4F BF6A01   294          D T  >100+ONOFF,@MOTOR+2          NUMBER OF BITS
1552 6C
1553 F66803   295          I/O  @MOTOR,3               CRU I/O COMMAND
1556 8679     296          CLR  @TIMER                 WAIT A  HALF SECOND
              297          $R PEAT
155  CE791E   298          $UNTIL @TIMER .GT. 30
155B 5558
155D 00       299          RTN
155  866C     300  MTROFF  CLR  @ONOFF                TURN OFF MOTOR
1560 554C     301          BR   MTRING
              302  *
              303  STALL
1562 8662     304          CLR  @TEMP
              305          $R PEAT                     WAIT FOR TEN  SECONDS
1564 8679     306          CLR  @TIMER                 TO GET PAST LEADER
              307            $REPEAT
1566 CE793C   308            $UNTIL @TIMER .GT.  60
1569 5566
156B 9062     309          INC  TEMP
156D C 620A   310          $UNTIL @TEMP .GT. 10
1570 5564
1572 00       311          RTN
```

```
1573 330002   313  %1     MOVE 2 FROM ROM(MTABLE(TEMP)) TO @TEMP1 ADDRESS O
1576 5815A0
1579 62
157A 330002   314         MOVE 2 FROM ROM(@TEMP1) TO @TEMP  LENGTH AND RE..
157D 620000
1580 58
1581 BC7563   315         ST   @TEMP+1,@KEYNUM   STORE RETURN CHARACTER
1584 E76200   316         DSRL @TEMP,8           RIGHT JUSTIFY LENGTH
1587 08
1588 3262B0   317         MOV  @TEMP FROM ROM(2(TEMP1)) TO RAM(@ADDR)
158B 640002
158E 5
15 F A16264   318         DADD @ADDR,@TEMP       LAST CHARACTER ADDRESS
1592 A0B064   319  %2     ADD @BIAS,RAM(@ADDR)   ADD BIAS
1595 52
1596 9164     320         DINC  @ADDR
1598 D16462   321         $IF @ADDR .DLT. @TEMP  GOTO %2
159B 5592
159D 060012   322         CALL RETN$$            RETURN
              323  ********************************************************
              324  *       MESSAGES AND DATA
              325  ********************************************************
15A0 15C015   326  MTABLE DATA #M1,#M2,#M3,#M4,#M5,#M6,#M7,#M8
15A3 C915E1
15A6 15F616
15A9 01160B
15AC 162116
15AF 38
15B0 164D16   327         DATA #M9,#M10,#M11,#M12,#M13,#M14,#M15,#M16
15B3 631675
15B6 168616
15B9 9716A0
15BC 16B716
15BF C9
15C0 0700     328  M1     DATA #>0700
15C2 524541   329  MESS1  DATA :READING:
15C5 44494E
15C  47
15C9 1600     330  M2     DATA #>1600
15CB 455252   331  MESS2  DATA :ERROR DETECTED IN DATA:
15CE 4F5220
15D1 444554
15D4 454354
15D7 454420
15DA 494E20
15DD 444154
15E0 41
15 1 13FF     332  M3     DATA #>13FF
15E3 505245   333  MESS3  DATA :PRESS CASSETTE STOP:
15 6 535320
15 9 434153
15 C 534554
15 F 544520
15F2 53544F
15F5 50
```

```
15F6 0900      334  M4      DATA #>0900
15F8 524543    335  MESS4   DATA :RECORDING:
   B 4F5244
  FE 494E47
1601 0800      336  M5      DATA #>0800
1603 434845    337  MESS5   DATA :CHECKING:
1606 434B49
1609 4E47
160B 14FF      338  M6      DATA #>14FF
160D 524557    339  MESS6   DATA :REWIND CASSETTE TAPE:
1610 494E44
1613 204341
1616 535345
1619 545445
161C 205441
161F 5045
1621 15FF      340  M7      DATA #>15FF
1623 505245    341  MESS7   DATA :PRESS CASSETTE RECORD:
1626 535320
1629 434153
162C 534554
162F 544520
1632 524543
1635 4F5244
1638 13FF      342  M8      DATA #>13FF
163A 505245    343  MESS8   DATA :PRESS CASSETTE PLAY:
163D 535320
  40 434153
  43 534554
1646 544520
1649 504C41
164C 59
164D 1400      344  M9      DATA #>1400
164F 434845    345  MESS9   DATA :CHECK TAPE (Y OR N)?:
1652 434B20
1655 544150
165  452028
165B 59204F
165E 52204E
1661 293F
1663 1000      346  M10     DATA #>1000
1665 544845    347  MESS10  DATA :THEN PRESS ENTER:
1668 4E2050
166B 524553
166E 532045
1671 4E5445
1674 52
1675 0FFE      34   M11     DATA #>0FFE
1677 505245    349  M SS11  DATA :PRESS R TO READ:
167A 535320
167D 522054
1680 4F2052
  83 454144
  86 0F00      350  M12     DATA #>0F00
1688 505245    351  MESS12  DATA :PRESS E TO EXIT:
```

```
168B 535320
168E 452054
1691 4F2045
1694 584954
1697 0700      352  M13     DATA #>0700
1699 444154    353  MESS13  DATA :DATA OK:
169C 41204F
169F 4B
16A0 1500      354  M14     DATA #>1500
16A2 455252    355  MESS14  DATA :ERROR - NO DATA FOUND:
16A5 4F5220
16A8 2D204E
16AB 4F2044
16AE 415441
16B1 20464F
16B4 554E44
16B7 1000      356  M15     DATA #>1000
16B9 505245    357  MESS15  DATA :PRESS C TO CHECK:
16BC 535320
16BF 432054
16C2 4F2043
16C5 484543
16C8 4B
16C9 1100      358  M16     DATA #>1100
16CB 505245    359  MESS16  DATA :PRESS R TO RECORD:
16CE 535320
16D1 522054
16D4 4F2052
16D7 45434F
16DA 5244
               360          END

 ERRORS=    0

 LENGTH=  972 (>03CC)

 126 SYMBOLS USED
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
			314 315 316 317 319 321
.MP1	0058	32	283 288 313 314 317
TIMER	0079	28	296 298 306 308
TLOAD	13F2	154	75
TOKAY	1408	162	199
TONE1	0034	23	276
TONE2	0036	24	
TOP	0003	1	
TOP01	1391	110	108
TOP10	13B5	126	113
TOP20	13C1	130	122
TOPEN	1387	108	70 74
TREAD	13CF	135	72
TSAVE	1489	230	76
TWRITE	13DA	138	73
TXFER	13DD	139	137
UNIT	0066	43	61 63 114 154 175 260 293
VDP	000C	1	
VEL	0007	1	
XPT	000E	1	86
YESKEY	0059	51	184
YPT	000D	1	180 258

TOTAL NUMBER OF SYMBOLS = 126
TOTAL NUMBER OF REFERENCES = 391

```
                1         TITLE EDIT-HC
                2 ********************************************************
4D00            3 M$EXEC EQU  >4D00         Module EXEC branch table address
2828            4 M$PSCN EQU  >2828         Module PSCAN branch table address
4000            5 M$FLMG EQU  >4000         Module FLMGR branch table address
                6 * Remove from 99/4A u/o equation calculator
                7 *M$COMP EQU  >1158         Start of COMPULATOR
                8 ********************************************************
0080            9 CRUADD EQU  >80
00C0           10 PRTNFN EQU  >C0
               11 ********************************************************
4D00           12 SCROLL EQU  M$EXEC        EQUATES FOR ROUTINES FROM OTHER
4018           13 LIST   EQU  M$FLMG+>18    SECTIONS
4D0C           14 EXEC1  EQU  M$EXEC+>0C
4D10           15 RUN    EQU  M$EXEC+>10
2828           16 PRGLST EQU  M$PSCN+>00
285C           17 KEYTAB EQU  M$PSCN+>34    Keyword table location
283A           18 PTLNE  EQU  M$PSCN+>12
282E           19 LLIST  EQU  M$PSCN+>06
283E           20 SEETWO EQU  M$PSCN+>16    FIND LINE NUMBER IN PROGRAM
2842           21 DISO   EQU  M$PSCN+>1A    DISPLAY LINE NUMBER
2846           22 PROP   EQU  M$PSCN+>1E    CHARACTER PROPERTY
282C           23 GETNB  EQU  M$PSCN+>04
283C           24 GETLN  EQU  M$PSCN+>14
2830           25 GETCHR EQU  M$PSCN+>08
2832           26 READLN EQU  M$PSCN+>0A
284E           27 ERR$$  EQU  M$PSCN+>26
2850           28 STKCHR EQU  M$PSCN+>28
2852           29 PUSHCH EQU  M$PSCN+>2A
2854           30 UNQSTR EQU  M$PSCN+>2C
2856           31 GETNB2 EQU  M$PSCN+>2E
2858           32 READL1 EQU  M$PSCN+>30
2838           33 DELREP EQU  M$PSCN+>10
2840           34 AUTON  EQU  M$PSCN+>18
4014           35 SAVE   EQU  M$FLMG+>14
4016           36 OLD    EQU  M$FLMG+>16
4012           37 CLSALL EQU  M$FLMG+>12    CLOSE ALL OPENED FILES
4D1A           38 LINK1  EQU  M$EXEC+>1A    LINK TO SUBPROGRAMS
               39 ********************************************************
0010           40 CPL    EQU  >10           CALL LINK    FLOATING-POINT PACKA
0012           41 RPL    EQU  >12           RETURN LINK
0014           42 CNS    EQU  >14           NUMBER TO STRING
0018           43 CHAR2$ EQU  >18           CHARACTER TABLE ADDRESS(UPPER CAS
004A           44 CHAR3$ EQU  >4A           CHARACTER TABLE ADDRESS(LOWER CAS
001C           45 PUSH   EQU  >1C           PUSH FLOATING ACC
001E           46 POPS   EQU  >1E           POP FLOATING ACC
0020           47 EXIT   EQU  >20           PROGRAM EXIT
0022           48 GSCAN  EQU  >22           GROM SCAN ROUTINE
0024           49 PWR    EQU  >24           EXPONENTIATION
0026           50 SQR    EQU  >26           SQUARE ROOT
0028           51 EXP    EQU  >28           EXPONENTIAL
002A           52 LOG    EQU  >2A           NATURAL LOG
002C           53 COS    EQU  >2C           COSINE
002E           54 SIN    EQU  >2E           SINE
0030           55 TAN    EQU  >30           TANGENT
```

```
0042     CHAT   EQU  >42
0043     BASE   EQU  >43
0044     BUFFY  EQU  >44
0046     CURLIN EQU  >46                Current line number (AUTONUM)
0048     CURINC EQU  >48                Current increment (AUTONUM)
004A     FAC    EQU  >4A
005C     ARG    EQU  >5C
006E     VSPTR  EQU  >6E
0076     EXP$   EQU  >76
         *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
         *  IT WAS USED FOR CALCULATOR FLAG REMOVED FROM 99/4A
         *CALCFL EQU  >82
         * REPLACE BY ONE BYTE FLAG ALLUP FOR ADDING LOWERCASE CHAR.
         * SET IN 99/4A 3/3/81
0082     ALLUP  EQU  >82
         *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
0088     FLAG   EQU  >88
0088     RSTK   EQU  >88                STARTS AT >8A
0089     GROMFL EQU  >89
         ****************************************************************
02E2     NLNADD EQU  >2E2                    LOCATIONS IN VDP
02FE     ENDSCR EQU  >2FE
0320     CRNBUF EQU  >320                    CRUNCHED INPUT LINE BUFFER
03BE     CRNEND EQU  >3BE                    END OF BUFFER
03EC     SAVEPC EQU  >3EC                    SAVED PROGRAM POINTER 2 BYTES
         *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
         *   REPLACE    VRAMVS EQU >5F8   TO ADD LOWERCASE CHAR. SET
         *   FOR 99/4A BY CAROLYN LEE 3/3/81
         *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
06F8     VRAMVS EQU  >6F8                    DEFAULT BASE OF VALUE STACK
0800     SYMPT  EQU  >800
         ****************************************************************
0002     X1     EQU  >02                IMMEDIATE VALUES
0017     Y1     EQU  >17
0002     DISPY  EQU  >02
000A     UPARR  EQU  >0A
000B     DWNARR EQU  >0B
000D     CHRTN  EQU  >0D
0012     LIMIT  EQU  >12
0020     BKGD   EQU  >20
000B     MAXLEN EQU  >0B
0060     OFFSET EQU  >60
0005     MAXET  EQU  >05
0003     LIST$  EQU  >03
0006     OLD$   EQU  >06
0007     RESEQ$ EQU  >07
000      SAVE$  EQU  >08
0009     EDIT$  EQU  >09
```

```
313  ********************************************************************
314  ********************************************************************
315  *       FLAGS IN TEXT EDITTING:
316  *       @FLAG  BIT  RESET                      SET
317  *                0   AUTONUM OFF               *AUTONUM ON
318  *                1   SYMBOL WAS DECLARED       *SYMBOL HAS NOT BEI
319  *                                              *DECLARED
320  *                2   NO SYMBOLS ON SYMBOL      *THERE ARE SYMBOLS
321  *                    TABLE YET                 *ON SYMBOL TABLE
322  *                4   UNTRACE IS ON             *TRACE IS ON
323  *                5   PRESCAN OR RUN MODE       *EDIT MODE
324  ********************************************************************
325  ********************************************************************
```

```
                    344  ********************************************************************
                    345  **      PROGRAM STARTS HERE.
                    346  ********************************************************************
2010 4417           347          BR    GROMBS.
2012 4195           348          BR    MAO
2014 460B           349          BR    SCDAT$
2016 466C           350          BR    SCANER
2018 467E           351          BR    ILL1                WAS SYNERR
201A 4192           352          BR    MAIN1
201C 47F1           353          BR    CHRTAB
201E 436D           354          BR    MVDN
2020 46AB           355          BR    GETLNB
```

```
                    357  *
                    358  *       MESSAGES
                    359  *
                    360          BASE 0,0,>300,>300,0,0,>60
2022 8A80B7         361  MSG0    DATA :* WARNING::
2025 A1B2AE
2028 A9AEA7
202B 9A
202C 13A9AE         362  MSG1    DATA 19,:INCORRECT STATEMENT:
202F A3AFB2
2032 B2A5A3
2035 B480B3
203  B4A1B4
203B A5ADA5
203E AEB4
2040 08A2A1         363  MSG4    DATA 8,:BAD NAME:
2043 A480AE
2046 A1ADA5
2049 0BADA5         364  MSG5    DATA 11,:MEMORY FULL:
204C ADAFB2
204F B980A6
2052 B5ACAC
2055 0EA3A1         365  MSG6    DATA 14,:CAN'T CONTINUE:
2058 AE87B4
205B 80A3AF
205E AEB4A9
2061 AEB5A5
                    366  MSG3
                    367  MSG14
2064 09A2A1         368  MSG7    DATA 9,:BAD VALUE:
2067 A480B6
206A A1ACB5
206D A5
206E 0EAEB5         369  MSG8    DATA 14,:NUMBER TOO SIG:
2071 ADA2A5
2074 B280B4
2077 AFAF80
207A A2A9A7
207D 16B3B4         370  MSG11   DATA 22,:STRING-NUMBER MISMATCH:
20 0 B2A9AE
2083 A78DAE
2086 B5ADA2
2089 A5B280
208C ADA9B3
20 F ADA1B4
2092 A3A8
2094 0CA2A1         371  MSG12   DATA 12,:BAD ARGUMENT:
2097 A480A1
209A B2A7B5
209D ADA5AE
20A0 B4
20A1 0DA2A1         372  MSG13   DATA 13,:BAD SUBSCRIPT:
20A4 A480B3
20A7 B5A2B3
20AA A3B2A9
```

```
      089C 83EF
1167  089E D7C7          MOVB R7,*R15       STORE ADDRESS
1168  08A0 61CA          S    R10,R7
1169  08A2 045B  GETRTN  RT
1170  08A4 0207  SETV    LI   R7,WRVDP      WRITE TO VDP MEMORY
      08A6 4000
1171  08A8 10F8          JMP  SETVDP
1172               *          MULTICOLOR MODE ADDRESS COMPUTATION
1173  08AA D020  MCMDA   MOVB @YPT,R0       LOAD YPT
      08AC 837E
1174  08AE C200          MOV  R0,R8
1175  08B0 0A58          SLA  R8,5          GET 3 LOW BITS AS LSB
1176  08B2 09D8          SRL  R8,13
1177  08B4 09B0          SRL  R0,11         POSITION 5 HIGH BITS
1178  08B6 0A80          SLA  R0,8
1179  08B8 A008          A    R8,R0
1180  08BA C207          MOV  R7,R8         SAVE XPT
1181  08BC 0247          ANDI R7,>3E00      POSITION XPT BITS
      08BE 3E00
1182  08C0 0967          SRL  R7,6
1183  08C2 A1C0          A    R0,R7         ADD X VALUE TO DISPLAY ADDRESS
1184  08C4 0227          AI   R7,>800
      08C6 0800
1185  08C8 D7E0          MOVB @R7LB,*R15    LOAD VDP ADDRESS
      08CA 83EF
1186  08CC 0A88          SLA  R8,8          SET CARRY TO LSD OF XPT
1187  08CE D7C7          MOVB R7,*R15
1188  08D0 028B          CI   R11,RTN       STORE OR RETRIEVE CB ?
      08D2 026A'
1189  08D4 16E6          JNE  GETRTN        STORE
1190  08D6 D02F          MOVB @VDPRD(R15),R0 LOAD BYTE FROM VDP
      08D8 FBFE
1191  08DA D220          MOVB @CHRBUF,R8    LOAD CB
      08DC 837D
1192  08DE 0248          ANDI R8,>0F00      MASK OUT LOW DIGIT
      08E0 0F00
1193  08E2 1804          JOC  USTRCB        JMP IF CB IS LOW DIGIT
1194  08E4 0240          ANDI R0,>0F00      MASK OUT LOW DIGIT
      08E6 0F00
1195  08E8 0A48          SLA  R8,4          MOVE CB TO HIGH DIGIT
1196  08EA 1002          JMP  VSTRCB
1197  08EC 0240  USTRCB  ANDI R0,>F000      MASK OUT HIGH DIGIT
      08EE F000
1198  08F0 0267  VSTRCB  ORI  R7,WRVDP      SET BIT TO WRITE TO VDP
      08F2 4000
1199  08F4 06A0          BL   @SETVDP       LOAD VDP ADDRESS
      08F6 089A'
1200  08F8 A008          A    R ,R0         COMBINE DIGITS
1201  0 FA DBC0          MOVB R0,@VDPWD(R15) STORE BYTE IN VDP
      08FC FFFE
1202  08FE 0456          B    *R6                  RETURN TO NEXT
1203               *=============================================================*
1204               *      HAND HELD UNIT DSR
1205               *      THERE IS A SEPARATE VERSION OF THE INTERPRETER FOR
1206               *      THE TI VERSION AND THE MILTON BRADLEY VERSION OF THE
1207               *      HAND HELD UNITS.
1208               *      THE MB VERSION OF THE INTERPRETER IS CALLED :
1209               *           HC2.ALBERT.SRC.MBINTR
1210               *
```

```
0507                    *       DIVIDE LOOP
0508                    *
0509                    *       U(J) IS THE HIGHEST ORDER BYTE OF WHAT IS LEFT OF
0510                    *       THE DIVIDEND.  EACH QUOTIENT DIGIT IS ESTIMATED A
0511                    *       FOLLOWS:
0512                    *         IF U(J) = V1  THEN Q := 99
0513                    *            ELSE Q := INT((100*U(J)+U(J+1))/V1)
0514                    *         IF V2*Q GT (100*U(J)+U(J+1)-Q*V1)*100+U(J+2)
0515                    *            THEN Q := Q - 1 AND THE TEST IS REPEATED.
0516                    *       THIS WILL ENSURE THAT Q-1 LE NEXT-QUOTIENT-DIGIT
0517                    *       LE Q.
0518                    *       NOT  THAT 100*U(J)+U(J+1)-Q*V1 =
0519                    *       REMAINDER((100*U(J)+U(J+1))/V1)
0520                    *       Q*V IS THEN SUBTRACTED FROM U.
0521                    *       IF THE RESULT IS NEGATIVE, V IS ADDED BACK IN AND
0522                    *       Q := Q-1  (THE PROBABILITY OF ADDING BACK IS APPROX
0523                    *       .03)
0524                    *
0525                    *       R0-R1   TEMPORARY
0526                    *       R2      NEXT QUOTIENT DIGIT
0527                    *       R3-R4   TEMPORARY
0528                    *       R5      QUOTIENT BYTE LOOP COUNT
0529                    *       R6      NUMBER OF SIGNIFICANT BYTES IN DIVISOR
0530                    *       R7      V1
0531                    *       R8      V2
0532                    *       R9      100*V1+V2
0533                    *       R11     POINTER INTO DIVIDEND (USUALLY POINTS
0534                    *               TO U(J))
0535                    *
0536 0376 0206  FDIV06 LI    R6,3           NUMBER DIVISOR BYTES + 1
     0378 0008
0537 037A 0606  FDIV07 DEC   R6             COMPUTE NUMBER OF SIG BYTES
0538                    *                   IN DIVISOR
0539 037C D026         MOVB  @FDVSR(R6),R0  GET NEXT HIGHER ORDER BYTE
     037E 8354
0540                    *                    OF DIVISOR
0541 0380 13FC         JEQ   FDIV07         IGNORE IT IF IT IS ZERO
0542 0382 04C7         CLR   R7             CLR HIGH BYTE OF WHERE V1 WILL BE
0543 0384 D820         MOVB  @FDVSR+1,@WS+15 RB(R7) IS V1
     0386 8355
     038  83EF
0544 038A C207         MOV   R7,R8          COPY V1 TO COMPUTE 100*V1
0545 038C 3A20         MPY   @LW100,R8      COMPUTE 100*V1
     038E 032A'
0546 0390 D820         MOVB  @FDVSR+2,@WS+17 GET V2 (HIGH BYTE IS ZERO)
     0392 8356
     0394 83F1
0547 0396 A24          A     R8,R9          COMPUTE 100*V1+V2
0548 0398 0205         LI    R5,-9          COMPUTE 9 BYTES OF QUOTIENT
     039A FFF7
0549 039C 020B         LI    R11,ARG        PTR TO HIGH BYTE OF DIVIDEND
     039E 835C
```

```
FLTERR      836C    A0019
FLTNDX      8354    A0017
FLTTAB  D   0000'    0023  0007
FMCLR       01A4'    0216  0218
FMEND       01FE'    0269  0629
FMUL02      01B0'    0233  0236
FMUL03      01BA'    0238  0240
FMUL04      01D0'    0247  0268
FMUL05      01DA'    0252  0264
FMULT   D   016E'    0196  0008  0024
FMULT1      017 '    0200  0197
FMULZR      0210'    0294  0203  0206  0426
FSUB    D   0062'    0080  0008  0024
FZERO   D   0210'    0295  0010  0384
GETCH       0020+   A0048
GETCH1      002E+   A0053 A0060
GETCHG      0032+   A0056
GRMRA   R           A0035
GRMWA   R   003A+   A0035 A0056 A0057
HIBYTE      0100     0018  0129  0130  0327  0331  0332
I           0004    A0005
J           0005    A0006 A0037 A0042 A0044
JOYX        8377    A0027
JOYY        8376    A0021
K           0006    A0007 A003  A0043
L           0007    A000  A0040 A0043
LB1         013F'    0162  0621
LB100       032B'    0472  0601  0618  0620
LLB         83EF    A0031
LW100       032A'    0471  0545  0554  0559  0564  0596
N           000     A0009
NORM01      0206'    0282  0285
NORM02      021A'    029   0283
NORM03      022A'    0303  0306
NORM04      023 '    030   0310
NORMAL      0202'    02 0  0184
OFERR       0100     0017  03 7
OV          02 2'    03 6  0023
OV1         02B4'    03 7  0385
OVEXP       02A '    03 2  0023
OVEXP1  D   02AA'    03 3  0010  0353
PACK01      028 '    0359  0357
PACKUP  D   026C'    0351  0007  01 5  0329  0335
PAD         8300    A0004 A0015 A0019 A0020 A0021 A0022 A0025 A0026 A0027 A0028
                          A0029 A0030 A0031  0014  0015
POPSTK      0000+   A0037  0046  0078  00 5  0199  0417
PROA        8310    A0022
R0          0000           0106  010   012   0133  0134  0235  0239  0241  0242
                           0253  0257  029   0299  0301  0309  0327  0328  0332
                           0334  0336  0390  0393  0395  0396  0397  039   0399
                           0420  0421  0425  0427  0488  0490  0491  0493  0495
                           0539  0555  0557  0561  0562  0563  0592  0594  0596
                           0602
R1          0001           0090  0091  0091  0092  0136  0157  0160  0163  0163
                           0164  0256  0280  0282  0284  0298  0303  0305  0308
                           0330  0333  0334  0336  0337  0362  0369  0370  0494
                           0567  0569  0571  0595
R10         000A           0033  0045  0077  0079  0081  0084  0196  0198  0326
                           0361  0365  0368  0382  0386  0414  0416
R11         000B           0033  0039  0045  0077  0079  0081  0084  0196  0198
```

\		03C0	A0023								
TR		936E	A0020	A0039	A0041	A0042					
)P	R		A0035								
.		83E0	A0030	0118	0122	0244	0250	0254	0259	0354	0468	0470
				0475	0543	0546	0553	0556	0560	0593	0598	0607
				0625								

SOURCE ACCESS NAME= BASC1.TI994A.V080581.SRC.CSN
OBJECT ACCESS NAME= PCD2.AEM.OBJ4A.CSN
LISTING ACCESS NAME= PCD2.AEM.LST4A.CSN
ERROR ACCESS NAME= PCD2.AEM.SRC.ERROR
OPTIONS= XREF
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
0012	A	FPEQ

=>BASC1.TI994A.V080581.SRC.FPEQ

```
0106 0030 10F4          JMP   CSI02           CONTINUE WITH STRING
0107                    ***************************************************
0108 0032               CSN10
0109 0032 0460  CSNZER  B     @FZERO
     0034 0000
0110                    *
0111 0036 045A  CSNZ10  B     *R10            RETURN
0112                    *
0113                    *
0114 0038               CSNOFL
0115 0038 0606          DEC   TP
0116 003A C806          MOV   TP,@FAC+12
     003C 8356
0117 003E 808C          C     TPNZ,TP1            IS MANTISSA ZERO ?
0118 0040 13F8          JEQ   CSNZER              YES, NUMBER IS ZERO
0119 0042 C801          MOV   R1,@EXP             SET EXPONENT SIGN
     0044 8376
0120 0046 0460          B     @OVEXP1          GO TO ERROR ROUTINE
     0048 0000
0121                    ***************************************************
0122                    *
0123                    *       CONVERT STRING TO NUMBER
0124                    *
0125                    *             GROM ENTRY
0126                    *
0127 004A DCE0  CSNGR   MOVB  @GROMFL,R3      TEST FOR GROM
     004C 8389
0128 004E 1303          JEQ   CSN
0129 0050 0203          LI    R3,GETCHG       ROUTINE TO GET GROM CHARACTER
     0052 0032+
0130 0054 1002          JMP   CSN01
0131                    *
0132 0056 0203  CSN     LI    R3,GETCH        ADDRESS OF GETCH ROUTINE
     0058 0020+
0133 005A C28B  CSN01   MOV   R11,R10         SAVE RETURN ADDRESS
0134 005C C1A0          MOV   @FAC+12,TP     INITIALIZE TEXT POINTER
     005E 8356
0135 0060 0693          BL    *R3              GET CHARACTER
0136 0062 04C7          CLR   R7              SIGN
0137 0064 C086          MOV   TP,TP1         SAVE REQUIRED DIGIT POINTER
0138 0066 0288          CI    CC,XPLUS         IS CHARACTER A PLUS
     0068 002B
0139 006A 1304          JEQ   CSN02            YES, IGNORE
0140 006C 0288          CI    CC,XMINUS        IS CHARACTER A MINUS SIGN
     006E 002D
0141 0070 1603          JNE   CSN04            NO MINUS
0142 0072 0707          SETO  R7               NEGATIVE
0143 0074 0582  CSN02   INC   TP1
0144      0076' CSN03   EQU   $                GET NEXT CHARACTER
0145 0076 0693          BL    *R3              BUMP TO NEXT CHARACTER
0146 0078               CSN04
0147 0078 0288          CI    CC,X3000
     007A 0030
0148 007C 13FC          JEQ   CSN03            IGNORE LEADING ZEROES
0149 007E D807          MOVB  R7,@SIGN         SAVE SIGN
     0080 8375
0150 0082 C306          MOV   TP,TPNZ
0151 0084 060C          DEC   TPNZ            FIRST NONZERO CHARACTER
0152 0086 0707          SETO  R7               INITIALIZE RELATIVE POSITION
0153 0088 1002          JMP   CSN06            FIND END OF INTEGER PART
```

```
$          01EC'          0088  0096  0099  0144  0157  0162  0271  0318  0328
A2         0100    0033
ARG        835C    A0018 A0038
ASSGNV R   0152'   0005  0256
C10        0022'   0099  0089  0244
CC         0008    A0024 A0052 A0053 A0059  0098  0100  0138  0140  0147  0158
                           0160  0163  0178  0187  0189  0200  0204  0207  0238
                           0241
CFI     D  0160'   0271  0011  0256
CFI01      0198'   0291  0285
CFI02      01AA'   0298  0284
CFI03      01B0'   0300  0282
CFI04      01BC'   0305  0308
CFI05      01CA'   0312  0302  0304  0306
CFI06      01CC'   0313  0301  0309
CFI08      01D8'   0318  0287  0295  0297  0315
CFIRI      01E6'   0327  0281  0325
CFIRS2     01E4'   0326  0317
CFIRSI     01E0'   0324  0314
CFISI1     01EA'   0328  0273
CSI01      000A'   0088  0101
CSI02      001A'   0096  0087  0106
CSI05      002C'   0105  0091  0095
CSINT      0002'   0084  0213
CSN     D  0056'   0132  0010  0128  0256
CSN01      005A'   0133  0130
CSN02      0074'   0143  0139
CSN03      0076'   0144  0148
CSN04      0078'   0146  0141
CSN05      008A'   0155  0161
CSN06      008E'   0157  0153
CSN07      009A'   0162  0159
CSN10      0032'   0108  0192
CSNF01     00A0'   0170
CSNF02     00A8'   0175  0179
CSNF03     00AA'   0176  0173
CSNF04     00B6'   0185  0174  0190
CSNF05     00C4'   0191  0188
CSNG       00C8'   0196  0164
CSNG02     00E8'   0212  0208
CSNG03     00EA'   0213  0205  0210
CSNGR   D  004A'   0127  0010  0256
CSNH       00F4'   0220  0201  0215
CSNH01     011A'   0234  0239  0246  0251
CSNH02     0138'   0247  0243
CSNH03     013C'   0248  0235
CSNH04     0144'   0252
CSNOFL     0038'   0114  0105
CSNZ10     0036'   0111  0103
CSNZ R     0032'   0109  0118  0223
DATA1      0000+   A0001
ERRI       0001'   0026  0319
ERRIOV     0003    0322  0026
EXP        8376    A0028 0119  0229
FAC        834A    A0015 A0016 A0017 A0018  0116  0134  0221  0232  0272  02
                           0277  0279  0307  0319  0327
FADD    R          0004
FBSYMB R   0154'   0005  0256
FDVSR      8354    A0016
FLTERR     836C    A0019
```

```
0001                ********** MOD'D FOR MACROS 8/4/78 **********
0002                **********MOD'D FOR MACROS(MACD) 3/5/81 FOR 99/4A ******
0003                       IDT  'PARSE3S'
0004                *
0005                *      BASIC PARSE CODE
0006                *
0007                *
0008                *      REGISTER USAGE
0009                *           RESERVED FOR GPL INTERPRETER
0010                *              R13, R14, R15
0011                *              R13 CONTAINS THE READ ADDRESS FOR GROM
0012                *              R14 IS USED IN BASSUP/10 FOR THE VDPRAM POIN
0013                *           RESERVED IN BASIC SUPPORT
0014                *              R8  MSBy CURRENT CHARACTER (LIKE CHAT IN GPL
0015                *              R8  LSBy ZERO
0016                *              R9  STACK POINTER PAD+@STKADD
0017                *              R10 READ DATA PORT ADDRESS FOR PROGRAM DATA.
0018                *      ALL EXITS TO GPL MUST GO THROUGH "NUDGO5"
0019                *
0020                *
0021                       COPY FPEQ
A0001 0000         DATA1  CSEG
A0002              ********************************************************
A0003              *  LOCATIONS USED BY THE INTERPRETER IN HOME COMPUTER
A0004       8300   PAD    EQU  >8300
A0005       0004   I      EQU  4
A0006       0005   J      EQU  5
A0007       0006   K      EQU  6
A0008       0007   L      EQU  7
A0009       0008   N      EQU  8
A0010              ********************************************************
A0011              *      PROCESSOR RAM FIXED LOCATIONS
A0012              *
A0013       8800   VDPRD  EQU  >8800        ADDRESS TO READ VDP
A0014       8C00   VDPWD  EQU  >8C00        ADDRESS TO WRITE VDP
A0015       834A   FAC    EQU  PAD+>4A      FLOATING ACCUMULATOR
A0016       8354   FDVSR  EQU  FAC+10       DIVISOR STORAGE DURING DIVISION
A0017       8354   FLTNDX EQU  FAC+10       INSTRUCTION INDEX SAVE
A0018       835C   ARG    EQU  FAC+18       FLOATING ARGUMENT
A0019       836C   FLTERR EQU  PAD+>6C      ERROR ADDRESS FOR MATH ROUTINES
A0020       836E   VSPTR  EQU  PAD+>6E      VALUE STACK POINTER
A0021       8376   JOYY   EQU  PAD+>76
A0022       8310   PROA   EQU  PAD+>10      PROCESSOR ROLLOUT AREA
A0023       03C0   VROA   EQU  >3C0         VIDEO RAM ROLLOUT AREA
A0024       0008   CC     EQU  R8
A0025       8373   STKADD EQU  PAD+>73
A0026       8375   SIGN   EQU  PAD+>75      TEMPORARY SIGN STORAGE
A0027       8377   JOYX   EQU  PAD+>77
A0028       8376   EXP    EQU  PAD+>76      TEMPORARY EXPONENT STORAGE
A0029       837C   STATUS EQU  PAD+>7C      STATUS REGISTER
A0030       83E0   WS     EQU  PAD+>E0      WORKSPACE MNEMONIC
A0031       83EF   LLB    EQU  PAD+>EF      LOWER HALF OF L
A0032              ********************************************************
A0033              *      SUBROUTINE TO POP VALUE STACK
A0034              *
A0035                      REF  GRMWA,GRMRA,WRVDP
A0036              *
A0037 0000 0205    POPSTK LI   J,-8          COUNTER FOR LOOP
      0002 FFF8
A0038 0004 0206           LI   K,ARG         ADDRESS TO STORE OPERAND
```

```
 0157                    *
 0158                    *       EXECUTE ONE OR MORE LINES OF BASIC
 0159                    *
 0160 00A2               EXECG
 0161 00A2 06A0                  BL    @SETREG
      00A4 05B4'
 0162 00A6 04E0                  CLR   @ERRCOD
      00A8 8322
 0163 00AA C020                  MOV   @BUFFY,R0           IMPERATIVE ?
      00AC 8344
 0164 00AE 1317                  JEQ   EXEC15              YES
 0165                    *
 0166                    *       LOOP FOR EACH STATEMENT
 0167                    *
 0168      00B0' EXEC10  EQU   $
 0169 00B0             MACD
*0001                  REF   C1000
*0002      00B3' CB3    EQU   $+3             CONSTANT BYTE 3
*0003 00B0 0300               LIMI  3                   LET INTERRUPTS LOOSE
      00B2 0003
*0004      00B6' C0     EQU   $+2             CONSTANT DATA 0
*0005 00B4 0300               LIMI  0
      00B6 0000
*0006 00B8 04E0               CLR   @>83D6              RESET VDP TIMEOUT
      00BA 83D6
*0007 00BC 06A0               BL    @BRKKEY
      00BE 0020
*0008 00C0 134F               JEQ   BRKPN1              IF SO TAKE BASIC BREAKPOINT
 0170 00C2 D020   EXEC12 MOVB  @FLAG,R0                  TEST TRACE FLAG
      00C4 8388
 0171 00C6 0A30               SLA   R0,3
 0172 00C8 1160               JLT   TRACE
 0173 00CA C820   EXEC11 MOV   @EXTRAM,@PGMPTR           GET LINE POINTER
      00CC 832E
      00CE 832C
 0174 00D0 06A0               BL    @PGMCHR
      00D2 06B8'
 0175 00D4 1142               JLT   BRKPNT
 0176 00D6 D808   EXEC14 MOVB  R8,@PGMPTR
      00D8 832C
 0177 00DA D81A               MOVB  *R10,@PGMPTR+1      GET SECOND BYTE OF POINTER
      00DC 832D
 0178 00DE 05C9   EXEC15 INCT  R9                        SAVE A RETURN ADDRESS
 0179 00E0 C660               MOV   @EXRTNA,*R9
      00E2 0000'
 0180 00E4 06A0               BL    @PGMCHR             GET FIRST CHARACTER.
      00E6 06B8'
 0181 00E8 1102               JLT   EXEC20              TOKEN
 0182 00EA 0460               B     @NL T               NO, FAKE "LET"
      00EC 0324'
 0183                    *
 0184 00EE C1C8   EXEC20 MOV   R8,R7                     SAVE FIRST TOKEN
 0185 00F0 06A0               BL    @PGMNXT             GET SECOND TOKEN
      00F2 06DA'
 0186 00F4 0977               SRL   R7,7                GET TABLE OFFSET
 0187 00F6 0227               AI    R7,->A2*2           NEGATIVE OR ZERO VALUES ONLY
      00F8 FE8C
 0188 00FA 1535               JGT   ERR1
 0189 00FC C1E7               MOV   @STMTTB(R7),R7      GET BRANCH ADDRESS
      00FE 041A'
```

```
0222                *     Subroutine called from SMB in 'BASSUP3S' for checking
0223                *     if the UDF bit is set.  Because there is no room in
0224                *     'BASSUP3S' and a few routines in 'BASSUP3S' used by
0225                *     Extended BASIC by absolute address scheme, the
0226                *     addresses in 'BASSUP3S' can not be modified.
0227                *     This routine is patched outside of 'BASSUP3S'.
0228                *
0229                *     This routine does the UDF checking in SMB instead of
0230                *     ASSG.  In ASSG : When in string case, pointer in
0231                *     ARG points to the value space, not the first byte,
0232                *     of the symbol table entry.  Therefore the general
0233                *     UDF check for both numeric and string case is changed
0234                *     to put in the beginning of SMB.
0235                *
0236                          DEF  PATCH1
0237                *
0238                *
0239 0130 C08B     PATCH1       MOV  R11,R2        SAVE THE RETURN ADDRESS
0240 0132 06A0               BL   @GETV         GET THE 1ST BYTE OF THE SYMBO
     0134 0000
0241                *                            TABLE ENTRY
0242 0136 834A               DATA FAC
0243 0138 C101               MOV  R1,R4         COPY FOR LATER USE
0244 013A 0A21               SLA  R1,2          CHECK FOR UDF ENTRY
0245 013C 1814               JOC  ERR1          IF UDF = THEN ERROR
0246 013E C044               MOV  R4,R1         RETURNS STATUS OF CHARACTER :
0247                *                            STRING OR NUMERIC VALUE CHECK
0248 0140 0452               B *R2              RETURN
0249                *
0250                * NOW MAINTAIN ADDRESSES IN THIS MODULE
0251 0142                    BSS 24
```

```
0319                    *
0320                    *       NUD ROUTINE FOR NUMERIC CONSTANT
0321                    *
0322 0196               NUMCON
0323 0196 C820                  MOV   @PGMPTR,@FAC+12   POINTER FOR CSN
     0198 832C
     019A 8356
0324 019C 06C8                  SWPB  R8
0325 019E A808                  A     R8,@PGMPTR
     01A0 832C
0326 01A2 04E0                  CLR   @FAC+10           ERROR INDICATOR
     01A4 8354
0327 01A6 06A0                  BL    @SAVRE2           SAVE REGISTERS
     01A8 05CA'
0328 01AA 06A0                  BL    @CSNGR            CONVERT STRING TO NUMBER
     01AC 0000
0329 01AE 06A0                  BL    @SETREG           SET UP REGISTERS
     01B0 05B4'
0330 01B2 8820                  C     @FAC+12,@PGMPTR   IS POINTER AS EXPECTED
     01B4 8356
     01B6 832C
0331 01B8 16D6                  JNE   ERR1              NO, SYNTAX ERROR
0332 01BA 06A0                  BL    @PGMCHR           GET NEXT CHAR. FROM PROGRAM
     01BC 06B8'
0333 01BE D020                  MOVB  @FAC+10,R0        OVERFLOW ?
     01C0 8354
0334 01C2 16DB                  JNE   WARN$$            YES, HAVE CPL ISSUE WARNING
0335 01C4 0460  NUMC50          B     @CONT             CONTINUE PARSE
     01C6 005E'
0336                    *
```

```
0778                    *
0779 0552               MINUS
0780 0552 06A0                  BL    @PSHPRS
     0554 05D6'
0781 0556   C2                  BYTE MINUS$,0
     0557   00
0782 0558 0202                  LI    R2,SSUB
     055A 0000
0783 055C 10E9                  JMP   LEDEX
0784                    *
0785                    *
0786 055E               TIMES
0787 055E 06A0                  BL    @PSHPRS
     0560 05D6'
0788 0562   C4                  BYTE DIVI$,0
     0563   00
0789 0564 0202                  LI    R2,SMULT
     0566 0000
0790 0568 10E3                  JMP   LEDEX
0791                    *
0792                    *
0793 056A               DIVIDE
0794 056A 06A0                  BL    @PSHPRS
     056C 05D6'
0795 056E   C4                  BYTE DIVI$,0
     056F   00
0796 0570 0202                  LI    R2,SDIV
     0572 0000
0797 0574 10DD                  JMP   LEDEX
0798                    *
0799                    *
0800 0576               EXPON
0801 0576 06A0                  BL    @PSHPRS
     0578 05D6'
0802 057A   C5                  BYTE EXPON$,0
     057B   00
0803 057C 0200                  LI    R0,LEXPON         RETURN VECTOR FOR EXPON NUD
     057E 0005
0804 0580 0460                  B     @EXIT             RETURN TO GPL
     0582 016A'
0805                    *
```

```
                454  ************************************************************
                455  *       DEFAULT TABLES AND ASCII CHARACTER SET
                456  *
044F 80         458  H80     DATA >80
                459  * The following line has been changed 8/07/81 to reset
                460  * VDP register 0 at powerup.
                461  VDPRG$
                462  *       DATA >60,>20,>F0,>0E,>F9,>86,>F8,>F7
0450 600020     463          DATA >60,>00,>20,>F0,>0E,>F9,>86,>F8,>F7
                464  TABLE$
0459 171717     465          DATA >17,>17,>17,>17
045D 171717     466          DATA >17,>17,>17,>17
0461 171717     467          DATA >17,>17,>17,>17
0465 060301     468          DATA >06,>03,>01,>0B
0469 0C0D0F     469          DATA >0C,>0D,>0F,>04
046D 020D08     470          DATA >02,>0D,>08,>0E
0471 05090A     471          DATA >05,>09,>0A,>06
0475 272722     472          DATA >27,>27,>22,>22
                473  BEEP$
0479 06BFDF     474          DATA 6,>BF,>DF,>FF,>80,5,>92,10,1,>9F,0
                475  GONG$
0484 06BFDF     476          DATA 6,>BF,>DF,>FF,>80,>20,>90,10,1,>9F,0
048F 0A3139     477  TI1979  DATA 10,:1981   :
0496 544558     478  TI      DATA :TEXAS INSTRUMENTS:
04A7 484F4D     479  HC      DATA :HOME COMPUTER:
                480  CHAR1$
04B4 000000     481          DATA #>0000,#>0000,#>0000,#>0000     BLANK
04BC 202020     482          DATA #>2020,#>2020,#>2020,#>0020     !
04C4 484848     483          DATA #>4848,#>4800,#>0000,#>0000     "
04CC 0048FC     484          DATA #>0048,#>FC48,#>48FC,#>4800     #
04D4 103C50     485          DATA #>103C,#>5038,#>1478,#>1000     $
04DC C0C408     486          DATA #>C0C4,#>0810,#>2040,#>8C0C     PERCENT
04E4 609090     487          DATA #>6090,#>9060,#>6094,#>8874     &
04EC 0 1020     488          DATA #>0810,#>2000,#>0000,#>0000     '
04F4 081020     489          DATA #>0810,#>2020,#>2020,#>1008     (
04FC 402010     490          DATA #>4020,#>1010,#>1010,#>2040     )
0504 000048     491          DATA #>0000,#>4830,#>CC30,#>4800     *
050C 000010     492          DATA #>0000,#>1010,#>7C10,#>1000     +
0514 000000     493          DATA #>0000,#>0000,#>0030,#>1020     ,
051C 000000     494          DATA #>0000,#>0000,#>7C00,#>0000     -
0524 000000     495          DATA #>0000,#>0000,#>0000,#>3030     .
052C 000408     496          DATA #>0004,#>0810,#>2040,#>8000     /
0534 384444     497          DATA #>3844,#>4444,#>4444,#>4438     0
053C 103050     498          DATA #>1030,#>5010,#>1010,#>107C     1
0544 788404     499          DATA #>7884,#>0408,#>1020,#>40FC     2
054C 788404     500          DATA #>78 4,#>0438,#>0404,#>8478     3
0554 0C1424     501          DATA #>0C14,#>2444,#>84FC,#>0404     4
055C F88080     502          DATA #>F880,#>80F8,#>0404,#>8478     5
0564 788080     503          DATA #>7880,#>80F8,#>8484,#>8478     6
056C FC0404     504          DATA #>FC04,#>0408,#>1020,#>4040     7
0574 788484     505          DATA #>7884,#>8478,#>8484,#>8478     8
057C 788484     506          DATA #>7884,#>8484,#>7C04,#>0478     9
0584 003030     507          DATA #>0030,#>3000,#>0030,#>3000     :
058C 003030     508          DATA #>0030,#>3000,#>0030,#>1020     ;
0594 000810     509          DATA #>0008,#>1020,#>4020,#>1008     <
```

```
 8B3 000038    620           DATA 000,000,03 ,010,010,010,038      i
08BA 000008    621           DATA 000,000,008,008,008,048,030      J
08C1 000024    622           DATA 000,000,024,028,030,028,024      k
08C8 000040    623           DATA 000,000,040,040,040,040,07C      l
08CF 000044    624           DATA 000,000,044,06C,054,044,044      m
08D6 000044    625           DATA 000,000,044,064,054,04C,044      n
08DD 00007C    626           DATA 000,000,07C,044,044,044,07C      o
08E4 000078    627           DATA 000,000,078,044,078,040,040      p
08EB 00003     628           DATA 000,000,03 ,044,054,048,034      q
08F2 000078    629           DATA 000,000,078,044,078,048,044      r
08F9 00003C    630           DATA 000,000,03C,040,038,004,078      s
0900 00007C    631           DATA 000,000,07C,010,010,010,010      t
0907 000044    632           DATA 000,000,044,044,044,044,038      u
090E 000044    633           DATA 000,000,044,044,028,028,010      v
0915 000044    634           DATA 000,000,044,044,054,054,028      w
091C 000044    635           DATA 000,000,044,028,010,028,044      x
0923 000044    636           DATA 000,000,044,028,010,010,010      y
092A 00007C    637           DATA 000,000,07C,008,010,020,07C      z
0931 182020    638           DATA 018,020,020,040,020,020,018      {
0938 101010    639           DATA 010,010,010,000,010,010,010      |
093F 300808    640           DATA 030,008,008,004,008,008,030      }
0946 002054    641           DATA 000,020,054,008,000,000,000      ~
               642 FOR$
094D 464F52    643           DATA :FOR:
               644 TIBUG$
0950 010303    645           DATA 1,3,3,3,3,3,3,3
0958 FC0405    646           DATA >FC,4,5,5,4,6,2,12
0960 008040    647           DATA 0,>80,>40,>40,>80,>0,12,>12
0968 FF80C0    648           DATA >FF,>80,>C0,>40,>60,>38,>1C,14
0970 192121    649           DATA >19,>21,>21,>3D,5,5,5,>C4
0978 BA8A8A    650           DATA >BA,>8A,>8A,>BA,>A1,>A1,>A1,>22
0980 030100    651           DATA 3,1,0,0,0,0,0,0
0988 E23110    652           DATA >E2,>31,>10,>18,12,7,3,0
0990 4C9020    653           DATA >4C,>90,>20,>40,>40,>20,>E0,0
0998 3C4299    654           DATA >3C,>42,>99,>A1,>A1,>99,>42,>3C
```

```
                729  *--------------- CONVERT NUMBER TO STRING ----------------
                730  *
                731  *       Work locations in processor roll-out area
                732  *       NOTE: VPTR$ and CH$ must be on even byte
                733  *       boundaries!
                734  *
0010            735  VPTR$   EQU  PROA$         Pointer into FAC, work register
0011            736  FLG$    EQU  PROA$+1       General work register
0012            737  MSD$CH  EQU  PROA$+2       Maximum # of significant digits
0013            738  FP$     EQU  PROA$+3
0014            739  OE$     EQU  PROA$+4       Odd/even indicator, work reg
0015            740  DP$     EQU  PROA$+5       Position of decimal point
0016            741  SPTR$   EQU  PROA$+6       ^ to ASCII string (result)
0017            742  SPSV$   EQU  PROA$+7       Save for above ^^^^^^^^^
0018            743  TYP$    EQU  PROA$+8       Type indicator
001A            744  CNSIT   EQU  PROA$+10      10-byte work area. ARG+6 or below
                745  *
                746  ********************************************************
                747  *
                748  *       Variable assignments within floating point area
                749  *
0055            750  CW$     EQU  FAC+11        Calculator width (0=free format)
0056            751  CE$     EQU  FAC+12        # sign. dgts for E-format - calc.
0057            752  FIX$    EQU  FAC+13        Fix length in calc mode (-=nofix)
```

```
 0A0A 6A1E
 0A0C D277FC   797        $IF @EXP$+1 .GE. -4 Nor is it too small.....
 0A0F 4A1E
 0A11 BE1209   798          ST   9,@MSD$CH   Could be fixed point rounding
 0A14 CE77FE   799          $IF @EXP$+1 .LE. -2 THEN Not maximum significance
 0A17 6A1E
 0A19 9012     800            INC  @MSD$CH   Round for max significant digits
 0A1B A01277   801            ADD  @EXP$+1,@MSD$CH
               802          $ ND IF
               803        $END IF
               804      $END IF
 0A1E 060B73   805      CALL CNSRND          Round number according to MSD$CH
               806      $
               807      $  Note : rounding can change exponent and print forma
               808      $
 0A21 BE18FF   809      ST   -1,@TYP$        Set non-integer flag
               810    $END IF
 0A24 CE7709   811    $IF @EXP$+1 .GT. 9 GOTO CNSG Use E-format for display
 0A27 6A64
 0A29 CE77FA   812    $IF @EXP$+1 .LE. -6 THEN Too small for fixed point outpu
 0A2C 6A59
 0A2E D277F6   813      $IF @EXP$+1 .LT. -10 GOTO CNSG So use E-format
 0A31 4A64
               814  *
               815  *      Let #  of sign. dgts determine format
               816  *
 0A33 BE1052   817      ST   FAC+ ,@VPTR$    Get pointer to last byte of numbe
 0A36 BC1214   818      ST   @0E$,@MSD$CH    Start of with odd/even exponent
 0A39 A21204   819      ADD  4,@MSD$CH         plus pre-update of actual pos.
               820      $REPEAT                for finding last 0 byte
 0A3C 9612     821        DECT @MSD$CH       Update sign  exp. count
 0A3E 9210     822        DEC  @VPTR$        Point to next higher byte of FAC
 0A40 8E9010   823      $UNTIL *VPTR$ .NE. 0 Found LSByte...FAC+1 <> 0 !!!!
 0A43 6A3C
 0A45 BC1190   824      ST   *VPTR$,@VPTR$+1 Store non-zero byte
 0A48 10
 0A49 8610     825      CLR  @VPTR$          LSByte might be divisible by 10
 0A4B AE100A   826      DIV  10,@VPTR$       Compute remainder after / 10
 0A4E 8E114A   827      $IF @VPTR$+1 .EQ. 0 THEN Remainder was zero.....
 0A51 54
 0A52 9212     828        DEC  @MSD$CH       Correct for trailing zero
               829      $END IF
 0A54 CC1277   830      $IF @MSD$CH .GT. @EXP$+1 GOTO CNSG Too many sign. digi
 0A57 6A64
               831    $END IF
               832  *
               833  *  Free format fixed point and integer floating output
               834  *
 0A59 BE120C   835    ST   12,@MSD$CH        Convert max. number of decimal dg
 0A5C 060BEC   836    CALL CNSD$1            Convert number to printable digit
               837  CNSF10
 0A5F 060C59   838    CALL CNSUTR            remove any trailing zeroes
 0A62 4A9F     839    BR   CNSLEA              and finish up
               840  *
               841  *      Free format E-notation floating output
```

```
 )05 006D1F
 0D08 BC564A  1263           ST   @FAC,@FAC+12    COMPUTE SIGN OF NEG. INTEGER
 0D0B 8056    1264           ABS  @FAC+12            RAISED TO INTEGER POWER
 0D0D CE5646  1265           $IF @FAC+12 .LE. >46 THEN
 0D10 6D1F
 0D12 BC7556  1266             ST   @FAC+12,@SGN$   GET EXPONENT
 0D15 A2750B  1267             A    >0B,@SGN$     ADDR. OF BYTE TO LEFT OF DEC.
 0D18 BC7590  126              ST   *SGN$,@SGN$   GET ONE'S RADIX DIGIT
 0D1B 75
 0D1C EA7501  1269             SRC  @SGN$,1       RESULT IS NEG IF E IS ODD
              1270           $END IF
              1271         $END IF
 0D1F BCA3DC  1272 PWR$$2  ST   @SGN$,RAM(FPSIGN)    SAVE SIGN
 0D22 75
 0D23 0611E0  1273         CALL XTFAC$          B IN FAC, E ON STACK
 0D26 814A    1274         DABS @FAC            MUST BE POSITIVE NUMBER
 0D28 060E64  1275         CALL LOG$$           COMPUTE LOG(B) IN FAC
 0D2B 0F0D    1276         FLTPT SMULT          COMPUTE E * LOG(B) IN FAC
 0D2D 060DB4  1277         CALL EXP$$           LET EXP GIVE ERROR WARNING
 0D30 D2A3DC  1278         $IF RAM(FPSIGN) .LT. 0 THEN
 0D33 006D38
 0D36 834A    1279           DNEG @FAC            NEGATE RESULT FOR NEG SIGN
              1280         $END
 0D38 00      1281         RTN
 0D39 D24A00  1282 PWRG02  CGE  0,@FAC          IS E NEGATIVE?
 0D3C 4D48    1283         BR   PWRG05          YES, DIVIDE BY ZERO
 0D3E 874A    1284         DCLR @FAC            RESULT IS ZERO
 0D40 00      1285         RTN
 0D41 310008  1286 PWRG01  MOVE 8 FROM ROM(#FPOS1) TO @FAC
 0D44 4A1045
 0D47 00      1287         RTN
 0D48 0F05    1288 PWRG05  FLTPT OV             RETURN OVERFLOW
 0D4A 00      1289         RTN
              1290 *
 0D4B D2B06E  1291 PWR$$3  $IF  RAM(@VSPTR) .GE. 0 GOTO PWR$$2
 0D4E 006D1F
 0D51 A36EFF  1292         DADD -8,@VSPTR       POP 8
 0D54 F8
 0D55 BE5405  1293         ST   ERRNIP,@FAC+10
 0D58 00      1294         RTN
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
			352 352 381 386 387 412 417 427 755 915
			1158 1161 1220 1230 1234 1238 1239 1248 1262 1272
			1278 1291 1342 1482 1494 1502 1569 1596 1643 1668
			1834 1849 1850 1850 1853
RANDOM	0078	57	336 341 360
RETN$$	043C	445	79
RMNGRM	DD00	32	254 255
RNDIT	02F7	322	324 328
ROLIN	0AE6	914	1324 1354 1399 1441 1510 1573 1669 1671
ROLOUT	09A0	754	760 1322 1380 1471 1566 1642
ROM	000A	1	135 168 174 177 178 179 199 200 201 224
			241 243 272 276 280 281 292 305 307 317
			344 368 381 387 1250 1286 1346 1381 1385 1406
			1431 1434 1474 1483 1495 1505 1507 1528 1567 1591
			1646 1651 1655 1666 1788 1793 1831 1836 1840 1898
ROUND$	0001	700	
ROUNU$	0002	701	1024 1881
RPI2	100B	1681	1567
RSCAN	0108	182	186
SADD	000B	710	1345 1509
SC$120	042C	438	433
SC$121	0438	443	421
SCLEN	0055	42	212 411 412 416 422 423 428
SCNAME	0056	43	412 414 426 427 428
SCOMP	000F	714	1210
SCRTCH	00C2	58	145
SCTEMP	0054	41	424 427
SCTMP2	0064	48	
SDIV	000E	713	1339 1492 1790
SGN$	0075	693	1018 1205 1208 1266 1267 1268 1268 1269 1272 1397
			1618 1861 1878 1884 1891
SGNBIT	8000	1174	1689 1693 1701 1710 1722 1734 1755 1778 1841
SGROM	001A	29	218 268 432
SIN$$	0F01	1565	93 1609
SINO1	0F3A	1587	1584
SINO2	0F3D	1588	1579
SINO3	0F4C	1593	1590
SINO4	0F56	1597	1595
SINP	10E7	1736	1597
SMULT	000D	712	1236 1242 1276 1805
SPRITE	0006	1	
SPSV$	0017	742	1102 1150
SPTR$	0016	741	767 770 771 858 860 863 864 868 870 871
			969 970 973 1102 1104 1105 1109 1114 1128 1130
			1133 1146 1147 114 1150
SQR$$	0D59	1321	89
SQRO1	0D90	1342	1349
SQRO2	0DAF	1353	1326
SQRP	102B	1686	1334
SQRQ	1045	1691	1337
SQRTEN	0FEB	1677	1431 1483
SROM	0019	28	214 429
SSUB	000C	711	1405 1426 1588
STKBAS	0072	53	128 216 217 219 220 237 249 250 251 252
			257 258 259 260 266 267 270 294 295 296

```
SYMBOL VALUE  DEF    REFERENCE TABLE

                      297  310  340  435  436
SUB    0007   706     1506 1592
SUBSTK 0073    54      219  361  371  372  430  431  434  435  440  441
                       446  447  448 1112
TABLE  000B     1      177   35  358
TABLE$ 0459   464      177
TAN$$  0F5F  1607       94
TAN01  0F7A  1618     1615
TAN3P8 1023  1684     1651
TANPI8 101B  1683     1646
TANRTN 0F79  1617     1613
TEMLEN 005A    46
TEMP   0059    45      245  272  273
TEMP1  0058    44      244  246  246  255  256  260  262  262  283  289
                       290  398  399  402  413  416  418  421  422
TEMP2  006C    50      239  253  261  284  326  333  334  335  336  337
                       33
TI     0496   478      199  280
TI1979 048F   477      200
TIBUG$ 0950   644      179
TON1$$ 03CE   397       96  163  235  332
TON2$$ 03D6   401       97  327
TONEX  03D2   399      403
TOP    0003     1
TRIGER 0007   727     1572 1613
TYP$   0018   743      792  809 1020 1027 1149
TYPE   006D    51      213  238  410
VDP    000C     1      135  168  174  224  276  317
VDPRG$ 0450   461      135  224  276  317
VEL    0007     1
VERR$  0449   451       84
VEXEC$ 044C   452       85
VPTR$  0010   735      773  775  786  787  789  790  791  817  822  823
                       824  824  825  826  827  944  945  946  947 1013
                      1015 1022 1029 1067 1068 1090 1091 1093 1094 1096
                      1096 1101 1104 1131 1132
VROA$  03C0   687      755  915
VSPTR  006E   688     1157 1158 1161 1162 1204 1237 1247 1248 1262 1291
                      1292 1342 1350 1395 1420 1834 1843 1848 1849 1850
                      1850 1853
VWARN$ 0446   450       83
WAIT   01AD   228
WAIT10 01AF   229      231
WARN$$ 284C    23      450
WRNOV  0001   720
XGROM1 018F   216      223
XPT    000E     1
XROM   018B   214      215
XTFAC$ 11E0  1847     1209 1225 1240 1243 1273 1336 1416 1425 1489 1610
                      1787
YPT    000D     1      181  183  185  187  192  195
ZER3   0FE5  1676     1495.

TOTAL NUMBER OF SYMBOLS =   268
TOTAL NUMBER OF REFERENCES =  1416
```

```
                    53  ********************************************************
                    54  *       HEADER FOR DSR LINKAGE
1310 131813         55          DATA #DSR2,#TAPE1,3,:CS1:
1313 260343
1316 5331
1318 000013         56  DSR2    DATA 0,0,#TAPE2,3,:CS2:
131B 2C0343
131E 5332
1320 000015         57          DATA 0,0,#%1,1,>03
1323 730103
                    5   ********************************************************
                    59  *       ENTRY POINTS FOR DEVICE SERVICE ROUTINE
                    60  *
1326 BF6600         61  TAPE1   DST  22,@UNIT            ENTRY FOR TAPE 1
1329 16
132A 5330           62          BR   TAPE
132C BF6600         63  TAPE2   DST  23,@UNIT            ENTRY FOR TAPE 2
132F 17
1330 BC5A73         64  TAPE    ST   @SUBSTK,@SAVSUB     SAVE SUBROUTINE POINTER
1333 A55654         65          DSUB @SCTEMP,@SCNAME     SUBTRACT LENGTH OF NAME
1336 B2EFFF         66          AND  >1F,RAM(-9(SCNAME)) CLEAR ERROR CODE
1339 F7561F
133C 35000A         67  RETRY   MOVE 10 FROM RAM(-10(SCNAME)) TO @FAC
133F 4AEFFF
1342 F656
1344 BD5E4C         68          DST  @BUFADD,@CONTRL+2   BUFFER ADDRESS
1347 8A4A           69          CAS  @FAC                BRANCH ON OPCODE
1349 5387           70          BR   TOPEN               OPEN FILE
134B 540E           71          BR   TCLOSE              CLOSE FILE
134D 53CF           72          BR   TREAD               READ RECORD
134F 53DA           73          BR   TWRITE              WRITE RECORD
1351 5387           74          BR   TOPEN               RESTORE/REWIND
1353 53F2           75          BR   TLOAD               LOAD FILE
1355 54 9           76          BR   TSAVE               SAVE FILE
1357 540E           77          BR   TCLOSE              DELETE FILE IS CLOSE OP
1359 535D           78          BR   DSRERR              NO SCRATCH RECORD
135B 5380           79          BR   EOF                 END OF  FILE CALL
                     0  ********************************************************
                     1  DSRSUB
                     2  DSRERR
135D B6EFFF         83          OR   >60,RAM(-9(SCNAME))  ERROR CODE 3 - ILLEGAL
1360 F75660
1363 061549         84  EXIT    CALL MTRON               TURN ON MOTOR
1366 061516          5  ENDIT   CALL SCROLL
1369 867F            6          CLR  XPT                 CLEAR SCREEN POINTER
136B BC735A         87          ST   @SAVSUB,@SUBSTK     RESTORE ORIGINAL SUBROUTI
136E 060012             EXRTN   CALL RETN$$              RETURN TO CALLING ROUTINE
                     9  *
1371 B6EFFF         90  DEVERR  OR   >C0,RAM(-9(SCNAME)) ERROR CODE 6 - DEVICE E
1374 F756C0
1377 0614A9         91          CALL MESS
137A 04             92          DATA 04                  STOP CASSETTE AND PRESS F
137B 061528         93          CALL ENTER
137E 5363           94          BR   EXIT
                    95  *
```

```
                     99  ********************************************************
                    100  *       CASSETTE RECORD ROUTINES
                    101  ********************************************************
                    102  *       LEGAL BIT COMBINATIONS IN FLAGS (FAC+1)
                    103  *
                    104  *        X  X  X  0  X  1  0  0    INPUT
                    105  *        X  X  X  0  X  0  1  0    OUTPUT
                    106  *  BIT   7  6  5  4  3  2  1  0
                    107  *
 13 7 8E4E53        108  TOPEN   $IF @FAC+4 .NE. 0 GOTO TOP01
 13 A 91
 13 B BEEFFF        109          ST  64,RAM(-6(SCNAME))  DEFAULT LENGTH IS 64 BYTE
 138E FA5640
 1391 A2EFFF        110  TOP01   A    63,RAM(-6(SCNAME))
 1394 FA563F
 1397 B2EFFF        111          AND  >C0,RAM(-6(SCNAME))
 139A FA56C0
 139D DA4B15        112          CLOG >15,@FAC+1          TEST "ZERO" BITS FOR OUTPU
 13A0 73B5          113          BS   TOP10               YES, ALL REQUIRED ARE ZERO
 13A2 D66717        114          $IF  @UNIT+1 .EQ. 23 GOTO DSRERR
 13A5 735D
 13A7 DA4B13        115          CLOG >13,@FAC+1          NO, TEST "ZERO" BITS FOR I
 13AA 535D          116          BR   DSRERR              NO, ERROR
                    117  *                               YES, INPUT
                    118  *                               BIT 2 IS ONE BY INFERENCE
 13AC 061503        119          CALL BEGIN
 13AF 0614A9        120          CALL MESS                DISPLAY MESSAGE
 13B2 0E            121          DATA 14                  "PRESS CASSETTE PLAY"
 13B3 53C1          122          BR   TOP20
                    123  *
                    124  *       MAY BE OUTPUT, TEST REQUIRED "ONE" BIT
                    125  *
 13B5 DA4B02        126  TOP10   $IF .BIT1 @FAC+1 .NE. 1 GOTO DSRERR
 13B8 735D
 13BA 061503        127          CALL BEGIN               REWIND CASSETTE TAPE
 13BD 0614A9        12           CALL M SS                DISPLAY MESSAGE
 13C0 0C            129          DATA 12                  "PRESS CASSETTE RECORD"
 13C1 061528        130  TOP20   CALL ENTER               WAIT FOR ENTER KEY
 13C4 061549        131          CALL MTRON               TURN ON MOTOR
 13C7 061562        132          CALL STALL               WAIT FOR TEN SECONDS
 13CA 06155E        133          CALL MTROFF              SHUT OFF MOTOR
 13CD 5366          134          BR   ENDIT               AND RETURN TO CALLING ROUT
 13CF BE6205        135  TREAD   ST   5,@TEMP             READ RECORD
 13D2 BCEFFF        136          ST   @>4E,RAM(-5(SCNAME))
 13D5 FB564E
 13D  53DD          137          BR   TXF R
 13DA BE6204        13   TWRITE  ST   4,@TEMP             WRITE RECORD
 13DD 865C          139  TXFER   CLR  @CONTRL             LENGTH OF RECORD
 13DF BC5DEF        140          ST   RAM(-6(SCNAME)),@CONTRL+1
 13E2 FFFA56
 13 5 061549        141          CALL MTRON               TURN ON MOTOR
 13E  F45C62        142          I/O  @CONTRL,@TEMP       I/O COMMAND FOR CASSETTE
 13 B 7371          143          BS   DEVERR              ERROR IN READ, EXIT
 13ED 06155E        144          CALL MTROFF              TURN OFF MOTOR
 13F0 536E          145          BR   EXRTN               RETURN TO CALLING ROUTINE
```

```
                    147  ************************************************************
                    14   *       CASSETTE LOAD ROUTINE
                    149  ************************************************************
                    150  *
                    151  *       LOAD CONTENTS OF DATA BUFFER FROM TAPE.  THE MAXIMU
                    152  *       NUMBER OF BYTES TO TRANSFER IS IN THE RECORD NUMBER
                    153  ************************************************************
13F2 D66717         154  TLOAD   $IF   @UNIT+1 .EQ. 23 GOTO DSRERR
13F5 735D
13F7 061503         155          CALL BEGIN                 POSITION TAPE MESSAGE, MOTOR
13FA 06149F         156          CALL MPLAY                 CASSETTE PLAY MESSAGE
13FD 00             157          DATA 00                    "READING"
                    15           $REPEAT
13FE 8E80CE         159          $UNTIL @>CE .EQ. 0         WAIT FOR SOUND TO COMPLETE
1401 53FE
1403 F65C05         160          I/O   @CONTRL,5           READ CASSETTE I/O CALL
1406 7455           161          BS    ERROR                CONDITION SET IF ERROR
1408 0614A9         162  TOKAY   CALL MESS                  COMPLETE AND SUCCESSFUL
140B 18             163          DATA 24                    "NO ERROR DETECTED"
140C 8653           164          CLR   @FAC+9               NO REATTEMPT NEEDED
                    165  TCLOSE
140E 06155E         166  DONE    CALL MTROFF                TURN OFF MOTOR
1411 0614A9         167          CALL MESS                  DISPLAY MESSAGE
1414 04             168          DATA 4                     "PRESS CASSETTE STOP"
1415 061528         169          CALL ENTER                 WAIT FOR ENTER KEY
1418 8E5373         170          $IF @FAC+9 .EQ. 0 GOTO EXIT  NO ERROR DETECTED
141B 63
  1C D6EFFF         171          $IF RAM(-10(SCNAME))  .NE. 6 GOTO EXIT NOT A SAVE
..+1F F65606
1422 5363
                    172  *
                    173  *       can't verify on CS2 because of cable
                    174  *
1424 D66717         175          $IF @UNIT+1 .EQ. 23 GOTO EXIT
1427 7363
1429 0614A9         176          CALL MESS                  DISPLAY MESSAGE
142C 10             177          DATA 16                    "CHECK TAPE (Y OR N)?"
142D 03             178  DCHECK  SCAN                       WAIT FOR KEY
142E 542D           179          BR    DCH CK
1430 BF7E17         180          DST   >171B,YPT            ADJUST SCREEN POINTER
1433 1B
1434 BC7D75         181          ST    @KEYNUM,CB
1437 A07D52         182          ADD   @BIAS,CB
143A D6754E         183          $IF @KEYNUM . Q. NOKEY GOTO EXIT
143D 7363
143F D67559         1 4          $IF   K YNUM .NE. YESKEY GOTO DCHECK
1442 542D
```

```
                223  ********************************************************
                224  *      CASSETTE SAVE ROUTINE
                225  ********************************************************
                226  *
                227  *      LENGTH OF THE BUFFER TO SAVE ONTO CASSETTE IS IN TH
                228  *      RECORD NUMBER.
                229  ********************************************************
1489 061503     230  TSAVE  CALL BEGIN              SETUP CASSETTE RECORDER
148C 061499     231         CALL MREC               DISPLAY RECORD MESSAGE
148F 06         232         DATA 6                  "RECORDING"
1490 061562     233         CALL STALL              WAIT FOR TEN SECONDS
1493 F65C04     234         I/O  @CONTRL,4          WRITE CASSETTE I/O CALL
1496 05140E     235         B    DONE               NO ERROR EXIT FOR RECORDING
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
%1	1573	313	57
%2	1592	319	321
ADDR	0064	42	251 317 318 319 320 321
BAUDRW	000D	44	
BEGIN	1503	269	119 127 155 193 230
BIAS	0052	40	182 248 259 278 319
BUFADD	004C	38	68
CB	0008	1	181 182
CHAR	0004	1	
CONTRL	005C	37	68 139 140 142 160 198 234 274
DCHECK	142D	178	179 184
DEVERR	1371	90	143 285
DISPLA	0009	1	
DONE	140E	166	235
DSR2	1318	56	55
DSRERR	135D	82	78 114 116 126 154
DSRSUB	135D	81	
ENDIT	1366	85	134
ENT1	1548	290	
ENTER	1528	281	93 130 169 220 245 272 282 289
ENTKEY	000D	46	289
EOF	1380	96	79
ER1	1480	218	
ERROR	1455	200	161
EXIT	1363	84	94 97 170 171 175 183
EXRTN	136E	88	145
FAC	004A	30	31 32 33 34 35 36 37 38 39 40 41 42 43 67 69 108 112 115 126 164 170 252
FIGURE	0002	1	
KEYNUM	0075	27	181 183 184 257 262 285 286 287 289 315
LENGTH	0050	39	274
LMESS	14C8	252	
LNK$LQ	003D	22	256
M1	15C0	328	326
M10	1663	346	327
M11	1675	348	327
M12	1686	350	327
M13	1697	352	327
M14	16A0	354	327
M15	16B7	356	327
M16	16C9	358	327
M2	15C9	330	326
M3	15E1	332	326
M4	15F6	334	326
M5	1601	336	326
M6	160B	338	326
M7	1621	340	326
M8	1638	342	326
M9	164D	344	327
MESS	14A9	247	91 120 128 162 167 176 202 206 240 243 270
MESS1	15C2	329	
MESS10	1665	347	
MESS11	1677	349	

```
0032      56  ATN     EQU  >32            ARCTANGENT
          57  ********************************************************
0018      58  VPOP    EQU  24             POP VALUE FROM VALUE STACK
001B      59  PGMCH   EQU  27             GET PROGRAM CHARACTER
          60  ********************************************************
0072      61  STACK   EQU  >72            STACK FOR DATA          STATUS BL
0073      62  SUBSTK  EQU  >73            SUBROUTINE STACK
0074      63  KEYBD   EQU  >74            KEYBOARD SELECTION
0075      64  RKEY    EQU  >75            KEY CODE
0078      65  RANDOM  EQU  >78            RANDOM NUMBER GENERATOR
0079      66  TIMER   EQU  >79            TIMING REGISTER
007B      67  VDPSTS  EQU  >7B            VDP STATUS REGISTER
007C      68  ERCODE  EQU  >7C            STATUS REGISTER
          69  ********************************************************
0000      70  VAR0    EQU  >00            TEMPORARY WORKSPACES IN
0001      71  VARV    EQU  >01            EDIT
0002      72  STPT    EQU  >02                 TWO BYTES
0004      73  PABPTR  EQU  >04
0004      74  VARY    EQU  >04
0005      75  VARZ    EQU  >05
0006      76  CCPPTR  EQU  >06
0006      77  ENPT    EQU  >06                 TWO BYTES
0007      78  RECLEN  EQU  >07
0008      79  CCPADR  EQU  >08
0008      80  VARC    EQU  >08
0009      81  VARD    EQU  >09
000B      82  VAR2    EQU  >0B
000C      83  BYTE    EQU  >0C
000C      84  BC      EQU  BYTE
000D      85  VAR1    EQU  >0D
000E      86  VAR4    EQU  >0E
0010      87  VAR5    EQU  >10
0011      88  VAR6    EQU  >11
0012      89  START   EQU  >12
0013      90  VAR8    EQU  >13
0014      91  NBC     EQU  >14                 TWO BYTES
0016      92  VAR9    EQU  >16
0017      93  DSRFLG  EQU  >17
          94  ********************************************************
0018      95  STRSP   EQU  >18
001A      96  STREND  EQU  >1A
001C      97  SREF    EQU  >1C
001E      98  NBD     EQU  >1E                 TWO BYTES
0020      99  VARW    EQU  >20            PERMANENT WORKING SPACES IN
0024     100  STVSPT  EQU  >24
002A     101  VARA    EQU  >2A
002C     102  VAR81   EQU  >2C
002E     103  EXTRAM  EQU  >2E
0030     104  STLN    EQU  >30
0032     105  ENLN    EQU  >32
0034     106  DATA    EQU  >34
0036     107  LNBUF   EQU  >36
0038     108  RAMPTR  EQU  >38
003E     109  SYMTAB  EQU  >3E
0040     110  SYMPTR  EQU  >40
```

```
                    160  *
                    161  *              B A S I C   T O K E N   T A B L E
                    162  *
                    163  *                                      REVISED 8/23/78
                    164  *       EQU   >80          SPARE
0081                165  ELSE$    EQU   >81
0082                166  SSEP$    EQU   >82          ":"              (62)
0083                167  TREM$    EQU   >83          "!"              (62)
0084                168  IF$      EQU   >84          "IF"
0085                169  GO$      EQU   >85          "GO"
0086                170  GOTO$    EQU   >86          "GOTO"
0087                171  GOSUB$   EQU   >87          "GOSUB"
0088                172  RETUR$   EQU   >88          "RETURN"
0089                173  DEF$     EQU   >89          "DEF"
008A                174  DIM$     EQU   >8A          "DIM"
008B                175  END$     EQU   >8B          "END"
008C                176  FOR$     EQU   >8C          "FOR"
008D                177  LET$     EQU   >8D          "LET"
008E                178  BREAK$   EQU   >8E          "BREAK"
008F                179  UNBRE$   EQU   >8F          "UNBREAK"
0090                180  TRACE$   EQU   >90          "TRACE"
0091                181  UNTRA$   EQU   >91          "UNTRACE"
0092                182  INPUT$   EQU   >92          "INPUT"
0093                183  DATA$    EQU   >93          "DATA"
0094                184  RESTO$   EQU   >94          "RESTORE"
0095                185  RANDO$   EQU   >95          "RANDOMIZE"
0096                186  NEXT$    EQU   >96          "NEXT"
0097                187  READ$    EQU   >97          "READ"
0098                188  STOP$    EQU   >98          "STOP"
0099                189  DELET$   EQU   >99          "DELETE"
009A                190  REM$     EQU   >9A          "REM"
009B                191  ON$      EQU   >9B          "ON"
009C                192  PRINT$   EQU   >9C          "PRINT"
009D                193  CALL$    EQU   >9D          "CALL"
009E                194  OPTIO$   EQU   >9E          "OPTION"
009F                195  OPEN$    EQU   >9F          "OPEN"
00A0                196  CLOSE$   EQU   >A0          "CLOSE"
00A1                197  SUB$     EQU   >A1          "SUB"
00A2                19   DISPL$   EQU   >A2          "DISPLAY"
                    199  *        EQU   >A3          "SUBEXIT"        (62)
                    200  *        EQU   >A4          "SUBEND"         (62)
                    201  *        EQU   >A5          "REAL"           (62)
                    202  *        EQU   >A6          "INTEGER"        (62)
                    203  *        EQU   >A7          "SCRATCH"        (62)
                    204  *        EQU   >A8          "ACCEPT"         FUTURE
                    205  *        EQU   >A9          "IMAGE"          FUTURE
                    206  *        EQU   >AA          SPARES
                    207  *        EQU   >AB
                    208  *        EQU   >AC
                    209  *        EQU   >AD
                    210  *        EQU   >AE
                    211  *        EQU   >AF
00B0                212  THEN$    EQU   >B0          "THEN"
00B1                213  TO$      EQU   >B1          "TO"
00B2                214  STEP$    EQU   >B2          "STEP"
```

```
00B3    215  COMMA$ EQU  >B3          ","
00B4    216  SEMIC$ EQU  >B4          ";"
00B5    217  COLON$ EQU  >B5          ":"
00B6    218  RPAR$  EQU  >B6          ")"
00B7    219  LPAR$  EQU  >B7          "("
00B8    220  CONC$  EQU  >B8          "&"                (62)
        221  *      EQU  >B9          SPARE
        222  *      EQU  >BA          "OR"               (62)
        223  *      EQU  >BB          "AND"              (62)
        224  *      EQU  >BC          SPARE FOR "XOR"    (62)
        225  *      EQU  >BD          "NOT"              (62)
00BE    226  EQUAL$ EQU  >BE          "="
00BF    227  LESS$  EQU  >BF          "<"
00C0    228  GREAT$ EQU  >C0          ">"
00C1    229  PLUS$  EQU  >C1          "+"
00C2    230  MINUS$ EQU  >C2          "-"
00C3    231  MULT$  EQU  >C3          "*"
00C4    232  DIVI$  EQU  >C4          "/"
00C5    233  CIRCU$ EQU  >C5          "^"
        234  *      EQU  >C6          SPARE
00C7    235  STRIN$ EQU  >C7          QUOTED STRING
00C8    236  UNQST$ EQU  >C8          UNQUOTED STRING
00C8    237  NUM$   EQU  UNQST$       ALSO NUMERICAL STRING
00C9    238  LN$    EQU  >C9          LINE NUMBER CONSTANT
        239  *      EQU  >CA          SPARE
00CB    240  ABS$   EQU  >CB          "ABS"
00CC    241  ATN$   EQU  >CC          "ATN"
00CD    242  COS$   EQU  >CD          "COS"
00CE    243  EXP$$  EQU  >CE          "EXP"
00CF    244  INT$   EQU  >CF          "INT"
00D0    245  LOG$   EQU  >D0          "LOG"
00D1    246  SGN$$  EQU  >D1          "SGN"
00D2    247  SIN$   EQU  >D2          "SIN"
00D3    248  SQR$   EQU  >D3          "SQR"
00D4    249  TAN$   EQU  >D4          "TAN"
00D5    250  LEN$   EQU  >D5          "LEN"
00D6    251  CHR$$  EQU  >D6          "CHR$"
00D7    252  RND$   EQU  >D7          "RND"
00D8    253  SEG$$  EQU  >D8          "SEG$"
00D9    254  POS$   EQU  >D9          "POS"
00DA    255  VAL    EQU  >DA          "VAL"
00DB    256  STR$$  EQU  >DB          "STR$"
        257  *
        258  *
        259  *
        260  *      EQU  >EC          "ALL"              (62)
        261  *      EQU  >ED          "USING"            (62)
        262  *      EQU  >EE          "BEEP"             (62)
        263  *      EQU  >EF          "ERASE"            (62)
        264  *      EQU  >F0          "AT"               (62)
00F1    265  BASE$  EQU  >F1          "BASE"
        266  *      EQU  >F2          "VARIABLE"         (62)
        267  *      EQU  >F3          "TEMPORARY"        (62)
        268  *      EQU  >F4          "RELATIVE"         (62)
        269  *      EQU  >F5          "INTERNAL"         (62)
```

```
00F6      270  SEQUE$ EQU  >F6          "SEQUENTIAL"
00F7      271  OUTPU$ EQU  >F7          "OUTPUT"
00F8      272  UPDAT$ EQU  >F8          "UPDATE"
00F9      273  APPEN$ EQU  >F9          "APPEND"
00FA      274  FIXED$ EQU  >FA          "FIXED"
00FB      275  PERMA$ EQU  >FB          "PERMANENT"
00FC      276  TAB$   EQU  >FC          "TAB"
00FD      277  NUMBE$ EQU  >FD          "#"
          278  *      EQU  >FE          GROM LINE POINTER
          279  *      EQU  >FF          SPARE
          280  ********************************************************
0003      281  BREAK  EQU  >03          CONTROL C (BREAK) COMMAND
0008      282  BACK   EQU  >08          KEYBOARD COMMAND KEY CODE
0009      283  FORW   EQU  >09
          284  ********************************************************
          285  *           ASCII CODES FOR EACH CHARACTER
0030      286  ZERO   EQU  :0:          0
0039      287  NINE   EQU  :9:          9
003A      288  COLON  EQU  ::::         :
003B      289  SEMIC  EQU  :;:          ;
003C      290  LESS   EQU  :<:          <
003D      291  EQUAL  EQU  :=:          =
003E      292  GREAT  EQU  :>:          >
003F      293  QUEST  EQU  :?:          ?
0041      294  A      EQU  :A:          A
0045      295  E      EQU  :E:          E
005A      296  Z      EQU  :Z:          Z
005B      297  OPENB  EQU  :[:
005C      298  BACKS  EQU  :\:
005D      299  CLOSEB EQU  :]:
005F      300  UNLN   EQU  :_:
0020      301  SPACE  EQU  : :
0022      302  QUOTE  EQU  :":          "
0024      303  DOLLAR EQU  :$:          $
0028      304  LPAR   EQU  :(:          (
0029      305  RPAR   EQU  :):          )
002B      306  PLUS   EQU  :+:          +
002C      307  COMMA  EQU  :,:          ,
002D      308  MINUS  EQU  :-:          -
002E      309  DOT    EQU  :.:          .
007E      310  CURSOR EQU  >1E+OFFSET       SPECIAL CURSOR CHARACTER
007F      311  EDGECH EQU  >1F+OFFSET       SPECIAL SCREEN EDGE CHARA(
```

```
                    327  *
                    328  *       GROM HEADER
                    329  *
                    330          GROM 1
                    331          ORG  0
2000 AA             332          DATA >AA            GROM ID
                    333  * Change from 1 to 2 for 99/4A
2001 02             334          DATA 2              VERSION
                    335  * Change from 2 to 1 to remove equation calc. from 99/4A
2002 01             336          DATA 1              NUMBER OF USER PROGRAMS
2003 00             337          DATA 0              RESERVED
2004 0000           338          DATA #0
2006 214D           339          DATA #USER          USER PROGRAM HEADER POINTER
2008 0000           340          DATA #0             NO DSRS
200A 4D1A           341          DATA #LINK1         LINK TO SUBPROGRAMS
200C 000000         342          DATA #0,#0          RESERVED
200F 00
```

```
 20AD B0B4
 20AF 0DAEA1   373  MSG15  DATA 13,:NAME CONFLICT:
 20B2 ADA580
 20B5 A3AFAE
 20B8 A6ACA9
 20BB A3B4
               374  MSG2
 20BD 0DA3A1   375  MSG17  DATA 13,:CAN'T DO THAT:
 20C0 AE87B4
 20C3 80A4AF
 20C6 80B4A8
 20C9 A1B4
 20CB B4A980   376  MSGFST DATA :TI BASIC READY:
 20CE A2A1B3
 20D1 A9A380
 20D4 B2A5A1
 20D7 A4B9
 20D9 0FA2A1   377  MSGBLN DATA 15,:BAD LINE NUMBER:
 20DC A480AC
 20DF A9AEA5
 20E2 80AEB5
 20E5 ADA2A5
 20E8 B2
 20E9 8A80A2   378  MSGBRK DATA :* BREAKPOINT AT :
 20EC B2A5A1
 20EF ABB0AF
 20F2 A9AEB4
 20F5 80A1B4
 20F8 80
 20F9 0EA6AF   379  MSG18  DATA 14,:FOR-NEXT ERROR:
 20FC B28DAE
 20FF A5B8B4
 2102 80A5B2
 2105 B2AFB2
               380  *
               381  *      FILE MANAGER ERROR MESSAGES
               382  *
 2108 B4B2B9   383  MSG20  DATA :TRY AGAIN:: :
 210B 80A1A7
 210E A1A9AE
 2111 9A80
 2113 09A98F   384  MSG21  DATA 9,:I/O ERROR:
 2116 AF80A5
 2119 B2B2AF
 211C B2
 211D 0AA6A9   385  MSG22  DATA 10,:FILE ERROR:
 2120 ACA580
 2123 A5B2B2
 2126 AFB2
 2128 08A9AE   386  MSG23  DATA 11,:INPUT ERROR:
 212B B0B5B4
 212E 80A5B2
   31 B2AFB2
 2134 0AA4A1   387  MSG24  DATA 10,:DATA ERROR:
 2137 B4A180
```

```
213A A5B2B2
213D AFB2
              *
213F ODACA9   MSG19  DATA 13,:LINE TOO LONG:
2142 AEA580
2145 B4AFAF
2148 80ACAF
214B AEA7
              *
                     BASE   0,0,>300,>300,0,0,0
              *<><><><><><><><><><><><><><><><><><><><><><><><><><>
              *  The following is changed for 99/4A w/o equation calc.
              *   ***** This if for the compulator so the error ******
              *   ******* messages didn't have to be moved ***********
              *       BR   SCANKB
              *
              *USER   DATA #USER2,#M$COMP,19,:EQUATION CALCULATOR:
              *
              *USER2  DATA #0,#MAIN,15,:TI BASIC       :
214D 000021   USER    DATA #0,#MAIN,8,:TI BASIC:
2150 6F0854
2153 492042
2156 415349
2159 43
              *<><><><><><><><><><><><><><><><><><><><><><><><><><>
              *<><><><><><><><><><><><><><><><><><><><><><><><><><>
              *       BY STAN HUME 06/23/80 TO SAVE 2 BYTES TO ADD THE
              *       FOLLOWING BRANCH TABLE ENTRY SO KILSYM WILL NOT
              *       BE DIRECTLY REFERENCED BY FLMGR.
              *
215A 422B             BR   KILSYM
              *<><><><><><><><><><><><><><><><><><><><><><><><><><>
              *
              *
              SPCCHR
215C 007C7C           DATA >00,>7C,>7C,>7C,>7C,>7C,>7C,>7C  CURSOR CHAR
215F 7C7C7C
2162 7C7C
2164 000000           DATA 0,0,0,0,0,0,0,0    SCREEN EDGE CHAR.
2167 000000
216A 0000
              *
216C F00CF8   VDPREG DATA >F0,>0C,>F8
```

```
                    418  ************************************************
                    419  *                                              *
                    420  *       START OF BASIC INTERPRETER             *
                    421  *                                              *
                    422  ************************************************
                    423  MAIN
216F BE7388         424          ST   RSTK,@SUBSTK LOAD SUBROUTINE STACK
2172 0627E3         425          CALL CHRTA2       LOAD CHARACTER TABLE
2175 31000E         426          MOVE 14 FROM ROM(MSGFST) TO RAM(NLNADD-32)
2178 A2C220
217B CB
217C B68088         427          CLR  @FLAG
217F 064012         428          CALL CLSALL
2182 BE34FF         429          ST   >FF,@DATA    ASSUME EXECUTION OF THE PROGRAM
                    430  *                         STARTS FROM THE 1ST LINE.
2185 BF6E06         431          DST  VRAMVS,@VSPTR     INIT VALUE STACK
2188 F8
2189 BD246E         432          DST  @VSPTR,@STVSPT    BASE OF STACK
218C BD3270         433  MAINA   DST  @>70,@ENLN   END OF MEMORY
218F BD3032         434          DST  @ENLN,@STLN   STARTING ADDRESS OF EXTERNAL RAM
                    435  MAIN1
2192 06222B         436          CALL KILSYM       Kill the symbol table
                    437  MAO
2195 B28088         438          RB   @FLAG,3           IN CASE OF ERROR IN FUNCTION
2198 F7
2199 B68088         439          SB   @FLAG,5           EDIT MODE
219C 20
                    440  *  FOLLOWING 2 LINES ARE ADDED FOR 99/4A KEYBOARD, 3/3/81
219D BE7405         441          ST   5,@KEYBD          SELECT FULL KEYBOARD
21A0 03             442          SCAN
                    443  *
21A1 BE7388         444  MAIN2   ST   RSTK,@SUBSTK STARTING ADD. OF SUBROUTINE STACK
                    445  *
21A4 BF2002         446  MA1     DST  NLNADD,@VARN      FRONT OF LINE POINTER
21A7 E2
                    447  *
                    448  **      AUTONUM ON?
                    449  *
21A8 DA8088         450          TBR  @FLAG,0           'AUTONUM' MODE ON?
21AB 01
21AC 61D6           451          BS   MA16              NO
                    452  *       GET THE LINE #.
21AE A14648         453          DADD @CURINC,@CURLIN   NEW LINE #
21B1 D24600         454    $IF @CURLIN .LT. 0 THEN      >32767?
21B4 61BC
21B6 B28088         455      RB   @FLAG,0          YES, RESET NUM FLAG
21B9 FE
21BA 41D6           456      BR   MA16
                    457    $END IF
                    458  *
                    459  MA10
21BC D53032         460    $IF @STLN .DNE. @ENLN THEN Line might exist
21BF 61C9
21C1 BD4446         461      DST  @CURLIN,@BUFFY  Ready for program search
21C4 06283E         462      CALL SEETWO          Search for existance of AUTONUM
```

```
21C7 6399      463       BS   EDT$00              line - COND set = yes
               464     $END IF
21C9 064D00    465     CALL SCROLL               Scroll to the next line
21CC BD5E46    466     DST  @CURLIN,@ARG+2       NEW LINE #
21CF 062842    467     CALL DISO                 DISPLAY LINE NUMBER
21D2 9120      468     DINC @VARW                FOLLOWED BY A SPACE
21D4 41D9      469     BR   MA16A
               470   *
               471   **      GET A LINE FROM THE KEYBOARD.
               472   *
21D6 064D00    473   MA16    CALL SCROLL
21D9 BEA2E1    474   MA16A   ST   :>:+>60,RAM(NLNADD-1)
21DC 9E
21DD 062832    475           CALL READLN             READ IN A LINE.
21E0 B221E0    476           AND  >E0,@VARW+1  CORRECT STATE OF LINE IN CASE O
               477   *                         'AUTONUM'
21E3 9421      478           INCT @VARW+1            ALLOW FOR SCREEN EDGE
               479   *
               480   *******SCAN THE LINE.*********
               481   *
21E5 062457    482     CALL SCANKB               SCAN THE LINE.
21E8 8F4462    483     $IF @BUFFY .DNE. 0 THEN Line # present
21EB 03
21EC DA8088    484       $IF .BITO @FLAG .EQ. 0 THEN No AUTONUM
21EF 0141FC
21F2 D6750D    485         $IF @RKEY .NE. CHRTN THEN Has to be up or down
21F5 61FC
21F7 D64201    486           $IF @CHAT .EQ. 1 GOTO EDT$$0 Start EDIT mode
21FA 6388
               487         $END IF
               488       $END IF
21FC 0626B4    489       CALL EDITLN             EDIT THE LINE IN
21FF 61A4      490       BS   MA1                If not really an edit
2201 4192      491       BR   MAIN1
               492   *   Jump always
               493     $END IF                   AND EXIT
2203 D64201    494     $IF @CHAT .EQ. 1 GOTO MA1  If blank input line...ignor
2206 61A4
2208 C6A320    495     $IF RAM(CRNBUF) .H. EDIT$ GOTO MP0  Intercept imperati
220B 096266
220E BF2C03    496     DST  CRNBUF+1,@VAR81      ANTICIPATE USAGE OF "PGMCHR"
2211 21
2212 0F1B      497     XML  PGMCH                PREPARE CHAT FOR "OLD" AND "SA\
               498   *
2214 8AA320    499           CASE RAM(CRNBUF)
2217 424D      500           BR   MA1H           'RUN'
2219 416F      501           BR   MAIN           'NEW'
221B 4268      502           BR   MA1H1          'CONTINUE'
221D 4245      503           BR   MA1G           'LIST'
221F 4342      504           BR   MAEXIT         'BYE'
2221 428C      505           BR   MA1H7          'NUMBER'
2223 42A7      506           BR   MA1H9          'OLD'
2225 42AA      507           BR   MA1H10         'RESEQUENCE'
2227 429F      508           BR   MA1H8          'SAVE'
2229 4377      509           BR   S$EDIT         'EDIT'
```

```
                 *
                 *******KILL SYMBOL TABLE
                 *
                 KILSYM
2228 BD4030        DST   @STLN,@SYMPTR       SPACE
222E D54070        $IF @SYMPTR .DNE. @>70 THEN
2231 6235
2233 9340            DDEC @SYMPTR            Point to first free byte
                   $END IF
2235 873E          DCLR @SYMTAB              Kill symbol table
2237 BD1840        DST   @SYMPTR,@STRSP      Set up string handling vars
223A 9318          DDEC @STRSP
223C BD1A18        DST   @STRSP,@STREND
223F 8643          CLR   @BASE               RESET OPTION BASE TO 0
2241 87A3EC        DCLR RAM(SAVEPC)  Disallow CONTINUE
2244 00            RTN
                 *
                 *******LIST*********
                 *
                 MA1G
2245 D53032        $IF @STLN .DEQ. @ENLN GOTO ILLST Refuse LIST without prog
2248 6346
224A 054018        B     LIST                LIST the actual program
                 *
                 *******RUN***********+
                 *
                 MA1H
 24D D53032        $IF @STLN .DEQ. @ENLN GOTO ILLST Refuse without program
2250 6346
2252 064D10        CALL RUN                  Get line information
2255 8744          DCLR @BUFFY               So no line numbers if error in cl
2257 064012        CALL CLSALL
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                 *        CHANGED FROM:
                 *        ST    1,@BUFFY
                 *              BY STAN HUME 6/19/80 TO SAVE BYTES
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
225A 9044          INC   @BUFFY              Put it back in execution
                 *                 Entry for GROM programs
                 MA1HO
225C 87A3EC        DCLR RAM(SAVEPC)          Disallow CONTINUE after RUN
225F BF6E06        DST   VRAMVS,@VSPTR       Set the stack empty
2262 F8
2263 BD246E        DST   @VSPTR,@STVSPT      Reset stack base
                 MPO
2266 4828          BR    PRGLST
                 *
                 *******CONTINUE*******++
                 *
                 MA1H1
2268 06282C        CALL GETNB                Check for END-OF-LINE
226B 466C          BR    SCANER              Junk on end of command
 5D 8FA3EC         $IF RAM(SAVEPC) .DNE. 0 THEN Hit BREAKPOINT last time
2270 6285
2272 B28088          RB    @FLAG,5           Switch to RUN mode
```

```
2275 DF
2276 064D00      CALL SCROLL
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
            *     CHANGED FROM:
            *   DST  RAM(SAVEPC),@EXTRAM Copy last Program Counter
            *   ST   -1,@BUFFY         Indicate program mode in BUFFY
            *   DCLR RAM(SAVEPC)       Prevent unauthorized CONTINUEs
            *     BY STAN HUME 6/19/80 TO SAVE BYTES
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
2279 872E         DCLR @EXTRAM           Prevent unauthorized CONTINUEs
227B C1A3EC       DEX  @EXTRAM,RAM(SAVEPC) and get previous program cou
227E 2E
227F BE44FF       ST   -1,@BUFFY         Back to program mode
2282 054D0C        B    EXEC1            Resume normal execution
                $END IF
2285 06284E     CALL ERR$$               Indicate error
2288 2055       DATA #MSG6               "* CAN'T CONTINUE"
228A 41A4       BR   MA1                 Since we're in EDIT mode, we'll
            *
            *******NUMBER*********
            *
228C 062840 MA1H7  CALL AUTON            GET THE STARTING LINE # AND INC
228F BD4614        DST  @NBC,@CURLIN Set current line number
2292 BD481E        DST  @NBD,@CURINC       and current increment
2295 B68088        SB   @FLAG,0          Set AUTONUM bit for future plea
2298 01
2299 BF2002        DST  NLNADD,@VARW
229C E2
229D 41BC          BR   MA10
            *
            *******SAVE*********
            *
229F D53032 MA1H8  $IF  @STLN .DEQ. @ENLN GOTO ILLST   If no program
22A2 6346
22A4 054014        B    SAVE
            *
            *******OLD*********
            *
22A7 054016 MA1H9  B    OLD
            *
            *****RESEQUENCE*****
            *
            MA1H10
22AA D53032   $IF @STLN .DEQ. @ENLN GOTO ILLST  If no program
22AD 6346
22AF 062840      CALL AUTON              GET STARTING LINE # AND INCREME
22B2 BD4A32      DST  @ENLN,@FAC         COMPUTE NUMBER OF INCREMENTS RE
22B5 A54A30      DSUB @STLN,@FAC             ACTUAL NUMBER OF LINES - :
  B8 E74A00      DSRL @FAC,2             THIS ALSO TAKES CARE OF THIS ^
22BB 02
22BC A94A1E      DMUL @NBD,@FAC          COMPUTE SPACE OCCUPIED BY INCRE
22BF 8E4B62      $IF @FAC+1 .NE. 0 THEN
22C2 CA
            SEQ$1
22C3 06284E        CALL ERR$$            GIVE ERRORMESSAGE IN CASE OVERF
```

```
22C6 20D9     607          DATA #MSGBLN          "VALUE OUT OF RANGE"
 ?C8 41A4     608          BR   MA1             AND GET BACK YOYO...IN EDIT MODE
              609        $END IF
 22CA A1144C  610        DADD @FAC+2,@NBC       NOW COMPUTE HIGHEST ADDRESS USED
 22CD 0C62C3  611        $IF .CARRY. GOTO SEQ$1 AND WATCH OUT FOR OVERFLOW
 22D0 C6147F  612        $IF @NBC .H. >7F GOTO SEQ$1   OVERFLOW IS > 32767
 22D3 62C3
              613        $  SO FAR, SO GOOD....NOW WE'RE IN FOR THE REAL WORK
 22D5 8650    614        CLR  @FAC+6            IS GOING TO BE USED FOR DOUBLE ADI
 22D7 BD4A32  615        DST  @ENLN,@FAC        START AT END OF PROGRAM SEGMENT
              616        $REPEAT
 22DA 954A    617          DINCT @FAC           AND SKIP EOL AND COUNT
              618          $REPEAT
 22DC D6B04A  619            $IF RAM(@FAC) .EQ. STRIN$ GOTO SEQ$2 SKIP STRINGS
 22DF C762E8
 22E2 D6B04A  620            $IF RAM(@FAC) .EQ. NUM$ THEN
 22E5 C842F3
              621  SEQ$2
 22E8 914A    622              DINC @FAC        GET NEXT TOKEN (COUNT)
 22EA BC51B0  623              ST   RAM(@FAC),@FAC+7 READY FOR DOUBLE ADD
 22ED 4A
 22EE A14A50  624              DADD @FAC+6,@FAC       UP TO END OF STRING
 22F1 431E    625            $SELSE
 22F3 D6B04A  626              $IF RAM(@FAC) .EQ. LN$ THEN ELSE CHECK FOR LINE #
 22F6 C9431E
 22F9 914A    627                DINC @FAC     FOUND ONE...MOVE UP TO NEXT BYTE
 ?2FB BD4E14  628                DST  @NBC,@FAC+4  GET SET FOR SEARCH
  2FE BD4C30  629                DST  @STLN,@FAC+2 COMPARE WITH ENTRIES IN LINE
              630                $REPEAT              TABLE
 2301 D5B04C  631                  $IF RAM(@FAC+2) .DEQ. RAM(@FAC) GOTO SEQ$3 OK
 2304 B04A63
 2307 18
 2308 A54E1E  632                    DSUB @N8D,@FAC+4  UPDATE NEW LINE #
 230B A34C00  633                    DADD 4,@FAC+2   AND ENTRY IN LINE # TABLE
 230E 04
 230F C54C32  634                $UNTIL @FAC+2 .DH. @ENLN STOP IF OUTSIDE TABLE
 2312 4301
 2314 BF4E7F  635                DST  >7FFF,@FAC+4 DEFAULT = 32767
 2317 FF
              636  SEQ$3
 2318 BDB04A  637                DST  @FAC+4,RAM(@FAC) UPDATE LINE REFERENCE
 231B 4E
 231C 914A    638                DINC @FAC     MOVE ON TO END OF LINE #
              639              $END IF
              640            $SEND IF
 231E 914A    641            DINC @FAC          GET NEXT TOKEN ON LINE
 2320 8EB04A  642          $UNTIL RAM(@FAC) .EQ. 0     STOP ON END OF LINE
 2323 42DC
 2325 D54A70  643        $UNTIL @FAC .DEQ. @>70       AND ON END OF PROGRAM COD
 2328 42DA
              644        $  NOW UPDATE THE LINE # TABLE
 ?32A BD4A30  645        DST  @STLN,@FAC        START AT BEGINNING OF TABLE
  32D BD4C14  646        DST  @NBC,@FAC+2       WITH START ADDRESS OFF COURSE
              647        $REPEAT                AT LEAST ONE ENTRY
 2330 BDB04A  648          DST  @FAC+2,RAM(@FAC)   UPDATE LINE # IN TABLE
```

```
2333 4C
2334 A54C1E   649          DSUB @NBD,@FAC+2    COMPUTE NEXT LINE #
2337 A34A00   650          DADD 4,@FAC         AND NEXT ENTRY IN LINE # TABLE
233A 04
233B C54A32   651        $UNTIL @FAC .DH. @ENLN STOP AT END OF LINE # TABLE
233E 4330
              652      $END IF
2340 41A4     653      BR   MA1               AND CONTINUE WITH SOMETHING NEW
              654  ********************
              655  *
              656  MAEXIT
2342 064012   657      CALL CLSALL            Properly close all files
2345 0B       658      EXIT
              659  *
              660  ILLST
2346 064D00   661      CALL SCROLL            Scroll the screen for message
2349 8744     662      DCLR @BUFFY            Prevent line # printing
234B 06284E   663      CALL ERR$$
234E 20BD     664      DATA #MSG17            "* ILLEGAL STATEMENT"
2350 467E     665      BR   ILL1
              666  *
              667  ********************
              668  *
              669  MEMFUL
2352 A12A02   670      DADD @STPT,@VARA       Total # of bytes to be added
2355 A5302A   671      DSUB @VARA,@STLN
2358 CB3007   672      $IF @STLN .NOT. .DHE. VRAMVS+64 THEN    Memory full
235B 386368
235E A1302A   673        DADD @VARA,@STLN     Back to old start line # buffe
2361 06284E   674        CALL ERR$$
2364 2049     675        DATA #MSG5           "* MEMORY FULL"
2366 467E     676        BR   ILL1
              677      $END IF
2368 00       678      RTN
```

```
                    680  ************************************************
                    681  **      SUBROUTINE TO MOVE THE CONTENT IN EXTERNAL RAM FRO
                    682  **      A LOWER ADDRESS LOCATION TO A HIGHER ADDRESS LOCAT
                    683  ************************************************
2369 9306           684  MVDN1   DDEC @VARY+2            NEXT LOCATION TO MOVE INTO
236B 9300           685          DDEC @VARO              NEXT LOCATION TO MOVE FROM
236D BCB006         686  MVDN    ST   RAM(@VARO),RAM(@VARY+2) MOVE ONE BYTE
2370 B000
2372 935C           687          DDEC @ARG               # OF BYTES
2374 4369           688          BR   MVDN1              Continue if not end of move
2376 00             689          RTN
```

```
                    691  *
                    692  *       EDIT routine - display requested line and edit
                    693  *            any changes in the program segment
                    694  *
                    695  S$EDIT
2377 06282C         696    CALL GETNB              Try to get in a line number
237A 6679           697    BS   ERR$1              No line number = syntax error
237C 06283C         698    CALL GETLN              Found something - check line #
237F 8E0C66         699    $IF @BYTE .EQ. 0 GOTO ERR$1 However, it's not legal
2382 79
2383 062856         700    CALL GETNB2             Check for junk on end
2386 4679           701    BR   ERR$1
                    702    $
                    703    $     BUFFY contains the line number we just read
                    704    $
                    705  EDT$$0
2388 D53032         706    $IF @STLN .DEQ. @ENDLN GOTO ILLST Nothing to edit
238B 6346
                    707    $REPEAT
238D 06283E         708      CALL SEETWO           Try to locate the number
2390 6399           709      BS   EDT$00           Number has to be exact
                    710  ERR$2
2392 06284E         711      CALL ERR$$            Give error message
2395 20D9           712      DATA #MSGBLN          "* BAD LINE NUMBER"
2397 467E           713      BR   ILL1             And continue
                    714  EDT$00
2399 BE061D         715      ST   29,@CCPPTR       Force new record on first line
                    716      $
                    717      $  The entry in the line number table is in EXTRAM
                    718      $
239C BD142E         719      DST  @EXTRAM,@NBC     Prepare data for list routine
239F BE1760         720      ST   OFFSET,@DSRFLG   Set screen output mode
23A2 BE071C         721      ST   2 ,@RECLEN       Select standard record length
23A5 06282E         722      CALL LLIST            List the line
                    723      $
                    724      $  VARW contains the position of the first character
                    725      $  following the line number
                    726      $
23A8 C40607         727      $IF @CCPPTR .H. @RECLEN THEN Exactly at end of line
23AB 43B8
23AD 064D00         728        CALL SCROLL         Scroll up one line
23B0 A72000         729        DSUB 32,@VARW       And correct both VARW
23B3 20
23B4 A70800         730        DSUB 28,@CCPADR                 and CCPADR
23B7 1C
                    731      $END IF
23B  BD5E20         732      DST  @VARW,@ARG+2     Set cursor at start position
23BB B25FE0         733      AND  >E0,@ARG+3       Back to beginning of line
23BE A35E00         734      DADD 125,@ARG+2       Compute theoretically highest l
23C1 7D
23C2 BD2A08         735      DST  @CCPADR,@VARA    Use current high pos. as high p
23C5 C95E2A         736      $IF @ARG+2 .DL. @VARA THEN More than 4 lines....corr
23C8 63CE
23CA BF5E02         737        DST  >2FD,@ARG+2    Allow for one more line
23CD FD
```

```
                    738       $END IF
 23CE 062858        739       CALL READL1             Allow the user to make changes
  3D1 B221E0        740       AND  >E0,@VARW+1        Include line number in edit
 23D4 9421          741       INCT @VARW+1            Allow for edge characters
 23D6 DA8088        742       $IF .BITO @FLAG .EQ. 1 GOTO EDT$01
 23D9 0143FA
 23DC D6750A        743       $IF @RKEY .EQ. UPARR THEN Ended in UP arrow
 23DF 43EC
 23E1 A71400        744         DSUB 4,@NBC           Compute next line entry
 23E4 04
 23E5 C91430        745         $IF @NBC .DL. @STLN GOTO EDT$01 Doesn't exist
 23E8 43FA
 23EA 43FF          746         BR   EDT$02
                    747       $END IF
 23EC D6750B        748       $IF @RKEY .EQ. DWNARR THEN Want next program line
 23EF 4403
 23F1 A31400        749         DADD 4,@NBC           Compute entry to next line
 23F4 04
 23F5 C51432        750         $IF @NBC .DH. @ENLN THEN Passed high program limit
 23F8 43FF
                    751  EDT$01
 23FA BE750D        752           ST   CHRTN,@RKEY Assume no interest in other line
 23FD 4403          753         $SELSE
                    754  EDT$02
 23FF BD1EB0        755           DST  RAM(@NBC),@NBD Next time we display this lin
 2402 14
                    756         $SEND IF
                    757       $END IF
 2403 8E6044        758       $IF @ARG+4 .EQ. 0 THEN We have a change in this line
 2406 0D
 2407 062457        759         CALL SCANKB           Crunch the input line
 240A 0626B4        760         CALL EDITLN           And edit into program buffer
                    761       $END IF
 240D BD441E        762       DST  @NBD,@BUFFY        Copy next line # to BUFFY
 2410 D6750D        763     $UNTIL @RKEY .EQ. CHRTN   Stop on carriage return
 2413 438D
 2415 4195          764     BR   MAO                  Don't kill the symbol table
```

```
                    766  *
                    767  *       GROM PROGRAMS ENTRY POINT
                    768  *
                    769  GROMBS
2417 8830           770    FETCH @STLN              Get ENLN and STLN data
2419 8831           771    FETCH @STLN+1
241B 8832           772    FETCH @ENLN
241D 8833           773    FETCH @ENLN+1
241F BD4A90         774    DST   *SUBSTK,@FAC       Temporary save of subroutine stac
2422 73
2423 BE738A         775    ST    RSTK+2,@SUBSTK     Replace it with our stack
2426 BD9073         776    DST   @FAC,*SUBSTK       Copy return address to stack
2429 4A
242A 0627E5         777    CALL CHRTA3
242D 874A           778    DCLR @FAC
                    779    $REPEAT
242F A3B04A         780      DADD >6060,RAM(@FAC)     ADD OFFSET TO SCREEN
2432 6060
2434 954A           781      DINCT @FAC
2436 D74A03         782    $UNTIL @FAC .DEQ. >300
2439 00442F
243C 390001         783    MOVE 1 FROM ROM(#H60) TO VDP(1)  TURN ON VDP
243F 012456
2442 BE8089         784    ST    >FF,@GROMFL        SET GROM FLAG FOR FURTHER USE
2445 FF
2446 BD3432         785    DST   @ENLN,@DATA        Select start at first line
2449 A73400         786    DSUB >03,@DATA           Get actual line # address
244C 03
244D 330002         787    MOVE 2 FROM ROM(@DATA) TO @BUFFY Get line # in BUFFY
2450 440000
2453 34
2454 425C           788    BR    MA1H0              PRESCAN AND START EXECUTION
                    789  *
2456 60             790  H60     DATA >60
```

```
                    792  ********************************************************
                    793  **      Subroutine to scan the line typed in, check some
                    794  **      syntax errors, give a token to each character, and
                    795  **      store these series of tokens in text stack on
                    796  **      processor memory.
                    797  **
                    798  **      SCANKB-Scan the input line in the display buffer an
                    799  **      convert to compressed form.
                    800  **      Keywords are converted to tokens and constants
                    801  **      are flagged.
                    802  **
                    803  **      CALL SCANKB
                    804  **           VARW (2 BYTES)     Start of line pointer
                    805  **           VARA (2 BYTES)     End of line pointer
                    806  **           BUFFY (2 BUTES)    Line # or 0 if none
                    807  **           BUFFY+2 (1 BYTE)   Length of line
                    808  **           BUFFY+3            Start of text
                    809  **           STACK (1 BYTE)     Pointer to end of text
                    810  **           TEMPORARY- @BYTE,@CHAT,@VAR5,@FAC,@ARG
                    811  **           CALLS- GETNB,GETCHR,NUMC,STRING
                    812  ********************************************************
2457 860C           813  SCANKB CLR   @BYTE              # OF DIGITS IN LINE #
2459 8744           814         DCLR  @BUFFY             ASSUME NO LINE #
245B BF3803         815         DST   CRNBUF-1,@RAMPTR     POINTER FOR CRUNCH
245E 1F
245F 06282C         816         CALL  GETNB              SKIP LEADING SPACES
2462 665B           817         BS    SCANRT             IF EMPTY LINE
                    818  *
                    819  *       TRY TO GET A LINE #.
                    820  *
                    821  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    822  *    Remove following lines for 99/4A without equation calc
                    823  *    by CAROLYN LEE 2/26/81
                    824  *       $IF @CALCFL .NE. 0 GOTO SC9  Leading #s are not
                    825  **                                    line #'s in calc mode
                    826  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><
2464 C64239         827         CH    NINE,@CHAT         NUMBER?
2467 6471           828         BS    SC9                NO
2469 CA4230         829         CHE   ZERO,@CHAT         NUMBER?
246C 4471           830         BR    SC9                NO
246E 06283A         831         CALL  PTLNE              PUT LINE # IN 2 BYTES
                    832  *
2471 875C           833  SC9    DCLR  @ARG               Assume this is not a symbol
                    834  *                                 and not a tail remark
2473 C5202A         835  SC10   DCH   @VARA,@VARW        END OF LINE?
2476 665B           836         BS    SCANRT             YES
2478 062856         837         CALL  GETNB2             SKIP SPACES
247B 665B           838         BS    SCANRT
247D 860C           839         CLR   @BYTE              # OF BYTES IN THE STRING
247F CA4230         840         CHE   ZERO,@CHAT         NUMERIC?
2482 6497           841         BS    SC12               MAY BE
2484 D6422E         842         CEQ   DOT,@CHAT          DECIMAL POINT?
2487 649C           843         BS    SC13               YES
2489 D64222         844         CEQ   QUOTE,@CHAT        QUOTE?
248C 44A1           845         BR    SC14               NO, SYMBOL OR KEYWORD
```

```
248E 062684   846          CALL STRING              YES, QUOTED STRING
2491 062830   847          CALL GETCHR
2494 052471   848          B    SC9                 TEST NEXT FIELD.
              849  *
2497 C64239   850  SC12    CH   NINE,@CHAT          NUMERIC?
249A 64AF     851          BS   SC20                NO
              852  SC13
249C 0627AF   853          CALL NUMC
249F 44F6     854          BR   SC36                TEST NEXT FIELD.
              855  *                                AFTER SETTING FLAG
              856  *
              857  *       ONE CHARACTER KEYWORD
              858  *
24A1 870C     859  SC14    DCLR @BC
24A3 062850   860          CALL STKCHR
24A6 BD0238   861          DST  @RAMPTR,@STPT       SAVE THE ADDRESS
24A9 062830   862          CALL GETCHR              GET NEXT ONE
24AC 0524FE   863          B    SC37                Stop scanning on end-of-lin
              864  *
              865  *       Test for a symbol or one-character keyword > Z
              866  *       or a regular keyword
              867  *
24AF 062846   868  SC20    CALL PROP                Legal in symbol name?
24B2 44A1     869          BR   SC14                No, a one-character keyword
24B4 870C     870          DCLR @BC                 Init. char. count
24B6 062850   871          CALL STKCHR
24B9 BD0238   872          DST  @RAMPTR,@STPT
24BC 44C3     873          BR   SC32                JUMP INTO LOOP
              874  *
              875  *       Convert a keyword into a token
              876  *
24BE 062850   877  SC30    CALL STKCHR              PUT IT ON TEXT STACK
24C1 900D     878          INC  @BC+1               COUNT KEYWORD CHARACTER
24C3 062830   879  SC32    CALL GETCHR              NEXT ONE
24C6 64DA     880          BS   SC35                END OF LINE
24C8 062846   881          CALL PROP                CHECK LEGALITY OF CHARACTE
24CB 64BE     882          BS   SC30                YES-CONTINUE
24CD D64224   883          CEQ  DOLLAR,@CHAT        DOLLAR ALSO LEGAL IN SYMBO
24D0 44DA     884          BR   SC35
24D2 062850   885          CALL STKCHR
24D5 900D     886          INC  @BC+1
24D7 062830   887          CALL GETCHR
              888  SC35
              889  *                                IF ONE CHAR. THEN A SYMBOL
24DA 8E0D64   890          $IF  @BC+1 .EQ. 0 GOTO SC36
24DD F6
24DE CA0D0A   891          CHE  MAXLEN-1,@BC+1      > MAXIMUM KEYWORD LENGTH?
24E1 44FE     892          BR   SC37                NO, THIS IS A KEYWORD.
24E3 CE0D0E   893          $IF  @BC+1 .LE. 14 GOTO SC36  Not symbol if >15
24E6 44F6
24E8 8744     894  NAMERR  DCLR @BUFFY              DON'T PRINT L.N.
24EA 06284E   895          CALL ERR$$
24ED 2040     896          DATA #MSG4               NAME ERROR
24EF 467E     897          BR   ILL1
              898  *       Jump always
```

```
                899  *                              ENTRY FROM KEYWORD SEARCH IF NOT F
                900  *                              ERROR IF ONE CHARACTER NAME
24F1 D60102     901  SC36A  $IF   @VARO+1 .EQ. 2 GOTO NAMERR
24F4 64E8
24F6 8E5C46     902  SC36   $IF   @ARG .NE. 0 GOTO SCANER YES, IT IS A SYMBOL
24F9 6C
                903  *                                   CHECK IF PREVIOUS A SYMBOL?
                904  *                                   IF YES, ERROR
24FA 905C       905         INC   @ARG                    INDICATE A SYMBOL
24FC 4473       906         BR    SC10                    NEXT ONE
                907  *      Jump always
24FE BD000C     908  SC37   DST   @BC,@VARO               SAVE IT IN A DOUBLE LENGTH AD
2501 9401       909         INCT  @VARO+1                 # OF CHARACTERS IN THE KEYWOR
2503 E20D01     910         SLL   @BC+1,1
2506 330002     911         MOVE 2 FROM ROM(#KEYTAB(BC)) TO @BC KEYWORD
2509 0C285C
250C 0C
                912  *                                   TABLE ADDRESS
250D BD0602     913  KW3    DST   @STPT,@ENPT             WHERE 1ST CHARACTER STORED
2510 32004A     914         MOVE  @VARO FROM ROM(@BC) TO @FAC KEYWORD FROM
2513 00000C
                915  *                                   KEYWORD TABLE
2516 D64AFF     916         CEQ   >FF,@FAC                END OF TABLE?
2519 64F1       917         BS    SC36A                   THIS IS A SYMBOL NAME.
251B A10C00     918         DADD  @VARO,@BC
                919  *                                   1ST CHARACTER ADDRESS OF NEXT
                920  *                                   KEYWORD
251E BE044A     921         ST    FAC,@VARY
                922  KW3A
2521 D49004     923         CEQ   RAM(@ENPT),*VARY        COMPARE THE CORRESPONDING CH
2524 B006
2526 450D       924         BR    KW3                     NO MATCH
2528 9106       925         DINC  @ENPT                   BUMP POINTERS
252A 9004       926         INC   @VARY
252C C93806     927         DCHE  @ENPT,@RAMPTR           END OF KEYWORD?
252F 6521       928         BS    KW3A                    NO
2531 BD3802     929         DST   @STPT,@RAMPTR           WHERE THE TOKEN'LL BE STORED
2534 BC0290     930         ST    *VARY,@STPT                GET THE TOKEN
2537 04
2538 BCB038     931         ST    @STPT,RAM(@RAMPTR)      PUT IN BUFFER
253B 02
253C D60293     932         $IF @STPT .EQ. DATA$ GOTO SCDAT     'DATA'
253F 6604
2541 D6029D     933         $IF @STPT .EQ. CALL$ GOTO SCALL     'CALL'
2544 663E
2546 D6029A     934         $IF @STPT .EQ. REM$ GOTO SCREM      'REM'
2549 65DF
254B C60209     935         CH    EDIT$,@STPT             EDIT COMMAND ?
254E 45BA       936         BR    KW4                     YEP...TREAT IT LIKE ONE
2550 BF5E25     937    DST  #LNTAB,@ARG+2          STORE START OF COMPARISON TABLE
2553 D5
2554 45AA       938    BR   SC50                   AND GO FULL START AHEAD
                939    $REPEAT
2556 D45C02     940      $IF @ARG .EQ. @STPT THEN FOUND CORRECT KEYWORD
2559 45AA
```

```
255B D602B1    941         $IF @STPT .EQ. TO$ THEN COULD BE "TO" IN "FOR"
255E 4568
2560 D6EFFF    942           $IF RAM(-1(RAMPTR)) .NE. GO$ GOTO SC9   IT IS
2563 FF3885
2566 4471
               943         $END IF
2568 062856    944         CALL  GETNB2          GET NEXT NON-BLANK CHARACTER
256B 665B      945         BS    SCANRT          RETURN ON END-OF-LINE
               946  SC45
256D CA4230    947         $IF @CHAT .L. ZERO GOTO SC9 TEST NUMERICAL RANGE
2570 4471
2572 C64239    948         $IF @CHAT .H. NINE GOTO SC9 TOO HIGH FOR THAT
2575 6471
2577 BD4E44    949         DST  @BUFFY,@FAC+4 STORE BUFFY SOMEWHERE
257A 06283A    950         CALL PTLNE         GET LINE # (OR AT LEAST TRY)
257D BD444E    951         DST  @FAC+4,@BUFFY RESTORE GOOD OLD BUFFY
2580 BC4E42    952         ST   @CHAT,@FAC+4  SAVE CURRENT CHARACTER
2583 BE42C9    953         ST   LN$,@CHAT     STORE LINE # TOKEN
2586 062850    954         CALL STKCHR
2589 BC424A    955         ST   @FAC,@CHAT    STORE BINARY LINE NUMBER
258C 062850    956         CALL STKCHR
258F BC424B    957         ST   @FAC+1,@CHAT
2592 062850    958         CALL STKCHR
2595 BC424E    959         ST   @FAC+4,@CHAT
2598 D6422C    960         $IF  @CHAT .NE. COMMA GOTO SC9 DO ALLOW MULTIPLE RE
259B 4471
259D BE42B3    961         ST   COMMA$,@CHAT  STORE COMMA TOKEN FOR SEPARATOR
25A0 062850    962         CALL STKCHR        PUT TOKEN IN CRUNCH BUFFER
25A3 06282C    963         CALL GETNB         GET NEXT NON-BLANCK
25A6 665B      964         BS   SCANRT        IF END OF INPUT BUFFER
25A8 456D      965         BR   SC45          AND CONTINUE
               966       $END IF              END HANDLING OF LINE NUMBERS
               967  SC50
25AA 330001    968       MOVE 1 FROM ROM(@ARG+2) TO @ARG GET TOKEN VALUE
25AD 5C0000
25B0 5E
25B1 915E      969         DINC @ARG+2        AND GET SET FOR THE NEXT ONE
25B3 D65CFF    970      $UNTIL @ARG .EQ. >FF   STOP AT END OF TABLE
25B6 4556
25B8 4471      971      BR   SC9              NO, CONTINUE
               972  KW4
25BA D73803    973      $IF @RAMPTR .DNE. CRNBUF GOTO SCANER TOKEN MUST BE FIRS
25BD 20466C
25C0 8F4443    974      $IF @BUFFY .DNE. 0 GOTO ILLST  CAN'T BE A PROGRAM LINE
25C3 46
25C4 D60206    975      $IF @STPT .EQ. OLD$ GOTO SCDAT   'OLD' ? -> DATA SCAN
25C7 6604
25C9 D60208    976      $IF @STPT .EQ. SAVE$ GOTO SCDAT  JUST LIKE 'SAVE'
25CC 6604
25CE D60203    977      $IF @STPT .EQ. LIST$ GOTO SCLIST Special scan command
25D1 65E9
25D3 465B      978      BR   SCANRT           COMMAND OK, STOP SCAN
               979  LNTAB
25D5 B081A1    980      DATA THEN$,ELSE$,SUB$,TO$,GOSUB$
25D8 B187
```

```
- 25DA 868E8F    981     DATA GOTO$,BREAK$,UNBRE$,RESTO$
  25DD 94
  25DE FF        982     DATA >FF
                 983   ***************************************************
                 984   *       SCAN 'REM' STATEMENT
                 985   *
                 986   ***************************************************
  25DF 062850    987   SCREM   CALL STKCHR
  25E2 062830    988           CALL GETCHR             Get next character
  25E5 45DF      989           BR   SCREM              If not done
  25E7 465B      990           BR   SCANRT             If done
                 991   *       Jump always
                 992   ***************************************************
                 993   *       SCAN 'LIST' STATEMENT
                 994   ***************************************************
                 995   SCLIST
  25E9 062856    996     CALL GETNB2                   Get first non-blank character
  25EC D64222    997     $IF @CHAT .EQ. QUOTE THEN Devicename
  25EF 4601
  25F1 062684    998       CALL STRING                 Collect the quoted string
  25F4 06282C    999       CALL GETNB                  Get character behind string
  25F7 665B     1000       BS   SCANRT                 Return from scan if blank
  25F9 D6423A   1001       $IF @CHAT .NE. COLON GOTO SCANER
  25FC 466C
  25FE 06282C   1002       CALL GETNB                  Get next non-blank
                1003     $END IF
- 2601 05265B   1004     B     SCANRT                  Get out (MUST BE LONG BRANCH! - GE
                1005   ***************************************************
                1006   *       SCAN 'DATA' STATEMENT
                1007   ***************************************************
                1008   *
  2604 BE4A01   1009   SCDAT  ST    1,@FAC            Non-INPUT scan for UNQSTR
  2607 9320     1010          DDEC @VARW              Get back to previous character
  2609 460E     1011          BR   SCDATA             And continue
                1012   SCDAT$
  260B BC0073   1013     ST    @SUBSTK,@VAR0          Save subroutine ^ for errors
                1014   SCDATA
  260E 8611     1015     CLR   @VAR6                   Counter for data fields
                1016   SCD01
  2610 06282C   1017     CALL GETNB                    Skip spaces
  2613 665B     1018     BS   SCANRT                   End of line
                1019     $REPEAT
  2615 D6422C   1020       $IF @CHAT .EQ. COMMA THEN
  2618 4624
  261A BE42B3   1021         ST   COMMA$,@CHAT         Yes, store comma token
  261D 062850   1022         CALL STKCHR
  2620 9011     1023         INC  @VAR6                Count fields for "INPUT"
  2622 4610     1024         BR   SCD01                (UNCONDITIONAL)
                1025       $END IF
  2624 D64222   1026       $IF @CHAT .EQ. QUOTE THEN
  2627 462E
  2629 062684   1027         CALL STRING               Yes, quoted string
  262C 4610     1028         BR   SCD01
                1029       $END IF
  262E BE132C   1030       ST   COMMA,@VAR8            UNQSTR will stop on a comma
```

```
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
             *   FOLLOWING LINE IS ADDED FOR LOWERCASE CHAR. SET IN
             *   99/4A 3/3/81
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
2631 868082      CLR  @ALLUP           Do not translate lower to upper
2634 062854      CALL UNQSTR           Put the unquoted string on stack
2637 D6422C    $UNTIL @CHAT .NE. COMMA  End of line exit
263A 6615
263C 465B      BR   SCANRT             But do count EOL commas
             *       Jump always
             ************************************************************
             *       SCAN CALL STATEMENT
             ************************************************************
263E 8E4266  SCALL  $IF @CHAT .EQ. 0 GOTO SCANER
2641 6C
2642 062856         CALL GETNB2             Skip spaces
2645 BE4A01         ST   1,@FAC             Non-INPUT scan for UNQSTR
2648 BE1328         ST   LPAR,@VAR8         UNQSTR will stop on a left
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
             *     THE FOLLOWING LINE IS FOR ADDING LOWERCASE CHAR. SE
             *     IN 99/4A 3/3/81
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
264B BE8082         ST   1,@ALLUP           Translate lower to upper
264E 01
264F 062854         CALL UNQSTR             Put the name on the stack
2652 665B           BS   SCANRT
2654 9320           DDEC @VARW
2656 06282C         CALL GETNB              Skip spaces
2659 44A1           BR   SC14               IF '('
             *
             *      End of line, put the length in length byte.
             *
265B 8642    SCANRT CLR  @CHAT              PUT END OF LINE TOKEN IN
265D 062850         CALL STKCHR
2660 BD0238         DST  @RAMPTR,@STPT
2663 A70203         DSUB CRNBUF-1,@STPT   GET LINE LENGTH
2666 1F
2667 BC4203         ST   @STPT+1,@CHAT      STORE LENGTH IN LENGTH BYTE
266A 4850           BR   STKCHR             STORE LENGTH AND RETURN
             *
             *      SCAN ERROR - PRINT "SYNTAX ERROR"
             *
             SCANER
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
             *   Remove following lines for 99/4A without equation cal
             *   by CAROLYN LEE 2/26/81
             * $IF @CALCFL .EQ. 0 THEN If not calc mode
             * $END IF     (at line ERR$1-1)
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
266C DA8088      $IF .BIT5 @FLAG .EQ. 0 THEN
266F 204677
2672 BC7300        ST   @VAR0,@SUBSTK   RESTORE SUBROUTINE STACK ENTR
2675 46CE          BR   RTNSET          RETURN WITH CONDITION SET
                 $END IF
2677 8744        DCLR @BUFFY                W/o a line number
```

```
                    1081
                    1082  ERR$1
2679 06284E         1083    CALL ERR$$
267C 202C           1084    DATA #MSG1                SYNTAX ERROR
                    1085  ILL1
267E A54648         1086    DSUB @CURINC,@CURLIN      Back up one line
2681 0521A1         1087    B    MAIN2
                    1088  *
                    1089  ***************************************************************
                    1090  **      Subroutine to verify a quoted string and store it
                    1091  **      in the crunch buffer.
                    1092  ***************************************************************
                    1093  STRING
2684 BE42C7         1094    ST   STRIN$,@CHAT         Push string token
2687 062850         1095    CALL STKCHR
268A 0626AB         1096    CALL GETLNB               Set up length byte
268D 062830         1097    CALL GETCHR
2690 666C           1098    BS   SCANER               End of line without closing " !!!
2692 D64222         1099    $WHILE @CHAT .NE. QUOTE Just keep on pushing
2695 669E
                    1100  STRI$1
2697 062852         1101      CALL PUSHCH             Push the character
269A 666C           1102      BS   SCANER            End of line before end of string !
269C 4692           1103    $SEND WHILE               Make it a tight loop
269E 062830         1104    CALL GETCHR               Check next character for second "
26A1 66AA           1105    BS   STRI$2               End-of-line
26A3 D64222         1106    $IF @CHAT .EQ. QUOTE GOTO STRI$1 Found a second "
26A6 6697
26A8 9320           1107    DDEC @VARW                Backup if no two ""s
                    1108  STRI$2
26AA 00             1109    RTN
                    1110  *
                    1111  *
                    1112  *
                    1113  *
                    1114  *       Get length byte address for a constant (strings and
                    1115  *       numerical contants), and set it to zero
                    1116  *
                    1117  GETLNB
26AB 9138           1118    DINC @RAMPTR              Move to length byte position
26AD BD0238         1119    DST  @RAMPTR,@STPT        And save the address in STPT
26B0 86B038         1120    CLR  RAM(@RAMPTR)         Initialize the length at 0
26B3 00             1121    RTN
                    1122  *
                    1123  *
```

```
             *
             *       A line with only a line #
             *       Delete the line or AUTONUM off?
             *
             EDITLN
26B4 064012    CALL CLSALL
26B7 06222B    CALL KILSYM
26BA 8602      CLR  @STPT                 Restore STPT
26BC BC0342    ST   @CHAT,@STPT+1
26BF D64201    $IF @CHAT .NE. 1 GOTO MA$ Something besides line #
26C2 4703
26C4 DA8088    $IF .BITO @FLAG .EQ. 1 THEN AUTONUM mode on
26C7 0166D2
26CA B28088      RB   @FLAG,0             Reset AUTONUM mode
26CD FE
             EDIT$1
             RTNSET
26CE D40000      CEQ  @0,@0               Set condition bit
26D1 01          RTNC                     And return
               $END IF
               $
26D2 D53032    $IF @STLN .DEQ. @ENLN GOTO EDIT$1 If nothing in memory
26D5 66CE
               $
               $    Delete the line # from line-#-buffer.
               $    Delete the text from program text area.
               $
26D7 06283E    CALL SEETWO                Try to find the given line #
26DA 46CE      BR   EDIT$1                Return if not found
26DC 062838    CALL DELREP                Remove it's text from program ar
               $
               $    Delete the 4 bytes from the line # table
               $
26DF BD062E    DST  @EXTRAM,@VARY+2       Pointer to line pointer
26E2 9106      DINC @VARY+2               Advance to last byte of entry
26E4 972E      DDECT @EXTRAM              Point to first byte of entry
26E6 BD002E    DST  @EXTRAM,@VARO
26E9 9300      DDEC @VARO                 Last byte of next line entry
               $                          Move down 4 bytes from here
26EB A52E30    DSUB @STLN,@EXTRAM         # of bytes to move down
26EE 8F2E66    $IF @EXTRAM .DNE. 0 THEN
26F1 F8
26F2 BD5C2E      DST  @EXTRAM,@ARC        Put in arg for MVDN
26F5 06236D      CALL MVDN                Move one byte at a time
               $END IF
26F8 A33000    DADD >04,@STLN     NEW STARTING ADD. OF LINE-#-BUFFER
26FB 04
26FC C53070    $IF @STLN .DH. @>70 GOTO MAINA   Deleted the one and on
26FF 618C
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
             *       CHANGED FROM:
             * RTN                        Return to sender
             *       BY STAN HUME ON 06/19/80 TO RESET SYMPTR AFTER EDI
             *       EDITING A LINE SO IMPERATIVELY CREATED VARIABLE
             *       WON'T DESTROY THE LINE NUMBER TABLE
```

```
                1173  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
2701 422B       1174    BR    KILSYM                Return to sender w/SYMPTR reset
                1175    $
                1176    $     Copy the line # and program text to RAM.
                1177    $
                1178  MA$
2703 BD0444     1179    DST   @BUFFY,@VARY          2 bytes of line #
2706 BD0632     1180    DST   @ENLN,@VARY+2         Last RAM address of line-#-buffer
2709 BD2E32     1181    DST   @ENLN,@EXTRAM         Last RAM address of line-#-buffer
270C D53032     1182    $IF @STLN .DEQ. @ENLN GOTO EDIT$5 1st text on RAM?
270F 6749
2711 912E       1183    DINC @EXTRAM                Get line
                1184    $
                1185  EDIT$3
2713 A72E00     1186    DSUB 4,@EXTRAM
2716 04
2717 D544B0     1187    $IF @BUFFY .DEQ. RAM(@EXTRAM) GOTO MA$5 Compare 2 line #'
271A 2E678A
271D BF2A00     1188    DST   4,@VARA               For MEMFUL
2720 04
2721 09472B     1189    $IF .H. THEN                New line # is greater
2724 D52E30     1190      $IF @EXTRAM .DEQ. @STLN GOTO EDIT$6 End of RAM line # b
2727 674D
2729 4713       1191      BR    EDIT$3
                1192    $END IF
                1193    $
                1194    $     Insert the current line # in RAM line # buffer
                1195    $     Move the higher line refs up. The program text
                1196    $     will be written behind the current text buffer
                1197    $
272B BD5C2E     1198    DST   @EXTRAM,@ARG          2nd byte of the length address
272E A35C00     1199    DADD 4,@ARG                 Point to last byte
2731 04
2732 A55C30     1200    DSUB @STLN,@ARG             # of bytes in between
2735 BD1630     1201    DST   @STLN,@VAR9           Copy old start address of line #'s
2738 062352     1202    CALL MEMFUL                 Check for memory full
273B A13002     1203    DADD @STPT,@STLN
273E 345CB0     1204    MOVE @ARG FROM RAM(@VAR9) TO RAM(@STLN) Move the contents
2741 30B016
2744 BD0632     1205    DST`  @ENLN,@VARY+2
2747 4756       1206    BR    EDIT$8                to new RAM area
                1207    $
                1208    $     Write the line # behind last line # in line # buffer
                1209    $
                1210  EDIT$5
2749 BF2A00     1211    DST   >03,@VARA             Subtract >03 from STLN to get
274C 03
                1212  EDIT$6
274D 062352     1213    CALL MEMFUL                 Pointer to new start addr. of #'s
                1214    $                           Check for memory full
2750 A13002     1215    DADD @STPT,@STLN
2753 BD2E30     1216    DST   @STLN,@EXTRAM
                1217  EDIT$8
2756 A50602     1218    DSUB @STPT,@VARY+2
2759 9106       1219    DINC @VARY+2                Pointer to 1st token address
```

```
275B 350004  1220    MOVE 4 FROM @VARY TO RAM(@EXTRAM) Copy line # and point
275E B02E04
             1221    $
             1222    $     WRITE THE LINE STATEMENT AFTER THE LAST TEXT ON RA
             1223    $
             1224  MA$4
2761 BD5C32  1225    DST  @ENLN,@ARG          Highest address for line # buffe
2764 A55C30  1226    DSUB @STLN,@ARG          Compute total length of line # b
2767 915C    1227    DINC @ARG                Add one for extra offset
2769 BD1630  1228    DST  @STLN,@VAR9         Old start addr. of line # buffer
276C 9102    1229    DINC @STPT               Point to next free byte in RAM
276E A53002  1230    DSUB @STPT,@STLN         New entry to line # table
2771 A53202  1231    DSUB @STPT,@ENLN
2774 345CB0  1232    MOVE @ARG FROM RAM(@VAR9) TO RAM(@STLN) Move STPT+1 byt
2777 30B016
277A 9302    1233    DDEC @STPT               Update length of text
277C BCEFFF  1234    ST   @STPT+1,RAM(-1(VARY+2)) Copy length for program se
277F FF0603
2782 340280  1235    MOVE @STPT FROM RAM(CRNBUF) TO RAM(@VARY+2) Move text
2785 06A320
             1236  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
             1237  *       CHANGED FROM:
             1238  * RTN
             1239  *       BY STAN HUME 06/19/80 TO RESET SYMPTR AFTER EDITIN
             1240  *       A LINE SO AN IMPERATIVELY CREATED VARIABLE WILL NO
             1241  *       DESTROY THE LINE NUMBER TABLE
             1242  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
2788 4228    1243    BR   KILSYM              Return to caller w/SYMPTR reset
             1244  *
             1245  **      REPLACE THE OLD TEXT.
             1246  *
             1247  MA$5
             1248  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
             1249  *       CHANGED FROM:
             1250  * CALL COCULE              Compute length of old entry
             1251  *       BY STAN HUME 06/19/80 TO PUT COCULE IN-LINE TO SAV
             1252  *       BYTES
             1253  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
278A BD5CE0  1254    DST  RAM(2(EXTRAM)),@ARG  Pointer to first token
278D 022E
278F B25C7F  1255    RB   @ARG,7              Reset possible breakpoint
2792 935C    1256    DDEC @ARG                Back up to length byte in line
2794 8608    1257    CLR  @VARC
2796 BC09B0  1258    ST   RAM(@ARG),@VARC+1 Create a double length in VARC
2799 5C
             1259  *<><><><><><><><><CHANGE ENDS HERE<><><><><><><><><><><>
279A BD2A08  1260    DST  @VARC,@VARA         Store for MEMFUL
279D 832A    1261    DNEG @VARA               And get length difference
279F 062352  1262    CALL MEMFUL              Check for memory full
27A2 A1302A  1263    DADD @VARA,@STLN
27A5 062838  1264    CALL DELREP              Remove old text (same line #)
27A8 972E    1265    DDECT @EXTRAM            Correct pointer
27AA BD0632  1266    DST  @ENLN,@VARY+2       Compute new last address of #'s
27AD 4756    1267    BR   EDIT$8              Get text in program segment
             1268  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
```

```
*       CHANGED FROM:
*COCULE,
* DST  RAM(2(EXTRAM)),@ARG  Pointer to first token
* RB   @ARG,7              Reset possible breakpoint
* DDEC @ARG                Back up to length byte in line
* CLR  @VARC
* ST   RAM(@ARG),@VARC+1 Create a double length in VARC
* RTN
*       BY STAN HUME 06/19/80 TO PUT IN-LINE ABOVE TO SAVE
*       BYTES
*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
```

```
                1281  ********************************************************
                1282  **     Subroutines to recognize numbers and store them
                1283  ********************************************************
                1284  *
                1285  **     The main subroutine recognizes "." and "E". In
                1286  **     between CALLs are being made to NUM1 and NUM2 to
                1287  **     recognize strings of numerical characters. NUM1
                1288  **     allows optional reconition of +/- signs preceding
                1289  **     the numerical strings
                1290  *
                1291  NUMC
27AF 9138       1292    DINC @RAMPTR              Update the memory pointer
27B1 BEB038     1293    ST   NUM$,RAM(@RAMPTR) Push token for numeric cons.
27B4 C8
27B5 0626AB     1294    CALL GETLNB               Set up length byte parameters
27B8 0627CB     1295    CALL NUMC$1               Allow recognition of +/- and num
27BB D6422E     1296    $IF @CHAT .EQ. DOT THEN Also allow decimal fraction
27BE 47C3
27C0 0627D5     1297      CALL NUMC$2             Store "." and recognize
                1298    $END IF
27C3 D64245     1299    $IF @CHAT .NE. E GOTO NUMC$4 Check exponential part
27C6 47E2
27C8 062852     1300    CALL PUSHCH               Push "E" character
                1301    $
                1302  *
                1303  *
                1304  NUMC$1
27CB D6422B     1305    $IF @CHAT .NE. PLUS THEN Allow "+" and "-" signs
27CE 67D5
27D0 D6422D     1306      $IF @CHAT .NE. MINUS GOTO NUMC$3
27D3 47D8
                1307    $END IF
                1308  NUMC$2
27D5 062852     1309    CALL PUSHCH               Push character in memory
                1310  NUMC$3
27D8 CA4230     1311    $IF @CHAT .HE. ZERO THEN Could be numeric
27DB 47E2
27DD CA423A     1312      $IF @CHAT .L. NINE+1 GOTO NUMC$2 It is !!!!!
27E0 47D5
                1313    $END IF
                1314  NUMC$4
27E2 00         1315    RTN
                1316  *
```

```
                *
                *       The color tables could be represented
                *       by the following DATA statements
                *
                *       DATA >D0,>00,>00,>00,>00,>00,>00,>00
                *       DATA >00,>00,>00,>00,>00,>00,>00,>17
                *       DATA >17,>17,>17,>17,>17,>17,>17,>17
                *       DATA >17,>17,>17,>17,>17,>17,>17,>17
                *
                CHRTA2
27E3 0780               ALL   :  :+OFFSET
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *   Remove following line for 99/4A without equation calc.
                *   by CAROLYN LEE 2/26/81
                *CHRTA3 CLR  @CALCFL             Set BASIC mode
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
27E5 BF80C0     CHRTA3 DST   >3567,@>C0         INIT RANDOM NUMBER
27E8 3567
27EA 310010             MOVE 16 FROM ROM(#SPCCHR) TO RAM(>3F0)
27ED A3F021
27F0 5C
27F1 BF4A04     CHRTAB DST   >400,@FAC
27F4 00
27F5 060018             CALL CHAR2$
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *    ADD FOLLOWING TWO LINES FOR LOWERCASE CHAR. SET FOR
                *    99/4A BY CAROLYN LEE 3/3/81
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
27F8 BF4A06             DST >600,@FAC
27FB 00
27FC 06004A             CALL CHAR3$
27FF 0407               BACK 7               BORDER COLOR
2801 BFA300             DST  >D000,RAM(>300)
2804 D000
2806 35000D             MOVE 13 FROM RAM(>301) TO RAM(>302)
2809 A302A3
280C 01
280D BEA30F             ST   >17,RAM(>30F)
2810 17
2811 350010             MOVE 16 FROM RAM(>30F) TO RAM(>310)
2814 A310A3
2817 0F
2818 390003             MOVE 3 FROM ROM(#VDPREG) TO VDP(2)
281B 02216C
281E 00                 RTN
                *
                *
                        END
```

ERRORS= 0

LENGTH= 2079 (>081F)

375 SYMBOLS USED

SYMBOL	VALUE	DEF	REFERENCE TABLE
A	0041	294	
ABS$	00CB	240	
ALLUP	0082	125	1035 1051
APPEN$	00F9	273	
ARG	005C	117	466 687 732 733 734 736 737 758 833 902 905 937 940 968 968 969 970 1162 1198 1199 1200 1204 1225 1226 1227 1232 1254 1255 1256 1258
ATN	0032	56	
ATN$	00CC	241	
AUTON	2840	34	579 599
BACK	0008	282	
BACKS	005C	298	
BASE	0043	112	522
BASE$	00F1	265	
BC	000C	84	859 870 878 886 890 891 893 908 910 911 911 914 918
BKGD	0020	150	
BREAK	0003	281	
BREAK$	008E	178	981
BUFFY	0044	113	461 483 537 544 570 662 762 787 814 894 949 951 974 1080 1179 1187
BYTE	000C	83	84 699 813 839
CALL$	009D	193	933
CB	0008	1	
CCPADR	0008	79	730 735
CCPPTR	0006	76	715 727
CHAR	0004	1	
CHAR2$	0018	43	1337
CHAR3$	004A	44	1343
CHAT	0042	111	486 494 827 829 840 842 844 850 883 947 948 952 953 955 957 959 960 961 997 1001 1020 1021 1026 1037 1043 1060 1064 1094 1099 1106 1133 1134 1296 1299 1305 1306 1311 1312
CHR$$	00D6	251	
CHRTA2	27E3	1327	425
CHRTA3	27E5	1334	777
CHRTAB	27F1	1336	353
CHRTN	000D	148	485 752 763
CIRCU$	00C5	233	
CLOSE$	00A0	196	
CLOSEB	005D	299	
CLSALL	4012	37	428 538 657 1130
CNS	0014	42	
COLON	003A	288	1001
COLON$	00B5	217	
COMMA	002C	307	960 1020 1030 1037
COMMA$	00B3	215	961 1021
CONC$	00B8	220	
COS	002C	53	
COS$	00CD	242	
CPL	0010	40	
CRNBUF	0320	133	495 496 499 815 973 1063 1235
CRNEND	03BE	134	
CRUADD	0080	9	
CURINC	0048	115	453 581 1086

SYMBOL	VALUE	DEF	REFERENCE TABLE
CURLIN	0046	114	453 454 461 466 580 1086
CURSOR	007E	310	
DATA	0034	106	429 785 786 787
DATA$	0093	183	932
DEF$	0089	173	
DELET$	0099	189	
DELREP	2838	33	1150 1264
DIM$	008A	174	
DISO	2842	21	467
DISPL$	00A2	198	
DISPLA	0009	1	
DISPY	0002	145	
DIVI$	00C4	232	
DOLLAR	0024	303	883
DOT	002E	309	842 1296
DSRFLG	0017	93	720
DWNARR	000B	147	748
E	0045	295	1299
EDGECH	007F	311	
EDIT$	0009	158	495 935
EDIT$1	26CE	1137	1143 1149
EDIT$3	2713	1185	1191
EDIT$5	2749	1210	1182
EDIT$6	274D	1212	1190
EDIT$8	2756	1217	1206 1267
EDITLN	26B4	1129	489 760
EDT$$0	2388	705	486
EDT$00	2399	714	463 709
EDT$01	23FA	751	742 745
EDT$02	23FF	754	746
ELSE$	0081	165	980
END$	008B	175	
ENDSCR	02FE	132	
ENLN	0032	105	433 434 460 529 535 598 598 600 615 634 651 706 750 772 773 785 1143 1180 1181 1182 1205 1225 1231 1266
ENPT	0006	77	913 923 925 927
EQUAL	003D	291	
EQUAL$	00BE	226	
ERCODE	007C	68	
ERR$$	284E	27	573 606 663 674 711 895 1083
ERR$1	2679	1082	697 699 701
ERR$2	2392	710	
EXEC1	4D0C	14	571
EXIT	0020	47	
EXP	0028	51	
EXP$	0076	119	
EXP$$	00CE	243	
EXTRAM	002E	103	568 569 719 1154 1156 1157 1160 1161 1162 1181 1183 1186 1187 1190 1198 1216 1220 1254 1265
FAC	004A	116	600 601 602 603 604 610 614 615 617 619 620 622 623 623 624 624 626 627 628 629 631 631 632 633 634 635 637 637 638 641 642 643 645 646 648 648 649 650 651 774 776 778 780 781 782 914 916 921 949 951

SYMBOL	VALUE	DEF	REFERENCE TABLE
			952 955 957 959 1009 1045 1336 1342
FIGURE	0002	1	
FIXED$	00FA	274	
FLAG	0088	127	427 438 439 450 455 484 559 582 742 1076 1135 1136
FOR$	008C	176	
FORW	0009	283	
GETCHR	2830	25	847 862 879 887 989 1097 1104
GETLN	283C	24	698
GETLNB	26AB	1117	355 1096 1294
GETNB	282C	23	556 696 816 963 999 1002 1017 1055
GETNB2	2856	31	700 837 944 996 1044
GO$	0085	169	942
GOSUB$	0087	171	980
GOTO$	0086	170	981
GREAT	003E	292	
GREAT$	00C0	228	
GROMBS	2417	769	347
GROMFL	0089	129	784
GSCAN	0022	48	
H60	2456	790	783
IF$	0084	168	
ILL1	267E	1085	351 665 676 713 897
ILLST	2346	660	529 535 588 592 706 974
INPUT$	0092	182	
INT$	00CF	244	
KEYBD	0074	63	441
KEYTAB	285C	17	911
KILSYM	222B	513	408 436 1131 1174 1243
KW3	250D	913	924
KW3A	2521	922	928
KW4	25BA	972	936
LEN$	00D5	250	
LESS	003C	290	
LESS$	00BF	227	
LET$	008D	177	
LIMIT	0012	149	
LINK1	4D1A	38	341
LIST	4018	13	530
LIST$	0003	154	977
LLIST	282E	19	722
LN$	00C9	238	626 953
LNBUF	0036	107	
LNTAB	25D5	979	937
LOG	002A	52	
LOG$	00D0	245	
LPAR	0028	304	1046
LPAR$	00B7	219	
M$EXEC	4D00	3	12 14 15 38
M$FLMG	4000	5	13 35 36 37
M$PSCN	2828	4	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
MA$	2703	1178	1134
MA$4	2761	1224	
MA$5	278A	1247	1187

SYMBOL	VALUE	DEF	REFERENCE TABLE
MA0	2195	437	348 764
MA1	21A4	446	490 494 575 608 653
MA10	21BC	459	584
MA16	21D6	473	451 456
MA16A	21D9	474	469
MA1G	2245	528	503
MA1H	224D	534	500
MA1H0	225C	546	788
MA1H1	2268	555	502
MA1H10	22AA	597	507
MA1H7	228C	579	505
MA1H8	229F	588	508
MA1H9	22A7	593	506
MAEXIT	2342	656	504
MAIN	216F	423	401 501
MAIN1	2192	435	352 491
MAIN2	21A1	444	1087
MAINA	218C	433	1166
MAXET	0005	153	
MAXLEN	000B	151	891
MEMFUL	2352	669	1202 1213 1262
MINUS	002D	308	1306
MINUS$	00C2	230	
MPO	2266	550	495
MSG0	2022	361	
MSG1	202C	362	1084
MSG11	207D	370	
MSG12	2094	371	
MSG13	20A1	372	
MSG14	2064	367	
MSG15	20AF	373	
MSG17	20BD	375	664
MSG18	20F9	379	
MSG19	213F	389	
MSG2	20BD	374	
MSG20	2108	383	
MSG21	2113	384	
MSG22	211D	385	
MSG23	2128	386	
MSG24	2134	387	
MSG3	2064	366	
MSG4	2040	363	896
MSG5	2049	364	675
MSG6	2055	365	574
MSG7	2064	368	
MSG8	206E	369	
MSGBLN	20D9	377	607 712
MSGBRK	20E9	378	
MSGFST	20CB	376	426
MULT$	00C3	231	
MVDN	236D	686	354 1163
MVDN1	2369	684	688
NAMERR	24E8	894	901
NBC	0014	91	580 610 612 628 646 719 744 745 749 750 755

SYMBOL	VALUE	DEF	REFERENCE TABLE
NBD	001E	98	561 603 632 649 755 762
NEXT$	0096	186	
NINE	0039	287	827 850 948 1312
NLNADD	02E2	131	426 446 474 583
NUM$	00C8	237	620 1293
NUMBE$	00FD	277	
NUMC	27AF	1291	853
NUMC$1	27CB	1304	1295
NUMC$2	27D5	1308	1297 1312
NUMC$3	27D8	1310	1306
NUMC$4	27E2	1314	1299
OFFSET	0060	152	310 311 720 1328
OLD	4016	36	593
OLD$	0006	155	975
ON$	009B	191	
OPEN$	009F	195	
OPENB	005B	297	
OPTIO$	009E	194	
OUTPU$	00F7	271	
PABPTR	0004	73	
PERMA$	00FB	275	
PGMCH	001B	59	497
PLUS	002B	306	1305
PLUS$	00C1	229	
POP	0005	1	
POPS	001E	46	
POS$	00D9	254	
PRGLST	2828	16	551
PRINT$	009C	192	
PROP	2846	22	868 881
PRTNFN	00C0	10	
PTLNE	283A	18	831 950
PUSH	001C	45	
PUSHCH	2852	29	1101 1300 1309
PWR	0024	49	
QUEST	003F	293	
QUOTE	0022	302	844 997 1026 1099 1106
RAM	0001	1	426 474 495 499 523 547 558 569 619 620 623 626 631 631 637 642 648 686 686 755 780 923 931 942 1120 1187 1204 1204 1220 1232 1232 1234 1235 1235 1254 1258 1293 1335 1345 1346 1346 1347 1348 1348
RAMPTR	0038	108	815 861 872 927 929 931 942 973 1062 1118 1119 1120 1292 1293
RANDO$	0095	185	
RANDOM	0078	65	
READ$	0097	187	
READL1	2858	32	739
READLN	2832	26	475
RECLEN	0007	78	721 727
REM$	009A	190	934
RESEQ$	0007	156	
RESTO$	0094	184	981
RETUR$	0088	172	
RKEY	0075	64	485 743 748 752 763

SYMBOL	VALUE	DEF	REFERENCE TABLE
RND$	00D7	252	
ROM	C00A	1	426 783 787 911 914 968 1335 1349
RPAR	0029	305	
RPAR$	00B6	218	
RPL	0012	41	
RSTK	00B8	128	424 444 775
RTNSET	26CE	1138	1078
RUN	4D10	15	536
S$EDIT	2377	695	509
SAVE	4014	35	589
SAVE$	0008	157	976
SAVEPC	03EC	135	523 547 558 569
SC10	2473	835	906
SC12	2497	850	841
SC13	249C	852	843
SC14	24A1	859	845 869 1056
SC20	24AF	868	851
SC30	24BE	877	882
SC32	24C3	879	873
SC35	24DA	888	880 884
SC36	24F6	902	854 890 893
SC36A	24F1	901	917
SC37	24FE	908	863 892
SC45	256D	946	965
SC50	25AA	967	938
SC9	2471	833	828 830 848 942 947 948 960 971
SCALL	263E	1043	933
SCANER	266C	1069	350 557 902 973 1001 1043 1098 1102
SCANKB	2457	813	482 759
SCANRT	265B	1060	817 836 838 945 964 978 990 1000 1004 1018 1038 1053
SCDO1	2610	1016	1024 1028
SCDAT	2604	1009	932 975 976
SCDAT$	260B	1012	349
SCDATA	260E	1014	1011
SCLIST	25E9	995	977
SCREM	25DF	987	934 989
SCROLL	4D00	12	465 473 560 661 723
SEETWO	283E	20	462 708 1148
SEG$$	00D8	253	
SEMIC	003B	289	
SEMIC$	00B4	216	
SEQ$1	22C3	605	611 612
SEQ$2	22E8	621	619
SEQ$3	2318	636	631
SEQUE$	00F6	270	
SGN$$	00D1	246	
SIN	002E	54	
SIN$	00D2	247	
SPACE	0020	301	
SPCCHR	215C	412	1335
SPRITE	0006	1	
SQR	0026	50	
SQR$	00D3	248	
SREF	001C	97	

```
SYMBOL  VALUE  DEF   REFERENCE TABLE

SSEP$   0082   166
STACK   0072    61
START   0012    89
STEP$   00B2   214
STKCHR  2850    28     860  871  877  885  954  956  958  962  987 1032
                      1061 1065 1095
STLN    0030   104     434  460  514  529  535  588  598  601  629  645
                       671  672  673  706  745  770  771 1143 1160 1165
                      1166 1182 1190 1200 1201 1203 1204 1215 1216 1226
                      1228 1230 1232 1263
STOP$   0098   188
STPT    0002    72     670  861  872  913  929  930  931  932  933  934
                       935  940  941  975  976  977 1062 1063 1064 1119
                      1132 1133 1203 1215 1218 1229 1230 1231 1233 1234
                      1235
STR$$   00DB   256
STREND  001A    96     521
STRI$1  2697  1100    1106
STRI$2  26AA  1108    1105
STRIN$  00C7   235     619 1094
STRING  2684  1093     846  998 1027
STRSP   0018    95     519  520  521
STVSPT  0024   100     432  549
SUB$    00A1   197     980
SUBSTK  0073    62     424  444  774  775  776 1013 1077
SYMPT   0800   141
SYMPTR  0040   110     514  515  516  519
SYMTAB  003E   109     518
TAB$    00FC   276
TABLE   000B     1
TAN     0030    55
TAN$    00D4   249
THEN$   00B0   212     980
TIMER   0079    66
TO$     00B1   213     941  980
TOP     0003     1
TRACE$  0090   180
TREM$   0083   167
UNBRE$  008F   179     981
UNLN    005F   300
UNQST$  00C8   236     237
UNQSTR  2854    30    1036 1052
UNTRA$  0091   181
UPARR   000A   146     743
UPDAT$  00F8   272
USER    214D   401     339
VAL     00DA   255
VAR0    0000    70     685  686  901  908  909  914  918 1013 1077 1157
                      1158
VAR1    000D    85
VAR2    000B    82
VAR4    000E    86
VAR5    0010    87
VAR6    0011    88    1015 1023
VAR8    0013    90    1030 1046
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
VAR81	002C	102	496
VAR9	0016	92	1201 1204 1228 1232
VARA	002A	101	670 671 673 735 736 835 1188 1211 1260 1261 1263
VARC	0008	80	1257 1258 1260
VARD	0009	81	
VARV	0001	71	
VARW	0020	99	446 468 476 478 583 729 732 740 741 835 1010 1054 1107
VARY	0004	74	684 686 921 923 926 930 1154 1155 1179 1180 1205 1218 1219 1220 1234 1235 1266
VARZ	0005	75	
VDP	000C	1	783 1349
VDPREG	216C	416	1349
VDPSTS	007B	67	
VEL	0007	1	
VPOP	0018	58	
VRAMVS	06F8	140	431 548 672
VSPTR	006E	118	431 432 548 549
X1	0002	143	
XPT	000E	1	
Y1	0017	144	
YPT	000D	1	
Z	005A	296	
ZERO	0030	286	829 840 947 1311

TOTAL NUMBER OF SYMBOLS = 375
TOTAL NUMBER OF REFERENCES = 1221

```
                1          TITLE PSCAN
                2  ********************************************************
4D00            3  M$EXEC  EQU  >4D00          Module EXEC branch table address
2010            4  M$EDIT  EQU  >2010          Module EDIT branch table address
4000            5  M$FLMG  EQU  >4000          Module FLMGR branch table address
2022            6  M$MSG   EQU  M$EDIT+>12     Start of message area
                7  ********************************************************
4D04            8  EXEC    EQU  M$EXEC+>04
4D00            9  SCROLL  EQU  M$EXEC
4D86           10  SCRO10  EQU  M$EXEC+>86
2012           11  MAO     EQU  M$EDIT+>02
2016           12  SCANER  EQU  M$EDIT+>06
2018           13  ILL1    EQU  M$EDIT+>08
4012           14  CLSALL  EQU  M$FLMG+>12
201C           15  CHRTAB  EQU  M$EDIT+>CC
2020           16  GETLNB  EQU  M$EDIT+>10
201E           17  MVDN    EQU  M$EDIT+>CE
4D0A           18  ASC     EQU  M$EXEC+>0A
4D0E           19  EXEC6C  EQU  M$EXEC+>CE
4D14           20  DELINK  EQU  M$EXEC+>14
2022           21  MSG0    EQU  M$MSG
202C           22  MSG1    EQU  M$MSG+>0A
2040           23  MSG4    EQU  M$MSG+>1E
2049           24  MSG5    EQU  M$MSG+>27
2064           25  MSG7    EQU  M$MSG+>42
20AF           26  MSG15   EQU  M$MSG+>8D
20BD           27  MSG17   EQU  M$MSG+>98
20D9           28  MSGBLN  EQU  M$MSG+>B7
20F9           29  MSG18   EQU  M$MSG+>D7
213F           30  MSG19   EQU  M$MSG+>11D
2113           31  MSG21   EQU  M$MSG+>F1
4D18           32  COMPCT  EQU  M$EXEC+>18
401A           33  OUTREC  EQU  M$FLMG+>1A
               34  ********************************************************
0010           35  CPL     EQU  >10            CALL LINK    FLOATING-POINT PACKAGE
0012           36  RPL     EQU  >12            RETURN LINK
0014           37  CNS     EQU  >14            NUMBER TO STRING
0018           38  CHAR2$  EQU  >18            CHARACTER TABLE ADD.
001C           39  PUSH    EQU  >1C            PUSH FLOATING ACC
001E           40  POPS    EQU  >1E            POP FLOATING ACC
0020           41  EXIT    EQU  >20            PROGRAM EXIT
0022           42  GSCAN   EQU  >22            GROM SCAN ROUTINE
0024           43  PWR     EQU  >24            EXPONENTIATION
0026           44  SQR     EQU  >26            SQUARE ROOT
0028           45  EXP     EQU  >28            EXPONENTIAL
002A           46  LOG     EQU  >2A            NATURAL LOG
002C           47  COS     EQU  >2C            COSINE
002E           48  SIN     EQU  >2E            SINE
0030           49  TAN     EQU  >30            TANGENT
0032           50  ATN     EQU  >32            ARCTANGENT
0036           51  TONE2   EQU  >36            BAD BEEP
               52  ********************************************************
0072           53  STACK   EQU  >72            STACK FOR DATA          STATUS BLOCK
0073           54  SUBSTK  EQU  >73            SUBROUTINE STACK
0074           55  KEYBD   EQU  >74            KEYBOARD SELECTION
```

```
0075      56  RKEY    EQU  >75          KEY CODE
0078      57  RANDOM  EQU  >78          RANDOM NUMBER GENERATOR
0079      58  TIMER   EQU  >79          TIMING REGISTER
007B      59  VDPSTS  EQU  >7B          VDP STATUS REGISTER
007C      60  ERCODE  EQU  >7C          STATUS REGISTER
007E      61  XPOINT  EQU  >7E          X POINTER
007F      62  YPOINT  EQU  >7F          Y POINTER
          63  *
          64  *>>>>>>>>>>>>>>>   LOCAL VARIABLES   <<<<<<<<<<<<<<<<<<
          65  *
0000      66  VAR0    EQU  >00              TEMPORARY
0001      67  VARV    EQU  >01              TEMPORARY
0002      68  ACCUM   EQU  >02              #OF BYTES ACCUMULATOR (4 BYT
0002      69  STPT    EQU  >02              START POINT
0004      70  PABPTR  EQU  >04
0004      71  VARY    EQU  >04
0005      72  VARZ    EQU  >05
0006      73  DFLTLM  EQU  >06              DEFAULT LIMIT
0006      74  ENPT    EQU  >06              END POINT
0006      75  CCPPTR  EQU  >06              OFFSET WITHIN RECORD (1)
0007      76  RECLEN  EQU  >07              LENGTH OF CURRENT RECORD (1
0008      77  VARC    EQU  >08
0008      78  CCPADD  EQU  >08              RAM ADDRESS OF CURRENT COLU
0009      79  VARD    EQU  >09
000C      80  BYTE    EQU  >0C              BYTE COUNTER
000C      81  NMPTR   EQU  BYTE
000D      82  VAR1    EQU  >0D
000E      83  VAR4    EQU  >0E
000E      84  CHSAV   EQU  VAR4
0010      85  TOPSTK  EQU  >10              TOP OF STACK POINTER
0012      86  START   EQU  >12
0012      87  LINUM   EQU  START
0013      88  VAR8    EQU  >13
0014      89  NMLEN   EQU  >14              # OF BYTES NEEDED
0014      90  NBC     EQU  >14
0016      91  XFLAG   EQU  >16              SCAN FLAG-BITS USED AS BELO
          92  *                             BIT 7  ENTER(0)  ENTERX(1)
          93  *                             BIT 6  ALPHA(1)
          94  *                             BIT 5  NUMERIC(1)
          95  *                             BIT 4  STRING(1)
          96  *                             BIT 3  SPECIAL(1)
          97  *                             BIT 2  FUNCTION FLAG(1)
          98  *                             BIT 1  OPTION BASE DECLARED
          99  *                             BIT 0  IMPERATIVE(1) EXECUT
0016     100  VAR9    EQU  >16
0017     101  DSRFLG  EQU  >17              INTERNAL=60, EXTERNAL=0 (1)
0017     102  FORNET  EQU  >17
         103  *
         104  *>>>>>>>>>>>   GLOBAL  VARIABLES   <<<<<<<<<<<<<<<<
         105  *
0018     106  STRSP   EQU  >18              BEGINNING OF STRING SPACE
001A     107  STREND  EQU  >1A              END OF STRING SPACE
001C     108  SREF    EQU  >1C              TEMPORARY STRING POINTER
001E     109  NBD     EQU  >1E
0020     110  VARW    EQU  >20
```

```
0022   ERRCOD EQU  >22
0024   STVSPT EQU  >24
       *                             NOT USED IN PRESCAN
       *
002A   VARA   EQU  >2A
002C   VARB1  EQU  >2C
002C   CHPTR  EQU  VARB1             CHARACTER POINTER
002E   EXTRAM EQU  >2E               RAM LINE POINTER
0030   STLN   EQU  >30
0030   LLPTR  EQU  STLN              LAST LINE POINTER
0032   ENLN   EQU  >32
0032   NUMBUF EQU  ENLN              1ST ADDRESS OF LINE # BUFFER
0034   DATA   EQU  >34
0036   LNBUF  EQU  >36
0038   RAMPTR EQU  >38
003E   SYMTAB EQU  >3E               SYMBOL TABLE POINTER
0040   SYMPTR EQU  >40               FREE SPACE PTR
0042   CHAT   EQU  >42               CHARACTER
0043   BASE   EQU  >43               OPTION BASE VALUE
0044   BUFFY  EQU  >44
004A   FAC    EQU  >4A
005C   ARG    EQU  >5C
006C   FPERAD EQU  >6C
006E   VSPTR  EQU  >6E
0075   SGN$   EQU  >75
0076   EXP$   EQU  >76
       *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
       * THIS WAS USED AS CALCULATOR FLAG REMOVED FROM 99/4A
       *CALCFL EQU  >82
       * REPLACE BY ONE BYTE FLAG ALLUP FOR LOWERCASE CHAR. SET IN
       * 99/4A  3/3/81
0082   ALLUP  EQU  >82
       *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
0088   FLAG   EQU  >88
0089   GROMFL EQU  >89
0088   RSTK   EQU  >88
00B7   STKMIN EQU  >B7               BOTTOM OF DATA STACK
00BD   STKMAX EQU  >BD               TOP OF DATA STACK
       ***********************************************************
02E2   NLNADD EQU  >2E2                        VDP RAM EQUATES
02FE   ENDSCR EQU  >2FE
0320   CRNBUF EQU  >320
03BE   CRNEND EQU  >3BE
03E0   SYMBOL EQU  >3E0              2 BYTES
03EB   WARNTP EQU  >3EB              TEMP FOR WARNING; 1 BYTE
       *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
       * REPLACE  VRAMVS EQU >5F8   FOR LOWERCASE CHAR. SET IN
       * 99/4A  3/3/81
       *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
06F8   VRAMVS EQU  >6F8
0800   SYMPT  EQU  >800
       ***********************************************************
0002   X1     EQU  >02                         IMMEDIATE VALUES
0003   X2     EQU  >03
0017   Y1     EQU  >17
```

```
0002    166  DISPY   EQU  >02
000D    167  CHRTN   EQU  >0D
0012    168  LIMIT   EQU  >12
001C    169  XN      EQU  >1C
0020    170  BKGD    EQU  >20
000B    171  MAXLEN  EQU  >0B
0005    172  MAXET   EQU  >05
        173  ********************************************************
0002    174  BREAK   EQU  >02
0003    175  DLETE   EQU  >03
0004    176  INSRT   EQU  >04
0007    177  CLRLN   EQU  >07
0008    178  BACK    EQU  >08          KEYBOARD COMMAND KEY CODE
0009    179  FORW    EQU  >09
000A    180  DOWN    EQU  >0A
000B    181  MVUP    EQU  >0B
        182  ********************************************************
0006    183  HELP    EQU  >06          EDITTING COMMAND CODES
0002    184  CONTCM  EQU  >02          (WHEN ENTERED WITH A LINE #, THE
0005    185  TRACE   EQU  >05           CODE IS ITS INTERNAL TOKEN. )
007E    186  ESC     EQU  >7E          ('EDIT', 'LIST', 'NEW', AND 'RUN'
0000    187  LISTCM  EQU  >00           WON'T BE ON TEXT STACK OF EXTERN
000F    188  EDIT    EQU  >0F           RAM. )
000E    189  NEW     EQU  >0E
0003    190  LOAD    EQU  >03
0004    191  SAVECM  EQU  >04
0001    192  RUNCM   EQU  >01
        193  ********************************************************
        194  *             ASCII CODES FOR EACH CHARACTER
0030    195  ZERO    EQU  >30          0
0039    196  NINE    EQU  >39          9
003A    197  COLON   EQU  >3A          :
003B    198  SEMIC   EQU  >3B          ;
003C    199  LESS    EQU  >3C          <
003D    200  EQUAL   EQU  >3D          =
003E    201  GREAT   EQU  >3E          >
003F    202  QUEST   EQU  >3F          ?
0040    203  AT      EQU  >40          @
0041    204  A       EQU  >41          A
005A    205  Z       EQU  >5A          Z
005B    206  OPENB   EQU  >5B          [
005C    207  BACKS   EQU  >5C
005D    208  CLOSEB  EQU  >5D          ]
005E    209  CIRCUM  EQU  >5E          ^
005F    210  UNLN    EQU  >5F          _
0020    211  SPACE   EQU  >20
0021    212  EXCL    EQU  >21          !
0022    213  QUOTE   EQU  >22          "
0023    214  NUMBER  EQU  >23          #
0024    215  DOLLAR  EQU  >24          $
0025    216  PERC    EQU  >25          %
0026    217  AMPER   EQU  >26          &
0027    218  APOS    EQU  >27          '
0028    219  LPAR    EQU  >28          (
0029    220  RPAR    EQU  >29          )
```

```
002A      221  MULT    EQU  >2A              *
002B      222  PLUS    EQU  >2B              +
002C      223  COMMA   EQU  >2C              ,
002D      224  MINUS   EQU  >2D              -
002E      225  DOT     EQU  >2E              .
002F      226  DIVI    EQU  >2F              /
0060      227  OFFSET  EQU  >60                    OFFSET FOR SCREEN
007E      228  CURSOR  EQU  >1E+OFFSET
007F      229  EDGECH  EQU  >1F+OFFSET             EDGE CHARACTER
          230  ***************************************************************
0001      231  ROUND$  EQU  1                 ROUND
0002      232  ROUNU$  EQU  2                 ROUND UP
0003      233  FACSTS  EQU  3                 STATUS
0004      234  OVUN    EQU  4                 OVER OR UNDER FLOW
0005      235  OV      EQU  5                 OVERFLOW
0006      236  FADD    EQU  6                 ADDITION
0007      237  FSUB    EQU  7                 SUBTRACTION
0008      238  FMULT   EQU  8                 MULTIPLICATION
0009      239  FDIV    EQU  9                 DIVISION
000A      240  FCOMP   EQU  10                COMPARE
000B      241  FSADD   EQU  11                STACK ADD
000C      242  FSSUB   EQU  12                STACK SUBTRACT
000D      243  FSMULT  EQU  13                STACK MULTIPLY
000E      244  FSDIV   EQU  14                STACK DIVIDE
000F      245  FSCOMP  EQU  15                STACK COMPARE
0010      246  CSNUM   EQU  16                STRING TO NUMBER
0011      247  GRINT   EQU  17                GREATEST INTEGER
0012      248  FLTINT  EQU  18                FLOATING TO INTEGER
0013      249  SYM     EQU  19                GET SYMBOL TABLE ENTRY
0014      250  SMB     EQU  20                GET SYMBOL VALUE
0016      251  SCHSYM  EQU  22                SEARCH SYMBOL TABLE
0017      252  VPUSH   EQU  23                PUSH ON VALUE STACK
0018      253  VPOP    EQU  24                POP FROM VALUE STACK
001B      254  PGMCH   EQU  27                GET PROGRAM CHARACTER
```

```
                    256  *
                    257  *              B A S I C   T O K E N   T A B L E
                    258  *
                    259  *                                      REVISED 8/23/78
                    260  *       EQU  >80          SPARE
0081                261  ELSE$    EQU  >81
                    262  *       EQU  >82          ":"               (62)
                    263  *       EQU  >83          "!"               (62)
0084                264  IF$      EQU  >84          "IF"
0085                265  GO$      EQU  >85          "GO"
0086                266  GOTO$    EQU  >86          "GOTO"
0087                267  GOSUB$   EQU  >87          "GOSUB"
0088                268  RETUR$   EQU  >88          "RETURN"
0089                269  DEF$     EQU  >89          "DEF"
008A                270  DIM$     EQU  >8A          "DIM"
008B                271  END$     EQU  >8B          "END"
008C                272  FOR$     EQU  >8C          "FOR"
008D                273  LET$     EQU  >8D          "LET"
008E                274  BREAK$   EQU  >8E          "BREAK"
008F                275  UNBRE$   EQU  >8F          "UNBREAK"
0090                276  TRACE$   EQU  >90          "TRACE"
0091                277  UNTRA$   EQU  >91          "UNTRACE"
0092                278  INPUT$   EQU  >92          "INPUT"
0093                279  DATA$    EQU  >93          "DATA"
0094                280  RESTO$   EQU  >94          "RESTORE"
0095                281  RANDO$   EQU  >95          "RANDOMIZE"
0096                282  NEXT$    EQU  >96          "NEXT"
0097                283  READ$    EQU  >97          "READ"
0098                284  STOP$    EQU  >98          "STOP"
0099                285  DELET$   EQU  >99          "DELETE"
009A                286  REM$     EQU  >9A          "REM"
009B                287  ON$      EQU  >9B          "ON"
009C                288  PRINT$   EQU  >9C          "PRINT"
009D                289  CALL$    EQU  >9D          "CALL"
009E                290  OPTIO$   EQU  >9E          "OPTION"
009F                291  OPEN$    EQU  >9F          "OPEN"
00A0                292  CLOSE$   EQU  >A0          "CLOSE"
00A1                293  SUB$     EQU  >A1          "SUB"
00A2                294  DISPL$   EQU  >A2          "DISPLAY"
                    295  *        EQU  >A3          "SUBEXIT"         (62)
                    296  *        EQU  >A4          "SUBEND"          (62)
                    297  *        EQU  >A5          "REAL"            (62)
                    298  *        EQU  >A6          "INTEGER"         (62)
                    299  *        EQU  >A7          "SCRATCH"         (62)
                    300  *        EQU  >A8          "ACCEPT"          FUTURE
                    301  *        EQU  >A9          "IMAGE"           FUTURE
                    302  *        EQU  >AA          SPARES
                    303  *        EQU  >AB
                    304  *        EQU  >AC
                    305  *        EQU  >AD
                    306  *        EQU  >AE
                    307  *        EQU  >AF
00B0                308  THEN$    EQU  >B0          "THEN"
00B1                309  TO$      EQU  >B1          "TO"
00B2                310  STEP$    EQU  >B2          "STEP"
```

```
00B3      311  COMMA$  EQU  >B3          ","
00B4      312  SEMIC$  EQU  >B4          ";"
00B5      313  COLON$  EQU  >B5          ":"
00B6      314  RPAR$   EQU  >B6          ")"
00B7      315  LPAR$   EQU  >B7          "("
00B8      316  CONC$   EQU  >B8          "&"  CONCATENATE STRINGS
          317  *       EQU  >B9          SPARE
          318  *       EQU  >BA          "OR"              (62)
          319  *       EQU  >BB          "AND"             (62)
          320  *       EQU  >BC          SPARE FOR "XOR"   (62)
          321  *       EQU  >BD          "NOT"             (62)
00BE      322  EQUAL$  EQU  >BE          "="
00BF      323  LESS$   EQU  >BF          "<"
00C0      324  GREAT$  EQU  >C0          ">"
00C1      325  PLUS$   EQU  >C1          "+"
00C2      326  MINUS$  EQU  >C2          "-"
00C3      327  MULT$   EQU  >C3          "*"
00C4      328  DIVI$   EQU  >C4          "/"
00C5      329  CIRCU$  EQU  >C5          "^"
          330  *       EQU  >C6          SPARE
00C7      331  STRIN$  EQU  >C7          QUOTED STRING
00C8      332  UNQST$  EQU  >C8          UNQUOTED STRING
00C8      333  NUM$    EQU  UNQST$       ALSO NUMERICAL STRING
00C8      334  NUMCO$  EQU  UNQST$            AND EVEN NUMERICAL CONSTANT
00C9      335  LN$     EQU  >C9          GROM NUMERIC CONSTANT
00CA      336  EOF$    EQU  >CA          "EOF"
00CB      337  ABS$    EQU  >CB          "ABS"
00CC      338  ATN$    EQU  >CC          "ATN"
00CD      339  COS$    EQU  >CD          "COS"
00CE      340  EXP$$   EQU  >CE          "EXP"
00CF      341  INT$    EQU  >CF          "INT"
00D0      342  LOG$    EQU  >D0          "LOG"
00D1      343  SGN$$   EQU  >D1          "SGN"
00D2      344  SIN$    EQU  >D2          "SIN"
00D3      345  SQR$    EQU  >D3          "SQR"
00D4      346  TAN$    EQU  >D4          "TAN"
00D5      347  LEN$    EQU  >D5          "LEN"
00D6      348  CHR$$   EQU  >D6          "CHR$"
00D7      349  RND$    EQU  >D7          "RND"
00D8      350  SEG$$   EQU  >D8          "SEG$"
00D9      351  POS$    EQU  >D9          "POS"
00DA      352  VAL$    EQU  >DA          "VAL"
00DB      353  STR$$   EQU  >DB          "STR$"
00DC      354  ASC$    EQU  >DC          "ASC"
          355  *       EQU  >DD          "PI"
00DE      356  REC$    EQU  >DE          "REC"
          357  *
          358  *
          359  *       EQU  >EC          "ALL"             (62)
          360  *       EQU  >ED          "USING"           (62)
          361  *       EQU  >EE          "BEEP"            (62)
          362  *       EQU  >EF          "ERASE"           (62)
          363  *       EQU  >F0          "AT"              (62)
00F1      364  BASE$   EQU  >F1          "BASE"
          365  *       EQU  >F2          "TEMPORARY"       (62)
```

```
00F3    366  VARIA$ EQU  >F3        "VARIABLE"      (62)
00F4    367  RELAT$ EQU  >F4        "RELATIVE"      (62)
00F5    368  INTER$ EQU  >F5        "INTERNAL"      (62)
00F6    369  SEQUE$ EQU  >F6        "SEQUENTIAL"
00F7    370  OUTPU$ EQU  >F7        "OUTPUT"
00F8    371  UPDAT$ EQU  >F8        "UPDATE"
00F9    372  APPEN$ EQU  >F9        "APPEND"
00FA    373  FIXED$ EQU  >FA        "FIXED"
00FB    374  PERMA$ EQU  >FB        "PERMANENT"
00FC    375  TAB$   EQU  >FC        "TAB"
00FD    376  NUMBE$ EQU  >FD        "#"
        377  *      EQU  >FE        GROM LINE POINTER
        378  *      EQU  >FF        SPARE
```

```
                380         GROM 1
                381         ORG  >0828
2828 4FFF       382         BR   PRGLST
282A 4F43       383         BR   PTLNER          Was PRA2
282C 4C75       384         BR   GETNB
282E 4DFA       385         BR   LLIST           Was PRB2
2830 4CA6       386         BR   GETCHR
2832 4A42       387         BR   READLN
2834 4C36       388         BR   AUTO1
2836 4FC4       389         BR   SEARCH
2838 4BD6       390         BR   DELREP
283A 4F12       391         BR   PTLNE
283C 4EF9       392         BR   GETLN
283E 4F5D       393         BR   SEETWO
2840 4C2B       394         BR   AUTON
2842 4FAF       395         BR   DISO
2844 5493       396         BR   MEMCHK          Was ENDERR
2846 5450       397         BR   PROP
2848 51E5       398         BR   ENTER
284A 522B       399         BR   ENTO9
284C 4D24       400         BR   WARN$$          REFERENCED BY MONITOR
284E 4D99       401         BR   ERR$$           REFERENCED BY MONITOR
2850 4C84       402         BR   STKCHR
2852 4CA0       403         BR   PUSHCH
2854 4CC0       404         BR   UNQSTR
2856 4C7A       405         BR   GETNB2
2858 4A49       406         BR   READL1
285A 4A4F       407         BR   READOO
```

```
                    409  ********************************************************
                    410  **      KEYWORD TABLE
                    411  **      THE TOKEN IS ITS LEFT BINDING POWER.
                    412  ********************************************************
                    413  KEYTAB DATA #CHAR1,#CHAR2,#CHAR3,#CHAR4;
                    414              #CHAR5,#CHAR6,#CHAR7,#CHAR8;
285C 287028         415              #CHAR9,#CHARA
285F 8F289C
2862 291D29
2865 73299E
2868 29D029
286B F12A16
286E 2A2B
2870 29B6           416  CHAR1   DATA RPAR,RPAR$          )
2872 28B7           417           DATA LPAR,LPAR$          (
2874 26B8           418           DATA AMPER,CONC$         & (CONCATENATE)
2876 5EC5           419           DATA CIRCUM,CIRCU$       ^
2878 3DBE           420           DATA EQUAL,EQUAL$        =
287A 2AC3           421           DATA MULT,MULT$          *
287C 2FC4           422           DATA DIVI,DIVI$          /
287E 2BC1           423           DATA PLUS,PLUS$          +
2880 2DC2           424           DATA MINUS,MINUS$        -
2882 3CBF           425           DATA LESS,LESS$          <
2884 3EC0           426           DATA GREAT,GREAT$        >
2886 3AB5           427           DATA COLON,COLON$        :
2888 3BB4           428           DATA SEMIC,SEMIC$        ;
288A 23FD           429           DATA NUMBER,NUMBE$       #
288C 2CB3           430           DATA COMMA,COMMA$        ,
288E FF             431           DATA >FF
288F 474F85         432  CHAR2   DATA :GO:,GO$
2892 494684         433           DATA :IF:,IF$
2895 4F4E9B         434           DATA :ON:,ON$
2898 544FB1         435           DATA :TO:,TO$
289B FF             436           DATA >FF
289C 444546         437  CHAR3   DATA :DEF:,DEF$
289F 89
28A0 44494D         438           DATA :DIM:,DIM$
28A3 8A
28A4 454E44         439           DATA :END:,END$
28A7 8B
28A8 454F46         440           DATA :EOF:,EOF$
28AB CA
28AC 464F52         441           DATA :FOR:,FOR$
28AF 8C
28B0 4C4554         442           DATA :LET:,LET$
28B3 8D
28B4 52454D         443           DATA :REM:,REM$
28B7 9A
28B8 535542         444           DATA :SUB:,SUB$
28BB A1
28BC 544142         445           DATA :TAB:,TAB$
28BF FC
28C0 414253         446           DATA :ABS:,ABS$
28C3 CB
28C4 41544E         447           DATA :ATN:,ATN$
```

```
 3C7 CC
28C8 434F53   448          DATA :COS:,COS$
28CB CD
28CC 455850   449          DATA :EXP:,EXP$$
28CF CE
28D0 494E54   450          DATA :INT:,INT$
28D3 CF
28D4 4C4F47   451          DATA :LOG:,LOG$
28D7 D0
28D8 524E44   452          DATA :RND:,RND$
28DB D7
28DC 53474E   453          DATA :SGN:,SGN$$
28DF D1
28E0 53494E   454          DATA :SIN:,SIN$
28E3 D2
28E4 535152   455          DATA :SQR:,SQR$
28E7 D3
28E8 54414E   456          DATA :TAN:,TAN$
28EB D4
28EC 4C454E   457          DATA :LEN:,LEN$
28EF D5
28F0 504F53   458          DATA :POS:,POS$
28F3 D9
28F4 56414C   459          DATA :VAL:,VAL$
28F7 DA
28F8 415343   460          DATA :ASC:,ASC$
28FB DC
28FC 524543   461          DATA :REC:,REC$
28FF DE
2900 4E4557   462          DATA :NEW:,>01
2903 01
2904 52554E   463          DATA :RUN:,>00
2907 00
2908 434F4E   464          DATA :CON:,>02     ABBREVIATION OF 'CONTINUE'
290B 02
290C 4E554D   465          DATA :NUM:,>05     ABBREVIATION OF 'NUMBER'
290F 05
2910 524553   466          DATA :RES:,>07     ABBREVIATION OF 'RESEQUENCE'
2913 07
2914 425945   467          DATA :BYE:,>04
2917 04
2918 4F4C44   468          DATA :OLD:,>06     'UNSAVE'
291B 06
291C FF       469          DATA >FF
291D 424153   470 CHAR4     DATA :BASE:,BASE$
2920 45F1
2922 444154   471          DATA :DATA:,DATA$
2925 4193
2927 454449   472          DATA :EDIT:,>09
292A 5409
292C 454C53   473          DATA :ELSE:,ELSE$
292F 4581
2931 474F54   474          DATA :GOTO:,GOTO$
2934 4F86
2936 4E4558   475          DATA :NEXT:,NEXT$
```

```
2939 5496
293B 524541   476          DATA :READ:,READ$
293E 4497
2940 535445   477          DATA :STEP:,STEP$
2943 50B2
2945 53544F   478          DATA :STOP:,STOP$
2948 5098
294A 544845   479          DATA :THEN:,THEN$
294D 4EB0
294F 434852   480          DATA :CHR$:,CHR$$
2952 24D6
2954 534547   481          DATA :SEG$:,SEG$$
2957 24D8
2959 535452   482          DATA :STR$:,STR$$
295C 24DB
295E 4C4953   483          DATA :LIST:,>03
2961 5403
2963 534156   484          DATA :SAVE:,>08
2966 4508
2968 43414C   485          DATA :CALL:,CALL$
296B 4C9D
296D 4F5045   486          DATA :OPEN:,OPEN$
2970 4E9F
2972 FF       487          DATA >FF
2973 425245   488  CHAR5   DATA :BREAK:,BREAK$
2976 414B8E
2979 474F53   489          DATA :GOSUB:,GOSUB$
297C 554287
297F 464958   490          DATA :FIXED:,FIXED$
2982 4544FA
2985 494E50   491          DATA :INPUT:,INPUT$
2988 555492
298B 505249   492          DATA :PRINT:,PRINT$
298E 4E549C
2991 545241   493          DATA :TRACE:,TRACE$
2994 434590
2997 434C4F   494          DATA :CLOSE:,CLOSE$
299A 5345A0
299D FF       495          DATA >FF
299E 4F5054   496  CHAR6   DATA :OPTION:,OPTIO$
29A1 494F4E
29A4 9E
29A5 524554   497          DATA :RETURN:,RETUR$
29A8 55524E
29AB 88
29AC 4E554D   498          DATA :NUMBER:,>05
29AF 424552
29B2 05
29B3 4F5554   499          DATA :OUTPUT:,OUTPU$
29B6 505554
29B9 F7
29BA 415050   500          DATA :APPEND:,APPEN$
29BD 454E44
29C0 F9
29C1 555044   501          DATA :UPDATE:,UPDAT$
```

```
 29C4 415445
29C7 F8
29C8 44454C    502           DATA :DELETE:,DELET$
29CB 455445
29CE 99
29CF FF        503           DATA >FF
29D0 554E54    504   CHAR7   DATA :UNTRACE:,UNTRA$
29D3 524143
29D6 4591
29D8 554E42    505           DATA :UNBREAK:,UNBRE$
29DB 524541
29DE 488F
29E0 524553    506           DATA :RESTORE:,RESTO$
29E3 544F52
29E6 4594
               507   *       DATA :REPLACE:,REPLA$  REMOVED FROM CURRENT VERSION
29E8 444953    508           DATA :DISPLAY:,DISPL$
29EB 504C41
29EE 59A2
29F0 FF        509           DATA >FF
29F1 434F4E    510   CHAR8   DATA :CONTINUE:,>02
29F4 54494E
29F7 554502
29FA 564152    511           DATA :VARIABLE:,VARIA$
29FD 494142
 A00 4C45F3
2A03 494E54    512           DATA :INTERNAL:,INTER$
2A06 45524E
2A09 414CF5
2A0C 52454C    513           DATA :RELATIVE:,RELAT$
2A0F 415449
2A12 5645F4
2A15 FF        514           DATA >FF
2A16 52414E    515   CHAR9   DATA :RANDOMIZE:,RANDO$
2A19 444F4D
2A1C 495A45
2A1F 95
2A20 504552    516           DATA :PERMANENT:,PERMA$
2A23 4D414E
2A26 454E54
2A29 FB
2A2A FF        517           DATA >FF
2A2B 534551    518   CHARA   DATA :SEQUENTIAL:,SEQUE$
2A2E 55454E
2A31 544941
2A34 4CF6
2A36 524553    519           DATA :RESEQUENCE:,>07
2A39 455155
2A3C 454E43
2A3F 4507
2A41 FF        520           DATA >FF
```

```
                    522  *
                    523  *       READLN - Read one logical line (up to four
                    524  *       physical lines) from the keyboard.  Interpret
                    525  *       things like BACKSPACE, INSERT, DELETE and
                    526  *       FORWARD.  The total number of characters can be
                    527  *       limited by changing the start value for ARG+2
                    528  *       (upper limit), and entering ar READL1.
                    529  *       VARW has to contain the start address of the field,
                    530  *       and VARA the current highest write address
                    531  *            Entering at READL1 also enables us to
                    532  *       prespecify the minimum number of characters to be
                    533  *       read.  Filling this field with a default answer
                    534  *       allows for default creation.
                    535  *            Entering at READ00 allows for specification
                    536  *       of the initial cursor-position.  In this case
                    537  *       ARG+5 has to be set to the cursor-position.
                    538  *            Please see to it that VARA, VARW, ARG+2,
                    539  *       ARG+4 have consistent values, i.e.
                    540  *             VARW <= ARG+5 <= VARA <= ARG+2
                    541  *       ARG+4 indicates if the line has been changed.
                    542  *       If so, it contains a 0.  If you enter READLN throu
                    543  *       READ00, you have to initialize ARG+4 to a nonzero
                    544  *       value, should you want to use this feature.
                    545  *
                    546  READLN
2A42 BF5E03         547    DST  >35D,@ARG+2        Set default upper limit
2A45 5D
2A46 BD2A20         548    DST  @VARW,@VARA        Default to "nothing entered yet"
                    549    $
                    550    $    Please make sure that VARA points at an space
                    551    $    location, or at the end-of-field
                    552    $
                    553  READL1
2A49 BE6001         554    ST   1,@ARG+4           This means "no change" in line
                    555  READL2
2A4C BD6120         556    DST  @VARW,@ARG+5       Position cursor at start of fiel
                    557  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                    558  *  Auto-repeat function is added by CAROLYN LEE 2/26/81
                    559  *  for 99/4A
                    560  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                    561  READ00
2A4F 860D           562    CLR  @VAR1              Counter for auto-repeat func.
                    563    $
                    564    $    To get out of insert mode, we usually return here
                    565    $
                    566  READ01
2A51 8663           567    CLR  @ARG+7             Indicate normal operation mode
2A53 BE017E         568    ST   CURSOR,@VARV       Use VARV for CURSOR/CHARACTER
                    569  READ$1
                    570    $
                    571    $    Input one character and alternate current
                    572    $    character position between normal and cursor
                    573    $
2A56 C0B061         574    EX   @VARV,RAM(@ARG+5) By alternating between the norma
2A59 01
```

```
A5A 8679        CLR  @TIMER                 character and the cursor, we mak
                $REPEAT                     the cursor "blink"
2A5C 03           SCAN                     Scan for a character
2A5D 6A78         BS   READ$2              Found a new one !!!!!
2A5F 900D         INC  @VAR1               Increment the auto-repeat counter
2A61 D675FF       $IF @RKEY .NE. >FF THEN   It is an old key
2A64 6A71
2A66 CA0DFE         $IF @VAR1 .HE. 254 THEN  Hold old key for a while
2A69 4A71
           *                                before starting repeat
2A6B A60D1E           SUB  30,@VAR1         Control repeat rate
2A6E 052A7A           B     READ$3
                    $END IF
                  $END IF
2A71 C6790E     $UNTIL @TIMER .H. 14   Time next character switch
2A74 4A5C
2A76 4A56       BR   READ$1              Restart character blink cycle
           $
           $    Correct if we ended up with a displayed cursor
           $
          READ$2
2A78 860D       CLR  @VAR1
          READ$3
2A7A D6017E     $IF @VARV .NE. CURSOR THEN Will have to change once more
2A7D 6A83
?A7F C0B061       EX    @VARV,RAM(@ARG+5) Exchange for current cursor
2A82 01
                $END IF
2A83 CA7520     $IF @RKEY .L. : : THEN Control character !!!!!
2A86 6B41
                  $
                  $  BREAK character handling comes first
                  $
2A88 D67502       $IF @RKEY .EQ. BREAK THEN Saw break character
2A8B 4AA1
2A8D B28088         RB    @FLAG,0        Reset AUTONUM mode
2A90 FE
          *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
          *  Delete this line for 99/4A without equation calculator
          *    by CAROLYN LEE 2/26/81
          *     $IF @CALCFL .NE. 0 GOTO CLRLIN If calc-clear the line
          *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
2A91 8E8089         $IF @GROMFL .NE. 0 GOTO READ$1 Ignore BREAK for GROMs
2A94 4A56
2A96 DA8088         $IF .BITS @FLAG .EQ. 0 THEN BREAK in run mode
2A99 204A9F
2A9C 054D0E           B    EXEC6C        Interrupt execution
                    $END IF
          *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
          *      CHANGED FROM:
          *     B    ILL1                Exit from edit mode
          *      BY STAN HUME 06/19/80 TO SAVE 1 BYTE
          *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
2A9F 4018           BR   ILL1            Exit from edit mode
                  $END IF
```

```
                    620       $
                    621       $  BACK-ARROW - Space back one position
                    622       $
2AA1 D67508         623       $IF @RKEY .EQ. BACK GOTO RBACK   Backup to previous po
2AA4 6BC3
                    624       $
                    625       $  RIGHT-ARROW - Forward space
                    626       $
2AA6 D67509         627       $IF @RKEY .EQ. FORW GOTO RFORW   Space one position
2AA9 6BBF
                    628       $
                    629       $  INSERT - Start INSERT mode here
                    630       $
2AAB D67504         631       $IF @RKEY .EQ. INSRT THEN        Set INSERT flag
2AAE 4AB3
2AB0 BE6301         632         ST   1,@ARG+7      Select INSERT mode
                    633       $END IF
                    634       $
                    635       $  DELETE - Delete the current character
                    636       $
2AB3 D67503         637       $IF @RKEY .EQ. DLETE THEN        DELETE all right
2AB6 4B08
2AB8 8660           638         CLR  @ARG+4        Indicate definite change in line
2ABA D52A61         639         $IF @VARA .DNE. @ARG+5 THEN Not an empty line
2ABD 6B02
                    640   *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    641   *       THE FOLLOWING THREE LINES ADDED BY STAN HUME ON
                    642   *       06/19/80 TO FIX PROBLEM OF DELETING-INSERTING-
                    643   *       DELETING-INSERTING CHARACTERS OVER A LINE BOUNDARY
                    644   *       MESSING UP THE END-OF-LINE POINTER AND DESTROYING
                    645   *       THE EDGE CHARACTERS
                    646   *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
2ABF D6B02A         647           $IF RAM(@VARA) .EQ. EDGECH  If pointing at edge-ch
2AC2 7F4AC7
2AC5 932A           648             DDEC @VARA       Back off on the end-of-line
                    649           $END IF
2AC7 BD5C2A         650           DST  @VARA,@ARG  Move everything from the right
2ACA A55C61         651           DSUB @ARG+5,@ARG   of the cursor to the left
2ACD 345CB0         652           MOVE @ARG FROM RAM(1(ARG+5)) TO RAM(@ARG+5)
2AD0 61E001
2AD3 61
2AD4 BD5C61         653           DST  @ARG+5,@ARG Start at the beginning
2AD7 B25DFC         654           AND  >FC,@ARG+1
2ADA B65D1D         655           OR   >1D,@ARG+1  Move over to the end of the line
2ADD C95C2A         656           $WHILE @ARG .DL. @VARA Update all errors
2AE0 6AEE
2AE2 C0E004         657             EX   RAM(@ARG),RAM(4(ARG))  Restore edge chars
2AE5 5CB05C
2AE8 A35C00         658             DADD 32,@ARG   Next victim please
2AEB 20
2AEC 4ADD           659           $END WHILE
2AEE 932A           660           DDEC @VARA       Pre-update end of string
2AF0 D6B02A         661           $IF RAM(@VARA) .EQ. EDGECH THEN Hit edge characte
2AF3 7F4AFA
2AF6 A72A00         662             DSUB 4,@VARA   Skip over edge characters
```

```
)AF9 04
                663           $END IF
2AFA D6B02A     664           $IF RAM(@VARA) .EQ. : :+OFFSET GOTO READ01
2AFD 806A51
2B00 912A       665           DINC @VARA        Locked at field position
                666         $END IF
2B02 BEB02A     667         ST   : :+OFFSET,RAM(@VARA) Clear last position
2B05 80
2B06 4A51       668         BR   READ01
                669       $END IF
                670       $
                671       $  CLEAR - Clear the entire input line
                672       $
2B08 D67507     673       $IF @RKEY .EQ. CLRLN THEN       Found CLEAR command
2B0B 4B24
                674   CLRLIN
                675         $REPEAT             Current maximum to minimum
2B0D D6B02A     676           $IF RAM(@VARA) .NE. EDGECH THEN Don't clear edges
2B10 7F6B17
2B13 BEB02A     677             ST   : :+OFFSET,RAM(@VARA) Blank line
2B16 80
                678           $END IF
2B17 932A       679           DDEC @VARA        Pre-update end-of-line
2B19 C92A20     680         $UNTIL @VARA .DL. @VARW Up to and including first pos
2B1C 6B0D
2B1E 912A       681         DINC @VARA          Undo last subtraction
2B20 8660       682         CLR  @ARG+4         Indicate change
2B22 4A4C       683         BR   READL2         And restart everything
                684       $END IF
                685       $
                686       $  General exit point.  Unidentified control codes
                687       $  don't have any effect !!!!!!!
                688       $
2B24 D6750D     689       $IF @RKEY .NE. CHRTN THEN Only react on CR/UP/DOWN
2B27 6B33
2B29 D6750B     690         $IF @RKEY .NE. MVUP THEN
2B2C 6B33
2B2E D6750A     691           $IF @RKEY .NE. DOWN GOTO READ#1
2B31 4A56
                692         $END IF
                693       $END IF
2B33 D52A5E     694       $IF @VARA .DEQ. @ARG+2 THEN Check for block on last pos
2B36 4B40
2B38 D6B02A     695         $IF RAM(@VARA) .NE. : :+OFFSET THEN    Blocked.....
2B3B 806B40
2B3E 912A       696           DINC @VARA        Point beyond last character in lin
                697         $END IF
                698       $END IF
2B40 00         699       RTN                   ENTER the current line
                700     $END IF
2B41 8E636B     701     $IF @ARG+7 .NE. 0 THEN INSERT mode
2B44 7D
                702       $
                703       $  INSERT is COMPLICATED!!!!! because of those edge
                704       $  characters.......Shift up all things... .continue
```

```
                  $  as a standard insert......VARA <= ARG+2
                  $
               *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
               *       THE FOLLOWING 4 LINES PLUS THE LABEL READ$4 ADDED
               *       BY STAN HUME ON 06/19/80 TO FIX MULTI-LINE DELETE
               *       CHARACTERS THEN INSERT CHARACTERS INTO MARGIN OVER
               *       THE EDGE CHARACTERS
               *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
2B45 D52A5E       $IF  @VARA .DEQ. @ARG+2 GOTO READ$4 OK if cursor at eo
2B48 6B54
2B4A D6B02A       $IF  RAM(@VARA) .EQ. EDGECH THEN  If pointing at edge
2B4D 7F4B54
2B50 A32A00         DADD 4,@VARA       Skip over the edge characters
2B53 04
                  $END IF
               READ$4
2B54 BD5C2A       DST  @VARA,@ARG      Use ARG as temp. for insert
2B57 C55C61       $WHILE @ARG .DH. @ARG+5 Move everything up to cur. loc
2B5A 4B76
2B5C 935C           DDEC @ARG          Copy lower location to higher one
2B5E BCE001         ST   RAM(@ARG),RAM(1(ARG)) going from high to low
2B61 5CB05C
2B64 D6B05C         $IF RAM(@ARG) .EQ. EDGECH THEN Bumped into wall agai
2B67 7F4B74
2B6A A75C00           DSUB 4,@ARG      Skip the wall
2B6D 04
2B6E BCE005           ST   RAM(@ARG),RAM(5(ARG)) And move character over
2B71 5CB05C
                    $END IF
2B74 4B57         $SEND WHILE
2B76 C92A5E       $IF @VARA .DL. @ARG+2 THEN Only update VARA is upper
2B79 6B7D
2B7B 912A           DINC @VARA            hasn't been reached yet
                  $END IF
                $END IF                Rest is standard copy-character
2B7D A27560     ADD  OFFSET,@RKEY      Create displayable character
2B80 BC3061     ST   @RKEY,RAM(@ARG+5) Display at current character pos.
2B83 75
2B84 8660       CLR  @ARG+4            Indicate change in line
               READ05
2B86 D5615E     $IF @ARG+5 .DEQ. @ARG+2 THEN Hit right margin
2B89 4B90
2B8B 060036       CALL TONE2           ---------- B E E P -------------
2B8E 4A56         BR   READ$1          Stay in current mode !!!!!!!
                $END IF
2B90 9161       DINC @ARG+5            Update current address
2B92 D6B061     $IF RAM(@ARG+5) .EQ. EDGECH THEN Correct for next line
2B95 7F4B9C
2B98 A36100       DADD 4,@ARG+5        By skipping border
2B9B 04
                $END IF
2B9C C5612A     $IF @ARG+5 .DH. @VARA THEN Check for last new high limi
2B9F 4BA4
2BA1 BD2A61       DST  @ARG+5,@VARA    Update new high limit
                $END IF
```

```
.BA4 CB2A02   746    $IF @VARA .DL. >2FE GOTO READ$1 Still some space to go
2BA7 FE4A56
2BAA 064D00   747    CALL SCROLL              Scroll the screen !!!!!
2BAD A72A00   748    DSUB 2 ,@VARA            Back to current start of line
2BB0 1C
2BB1 A72000   749    DSUB 32,@VARW            Backup line start address
2BB4 20
2BB5 A75E00   750    DSUB 32,@ARG+2           Absolute high limit backs up too
2BB8 20
2BB9 A76100   751    DSUB 32,@ARG+5           Current cursor position too
2BBC 20
2BBD 4A56     752    BR   READ$1              Start with something else
              753    $
              754    $    Something special for forward cursor move
              755    $
              756  RFORW
2BBF 8663     757    CLR  @ARG+7              Leave INSERT mode - don't copy
2BC1 4886     758    BR   READ05                but use rest of input logic.
              759  RBACK
2BC3 C56120   760    $IF @ARG+5 .DH. @VARW    This is the standard backup entry,
2BC6 4BD4
2BC8 9361     761      DDEC @ARG+5              so we backup the current positio
2BCA D6B061   762      $IF RAM(@ARG+5) .EQ. EDGECH THEN Skip border line
2BCD 7F4BD4
2BD0 A76100   763        DSUB 4,@ARG+5        Backup to previous line
2BD3 04
              764      $END IF
              765    $END IF
2BD4 4A51     766    BR  READ01               Go back for next char.
```

```
                    768  ********************************************************
                    769  **      SUBROUTINE TO MOVE CONTENTS IN RAM
                    770  ********************************************************
2BD6 952E           771  DELREP DINCT @EXTRAM      POINTER TO LENGTH LOCATION ON RAM
2BD8 BD5CB0         772         DST  RAM(@EXTRAM),@ARG POINTER TO 1ST TOKEN ADD.
2BDB 2E
2BDC B25C7F         773         RB   @ARG,7         ENSURE BREAKPOINT BIT IS OFF
2BDF 935C           774         DDEC @ARG           WHERE LENGTH IS STORED
2BE1 BC09B0         775         ST   RAM(@ARG),@VARC+1 LENGTH
2BE4 5C
2BE5 9009           776         INC  @VARC+1        TEXT LENGTH + 1 LENGTH BYTE
                    777  *                          = # OF BYTE TO DELETE
2BE7 8608           778         CLR  @VARC          1ST BYTE OF A DOUBLE ADDRESS
2BE9 BD0032         779         DST  @ENLN,@VARO    LAST ADD. OF LINE-#-BUFFER ON RAM
                    780  *
                    781  **      CHECK THROUGH THE LINE # BUFFER TO SEE IF THE TEXT
                    782  **      LOCATION IS AFTER THE REPLACED ONE.  IF YES, THE
                    783  **      LOCATION IS NOW @VARY DOWN.
                    784  *
2BEC 9100           785         DINC @VARO          ADJUST FOR LATER DDECT
                    786  *
2BEE 9700           787  DEREA  DDECT @VARO
2BF0 BD06B0         788         DST  RAM(@VARO),@VARY+2  POINTER TO 1ST TOKEN ADD.
2BF3 00
                    789  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    790  *      THE FOLLOWING LINE ADDED BY STAN HUME ON 06/19/80
                    791  *      TO FIX PROBLEM OF UPDATING LINE POINTERS CORRECTLY
                    792  *      WHEN A BREAKPOINT IS SET ON A LINE
                    793  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
2BF4 B2067F         794         RB   @VARY+2,7      CLEAR POSSIBLE BREAKPOINT!!!!!
2BF7 9306           795         DDEC @VARY+2        WHERE LENGTH IS STORED
2BF9 C55C06         796         DCH  @VARY+2,@ARG   WAS THIS ENTERED AFTER THE TO-BE
                    797  *                          -DELETED LINE STATEMENT?
2BFC 4C02           798         BR   DEREB          THE TO-BE-DELETED AFTER THIS
                    799  *
                    800  **      MOVE DOWN BY @VARC+1.
                    801  *
2BFE A1B000         802         DADD @VARC,RAM(@VARO)   UPDATE POINTER
2C01 08
2C02 9700           803  DEREB  DDECT @VARO         BACK TO POINTER TO THE LINE #
2C04 D53000         804         DCEQ  @VARO,@STLN   LAST LINE ON RAM?
2C07 4BEE           805         BR   DEREA          NO.
                    806  *
                    807  **      WRITE THE NEW TEXT AFTER THE LAST TEXT LOCATION,
                    808  **      AND PUT THE 1ST ADDRESS IN LINE # BUFFER.
                    809  *
2C09 BD065C         810         DST  @ARG,@VARY+2
2C0C BD0006         811         DST  @VARY+2,@VARO
2C0F 9300           812         DDEC @VARO
2C11 A10608         813         DADD @VARC,@VARY+2 POINTER TO 1ST TOKEN (DELETE TEX
2C14 9306           814         DDEC @VARY+2        POINTER TO 1ST DELETE TOKEN(LENGT
2C16 BD5C00         815         DST  @VARO,@ARG     1ST ADDRESS TO MOVE DOWN
2C19 A55C30         816         DSUB @STLN,@ARG
2C1C 915C           817         DINC @ARG           # OF BYTES TO MOVE DOWN
2C1E 06201E         818         CALL MVDN           DELETE THE TEXT.
```

```
2C21 A13008   819          DADD @VARC,@STLN  NEW STARTING POINTER ON RAM
2C24 A13208   820          DADD @VARC,@ENLN  NEW LAST POINTER TO LINE-#-BUFFER
              821  *                          ON RAM
2C27 A12E08   822          DADD @VARC,@EXTRAM NEW POINTER TO LENGTH LOCATION
              823  *                          OF THE REPLACED LINE TEXT
              824  *                          (OLD TEXT BEEN DELETED)
              825  AUTO$2
2C2A 00       826          RTN
```

```
                    828  ********************************************************
                    829  **      SUBROUTINE TO GET THE PARAMETERS OF 'NUMBER'
                    830  ********************************************************
                    831  AUTON
 2C2B BF1400        832    DST   100,@NBC           Set default start line #
 2C2E 64
 2C2F BF1E00        833    DST   10,@NBD              and default line increment
 2C32 0A
 2C33 BE082C        834    ST    COMMA,@VARC        Select default separator
                    835  AUTO1
 2C36 9320          836    DDEC @VARW               Not to miss first character
 2C38 062C75        837    CALL GETNB               Get first non-blank character
 2C3B 6C2A          838    BS    AUTO$2             Return on end-of-line
 2C3D 062EF9        839    CALL GETLN               Try to get start line # in
 2C40 8E0C4C        840    $IF @BYTE .EQ. 0 THEN    BYTE indicates presence of #
 2C43 4F
 2C44 8E004C        841      $IF @VARO .NE. 0 GOTO STK1  If not, "," separator ?
 2C47 4D
 2C48 062C75        842      CALL GETNB             Skip spaces again
 2C4B 4C65          843      BR    AUTO$1           If so, we might have an increment
                    844  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    845  *       CHANGED FROM:
                    846  *         B   SCANER
                    847  *       BY STAN HUME 9/6/79     TO SAVE BYTES
                    848  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
 2C4D 4016          849  STK1    BR    SCANER
                    850  *
                    851    $END IF
 2C4F BD1444        852    DST   @BUFFY,@NBC        Get start line # in NBC
 2C52 8E004C        853    $IF @VARO .EQ. 0 THEN    Followed by legal separator ?
 2C55 60
 2C56 062C75        854       CALL GETNB            Get first non-blank for separator
 2C59 6C2A          855       BS    AUTO$2          Version 'command # <CR>'
 2C5B BC0E09        856       ST    @VARD,@VAR4     Don't allow space = legal seperat
 2C5E 6C65          857       BS    AUTO$1          Set means found numeric digit
                    858    $END IF
 2C60 062C7A        859    CALL GETNB2              Skip spaces for possible version
 2C63 6C2A          860    BS    AUTO$2               'command  , <CR>' or
                    861    $                          'command  #, <CR>'
                    862  AUTO$1
 2C65 062EF9        863    CALL GETLN               Get second argument
 2C68 8E0C6C        864    $IF @BYTE .EQ. 0 GOTO STK1  If not there...error.....
 2C6B 4D
 2C6C BD1E44        865    DST   @BUFFY,@NBD        Else save it in NBD
 2C6F 8E006C        866    $IF @VARO .EQ. 0 GOTO STK1  Only end-of-line is legal no
 2C72 4D
 2C73 4F4F          867    BR    ENDERR             Check for that then
                    868  *                          Exit through ENDERR
```

```
                870  ********************************************************
                871  **      SUBROUTINE TO GET A NON-BLANK CHARACTER FROM LINE
                872  ********************************************************
                873  GETNB
                874    $REPEAT
2C75 062CA6     875      CALL GETCHR           GET A CHARACTER
2C78 6C80       876      BS   RTNSET           END OF LINE
                877  GETNB2
2C7A D64220     878    $UNTIL @CHAT .NE. SPACE
2C7D 6C75
2C7F 00         879    RTN                     NO
                880  *
                881  *       RETURN WITH CONDITION BIT SET
                882  *
                883  RTNSET
2C80 D40000     884    CEQ  @0,@0              SET CONDITION BIT
2C83 01         885    RTNC
                886  *
                887  ********************************************************
                888  **      SUBROUTINE TO CHECK FOR STACK FULL
                889  **      AND STACK THE CHARACTER THAT IS IN CHAT.
                890  ********************************************************
                891  STKCHR
2C84 C73803     892    $IF @RAMPTR .DH. CRNEND GOTO STKERR  STACK FULL ERROR
2C87 BE6C91
2C8A 9138       893    DINC @RAMPTR
2C8C BCB038     894    ST   @CHAT,RAM(@RAMPTR)
2C8F 42
2C90 00         895    RTN
                896  *
                897  STKERR
2C91 DA8088     898    $IF .BIT5 @FLAG .EQ. 0 GOTO STK1 EDIT mode ????
2C94 206C4D
2C97 8744       899    DCLR @BUFFY         DON'T PRINT LINE NUMBER
2C99 062D99     900    CALL ERR$$
2C9C 213F       901    DATA #MSG19         INPUT LINE TOO LONG
2C9E 4F4D       902    BR   PTLNR1
                903  *
                904  PUSHCH
2CA0 90B002     905    INC  RAM(@STPT)         ENTRY FOR 'NUMC' AND 'STRING'
2CA3 062C84     906    CALL STKCHR
                907  ********************************************************
                908  **      SUBROUTINE TO GET A CHARACTER FROM LINE
                909  ********************************************************
                910  GETCHR
2CA6 C5202A     911    $IF @VARW .DH. @VARA GOTO RTNSET
2CA9 6C80
2CAB BC42B0     912    ST   RAM(@VARW),@CHAT   NO, PUT CHARACTER IN @CHAT
2CAE 20
2CAF D6427F     913    $IF @CHAT .NE. EDGECH THEN  LOOK OUT FOR EDGE OF SCREEN
2CB2 6CBA
2CB4 A64260     914      S    OFFSET,@CHAT     SCREENCHARACTER INTO ASCII
2CB7 9120       915      DINC @VARW
2CB9 00         916      RTN
                917    $END IF
```

```
                *
2CBA A32000       DADD 4,@VARW            SKIP TO NEXT LINE
2CBD 04
2CBE 4CA6         BR   GETCHR             (UNCONDITIONAL)
                *
                *
                *
                UNQSTR
2CC0 9138         DINC @RAMPTR            PUSH UNQOTED STRING TOKEN
2CC2 BEB038       ST   UNQST$,RAM(@RAMPTR)
2CC5 C8
2CC6 062020       CALL GETLNB             SET UP LENGTH BYTE
                  $REPEAT
2CC9 C73803         $IF @RAMPTR .DH. CRNEND THEN Watch out...overflow
2CCC BE4CD9
2CCF 062C7A           CALL GETNB2         Next non-blank character has
2CD2 6CEB             BS   UNQ$1            to be end of line or
2CD4 D44213           $IF @CHAT .EQ. @VAR8 GOTO UNQ$1    delimiter
2CD7 6CEB
                    $END IF               Fall through means bomb off
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                * FOLLOWING 3 LINES ARE ADDED FOR LOWERCASE CHAR. SET IN
                * 99/4A  3/3/81
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
2CD9 8E8082           $IF @ALLUP .NE. 0 THEN
2CDC 6CE1
2CDE 063450             CALL PROP               Translate lower to upper
                *                               Ignore the condition bit
                      $END IF
2CE1 062CA0         CALL PUSHCH           PUSH CHARACTER AND GET NEXT
2CE4 6CEB           BS   UNQ$1            END OF LINE
2CE6 D44213       $UNTIL @CHAT .EQ. @VAR8
2CE9 4CC9
                UNQ$1
2CEB D6B038       $IF RAM(@RAMPTR) .EQ. SPACE THEN
2CEE 204CFB
2CF1 9338           DDEC @RAMPTR          Trace back trailing spaces
2CF3 92B002         DEC  RAM(@STPT)         and update count
2CF6 D53802         $IF @RAMPTR .DNE. @STPT GOTO UNQ$1 Don't overtrace
2CF9 4CEB
                  $END IF
2CFB 00           RTN
                *
                ***********************************************************************
                **     SUBROUTINE TO CONVERT BINARY INTO ASCII
                ***********************************************************************
                CSINT
2CFC 874A         DCLR @FAC               Start with clean FAC
                CSINT2
2CFE 0F1B         XML  PGMCH
2D00 A64230       S    >30,@CHAT          Subtract ASCII value for "0"
2D03 CA420A       $IF @CHAT .L. 10 THEN   Valid numeric
2D06 6D20
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *      CHANGED FROM:
```

```
                964  *    CLR   @VAR0                                          *
                965  *    ST    @CHAT,@VAR0+1    Save number in double byte loc.*
                966  *    DMUL  10,@FAC          Multiply previous result      *
                967  *    $IF @FAC+1 .NE. 0 GOTO RTNSET Overflow ?????       *
                968  *    DST   @FAC+2,@FAC      Get result back down          *
                969  *    DADD  @VAR0,@FAC       Add current digit             *
                970  *       TO FIX BUG OF DIM A(65536) BEING THE SAME AS      *
                971  *       DIM A(0).   ALSO TO SAVE BYTES                    *
                972  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><*
2D08 AB4A00     973       DMUL  10,@FAC          Multiply previous result
2D0B 0A
2D0C 8F4A4C     974       $IF   @FAC .DNE. 0 GOTO RTNSET Exit on overflow
2D0F 80
2D10 BC4B42     975       ST    @CHAT,@FAC+1     Put in current character
2D13 A14A4C     976       DADD  @FAC+2,@FAC      Add in previous result
2D16 0C6C80     977       $IF   .CARRY. GOTO RTNSET  If overflow-return
                978  *<><><><><><><><><><>CHANGES END HERE<><><><><><><><><><><*
2D19 D24A00     979       $IF @FAC .LT. 0 GOTO RTNSET Integer > 32767
2D1C 4C80
2D1E 4CFE       980       BR    CSINT2           And loop until done
                981     $END IF
2D20 A24230     982     A     >30,@CHAT
                983  CSINT4
2D23 00         984     RTN                      Also used somewhere else
                985  *
```

```
                    987  *
                    988  ***************************************************
                    989  **     SUBROUTINE TO PRINT OUT ERROR MESSAGES
                    990  ***************************************************
                    991  WARN$$
                    992  *<><><><><><><><><><><><><><><><><><><><><><><><><><>
                    993  *   Delete these line for 99/4A without equation calculat
                    994  *    by CAROLYN LEE 2/26/81
                    995  *    Also add an unconditional CALL SCROLL
                    996  * $IF @CALCFL .EQ. 0 THEN        If not calculator
                    997  *   CALL SCROLL
                    998  * $END IF
                    999  *<><><><><><><><><><><><><><><><><><><><><><><><><><>
2D24 064D00        1000    CALL SCROLL
2D27 31000A        1001    MOVE 10 FROM ROM(MSG0) TO RAM(NLNADD) '* WARNING-'
2D2A A2E220
2D2D 22
2D2E 8876          1002    FETCH @EXP$           1ST BYTE OF LENGTH ADD.
2D30 8877          1003    FETCH @EXP$+1         2ND BYTE OF LENGTH ADD.
2D32 064D00        1004    CALL SCROLL
                   1005  ERR$1
2D35 8674          1006    CLR @KEYBD              CONSOLE KEYBOARD
2D37 060036        1007    CALL TONE2              WAKE UP THE USER
2D3A BF2002        1008    DST  NLNADD+2,@VARW     STARTING DISPLAY LOCATION
2D3D E4
2D3E 330001        1009    MOVE 1 FROM ROM(@EXP$) TO @ARG+1 LENGTH OF MESSAGE
2D41 5D0000
2D44 76
2D45 865C          1010    CLR  @ARG
2D47 325CB0        1011    MOVE @ARG FROM ROM(1(EXP$)) TO RAM(@VARW) 'MESSAGE'
2D4A 200001
2D4D 76
2D4E A1205C        1012    DADD @ARG,@VARW         START ADDRESS FOR REST OF MESSA
2D51 D77621        1013    $IF  @EXP$ .DEQ. MSG21 THEN     "I/O ERROR xxy"
2D54 134D6C
2D57 9120          1014      DINC @VARW            ONE BLANK SPACE
2D59 BD5FE0        1015      DST  RAM(4(PABPTR)),@ARG+3   GET BOTH BYTES OF CODE
2D5C 0404
2D5E 865E          1016      CLR  @ARG+2                  CLEAR MSBYTE OF FIRST O
2D60 062FAF        1017      CALL DISO                    DISPLAY FIRST BYTE
2D63 BC5F60        1018      ST   @ARG+4,@ARG+3           GET SECOND BYTE
2D66 E65F05        1019      SRL  @ARG+3,5                GET BITS OF INTEREST
2D69 062FAF        1020      CALL DISO                    DISPLAY SECOND BYTE
                   1021    $END IF
2D6C DA8088        1022    $IF .BIT5 @FLAG .EQ. 0 THEN
2D6F 204D96
2D72 8F446D        1023      $IF @BUFFY .DNE. 0 THEN  NO IMPERATIVE STMTS
2D75 96
2D76 C72002        1024        $IF @VARW .DH. >2F5 THEN
2D79 F54D83
2D7C 064D00        1025          CALL SCROLL       LINE # WILL BE ON NEXT LINE
2D7F BF2002        1026          DST  NLNADD+2,@VARW   DISPLAY LOCATION FOR ' IN
2D82 E4
                   1027        $END IF
2D83 BFE001        1028        DST  :IN:+>6060,RAM(1(VARW))      ' IN '
```

```
2D86 20A9AE
2D89 A32000           DADD 4,@VARW              DISPLAY LOCATION FOR LINE #
2D8C 04
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                 *       CHANGED FROM:
                 *          ST    @CHAT,RAM(WRNTP)
                 *          CALL  ASC
                 *          ST    RAM(WRNTP),@CHAT
                 *       BY STAN HUM  9/6/79    TO SAVE BYTES
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
2D8D BC7642           ST    @CHAT,@EXP$
2D90 064D0A           CALL ASC                  DISPLAY THE LINE #
2D93 BC4276           ST    @EXP$,@CHAT
                    $END IF
                   $END IF
2D96 054D00        B     SCROLL
                 *
                 ERR$$
2D99 8876          FETCH @EXP$              GET LENGTH ADDRESS
2D9B 8877          FETCH @EXP$+1               BEFORE ENTERING A NEW LEVEL
2D9D D73E03        $IF  @SYMTAB .DEQ. CRNBUF THEN      If in function prescan
2DA0 204DA7
2DA3 BD3EA3            DST  RAM(CRNBUF+2),@SYMTAB    Restore  s.t. pointer
2DA6 22
                   $END IF
                 * Reset stack if NOT edit message (it returns)
                 * Also resets if in calculator mode (FLAG is cleared)
2DA7 DA8088        $IF .BIT5 @FLAG .EQ. 0 THEN
2DAA 204DB0
2DAD BE738A            ST   RSTK+2,@SUBSTK    Initialize subr. stack
                 *                          Allow 1 entry for GROM program
                 *                             OR calculator return address
                   $END IF
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                 *   Delete these line for 99/4A without equation calculator
                 *     by CAROLYN LEE 2/26/81
                 *   Replace by CALL SCROLL
                 * $IF @CALCFL .EQ. 0 THEN      If not calculator
                 *   CALL SCROLL                Scroll the screen
                 * $SELSE
                 *   CALL SCRO10                Only clear the bottom line
                 * $END IF
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
2DB0 064D00        CALL SCROLL                 Scroll the screen
2DB3 BEA2E2        ST   :*:+>60,RAM(NLNADD)    PUT :*: IN FRONT OF MESSAGE
2DB6 8A
                 *
2DB7 062D35        CALL ERR$1               ENTER WARNING WITHOUT WARNING
                 *
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                 *   Delete these line for 99/4A without equation calculator
                 *     by CAROLYN LEE 2/26/81
                 * $IF @CALCFL .EQ. 0 THEN      If not calculator
                 * $END IF     (at ERR$$1 - 1)
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
```

```
2DBA DA8088       TBR  @FLAG,5       EDITING MODE?
2DBD 20
2DBE 4D23         BR   CSINT4        YES
2DC0 06201C       CALL CHRTAB              LOAD CHARACTER TABLE
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
            *      CHANGED FROM:
            *        $IF @BUFFY .DNE. 0 THEN
            *          DST   @STVSPT,@VSPTR
            *          DCLR @BUFFY
            *          CALL CLSALL
            *        $END IF
            *      BY STAN HUME 9/6/79     TO FIX KILLING THE STACK
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
2DC3 8F446D       $IF @BUFFY .DNE. 0 THEN
2DC6 CE
2DC7 8744           DCLR @BUFFY            No line # print (error in CLSAL
2DC9 064012         CALL CLSALL            TERMINATE THE PROGRAM
2DCC 9344           DDEC @BUFFY            PERMIT STACK CLEAN-UP BELOW
                  $END IF
2DCE 8E8089       $IF @GROMFL .NE. 0 GOTO RTNSET
2DD1 4C80
            *                             GROM PROGRAM RETURNS WITH CONDITI
            *                               SET ON ERROR
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
            *      CHANGED FROM:
            *        $IF @VSPTR .NOT. .DLE. @STVSPT THEN
            *      BY STAN HUME 9/6/79     TO MAKE IT READABLE
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
2DD3 C56E24 ERR$$1 $IF @VSPTR .DH. @STVSPT THEN If not stack end
2DD6 4DF1
2DD8 CEE002          $IF RAM(2(VSPTR)) .LE. >65 THEN If number or stri
2DDB 6E656D
2DDE E3
2DDF 0F18              XML  VPOP             Get rid of it
2DE1 4DD3              BR   ERR$$1
                     $END IF
2DE3 D6E002          $IF RAM(2(VSPTR)) .EQ. >68 THEN  If UDF
2DE6 6E684D
2DE9 F1
2DEA 874E              DCLR @FAC+4           To skip part of DELINK
2DEC 064D14            CALL DELINK           Delink the s.t. entry
2DEF 4DD3              BR   ERR$$1
                     $END IF
                   $END IF
            *
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
            *   Delete these line for 99/4A without equation calculatc
            *     by CAROLYN LEE 2/26/81
            *      $IF @CALCFL .NE. 0 THEN If calculator
            *        ST >FF,@ERRCOD        Indicate error to calc
            *        RTN                      And go back to it
            *      $END IF
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
            *      CHANGED FROM:
```

```
                1125  *         B    MAO                                          *
                1126  *       BY STAN HUME 9/6/79     TO KILL STACK AFTER CLEAN-UP *
                1127  *                               IN CASE OF  '10 GOSUB 10'    *
                1128  *                               AND SAVE ONE BYTE            *
                1129  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
2DF1 8F446D     1130          $IF  @BUFFY .DNE. 0 THEN  If in program mode          1
2DF4 F8
2DF5 BD6E24     1131            DST  @STVSPT,@VSPTR  Kill the stack                  1
                1132          $END IF
2DF8 4012       1133          BR   MAO               Else if BASIC, reinitialize
```

```
                    1135  ********************************************************
                    1136  *
                    1137  *       LLIST - Lists one program line on the screen.  The
                    1138  *               entrypoint to the line is given in STPT.
                    1139  *
                    1140  *       In this routine, FAC+2 is used as a flag to indica
                    1141  *       that the most recent character outputted was an
                    1142  *       alfanumerical character.  If the next character is
                    1143  *       also an alfanumerical character, the two are
                    1144  *       separated by a space.
                    1145  *
                    1146  ********************************************************
                    1147  LLIST
 2DFA 06401A        1148    CALL OUTREC               Make room for a new line
 2DFD BD5EB0        1149    DST  RAM(@NBC),@ARG+2     Prepare for line # printing
 2E00 14
 2E01 062F9A        1150    CALL OUTLN                Display current line in free for
 2E04 BD2008        1151    DST  @CCPADD,@VARW        Copy position for editing
 2E07 9120          1152    DINC @VARW                Leave room for space
 2E09 BD2CE0        1153    DST  RAM(2(NBC)),@VAR81     Pick up entrypoint in code
 2E0C 0214
 2E0E B22C7F        1154    RB   @VAR81,7             Reset the BREAKPOINT bit
                    1155  LLIS$0
                    1156  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    1157  *       CHANGED FROM:
                    1158  *         ST  : :,@FAC+3
                    1159  *         CLR @FAC+2
                    1160  *       BY STAN HUME 9/6/79     TO SAVE BYTES
                    1161  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><
 2E11 BF4C00        1162    DST  : :,@FAC+2           Clear blank fill and set space
 2E14 20
                    1163  LLI$12
 2E15 062F58        1164    CALL DSPPG2               Get next token on line
 2E18 6F11          1165    BS   LLIS$9               Exit on end of line
 2E1A 8E4D6E        1166    $IF  @FAC+3 .NE. 0 THEN
 2E1D 27
 2E1E C04D42        1167      EX   @CHAT,@FAC+3
 2E21 062FE3        1168      CALL DSPCHR
 2E24 C04D42        1169      EX   @CHAT,@FAC+3
                    1170    $END IF
                    1171  LLI$15
                    1172    $
                    1173    $     Next thing to determine is whether or not we need
                    1174    $     a space for separation with the next stuff
                    1175    $
 2E27 864D          1176    CLR  @FAC+3               Assume we'll get an alphanumeri
 2E29 CA42B3        1177    $IF @CHAT .HE. COMMA$ THEN Figure out separator range
 2E2C 4E33
 2E2E CA42C8        1178      $IF @CHAT .L. NUMCO$ GOTO LLIS#2
 2E31 4E4B
                    1179    $END IF
 2E33 BE4D20        1180    ST   : :,@FAC+3             Prepare for alfa indication
 2E36 8E4C6E        1181    $IF @FAC+2 .NE. 0 THEN alfanum-alfanum combination
 2E39 4B
 2E3A D65420        1182      $IF @FAC+10 .NE. : : THEN Don't output 2 spaces
```

```
:3D 6E4B
:3F BC4C42         ST    @CHAT,@FAC+2     Save CHAT somewhere                2
:42 BE4220         ST    : :,@CHAT        And display a space                2
:45 062FE3         CALL  DSPCHR                                              2
:48 BC424C         ST    @FAC+2,@CHAT     Retrieve CHAT                      2
                 $END IF
               $END IF
             LLIS$2
:4B C04C4D     EX    @FAC+3,@FAC+2     Could be for the next time too
               $
               $     That takes care of all the extra spaces we might need
               $
:4E DA4280     $IF .BIT7 @CHAT .EQ. 0 THEN just copy variable names      i
:51 4E5F
                 $REPEAT
:53 062F55         CALL  DSPPGM           Copy each character                2
:56 6F11           BS    LLIS$9           And exit on end of line            2
:58 DA4280       $UNTIL .BIT7 @CHAT .EQ. 1                                   1
:5B 6E53
:5D 4E27         BR    LLI$15                                                1
               $END IF
:5F D642C8     $IF @CHAT .NE. NUM$ THEN                                      1
:62 6E6C
E64 D642C7       $IF @CHAT .NE. STRIN$ GOTO LLIS$3                           :
E67 4E96
:69 062FE0       CALL DSPQUO            Display first quote of string       :
               $END IF
               $
               $     This place is the general location for strings
               $     both quoted and unquoted.
               $
E6C 0F1B       XML   PGMCH              Get string length in CHAT
E6E BC4A42     ST    @CHAT,@FAC         Copy in temporary space
E71 8E4A6E     $WHILE @FAC .NE. 0 This also takes care of empty strings
E74 8A
E75 0F1B         XML   PGMCH            Get next character
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
             *       CHANGED FROM:
             *         $IF @CHAT .EQ. QUOTE THEN
             *           CALL DSPCHR
             *         $END IF
             *         CALL DSPCHR
             *       BY STAN HUME 9/6/79      TO FIX DISPLAYING "" IN DATA
             *                                STATEMENTS
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
:E77 8E4C4E      $IF @FAC+2 .EQ. 0 THEN Alpha means unquoted string
:E7A 83
:E7B D64222        $IF @CHAT .EQ. QUOTE THEN
:E7E 4E83
:E80 062FE3          CALL DSPCHR         Display two quotes for one
                   $END IF
                 $END IF
.E83 062FE3      CALL DSPCHR            Display second quote or normal chr
:E86 924A        DEC  @FAC              Update string length;get next toke
:E88 4E71      $SEND WHILE              We also fall through on 0 here
```

```
2E8A 8E4C4E  1230    $IF @FAC+2 .NE. 0 GOTO LLIS$1 non-alfa end means extra q
2E8D F5
2E8E 062FE0  1231    CALL DSPQUO              Display closing quote
             1232  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
             1233  *  INSERT FOLLOWING LINE TO FIX PROBLEM WITH
             1234  *  IF "XXX" THEN XX ->(AFTER LISTING) IF "XXX"THEN XX
             1235  *  FOR 99/4A 3/4/81
             1236  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
2E91 BE4C20  1237    ST  : :,@FAC+2           Cause space before following alpt
2E94 4EF5    1238    BR   LLIS$1
             1239    $
             1240    $    Try to decode line numbers and keywords
             1241    $
             1242  LLIS$3
2E96 D642C9  1243    $IF @CHAT .EQ. LN$ THEN Decode line #
2E99 4EAA
2E9B 0F1B    1244      XML  PGMCH             Get the high order byte first
2E9D BC5E42  1245      ST   @CHAT,@ARG+2      Store in BE,BE+1
2EA0 0F1B    1246      XML  PGMCH             I.e. BE+1 will contain low order
2EA2 BC5F42  1247      ST   @CHAT,@ARG+3        information as collected here
2EA5 062F9A  1248      CALL OUTLN             Display the actual information
2EA8 4EF5    1249      BR   LLIS$1            And continue
             1250    $END IF
             1251    $
             1252    $    Now it has to be a normal keyword
             1253    $
2EAA 874A    1254    DCLR @FAC                FAC will contain the offset in t
2EAC BF4E00  1255    DST  1,@FAC+4              token address table; FAC+3 the
2EAF 01
             1256    $                          current token length
             1257  LLIS$4
             1258    $
             1259    $    Use FAC+8 as a pointer within the token table
             1260    $
2EB0 330002  1261    MOVE 2 FROM ROM(KEYTAB(FAC)) TO @FAC+8
2EB3 52285C
2EB6 4A
             1262    $REPEAT
2EB7 A1524E  1263      DADD @FAC+4,@FAC+8   Point to actual token value
2EBA 330002  1264      MOVE 2 FROM ROM(@FAC+8) TO @FAC+6
2EBD 500000
2EC0 52
             1265      $
             1266      $  This will cause the current keyword to go to FAC+6
             1267      $  and the possible end of table to FAC+7
             1268      $       WATCH OUT ... Here we use the fact that no er
             1269      $  in the token table is empty !!!!!!!
             1270      $
2EC1 D45042  1271      $IF @FAC+6 .EQ. @CHAT GOTO LLIS$5 Get out if found
2EC4 6ED3
2EC6 9152    1272      DINC @FAC+8            Skip token value
2EC8 D651FF  1273    $UNTIL @FAC+7 .EQ. >FF Found end of current entry
2ECB 4EB7
2ECD 954A    1274    DINCT @FAC               Get next entry in table
2ECF 914E    1275    DINC  @FAC+4             Update string length
```

```
2ED1 4EB0     1276    BR    LLIS$4              And continue for the next entry
              1277    $
              1278    $     At this point FAC+8 contains a pointer to the
              1279    $     correct table entry
              1280    $
              1281  LLIS$5
2ED3 A5524E   1282    DSUB @FAC+4,@FAC+8        Backup to actual string start
2ED6 BC4A42   1283    ST    @CHAT,@FAC          Save current value of token
              1284    $
              1285    $     Make the actual transfer of the keyword
              1286    $
              1287  LLIS$6
2ED9 330001   1288    MOVE 1 FROM ROM(@FAC+8) TO @CHAT
2EDC 420000
2EDF 52
              1289    $
              1290    $     And output the thus found character
              1291    $
2EE0 062FE3   1292    CALL DSPCHR               Display the character on the screen
2EE3 9152     1293    DINC @FAC+8               Update FAC+8 for next reference
2EE5 924F     1294    DEC  @FAC+5               Count number of characters
2EE7 4ED9     1295    BR    LLIS$6              Always less than 255
2EE9 CA4AB3   1296    $IF @FAC .L. COMMA$ GOTO LLIS$0 Master stuff always space
2EEC 4E11
2EEE D64AFD   1297    $IF @FAC .EQ. NUMBE$ THEN                                    i
2EF1 4EF5
2EF3 864C     1298      CLR  @FAC+2             Avoid spaces behind here           i
              1299    $END IF
              1300  LLIS$1
2EF5 864D     1301    CLR  @FAC+3
2EF7 4E15     1302    BR    LLI$12              Continue for next keyword
```

```
                    1304  *
                    1305  *     Convert line number from ASCII to binary
                    1306  *          for all kinds of commands
                    1307  *
2EF9 8600           1308  GETLN  CLR  @VARO          ASSUME A CHARACTER '-', ', ', OR '
2EFB 860C           1309         CLR  @BYTE          # OF LINE # DIGIT
2EFD CA4230         1310         $IF  @CHAT .HE. :0: THEN
2F00 4F0A
2F02 CE4239         1311          $IF @CHAT .LE. :9: THEN
2F05 6F0A
2F07 062F12         1312           CALL PTLNE         LINE #
                    1313          $END IF
                    1314         $END IF
2F0A D44208         1315         CEQ  @VARC,@CHAT    FOLLOWED BY '-'?
2F0D 6F11           1316         BS   GETLN2         YES
2F0F 9000           1317         INC  @VARO          END OF LINE OR ILLEGAL CHARACTER
                    1318  LLIS$9
2F11 00             1319  GETLN2 RTN
                    1320  *
                    1321  *
                    1322  *
                    1323  *
                    1324  *
                    1325  *
                    1326  PTLNE
2F12 874A           1327    DCLR @FAC
                    1328  *
                    1329  PTLN$1
2F14 BC0142         1330    ST   @CHAT,@VARO+1      Make that two bytes for DADD
2F17 A60130         1331    S    :0:,@VARO+1
2F1A CA010A         1332    $IF @VARO+1 .L. 10 THEN Only perform if legal
2F1D 6F3B
2F1F AB4A00         1333      DMUL 10,@FAC
2F22 0A
2F23 8E4B4F         1334      $IF @FAC+1 .NE. 0 GOTO PTLNER No overflow allowed
2F26 43
2F27 BD4A4C         1335      DST  @FAC+2,@FAC      Get result back in FAC,FAC+1
2F2A 8600           1336      CLR  @VARO            Clear MSByte of VARO,VARO+1
2F2C A14A00         1337      DADD @VARO,@FAC       Compute result with next digit
2F2F 900C           1338      INC  @BYTE            # OF DIGITS
2F31 D24A00         1339      $IF @FAC .LT. 0 GOTO PTLNER >32767
2F34 4F43
2F36 062CA6         1340      CALL GETCHR           No, get next character
2F39 4F14           1341      BR   PTLN$1           Continue while not end of line
                    1342    $END IF
2F3B BD444A         1343    DST  @FAC,@BUFFY        Copy result in BUFFY
2F3E 8F4A6F         1344    $IF  @FAC .DEQ. 0 GOTO PTLNER   Zero is illegal l.n.
2F41 43
2F42 00             1345    RTN
                    1346  *
                    1347  *
2F43 064D00         1348  PTLNER CALL SCROLL
2F46 8744           1349         DCLR @BUFFY
2F48 062D99         1350         CALL ERR$$
2F4B 20D9           1351         DATA #MSGBLN       BAD LINE NUMBER
```

```
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *       CHANGED FROM:
                *         B    ILL1
                *       BY STAN HUME 9/6/79     TO SAVE BYTES
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
2F4D 4018       PTLNR1 BR    ILL1
                *
                *
                ENDERR
2F4F 062C75            CALL  GETNB            IF NOT END OF LINE
2F52 4C4D              BR    STK1
2F54 00                RTN
                *
                *
                *
2F55 062FE3     DSPPGM CALL  DSPCHR
2F58 0F1B       DSPPG2 XML   PGMCH
2F5A 8E42              CZ    @CHAT
2F5C 01                RTNC
```

```
                *
                *      LOCATE LINE
                **
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                *      CHANGED FROM:
                *        $IF @ENLN .DNE. @STLN THEN
                *              :
                *              :
                *        $END IF
                *        B   MAO
                *      BY STAN HUME 9/6/79    TO SAVE BYTES
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                *      CHANGED FROM:
                *SEEONE
                * $IF @ENLN .DEQ. @STLN GOTO MAO  No program - ignore all
                *      BY STAN HUME ON 06/19/80 TO NOTHING BECAUSE THE COI
                *      IS NEVER USED=>IT IS DEAD CODE!!!
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                SEETWO
2F5D BD2E32       DST   @ENLN,@EXTRAM
2F60 A72E00       DSUB  >03,@EXTRAM      Pointer to first line # in videora
2F63 03
                SEE1
                  $REPEAT
2F64 C9B02E         $IF RAM(@EXTRAM) .DHE. @BUFFY THEN Actual >= specifie
2F67 444F6F
2F6A D5B02E           DCEQ @BUFFY,RAM(@EXTRAM) Same line as specified ?
2F6D 44
2F6E 01               RTNC
                    $END IF
2F6F A72E00         DSUB >04,@EXTRAM   Next line # in videoram
2F72 04
2F73 C92E30       $UNTIL @EXTRAM .DL. @STLN Check for highest line
2F76 6F64
2F78 BD2E30       DST  @STLN,@EXTRAM Default to last line in program
2F7B 00           RTN
```

```
                    1405  *
                    1406  *       ROUTINE TO DISPLAY THE LINE NUMBERON THE SCREEN
                    1407  *
                    1408  *
                    1409  DISLN
2F7C 8661           1410    CLR  @ARG+5              Start with 0 characters
2F7E BE6767         1411    ST   ARG+11,@ARG+11      Select first address + 1
                    1412    $REPEAT
2F81 875C           1413      DCLR @ARG              Clear upper two bytes of 4-byte
2F83 9267           1414      DEC  @ARG+11           Go to next position
2F85 AF5C00         1415      DDIV 10,@ARG           Compute least significant remainde
2F88 0A
2F89 A25F30         1416      ADD  >30,@ARG+3        Always < 10 off course
2F8C BC9067         1417      ST   @ARG+3,*ARG+11    Store it in ARG
2F8F 5F
2F90 BD5E5C         1418      DST  @ARG,@ARG+2       Replace remainder by result
2F93 9061           1419      INC  @ARG+5            Update total # of characters
2F95 8F5E4F         1420    $UNTIL @ARG+2 .DEQ. 0
2F98 81
2F99 00             1421    RTN
                    1422  *
                    1423  OUTLN
2F9A 062F7C         1424    CALL DISLN               Convert from binary to chars
                    1425  OUTL$1
2F9D BC4290         1426    ST   *ARG+11,@CHAT       Prepare for display character
2FA0 67
2FA1 062FE3         1427    CALL DSPCHR
2FA4 9067           1428    INC  @ARG+11
2FA6 9261           1429    DEC  @ARG+5
2FA8 4F9D           1430    BR   OUTL$1
2FAA 00             1431    RTN
                    1432  *
                    1433  DISPLN
2FAB BD5EB0         1434    DST  RAM(@STPT),@ARG+2 Display the current line #
2FAE 02
                    1435  DISO
2FAF 062F7C         1436    CALL DISLN
                    1437  DISP$1
2FB2 BCB020         1438    ST   *ARG+11,RAM(@VARW) Get most significant character
2FB5 9067
2FB7 A2B020         1439    ADD  OFFSET,RAM(@VARW) Display the character somewhere
2FBA 60
2FBB 9120           1440    DINC @VARW
2FBD 9067           1441    INC  @ARG+11           Get next position
2FBF 9261           1442    DEC  @ARG+5            Update count
2FC1 4FB2           1443    BR   DISP$1            And loop until finished
2FC3 00             1444    RTN
```

```
                $ * * * * * * * * * * * * * * * * * * * * * * * * * * *
                $    ADAPTED FOR GROM PROGRAMS 9/18/78
                $ * * * * * * * * * * * * * * * * * * * * * * * * * * *
              SEARCH
2FC4 A72E00     DSUB  4,@EXTRAM
2FC7 04
2FC8 8E8089     $IF @GROMFL .EQ. 0 THEN
2FCB 4FD6
2FCD BD2CB0       DST  RAM(@EXTRAM),@VAR81  WHERE 1ST TOKEN IS STORED
2FD0 2E
2FD1 B22C7F       RB   @VAR81,7              RESET POSSIBLE BREAKPOINT
2FD4 4FDD       $SELSE
2FD6 330002       MOVE 2 FROM ROM(@EXTRAM) TO @VAR81 SAME FOR GROM
2FD9 2C0000
2FDC 2E
                $SEND IF
2FDD 0F1B       XML PGMCH
2FDF 00         RTN
              *
              *
              *
              DSPQUO
2FE0 BE4222     ST   QUOTE,@CHAT        DISPLAY A QUOTE
              *
              *
              *
              DSPCHR
2FE3 C40607     $IF @CCPPTR .H. @RECLEN THEN Action on end of screen
2FE6 4FEF
2FE8 06401A       CALL OUTREC           Output current record
2FEB A72000       DSUB 32,@VARW         Keep track of begin of line
2FEE 20
                $END IF
2FEF BCB008     ST   @DSRFLG,RAM(@CCPADD) Put offset on screen
2FF2 17
2FF3 A0B008     ADD  @CHAT,RAM(@CCPADD)  Add in the character
2FF6 42
2FF7 9108       DINC @CCPADD            Bump output pointer
2FF9 9006       INC  @CCPPTR            Update current line position
2FFB BC5442     ST   @CHAT,@FAC+10      FAC+10 may be used by OUTREC !!!
2FFE 00         RTN
```

```
                    1479  ********************************************************
                    1480  *       PRESCAN - Scans the entire statement or program     *
                    1481  *       building up the symbol table and checking some      *
                    1482  *       errors.                                             *
                    1483  ********************************************************
                    1484  PRGLST
2FFF BF0600         1485  PRA2    DST   10,@DFLTLM          SET DEFAULT ARRAY SIZE
3002 0A
3003 B2 088         1486          RB    @FLAG,5             IN PRESCAN MODE
3006 DF
3007 8F44           1487          DCZ   @BUFFY              IMPERATIVE?
3009 501C           1488          BR    PRA3                WITH LINE #
                    1489  *
                    1490  *******CODE FOR AN IMPERATIVE*******
                    1491  *
300B BF2C03         1492          DST   CRNBUF,@CHPTR  POINTER TO 1ST TOKEN
300E 20
300F 0F1B           1493          XML   PGMCH               1ST TOKEN
                    1494  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><*
                    1495  *   Delete these line for 99/4A without equation calculator*
                    1496  *     by CAROLYN LEE 2/26/81                              *
                    1497  *     Also add an unconditional CALL SCROLL               *
                    1498  * $IF @CALCFL .EQ. 0 THEN          If not calculator      *
                    1499  *   CALL SCROLL                                           *
                    1500  * $END IF                                                 *
                    1501  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><*
3011 064D00         1502          CALL SCROLL
3014 BE1601         1503          ST    1,@XFLAG            SET IMPERATIVE FLAG
3017 06306D         1504          CALL DIMDIM               DO PRESCAN
                    1505  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><*
                    1506  *       CHANGED FROM                                      *
                    1507  *         B     EXEC                                      *
                    1508  *       BY  STAN HUME 9/6/79  TO SAVE BYTES               *
                    1509  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><*
301A 5028           1510          BR    PRA4                EXECUTE IT.
                    1511  *
                    1512  *******CODE FOR A PROGRAM******
                    1513  *
301C 06302B         1514  PRA3    CALL DIMTAB               BUILD A SYMBOL TABLE ,RESERVE
                    1515  *                                 VARIABLE SPACES,AND CHECK SOME
                    1516  *                                 ERRORS CONCERNNING THE SYMBOL.
                    1517  *
                    1518  **      EXECUTE THE PROGRAM.
                    1519  *
301F B28088         1520          AND   >10,@FLAG           RESET ALL FLAGS BUT THE 'TRAC
3022 10
3023 BF80C0         1521          DST   >3567,@>C0       PUT AN INITIAL VALUE IN RANDOM
3026 3567
                    1522  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                    1523  *       CHANGED FROM:
                    1524  *         B     EXEC
                    1525  *       BY STAN HUME 9/6/79     TO SAVE ONE BYTE ABOVE
                    1526  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
3028 054D04         1527  PRA4    B     EXEC                EXECUTE THE PROGRAM.
```

```
                *************************************************************
                *      SUBROUTINE TO STORE SYMBOLS IN SYMBOL TABLE ALONG
                *      THEIR POINTER TO THE VARIABLE SPACE, RESERVE SPAC
                *      VARIABLES SPACE,CHECK SOME SYMBOL ERRORS
                *************************************************************
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *      CHANGED FROM:
                *DIMTAB CLR  @FORNET                INITIALIZE COUNTER OF 'FOR
                **                                  NEXT' LOOPS TO 0
                *      BY STAN HUME ON 06/19/80 TO COMBINE WITH CLR @XFLA
                *      BELOW TO SAVE TWO BYTES
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
302B 8716       DIMTAB DCLR @XFLAG                 Clear flag byte and for-nex
                *                                     counter
302D BD2E32            DST  @NUMBUF,@EXTRAM        1ST ADD. OF LINE # BUFFER
3030 A32E00            DADD 3,@EXTRAM
3033 03
3034 873E              DCLR @SYMTAB                1ST ADDRESS OF SYMBOL TABLE
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *      CHANGED FROM:
                *      $IF @GROMFL .EQ. 0 THEN
                *        DST  @STLN,@SYMPTR        INITIALIZE SYMBOL TABLE PTR
                *      $SELSE
                *        DST  @>70,@SYMPTR         INITIALIZE FOR EMPTY PROGRA
                *      $SEND IF
                *      BY STAN HUME ON 06/24/80 TO SAVE 2 BYTES
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
3036 BD4030            DST  @STLN,@SYMPTR          Assume VDP & init free spac
3039 8E8089            $IF  @GROMFL .NE. 0 THEN    But if GROM init free
303C 7041
303E BD4070              DST  @>70,@SYMPTR            space for GROM case
                       $END IF
3041 8643              CLR  @BASE                  ZERO OPTION BASE
3043 9340              DDEC @SYMPTR                BACK UP FROM LINE # BUFFER
3045 BD1840            DST  @SYMPTR,@STRSP         INITIALIZE POINTERS
3048 9318              DDEC @STRSP
304A BD1A18            DST  @STRSP,@STREND
304D B68088            SB   @FLAG,7                PRESCAN MODE
3050 80
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *      CHANGED FROM:
                *      CLR  @XFLAG                 RESET ALL FLAG BITS
                *      BY STAN HUME ON 06/19/80 TO COMBINE WITH CLR @FORN
                *      ABOVE TO SAVE BYTES.
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
3051 BD1230            DST  @STLN,@LINUM
3054 9512              DINCT @LINUM                PT TO PTR TO LAST LINE
                *
                *************************************************************
                *
3056 D52E12     DIM1   DCEQ @LINUM,@EXTRAM         SCANNED ALL LINES?
3059 5067              BR   DIM1A                  NOPE-GO ON
                *
305B 8E17              CZ   @FORNET                EQUAL NUMBER OF 'FOR'S,'NEX
305D 7066              BS   DIMRTN                 YES
```

```
305F 8744     1581          DCLR @BUFFY              ERROR MESSAGE WITHOUT LINE #
3061 062D99   1582  DIMFN   CALL ERR$$               ERROR
3064 20F9     1583          DATA #MSG18              'FOR-NEXT ERROR'
3066 00       1584  DIMRTN  RTN                      ALL DONE
              1585  *
3067 062FC4   1586  DIM1A   CALL SEARCH             GET THE TEXT OFF VDP
306A B21602   1587          AND  >02,@XFLAG          RESET ALL FLAG BITS EXCEPT OP
306D D6428A   1588  DIMDIM  CEQ  DIM$,@CHAT          'DIM'?
3070 5081     1589          BR   DIMOP               ANY STMT OTHER THAN 'DIM'
              1590  *******DIM NSION STMT
              1591  DIMDO1
3072 0631E5   1592          CALL ENTER               DECLARE THIS VARIABLE
3075 D642B3   1593          CEQ  COMMA$,@CHAT        ','?
3078 7072     1594          BS   DIMDO1              YES, LOOP
307A 50AF     1595          BR   DIM3A               Must have EOL
              1596  *
307C 062D99   1597  ENTO3   CALL ERR$$               * BAD NAME
307F 2040     1598          DATA #MSG4
3081 D6429E   1599  DIMOP   CEQ  OPTIO$,@CHAT        'OPTION'?
3084 50C5     1600          BR   DIMDEF              NO - Try DEF stmt
              1601  *******OPTION BASE STMT
3086 0631D1   1602          CALL IMPILL              IMPERATIVE?
3089 063481   1603          CALL PGMERR              'OPTION', THEREFORE MUST BE
308C DA1602   1604          TBR  @XFLAG,1            'BASE' ALREADY SET?
308F 51D6     1605          BR   DIM4A               YES, ITS AN ERROR
3091 0630B9   1606          CALL SYNCHK              Must have a 'BASE'
3094 F1       1607          DATA BASE$
3095 0630B9   1608          CALL SYNCHK              Must have a numeric constant
3098 C8       1609          DATA NUMCO$
3099 0630B9   1610          CALL SYNCHK              Must have 1-char numeric cons
309C 01       1611          DATA 1
309D 8643     1612          CLR  @BASE               ASSUME BASE 0
309F A64230   1613          SUB  >30,@CHAT           MUST BE 0 OR 1
30A2 70AA     1614          BS   DIM3                OK IF '0'
30A4 9242     1615          DEC  @CHAT               CHECK FOR A '1'
30A6 50C0     1616          BR   SYNTER              IF NO, BASE ERROR
30A8 9043     1617          INC  @BASE               OPTION BASE 1
30AA B61602   1618  DIM3    SB   @XFLAG,1            SET THE OPTION BASE FLAG
30AD 0F13     1619          XML  PGMCH               STATEMENT MUST END HERE
30AF 8E42     1620  DIM3A   CZ   @CHAT
30B1 50C0     1621          BR   SYNTER              Not EOL-error
30B3 DA1601   1622  DIM6    TBR  @XFLAG,0            Imperative?
30B6 7056     1623          BS   DIM1                No-Continue scanning
30B8 00       1624  DIM8    RTN
              1625  *
              1626  *       Syntax required token routine
              1627  *
30B9 884A     1628  SYNCHK  FETCH @FAC
30BB D4424A   1629          $IF  @CHAT .EQ. @FAC GOTO PGMERR
30BE 7481
30C0 062099   1630  SYNTER  CALL ERR$$               JUNK ON END OF LINE
30C3 202C     1631          DATA #MSG1               SYNTAX ERROR
              1632  *
              1633  *
30C5 D64289   1634  DIMDEF  CEQ  DEF$,@CHAT          'DEF' STATEMENT?
```

```
30C8 5155    1635          BR    DIMREM              NO
             1636  *
             1637  *******ALL THIS IS FOR A FUNCTION DEFINITION
             1638  *
30CA 0631D1  1639          CALL  IMPILL              CAN'T BE IMPERATIVE
30CD B61684  1640          OR    >84,@XFLAG          SET FUNCTION BIT
             1641  *                                 SET 'ENTERX' BIT
30D0 0631E5  1642          CALL  ENTER
             1643  *                          **ENTER RESETS FUNCTION BIT
30D3 DAB03E  1644          CLOG  >07,RAM(@SYMTAB)    Did function have parms?
30D6 07
30D7 7177    1645          BS    DDEF02              NO
30D9 B61680  1646          SB    @XFLAG,7            'ENTERX' CALL
30DC B68088  1647          SB    @FLAG,3             FAKE IT SO SYMBOL TABLE SEAR
30DF 08
             1648  *                                 WON'T BE MADE
30E0 0631E8  1649          CALL  ENTERW              ENTER THE PARAMETER
30E3 B28088  1650          RB    @FLAG,3
30E6 F7
30E7 0630B9  1651          CALL  SYNCHK              Complex symbol must be
30EA B6      1652          DATA  RPAR$               followed by ')='
30EB 0630B9  1653          CALL  SYNCHK
30EE BE      1654          DATA  EQUAL$
30EF 35002A  1655          MOVE  42 FROM RAM(@SYMTAB) TO RAM(CRNBUF)
30F2 A320B0
30F5 3E
             1656  *       FIGURE OUT WHY 42(ACTUALLY NEED 39, BUT WHY?)
30F6 BD00A3  1657          DST   RAM(CRNBUF+4),@VARO  GET PTR TO NAME
30F9 24
30FA 8E8089  1658          CZ    @GROMFL             GROM?
30FD 710A    1659          BS    DDEF03              NO
             1660  *       IF GROM MUST FIX UP THE NAME POINTER
             1661  *       BECAUSE THE NAME WAS MOVED TOO
30FF A5003E  1662          DSUB  @SYMTAB,@VARO       OFFSET INTO ENTRY
3102 A30003  1663          DADD  CRNBUF,@VARO        NEW LOCATION OF NAME
3105 20
3106 BDA324  1664          DST   @VARO,RAM(CRNBUF+4)      PUT IT IN
3109 00
310A BD40E0  1665  DDEF03  DST   RAM(2(SYMTAB)),@SYMPTR    RESET FREE PTR
310D 023E
310F BF3E03  1666          DST   CRNBUF,@SYMTAB      POINT INTO CRUNCH BUFFER
3112 20
3113 9340    1667          DDEC  @SYMPTR
3115 8E42    1668  DDEF06  CZ    @CHAT               END OF LINE?
3117 714F    1669          BS    DDEF12              YES
3119 DA4280  1670          $IF   .BIT7 @CHAT .EQ. 0 GOTO DDEF08
311C 712C
311E D642C8  1671          CEQ   NUM$,@CHAT          NUMERIC CONSTANT?
3121 7148    1672          BS    DDEF10              YES-SKIP IT
3123 D642C7  1673          CEQ   STRIN$,@CHAT        NO-QUOTED STRING?
3126 7148    1674          BS    DDEF10              YES SKIP IT
3128 0F13    1675          XML   PGMCH               GET NEXT CHARACTER
312A 5115    1676          BR    DDEF06
             1677  *       Jump always
312C B61680  1678  DDEF08  SB    @XFLAG,7            MAKE AN 'ENTERX'
```

```
 12F 0631ED  1679          CALL ENTERX              ENTER THE SYMBOL
             1680  *
             1681  *******RELINK TO KEEP PARAMETER AT THE BEGINNING OF THE TAB
             1682  *
3132 D73E03  1683          DCEQ CRNBUF,@SYMTAB      MAKE AN ENTRY?
3135 20
3136 7115    1684          BS   DDEF06              NOPE-DON'T RE-DO LINKS
3138 BDE002  1685          DST  RAM(CRNBUF+2),RAM(2(SYMTAB))
313B 3EA322
313E BDA322  1686          DST  @SYMTAB,RAM(CRNBUF+2)
3141 3E
3142 BF3E03  1687          DST  CRNBUF,@SYMTAB
3145 20
3146 5115    1688          BR   DDEF06              GOON
             1689  *       Jump always
3148 063488  1690  DDEF10  CALL SKPSTR              SKIP IT
314B 0F1B    1691          XML  PGMCH               GET NEXT CHARACTER
314D 5115    1692          BR   DDEF06              GO ON
314F BD3EA3  1693  DDEF12  DST  RAM(CRNBUF+2),@SYMTAB      DELINK
3152 22
3153 50B3    1694          BR   DIM6                SCAN ON
             1695  *******REM STMT
3155 D6429A  1696  DIMREM  CEQ  REM$,@CHAT          REM STATEMENT?
3158 70B3    1697          BS   DIM6                YES-SKIP IT
             1698  *******INPUT STMT
315A D64292  1699          CEQ  INPUT$,@CHAT        INPUT?
315D 71CC    1700          BS   ILLIMP              YES, CAN'T BE IMPERATIVE
             1701  *******DATA STMT
315F D64293  1702  DIMDAT  CEQ  DATA$,@CHAT         DATA STMT?
3162 5169    1703          BR   DIMSUB              NO-GO ON
3164 0631D1  1704          CALL IMPILL              IMPERATIVE?
3167 5056    1705          BR   DIM1                NO-SKIP LINE AND GET NEXT
             1706  *
3169 D64287  1707  DIMSUB  CEQ  GOSUB$,@CHAT        GOSUB?
316C 71CC    1708          BS   ILLIMP              YES, IMPERTIVE?
316E D6428C  1709  DIMFOR  CEQ  FOR$,@CHAT          FOR?
3171 51A2    1710          BR   DIMRET              NO
3173 9017    1711          INC  @FORNET             COUNTER + 1
3175 51CC    1712          BR   ILLIMP              IMPERATIVE?
             1713  *
3177 0630B9  1714  DDEF02  CALL SYNCHK
317A BE      1715          DATA EQUAL$
             1716  *******ALL OTHER STMTS
317B 8E42    1717  DIMDIF  CZ   @CHAT               END OF LINE?
317D 70B3    1718          BS   DIM6                YES
317F DA4280  1719          TBR  @CHAT,7             KEYWORD OR SYMBOL?
3182 7193    1720          BS   DIM5                SYMBOL
3184 D642C8  1721          CEQ  NUM$,@CHAT          NUMERIC CONSTANT?
3187 719B    1722          BS   DIM5A               YES-SKIP IT
3189 D642C7  1723          CEQ  STRIN$,@CHAT        NO-QUOTED STRING?
318C 719B    1724          BS   DIM5A               YES-SKIP IT
318E 0631DB  1725          CALL PGMUPT              GET NEXT CHARACTER
3191 517B    1726          BR   DIMDIF
3193 B61680  1727  DIM5    SB   @XFLAG,7            SET 'ENTERX' BIT
3196 0631ED  1728          CALL ENTERX
```

```
3199 517B    1729          BR   DIMDIF            SCAN REST OF LINE
319B 063488  1730  DIM5A   CALL SKPSTR            SKIP IT
319E 0F1B    1731          XML  PGMCH             GET THE NEXT CHARACTER
31A0 517B    1732          BR   DIMDIF            AND CONTINUE
             1733  *
31A2 D64296  1734  DIMRET  CEQ  NEXT$,@CHAT       NEXT?
31A5 51B3    1735          BR   DIMON             NO
31A7 0631D1  1736          CALL IMPILL            CAN'T BE DIRECT
31AA 9217    1737          DEC  @FORNET           COUNTER - 1
31AC D21700  1738          $IF @FORNET .LT. 0 GOTO DIMFN
31AF 5061
31B1 517B    1739          BR   DIMDIF
             1740  *
31B3 D64288  1741  DIMON   $IF  @CHAT .EQ. RETUR$ GOTO ILLIMP
31B6 71CC
31B8 D6429B  1742          $IF  @CHAT .EQ. ON$ GOTO ILLIMP
31BB 71CC
31BD D64284  1743          $IF  @CHAT .EQ. IF$ GOTO ILLIMP
31C0 71CC
31C2 D64285  1744          $IF  @CHAT .EQ. GO$ GOTO ILLIMP
31C5 71CC
31C7 D64286  1745          $IF  @CHAT .NE. GOTO$ GOTO DIMDIF
31CA 517B
31CC 0631D1  1746  ILLIMP  CALL IMPILL            IMPERATIVE?
31CF 517B    1747          BR   DIMDIF            NO
             1748  *
             1749  *
             1750  *
31D1 DA1601  1751  IMPILL  TBR  @XFLAG,0          IMPERATIVE?
31D4 70B8    1752          BS   DIM8              NO, RETURN
31D6 062D99  1753  DIM4A   CALL ERR$$             YES, ERROR
31D9 20BD    1754          DATA #MSG17            ILLEGAL STATEMENT
             1755  *
             1756  *
             1757  *
31DB D642C9  1758  PGMUPT  $IF @CHAT .EQ. LN$ THEN SKIP NUMERIC STUFF
31DE 51E2
31E0 952C    1759            DINCT @VAR81     UPDATE TOKEN POINTER
             1760          $END IF
31E2 0F1B    1761          XML  PGMCH
31E4 00      1762          RTN
```

```
              1764  **********************************************************
              1765  *                       ENTER AND ENTERX ROUTINES         *
              1766  **********************************************************
31E5 063481   1767  ENTER  CALL PGMERR            GET NEXT CHARACTER
31E8 DA4280   1768  ENTERW $IF .BIT7 @CHAT .NE. 0 GOTO SYNTER
31EB 50C0
              1769  ENTERX
31ED BE5949   1770         ST   FAC-1,@FAC+15
31F0 BD0C2C   1771         DST  @CHPTR,@NMPTR      Save pointer to name
31F3 930C     1772         DDEC @NMPTR
              1773  *
              1774  *******Accumulate the name of the symbol
              1775  *
31F5 D65958   1776  ENT01  CEQ  FAC+14,@FAC+15     Trying to save 16th char?
31F8 707C     1777         BS   ENT03              YES-ERROR
31FA 9059     1778         INC  @FAC+15            Count it
31FC BC9059   1779         ST   @CHAT,*FAC+15      Save it
31FF 42
3200 0F1B     1780         XML  PGMCH              GET NEXT CHARACTER
              1781  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
              1782  *      CHANGED FROM:                                     *
              1783  *      $IF  @CHAT .NE. 0 THEN    if not EOL              *
              1784  *        $IF  .BIT7 @CHAT .EQ. 0 GOTO ENT01  If not token *
              1785  *      $END IF                                           *
              1786  *      BY STAN HUME ON 06/19/80 TO SAVE BYTES AND TO     *
              1787  *      EXECUTE FASTER                                    *
              1788  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
3202 CE4200   1789         $IF  @CHAT .GT. 0 GOTO ENT01 Keep going if alpha
3205 71F5
3207 BD6C2C   1790         DST  @VAR81,@ARG+16     SAVE TEXT POINTER TO
320A 936C     1791         DDEC @ARG+16                PUT IN S.T. ENTRY
320C D69059   1792         $IF  *FAC+15 .EQ. DOLLAR THEN   String variable?  :
320F 245215
3212 B61610   1793              SB  @XFLAG,4       Set string flag        :
              1794         $END IF
3215 A6594A   1795         SUB  FAC,@FAC+15        Calc length of name
3218 9059     1796         INC  @FAC+15            + Offset of 1
321A D642B7   1797         $IF @CHAT .EQ. LPAR$ GOTO ENT22   If complex
321D 726B
321F DA1680   1798         TBR  @XFLAG,7           Simple - Called by ENTER?
3222 5229     1799         BR   ENT08              No, called by ENTERX
3224 DA1604   1800         TBR  @XFLAG,2           DIM statement?
3227 70C0     1801         BS   SYNTER             Yes-Error-'DIM' w/o subscript
```

```
                    1803  ********************************************************
                    1804  *                  CODE FOR SIMPLE ENTRY INTO TABLE
                    1805  *       This includes all non-dimensioned varaibles as well
                    1806  *       phony entries for no-parameter functions.  ENT09 is
                    1807  *       entry point for entering one of these phony entries.
                    1808  *       ENT10 is the code which checks for consistent use
                    1809  *       of symbols within the user's program.
                    1810  ********************************************************
3229 932C           1811  ENT08   DDEC @CHPTR              CORRECT PTR OVERSHOOT
322B BD0E2C         1812  ENT09   DST  @CHPTR,@CHSAV       SAVE CHARACTER PTR
322E 8680B8         1813          CLR  @STKMIN+1           ZERO DIMENSIONS
3231 BE10B8         1814          ST   STKMIN+1,@TOPSTK      SAVE TOP OF STACK
3234 DA8088         1815          TBR  @FLAG,3             RUN MODE?
3237 08
3238 525B           1816          BR   ENT16               YES-DON'T SEARCH TABLE
323A 0F16           1817          XML  SCHSYM              SEARCH SYMBOL TABLE
323C 525B           1818          BR   ENT16               NOT FOUND-MUST ENTER IT
323E 912C           1819          DINC @CHPTR              CORRECT PTR UNDERSHOOT
                    1820  *
                    1821  *       Common code used by SIMPLE and COMPLEX
                    1822  *       When the symbol appears in the SYMBOL
                    1823  *       TABLE.  It verifies that the declarations
                    1824  *       are the same(# of parms/dims,string,function)
                    1825  *
3240 DA1680         1826  ENT10   TBR  @XFLAG,7           CALLED BY ENTER?
3243 72E6           1827          BS   ENT05               YES-Attempted redeclaratio
3245 BC00B0         1828          ST   RAM(@FAC),@VAR0     FETCH DECLARATION
3248 4A
3249 DA1604         1829          TBR  @XFLAG,2            NO-FUNCTION NOW?
324C 52E6           1830          BR   ENT05               YES, ITS AN ERROR
324E B20007         1831          AND  >07,@VAR0           MASK FUNCTION AND STRING BIT
3251 D49010         1832          CEQ  @VAR0,*TOPSTK       Are they equal?
3254 00
3255 52E6           1833          BR   ENT05               NO-ERROR
3257 B21603         1834          AND  >03,@XFLAG          Reset all but OPT. BASE
                    1835  *                                 and imperative.
325A 00             1836          RTN                      All OK-Type matches perfectl
325B 350010         1837  ENT16   MOVE 16 BYTES FROM @FAC TO @ARG
325E 5C4A
3260 BF1400         1838          DST  14,@NMLEN           NEED 14 BYTES FOR SIMPLE VAR
3263 0E
3264 DA1614         1839          CLOG >14,@XFLAG          STRING OR FUNCTION?
3267 7384           1840          BS   ENT61               NO-ALLOCATE & UPDATE TABLE
3269 5380           1841          BR   ENT60               NEED 8 BYTES FOR STRING&FUNC
                    1842  *       THEN ALLOCATES AND UPDATES TABLE  >>JUMP ALWAYS<<
                    1843  *
```

```
                1845  ************************************************************
                1846  *                  CODE FOR A COMPLEX ENTER
                1847  ************************************************************
326B BD0E2C     1848  ENT22   DST   @CHPTR,@CHSAV      SAVE THE LINE PTR
326E BE72B7     1849          ST    STKMIN,@STACK
3271 350010     1850          MOVE 16 BYTES FROM @FAC TO @ARG
3274 5C4A
3276 DA1684     1851          CLOG > 4,@XFLAG          CALLED BY ENTERX OR INSID  A
3279 52A9       1852          BR    ENT2               YES-REQUIRES SPECIAL SCANNING
                1853  *******CODE FOR A DIMENSION STATEMENT STARTS HERE*******
327B 0F13       1854  ENT24   XML   PGMCH              GET NEXT CHARACTER
327D 0630B9     1855          CALL  SYNCHK             'NUMCON"?
3280 C8         1856          DATA  NUMCO$
3281 062CFC     1857          CALL  CSINT              CONVERT TO INTEGER
3284 728F       1858          BS    ENT25
3286 8E4A       1859          CZ    @FAC               BIG DIMENSION?
3288 5294       1860          BR    ENT26              YES-OK
328A C84B43     1861          CHE   @BASE,@FAC+1       DIMENSION >= BASE?
328D 7294       1862          BS    ENT26              YES
328F 062D99     1863  ENT25   CALL  ERR$$              ERROR
3292 2064       1864          DATA  #MSG7              BAD VALUE
3294 8C4B       1865  ENT26   PUSH  @FAC+1             PUSH THIS DIMENSION
3296 8C4A       1866          PUSH  @FAC
3298 C672BD     1867          CH    STKMAX,@STACK      TOO MANY DIMENSIONS?
329B 70C0       1868          BS    SYNTER             YES-ERROR
329D D642B3     1869          CEQ   COMMA$,@CHAT       COMMA?
32A0 727B       1870          BS    ENT24              KEEP READING DIMENSIONS
32A2 D642B6     1871          CEQ   RPAR$,@CHAT        ENDED W/RIGHT PAREN?
32A5 50C0       1872          BR    SYNTER             NO-ERROR
32A7 52F3       1873          BR    ENT40              YES, SO CONTINUE
                1874  *******CODE FOR A NON-DIM STATEMENT STARTS HERE*******
32A9 BE0001     1875  ENT28   ST    1,@VAR0            Parenthesis level counter
                1876  *                                  At first level
32AC 063481     1877  ENT29   CALL  PGMERR             GET NEXT CHARACTER
32AF DA4280     1878          $IF  .BIT7 @CHAT .EQ. 0 GOTO ENT29
32B2 72AC
32B4 D642B6     1879          $IF   @CHAT .EQ. RPAR$   GOTO ENT34
32B7 72EB
32B9 DA1604     1880          $IF  .BIT2 @XFLAG .EQ. 1 GOTO SYNTER
32BC 50C0
32BE D642C7     1881          $IF   @CHAT .EQ. STRIN$ GOTO ENT30
32C1 72DD
32C3 D642B7     1882          $IF   @CHAT .EQ. LPAR$  GOTO ENT32
32C6 72E2
32C8 D642B3     1883          $IF   @CHAT .NE. COMMA$ GOTO ENT29
32CB 52AC
32CD CE0001     1884          CGT   1,@VAR0            Was this a top level comma?
32D0 72AC       1885          BS    ENT29              No-Ignore it
32D2 8C07       1886          PUSH  @DFLTLM+1
32D4 8C06       1887          PUSH  @DFLTLM            Push a default limit
32D6 C672BD     1888          CH    STKMAX,@STACK      Too many dimensions?
32D9 70C0       1889          BS    SYNTER             YES-ERROR
32DB 52AC       1890          BR    ENT29              No-OK, so look for more commas
32DD 063488     1891  ENT30   CALL  SKPSTR             Skip the string/numeric const
32E0 52AC       1892          BR    ENT29
```

```
                *
32E2 9000       ENT32  INC  @VARO                Increase nesting level
32E4 52AC              BR   ENT29                LOOP
                *******Jump always
32E6 062D99     ENT05  CALL ERR$$
32E9 20AF              DATA #MSG15               * NAME CONFLICT
32EB 9200       ENT34  DEC  @VARO                DECREASE NESTING LEVEL
32ED 52AC              BR   ENT29                CONTINUE SCANNING UNLESS THRC
32EF 8C07              PUSH @DFLTLM+1
32F1 8C06              PUSH @DFLTLM              PUSH FINAL DIMENSION
                ********      CODE FOR CALCULATING MEMORY NEEDED         *****
32F3 BC0072     ENT40  ST   @STACK,@VARO         COMPUTE THE # OF DIMS
32F6 A600B7            SUB  STKMIN,@VARO
32F9 E60001            SRL  @VARO,1
32FC CE0003            CGT  3,@VARO               TOO MANY DIMS?
32FF 70C0              BS   SYNTER               YES
3301 8C00              PUSH @VARO                PUSH THE NUMBER OF DIMS
3303 BC1072            ST   @STACK,@TOPSTK
3306 350010            MOVE 16 BYTES FROM @ARG TO @FAC
3309 4A5C
330B 0F16              XML  SCHSYM               SEARCH SYMBOL TABLE
330D 5314              BR   ENT44                NOT FOUND IN TABLE-ENTER IT
330F BD2C0E            DST  @CHSAV,@CHPTR        RESTORE SCAN RESTART AT '('
3312 5240              BR   ENT10                And check for consistency
3314 DA1604     ENT44  TBR  @XFLAG,2             Have a function symbol?
3317 5380              BR   ENT60                Yes-Skip array size calcul.
3319 BC1072            ST   @STACK,@TOPSTK       SAVE STACK PTR
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *      CHANGED FROM:
                *      POP  @FAC                 SKIP # OF DIMS
                *      BY STAN HUME ON 06/19/80 TO SAVE 2 BYTES
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
331C 9272              DEC  @STACK               Skip the number of dimension
331E BC4A90            POP  @FAC                 ASSUME BASE 0
3321 7C
3322 BC4B90            POP  @FAC+1
3325 7C
3326 914A              DINC @FAC
3328 8608              CLR @VARC                 BUT CORRECT IF BASE 1
332A BC0943            ST   @BASE,@VARC+1        HANDLE 1ST DIM SPECIALLY TO
332D A54A08            DSUB @VARC,@FAC           AVOID 1 MPY
3330 BD144A            DST  @FAC,@NMLEN          FAC GETS # OF ELEMENTS IN AR
3333 053350            B    ENT53                MERGE INTO LOOP
3336 BC4A90     ENT50  POP  @FAC                 GET NEXT DIMENSION
3339 7C
333A BC4B90            POP  @FAC+1
333D 7C
333E 914A              DINC @FAC                 ASSUME BASE 0
3340 A54A08            DSUB @VARC,@FAC           BUT CORRECT IF BASE 1
3343 BD0214            DST  @NMLEN,@ACCUM
3346 A9024A            DMUL @FAC,@ACCUM          ACCUMULATE SIZE
3349 8F02              DCZ  @ACCUM               CHECK FOR OVERFLOW
334B 544B              BR   ENT70                OUT OF MEMORY
334D BD1404            DST  @ACCUM+2,@NMLEN
3350 D672B7     ENT53  CEQ  STKMIN,@STACK        END OF PUSHED STUFF?
```

```
 353 5336          BR   ENT50              NO-LOOP
3355 DA1610        TBR  @XFLAG,4           STRING ARRAY?
3358 7365          BS   ENT59              NO - NUMERIC ARRAY
335A DA14E0        CLOG >E0,@NMLEN         IF TOP BIT SET-OVERFLOW
335D 544B          BR   ENT70              * MEMORY FULL
335F E31400        DSLL @NMLEN,1           MPY BY 2
3362 01
3363 536E          BR   ENT59A             >>>>JUMP ALWAYS<<<<
3365 DA14F0 ENT59  CLOG >F0,@NMLEN         IF ANY OF THE TOP 3 BITS ARE
3368 544B          BR   ENT70              1 THERE WILL BE AN OVERFLOW
336A E31400        DSLL @NMLEN,3           MPY BY 8
336D 03
336E A31400 ENT59A DADD 6,@NMLEN           NEED 6 MORE BYTES
3371 06
3372 864A          CLR  @FAC
3374 BC4B90        ST   *TOPSTK,@FAC+1     GET # OF DIMENSION
3377 10
3378 E24B01        SLL  @FAC+1,1             MPY BY 2
337B A1144A        DADD @FAC,@NMLEN        TOTAL # NEEDED
337E 5384          BR   ENT61              >>>JUMP ALWAYS<<<
            *
3380 BF1400 ENT60  DST  8,@NMLEN           FUNCS&SIMPLE STRINGS NEED 8
3383 08
            *
            *  **********CHECK TO SEE IF ENOUGH MEMORY****************
            *
            *      PUT  SYMBOL IN SYMBOL TABLE IF IT IS AN
            *           IMPERATIVELY CREATED SYMBOL
            *
3384 BE8089 ENT61  CZ   @GROMFL            GROM?
3387 538E          BR   ENT61A             YES-ENTER NAME IN SYMBOL TABLE
3389 DA1601        TBR  @XFLAG,0           IMPERATIVE?
338C 73AA          BS   ENT62              NOPE
338E 8E6B   ENT61A CZ   @ARG+15            ZERO LENGTH?
3390 73AA          BS   ENT62              YEP-(FUNCTION EXECUTION)
            *      MOVE THE NAME INTO THE SYMBOL TABLE
3392 8600          CLR  @VAR0              RE-DO NAME AND POINTER
3394 BC016B        ST   @ARG+15,@VAR0+1    GET LENGTH OF NAME
3397 BD4A00        DST  @VAR0,@FAC
339A 063493        CALL MEMCHK             CHECK ENOUGH MEMORY
339D A54000        DSUB @VAR0,@SYMPTR      GET SPACE FOR NAME
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
            *      CHANGED FROM:                                           *
            *      MOVE @VAR0 FROM @ARG TO RAM(1(SYMPTR)) MOVE THE NAME*
            *      DST  @SYMPTR,@NMPTR     SET NEW PTR TO NAME            *
            *      DINC @NMPTR             NEW PTR TO NAME                *
            *      BY STAN HUME 06/19/80 TO SAVE 1 BYTE                    *
            *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
33A0 BD0C40        DST  @SYMPTR,@NMPTR     Set pointer to name
33A3 910C          DINC @NMPTR             We were off by one
33A5 3400B0        MOVE @VAR0 FROM @ARG TO RAM(@NMPTR) Move the name
33A8 0C5C
            *
33AA BD4A14 ENT62  DST  @NMLEN,@FAC        # OF BYTES NEEDED
33AD 063493        CALL MEMCHK             CHECK ENOUGH ROOM IN MEMORY
```

```
                *
                *       NOW CONSTRUCT THE ENTRY FOR THE SYMBOL TABLE
                *       IN THE FAC
                *
33B0 864A                CLR  @FAC                 CLEAR THE ACCUMULATION AREA
33B2 DA1610              TBR  @XFLAG,4             STRING?
33B5 73BA                BS   ENT63                NO
33B7 B64A80              SB   @FAC,7               SET STRING BIT
33BA DA1604     ENT63    TBR  @XFLAG,2             FUNCTION?
33BD 73C2                BS   ENT65                NO
33BF B64A40              SB   @FAC,6               SET FUNCTION BIT
33C2 BC7210     ENT65    ST   @TOPSTK,@STACK       GET # OF DIMENSIONS OR PARAM
33C5 BC5090              POP  @FAC+6
33C8 7C
33C9 8E50                CZ   @FAC+6               NO DIMENSIONS?
33CB 73D8                BS   ENT67                YES-SIMPLE OR STRING VARIABL
33CD B44A50              OR   @FAC+6,@FAC          OVERLAY # OF DIMENSIONS
33D0 DA1604              TBR  @XFLAG,2             DEF?
33D3 53D8                BR   ENT67                YES
33D5 B61602              SB   @XFLAG,1             SET OPT. BASE FLAG
33D8 BC4B6B     ENT67    ST   @ARG+15,@FAC+1       SAVE LENGTH OF NAME
33DB BD4C3E              DST  @SYMTAB,@FAC+2       LINK TO PREVIOUS ENTRY
33DE BD4E0C              DST  @NMPTR,@FAC+4        SAVE THE PTR TO NAME
33E1 A54014              DSUB @NMLEN,@SYMPTR       SET NEW TABLE POINTER
33E4 9140                DINC @SYMPTR
                *******MOVE THE ENTRY FROM THE FAC TO THE SYMBOL TABLE
33E6 350006              MOVE 6 BYTES FROM @FAC TO RAM(@SYMPTR)
33E9 B0404A
33EC BD3E40              DST  @SYMPTR,@SYMTAB      PTR TO BEGINNING OF TABLE
33EF DA8088              TBR  @FLAG,3              RUN FUNCTION MODE?
33F2 08
33F3 53F9                BR   ENT67A               YES-SKIP AHEAD
33F5 BDA3E0              DST  @SYMTAB,RAM(SYMBOL) SAVE PTR ON VDP RAM
33F8 3E
33F9 A34000     ENT67A DADD 6,@SYMPTR
33FC 06
33FD DA1604              TBR  @XFLAG,2             FUNCTION?
3400 5424                BR   ENT68A               YES-SKIP DIMENSION SAVE CODE
                *******NOW PREPARE TO ZERO THE REQUIRED AMMOUNT OF MEMORY
3402 C672B7              CH   STKMIN,@STACK        NON-ARRAY?
3405 542A                BR   ENT69'               YES
3407 BE72B7              ST   STKMIN,@STACK        GET TO BOTTOM OF STACK
340A 9072       ENT68    INC  @STACK
340C C87210              CHE  @TOPSTK,@STACK       FINISHED?
340F 742A                BS   ENT69                YES
3411 BCE001              ST   *STACK,RAM(1(SYMPTR))      REVERSE POP
3414 409072
3417 9072                INC  @STACK
3419 BCB040              ST   *STACK,RAM(@SYMPTR)
341C 9072
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *       CHANGED FROM:
                *       DINCT @SYMPTR
                *       DDECT @NMLEN
                *       B     ENT68                GET NEXT DIMENSION
```

```
                *       BY STAN HUME ON 06/19/80 TO SAVE 1 BYTE
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><>
341E 9714               DDECT @NMLEN             Decrement total # of bytes
3420 9540               DINCT @SYMPTR            Increment pointer to next dim
3422 540A               BR    ENT68              Loop to put next dim in
                *
                *       FUNCTION ENTRY CODE
                *
                ENT68A
3424 BDB040             DST   @ARG+16,RAM(@SYMPTR)  SAVE PTR TO '(' OR '='
3427 6C
3428 5438               BR    ENT69B
                *******Jump always
                *******NOW ZERO THE REQUIRED AMMOUNT OF MEMORY
342A A71400     ENT69   DSUB 7,@NMLEN            NOW CLEAR MEMORY
342D 07
342E 86B040             CLR   RAM(@SYMPTR)       CLEAR FIRST BYTE, THEN THE RE
3431 3414E0             MOVE @NMLEN BYTES FROM RAM(@SYMPTR) TO RAM(1(SYMPTR)
3434 0140B0
3437 40
3438 BD403E     ENT69B DST   @SYMTAB,@SYMPTR     SET NEW FREE PTR @ TABLE
343B 9340               DDEC @SYMPTR             NOW, SET IT AT 1ST FREE BYTE
343D B21683             AND   >83,@XFLAG         Clear all but OPT. BASE
                *                                and imperative.
3440 DA1680             TBR   @XFLAG,7           Called by ENTER?
3443 7448               BS    ENT71              Yes-Return through PGMCHR
3445 BD2C0E             DST   @CHSAV,@CHPTR      RESTORE CHARACTER POINTER
3448 0F1B       ENT71   XML   PGMCH              GET NEXT CHARACTER
344A 00                 RTN
                *
344B 062D99     ENT70   CALL ERR$$
344E 2049               DATA #MSG5               * OUT OF MEMORY
```

```
                    2069  ********************************************************
                    2070  *        ROUTINE TO TELL PROPERTY OF A CHARACTER
                    2071  ********************************************************
                    2072  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    2073  *    PROP IS CHANGED TO ADD LOWERCASE CHAR. SET IN 99/4A
                    2074  *    3/3/81
                    2075  *      ON RETURN THE CONDITION SPECIFIES THE PROPERTY
                    2076  *           SET MEANS CAN BE IN A VARIABLE NAME
                    2077  *           i.e. 0-9,@,A-Z,[,\,],_
                    2078  *PROP
                    2079  * $IF @CHAT .HE. ZERO THEN      >= 0?
                    2080  *   $IF @CHAT .LE. NINE GOTO RTNSET  =< 9 ?
                    2081  * $END IF
                    2082  * $IF  @CHAT .HE. :@: THEN
                    2083  *     $IF @CHAT .LE. :]: GOTO RTNSET
                    2084  * $END IF
                    2085  * $IF @CHAT .EQ. UNLN GOTO RTNSET
                    2086  * RTN
                    2087  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    2088  *  ON RETURN THE CONDITION SPECIFIES THE PROPERTY
                    2089  *     SET MEANS CAN BE IN A VARIABLE NAME
                    2090  *     i.e. 0-9,@,A-Z,a-z,[,\,],_
                    2091  PROP
3450 CA4230         2092    $IF @CHAT .HE. ZERO THEN
3453 545A
3455 CA423A         2093      $IF @CHAT .L. ::::  GOTO RTNSET     Digit 0-9
3458 4C80
                    2094    $END IF
345A CA4240         2095    $IF @CHAT .HE. :@: THEN
345D 5464
345F CA425E         2096      $IF @CHAT .L. :^: GOTO RTNSET      @,A-Z,[,],\
3462 4C80
                    2097    $END IF
3464 CA425F         2098    $IF @CHAT .HE. UNLN THEN
3467 5480
3469 CA4261         2099      $IF @CHAT .HE. :a: THEN
346C 5476
346E CA427B         2100        $IF @CHAT .L. :{: THEN
3471 7476
3473 A64220         2101          SUB >20,@CHAT           Change lower to upper
                    2102        $END IF
                    2103      $END IF
3476 D64260         2104      $IF @CHAT .EQ. :`: GOTO RTNRST
3479 7480
347B CA427B         2105      $IF @CHAT .L. :{: GOTO RTNSET      a-z,_
347E 4C80
                    2106    $END IF
3480 00             2107  RTNRST  RTN
                    2108  ********************************************************
                    2109  *        THIS ROUTINE READS A CHARACTER AND WILL GIVE
                    2110  *        AN ERROR IF IT READS AN END OF LINE(PREMATURE END
                    2111  ********************************************************
3481 0F1B           2112  PGMERR XML   PGMCH
3483 8E42           2113         CZ    @CHAT              END OF LINE?
3485 70C0           2114         BS    SYNTER             YES=>ERROR
```

```
3487 00                  RTN                        NO-OK GO ON
               ****************************************************************
               *                THIS ROUTINE SKIPS QUOTED STRINGS
               *             UNQUOTED STRINGS AND NUMERIC CONSTANTS
               ****************************************************************
3488 0F13      SKPSTR XML   PGMCH           GET THE BYTE COUNT
348A 8708             DCLR  @VARC
348C BC0942           ST    @CHAT,@VARC+1
348F A12C08           DADD  @VARC,@CHPTR SKIP THE STRING
               *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
               *      CHANGED FROM:
               *      RTN
               *      BY STAN HUME 06/24/80 TO SAVE 1 BYTE BELOW.
               *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
3492 00        MEMC08 RTN
```

```
                    2131  ********************************************************
                    2132  *
                    2133  *           THIS IS THE MEMORY CHECK ROUTINE
                    2134  *
                    2135  *       IT CHECKS TO SEE IF THERE IS ENOUGH ROOM TO
                    2136  *       INSERT A SYMBOL TABLE ENTRY OR A P.A.B.  INTO
                    2137  *       THE VDP BETWE N THE STATIC SYMBOL TABLE/PAB
                    2138  *       AREA AND THE DYNAMIC STRING AREA.   IF THERE
                    2139  *       IS ENOUGH SPACE IT SIMPLY RETURNS.   IF THERE
                    2140  *       IS NOT IT ATTEMPTS TO MOVE THE STRING SPACE
                    2141  *       DOWN(TO LOWER ADDRESS) AND THEN INSERT THE
                    2142  *       NEEDED AREA BETWEEN THE TWO.   NOTE: IT MAY
                    2143  *       INVOKE COMPCT TO DO A GARBAGE COLLECTION.   IF
                    2144  *       THERE IS NOT ENOUGH SPACE AFTER COMPCT THEN
                    2145  *       ISSUES   *MEMORY FULL    MESSAGE.
                    2146  *
                    2147  *       INPUT:   # OF BYTES NEEDED IN  FAC,FAC+1
                    2148  *       USES:    FAC+2,FAC+4      AS  TEMPORARIES
                    2149  *
                    2150  ********************************************************
                    2151  *
3493 BD4E4A         2152  MEMCHK DST   @FAC,@FAC+4
3496 BD4C40         2153         DST   @SYMPTR,@FAC+2     GET BEGINNING OF S.T.  FREE S
3499 A54C18         2154         DSUB  @STRSP,@FAC+2      CALCULATE SIZ OF GAP
349C D14C4A         2155         DCGE  @FAC,@FAC+2        ENOUGH SPACE ALREADY?
349F 7492           2156         BS    MEMC08             YES - DONE - RTN
34A1 A54A4C         2157         DSUB  @FAC+2,@FAC        GET AMOUNT NEEDED ABOVE WHAT
                    2158  *                               AVAILABLE
34A4 BD4C1A         2159         DST   @STREND,@FAC+2     GET BEGINNING OF STRING FREE
34A7 A54C6E         2160         DSUB  @VSPTR,@FAC+2      CALCULATE SIZE OF GAP
34AA A74C00         2161         DSUB  64,@FAC+2          VSPTR OFFSET TOO
34AD 40
34AE D14C4A         2162         DCGE  @FAC,@FAC+2        ENOUGH ROOM TO MOVE?
34B1 74C8           2163         BS    MEMC02             YES-MOVE IT
34B3 064D18         2164         CALL  COMPCT             NO-COMPACTIFY
34B6 BD4C1A         2165         DST   @STREND,@FAC+2     GET BEGINNING OF STRING FREE
34B9 A54C6E         2166         DSUB  @VSPTR,@FAC+2      CALCULATE SIZE OF GAP
34BC A74C00         2167         DSUB  64,@FAC+2          VSPTR OFFSET TOO
34BF 40
34C0 BD4A4E         2168         DST   @FAC+4,@FAC        RESTORE FAC
34C3 D14C4A         2169         DCGE  @FAC,@FAC+2        ENOUGH ROOM NOW?
34C6 544B           2170         BR    ENT70              NO-   *MEMORY FULL
                    2171  *
                    2172  *******NOW MOVE THE DYNAMIC STRING AREA DOWN IN MEMORY
                    2173  *
34C8 BD4C18         2174  MEMC02 DST   @STRSP,@FAC+2      CALCULATE # OF BYTES
34CB A54C1A         2175         DSUB  @STREND,@FAC+2     IN THE TOTAL STRING SPACE
34CE BD4E1A         2176         DST   @STREND,@FAC+4     BEGINNING OF MOVE ADDRESS
34D1 A51A4A         2177         DSUB  @FAC,@STREND       SET FREE PTR(COPY-TO ADDRESS
34D4 8F4C           2178         DCZ   @FAC+2             NO BYTES TO MOVE?
34D6 74E0           2179         BS    MEMC03             RIGHT
34D8 344CE0         2180         MOVE  @FAC+2 FROM RAM(1(FAC+4)) TO RAM(1(STREND))
34DB 011AE0
34DE 014E
                    2181  *                               MOVE IT
```

```
34E0 A5184A  2182  MEMC03 DSUB @FAC,@STRSP        SET NEW STRING SPACE PTR
             2183  *
             2184  *******NOW FIX UP STRING PTRS
             2185  *
34E3 BD4C18  2186         DST  @STRSP,@FAC+2      GET BEGINNING OF STRING SPAC
34E6 864E    2187         CLR  @FAC+4
34E8 BC4FB0  2188  MEMC04 ST   RAM(@FAC+2),@FAC+5   SAVE LENGTH OF ENTRY
34EB 4C
34EC C91A4C  2189         DCHE @FAC+2,@STREND     FINISHED?
34EF 7492    2190         BS   MEMC08             YES
34F1 A54C4E  2191         DSUB @FAC+4,@FAC+2      POINT AT STRING
34F4 8FEFFF  2192         DCZ  RAM(-3(FAC+2))     FREE STRING?
34F7 FD4C
34F9 7505    2193         BS   MEMC06             YES
34FB BD4AEF  2194         DST  RAM(-3(FAC+2)),@FAC      GET BACKPTR
34FE FFFD4C
3501 BDB04A  2195         DST  @FAC+2,RAM(@FAC)   SET FORWARD PTR
3504 4C
3505 A74C00  2196  MEMC06 DSUB 4,@FAC+2           NOW POINT AT NEXT LENGTH
3508 04
3509 54E8    2197         BR   MEMC04             CONTINUE ON(JUMP ALWAYS)
             2198  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
             2199  *       CHANGED FROM:
             2200  *MEMC08 RTN
             2201  *       BY STAN HUME ON 06/24/80 TO SAVE 1 BYTE.   LABEL
             2202  *       MOVED TO RTN IN SKPSTR ABOVE.
             2203  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
             2204         END
```

ERRORS= 0

LENGTH= 3299 (>0CE3)

482 SYMBOLS USED

SYMBOL	VALUE	DEF	REFERENCE TABLE
A	0041	204	
ABS$	00CB	337	446
ACCUM	0002	68	1937 1938 1939 1941
ALLUP	0082	142	938
AMPER	0026	217	418
APOS	0027	218	
APPEN$	00F9	372	500
ARG	005C	132	547 554 556 567 574 596 632 638 639 650 651 651 652 652 652 653 653 654 655 656 657 657 658 682 694 701 713 718 719 719 720 721 721 722 723 724 724 727 732 733 735 735 739 740 741 743 744 750 751 757 760 761 762 763 772 773 774 775 796 810 815 816 817 1009 1010 1011 1012 1015 1016 1018 1018 1019 1149 1245 1247 1410 1411 1411 1413 1414 1415 1416 1417 1417 1418 1418 1419 1420 1426 1428 1429 1434 1438 1441 1442 1790 1791 1837 1850 1911 1971 1975 1988 2011 2049
ASC	4D0A	18	1038
ASC$	00DC	354	460
AT	0040	203	
ATN	0032	50	
ATN$	00CC	338	447
AUTO$1	2C65	862	843 857
AUTO$2	2C2A	825	838 855 860
AUTO1	2C36	835	388
AUTON	2C2B	831	394
BACK	0008	178	623
BACKS	005C	207	
BASE	0043	129	1559 1612 1617 1861 1929
BASE$	00F1	364	470 1607
BKGD	0020	170	
BREAK	0002	174	602
BREAK$	008E	274	488
BUFFY	0044	130	852 865 899 1023 1090 1091 1093 1130 1343 1349 1396 1397 1487 1581
BYTE	000C	80	81 840 864 1309 1338
CALL$	009D	289	485
CB	0008	1	
CCPADD	0008	78	1151 1472 1473 1474
CCPPTR	0006	75	1468 1475
CHAR	0004	1	
CHAR1	2870	416	413
CHAR2	288F	432	413
CHAR2$	0018	38	
CHAR3	289C	437	413
CHAR4	291D	470	413
CHAR5	2973	488	413
CHAR6	299E	496	413
CHAR7	29D0	504	413
CHAR8	29F1	510	413
CHAR9	2A16	515	413
CHARA	2A2B	518	413
CHAT	0042	128	878 894 912 913 914 932 944 960 961 975 982 1037 1039 1167 1169 1177 1178 1183 1184 1186

SYMBOL	VALUE	DEF	REFERENCE TABLE
			1194 1198 1201 1202 1210 1223 1243 1245 1247 1271
			1283 1288 1310 1311 1315 1330 1369 1426 1463 1473
			1476 1588 1593 1599 1613 1615 1620 1629 1634 1668
			1670 1671 1673 1696 1699 1702 1707 1709 1717 1719
			1721 1723 1734 1741 1742 1743 1744 1745 1758 1768
			1779 1789 1797 1869 1871 1878 1879 1881 1882 1883
			2092 2093 2095 2096 209 2099 2100 2101 2104 2105
			2113 2122
CHPTR	002C	117	1492 1771 1811 1812 1819 1848 1914 2062 2123
CHR$$	00D6	348	480
CHRTAB	201C	15	1080
CHRTN	000D	167	689
CHSAV	000E	84	1812 1848 1914 2062
CIRCU$	00C5	329	419
CIRCUM	005E	209	419
CLOSE$	00A0	292	494
CLOSEB	005D	208	
CLRLIN	2B0D	674	
CLRLN	0007	177	673
CLSALL	4012	14	1092
CNS	0014	37	
COLON	003A	197	427
COLON$	00B5	313	427
COMMA	002C	223	430 834
COMMA$	00B3	311	430 1177 1296 1593 1869 1883
COMPCT	4D18	32	2164
CONC$	00B8	316	418
CONTCM	0002	184	
COS	002C	47	
COS$	00CD	339	448
CPL	0010	35	
CRNBUF	0320	152	1047 1048 1492 1655 1657 1663 1664 1666 1683 1685
			1686 1687 1693
CRNEND	03BE	153	892 929
CSINT	2CFC	956	1857
CSINT2	2CFE	958	980
CSINT4	2D23	983	1079
CSNUM	0010	246	
CURSOR	007E	228	568 595
DATA	0034	123	
DATA$	0093	279	471 1702
DDEF02	3177	1714	1645
DDEF03	310A	1665	1659
DDEF06	3115	1668	1676 1684 1688 1692
DDEF08	312C	1678	1670
DDEF10	3148	1690	1672 1674
DDEF12	314F	1693	1669
DEF$	0089	269	437 1634
DELET$	0099	285	502
DELINK	4D14	20	1110
DELREP	2BD6	771	390
DEREA	2BEE	787	805
DEREB	2C02	803	798
DFLTLM	0006	73	1485 1886 1887 1901 1902
DIM$	008A	270	438 1588

SYMBOL	VALUE	DEF	REFERENCE TABLE
DIM1	3056	1576	1623 1705
DIM1A	3067	1586	1577
DIM3	30AA	1618	1614
DIM3A	30AF	1620	1595
DIM4A	31D6	1753	1605
DIM5	3193	1727	1720
DIM5A	319B	1730	1722 1724
DIM6	30B3	1622	1694 1697 1718
DIM8	30B8	1624	1752
DIMDO1	3072	1591	1594
DIMDAT	315F	1702	
DIMDEF	30C5	1634	1600
DIMDIF	317B	1717	1726 1729 1732 1739 1745 1747
DIMDIM	306D	1588	1504
DIMFN	3061	1582	1738
DIMFOR	316E	1709	
DIMON	31B3	1741	1735
DIMOP	3081	1599	1589
DIMREM	3155	1696	1635
DIMRET	31A2	1734	1710
DIMRTN	3066	1584	1580
DIMSUB	3169	1707	1703
DIMTAB	302B	1541	1514
DISO	2FAF	1435	395 1017 1020
DISLN	2F7C	1409	1424 1436
DISP$1	2FB2	1437	1443
DISPL$	00A2	294	508
DISPLA	0009	1	
DISPLN	2FAB	1433	
DISPY	0002	166	
DIVI	002F	226	422
DIVI$	00C4	328	422
DLETE	0003	175	637
DOLLAR	0024	215	1792
DOT	002E	225	
DOWN	000A	180	691
DSPCHR	2FE3	1467	1168 1185 1224 1227 1292 1367 1427
DSPPG2	2F58	1368	1164
DSPPGM	2F55	1367	1196
DSPQUO	2FE0	1462	1203 1231
DSRFLG	0017	101	1472
EDGECH	007F	229	647 661 676 714 722 740 762 913
EDIT	000F	188	
ELSE$	0081	261	473
END$	008B	271	439
ENDERR	2F4F	1360	867
ENDSCR	02FE	151	
ENLN	0032	121	122 779 820 1392
ENPT	0006	74	
ENT01	31F5	1776	1789
ENT03	307C	1597	1777
ENT05	32E6	1897	1827 1830 1833
ENT08	3229	1811	1799
ENT09	322B	1812	399
ENT10	3240	1826	1915

SYMBOL	VALUE	DEF	REFERENCE TABLE
ENT16	325B	1837	1816 1818
ENT22	326B	1848	1797
ENT24	327B	1854	1870
ENT25	328F	1863	1858
ENT26	3294	1865	1860 1862
ENT28	32A9	1875	1852
ENT29	32AC	1877	1878 18 3 1885 1890 1892 1895 1900
ENT30	32DD	1891	1881
ENT32	32E2	1894	1882
ENT34	32EB	1899	1879
ENT40	32F3	1904	1873
ENT44	3314	1916	1913
ENT50	3336	1933	1943
ENT53	3350	1942	1932
ENT59	3365	1950	1945
ENT59A	336E	1953	1949
ENT60	3380	1960	1841 1917
ENT61	3384	1967	1840 1958
ENT61A	338E	1971	1968
ENT62	33AA	1990	1970 1972
ENT63	33BA	2000	1998
ENT65	33C2	2003	2001
ENT67	33D8	2011	2006 2009
ENT67A	33F9	2022	2020
NT68	340A	2029	2044
ENT68A	3424	2048	2024
ENT69	342A	2053	2027 2031
ENT69B	3438	2056	2050
ENT70	344B	2066	1940 1947 1951 2170
ENT71	3448	2063	2061
ENTER	31E5	1767	398 1592 1642
ENTERW	31E8	1768	1649
ENTERX	31ED	1769	1679 1728
EOF$	00CA	336	440
EQUAL	003D	200	420
EQUAL$	00BE	322	420 1654 1715
ERCODE	007C	60	
ERR$$	2D99	1044	401 900 1350 1582 1597 1630 1753 1863 1897 2066
ERR$$1	2DD3	1103	1106 1111
ERR$1	2D35	1005	1070
ERRCOD	0022	111	
ESC	007E	186	
EXCL	0021	212	
EXEC	4D04	8	1527
EXEC6C	4D0E	19	611
EXIT	0020	41	
EXP	0028	45	
EXP$	0076	136	1002 1003 1009 1011 1013 1037 1039 1045 1046
EXP$$	00CE	340	449
EXTRAM	002E	118	771 772 822 1392 1393 1396 1397 1400 1401 1402 1450 1452 1455 1543 1544 1576
AC	004A	131	957 973 974 975 976 976 979 1109 1162 1166 1167 1169 1176 1180 1181 1192 1193 1196 1190 1190 1210 1211 1222 1228 1230 1237 1254 1255 1261 1261 1263 1263 1264 1264 1271 1272 1273 1274 1275 1282

SYMBOL	VALUE	DEF	REFERENCE TABLE
			1282 1283 1288 1293 1294 1296 1297 1298 1301 1327
			1333 1334 1335 1335 1337 1339 1343 1344 1476 1628
			1629 1770 1770 1776 1776 1778 1779 1792 1795 1795
			1796 1828 1837 1850 1859 1861 1865 1866 1911 1925
			1926 1927 1930 1931 1933 1934 1935 1936 1938 1954
			1955 1956 1957 1976 1990 1996 1999 2002 2004 2005
			2007 2007 2011 2012 2013 2017 2152 2152 2153 2154
			2155 2155 2157 2157 2159 2160 2161 2162 2162 2165
			2166 2167 2168 2168 2169 2169 2174 2175 2176 2177
			2178 2180 2180 2182 2186 2187 2188 2188 2189 2191
			2191 2192 2194 2194 2195 2195 2196
FACSTS	0003	233	
FADD	0006	236	
FCOMP	000A	240	
FDIV	0009	239	
FIGURE	0002	1	
FIXED$	00FA	373	490
FLAG	0088	144	603 610 898 1022 1052 1078 1486 1520 1564 1647
			1650 1815 2019
FLTINT	0012	248	
FMULT	0008	238	
FOR$	008C	272	441 1709
FORNET	0017	102	1579 1711 1737 1738
FORW	0009	179	627
FPERAD	006C	133	
FSADD	000B	241	
FSCOMP	000F	245	
FSDIV	000E	244	
FSMULT	000D	243	
FSSUB	000C	242	
FSUB	0007	237	
GETCHR	2CA6	910	386 875 920 1340
GETLN	2EF9	1308	392 839 863
GETLN2	2F11	1319	1316
GETLNB	2020	16	927
GETNB	2C75	873	384 837 842 854 1361
GETNB2	2C7A	877	405 859 930
GO$	0085	265	432 1744
GOSUB$	0087	267	489 1707
GOTO$	0086	266	474 1745
GREAT	003E	201	426
GREAT$	00C0	324	426
GRINT	0011	247	
GROMFL	0089	145	609 1095 1451 1556 1658 1967
GSCAN	0022	42	
HELP	0006	183	
IF$	0084	264	433 1743
ILL1	2018	13	618 1357
ILLIMP	31CC	1746	1700 1708 1712 1741 1742 1743 1744
IMPILL	31D1	1751	1602 1639 1704 1736 1746
INPUT$	0092	278	491 1699
INSRT	0004	176	631
INT$	00CF	341	450
INTER$	00F5	368	512
KEYBD	0074	55	1006

SYMBOL	VALUE	DEF	REFERENCE TABLE
.EYTAB	285C	413	1261
LEN$	00D5	347	457
LESS	003C	199	425
LESS$	00BF	323	425
LET$	008D	273	442
LIMIT	0012	168	
LINUM	0012	87	1571 1572 1576
LISTCM	0000	187	
LLI$12	2E15	1163	1302
LLI$15	2E27	1171	1199
LLIS$0	2E11	1155	1296
LLIS$1	2EF5	1300	1230 1238 1249
LLIS$2	2E4B	1189	1178
LLIS$3	2E96	1242	1202
LLIS$4	2EB0	1257	1276
LLIS$5	2ED3	1281	1271
LLIS$6	2ED9	1287	1295
LLIS$9	2F11	1318	1165 1197
LLIST	2DFA	1147	385
LLPTR	0030	120	
LN$	00C9	335	1243 1758
LNBUF	0036	124	
LOAD	0003	190	
LOG	002A	46	
LOG$	00D0	342	451
.PAR	0028	219	417
LPAR$	00B7	315	417 1797 1882
M$EDIT	2010	4	6 11 12 13 15 16 17
M$EXEC	4D00	3	8 9 10 18 19 20 32
M$FLMG	4000	5	14 33
M$MSG	2022	6	21 22 23 24 25 26 27 28 29 30 31
MAO	2012	11	1133
MAXET	0005	172	
MAXLEN	000B	171	
MEMC02	34C8	2174	2163
MEMC03	34E0	2182	2179
MEMC04	34E8	2188	2197
MEMC06	3505	2196	2193
MEMC08	3492	2129	2156 2190
MEMCHK	3493	2152	396 1977 1991
MINUS	002D	224	424
MINUS$	00C2	326	424
MSG0	2022	21	1001
MSG1	202C	22	1631
MSG15	20AF	26	1898
MSG17	20BD	27	1754
MSG18	20F9	29	1583
MSG19	213F	30	901
MSG21	2113	31	1013
MSG4	2040	23	1598
MSG5	2049	24	2067
MSG7	2064	25	1864
MSGBLN	20D9	28	1351
MULT	002A	221	421

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

MULT$   00C3   327     421
MVDN    201E    17     818
MVUP    000B   181     690
NBC     0014    90     832  852 1149 1153
NBD     001E   109     833  865
NEW     000E   189
NEXT$   0096   282     475 1734
NINE    0039   196
NLNADD  02E2   150    1001 1008 1026 1068
NMLEN   0014    89    1838 1931 1937 1941 1946 1948 1950 1952 1953 1957
                      1960 1990 2014 2042 2053 2055
NMPTR   000C    81    1771 1772 1986 1987 1988 2013
NUM$    00C8   333    1201 1671 1721
NUMBE$  00FD   376     429 1297
NUMBER  0023   214     429
NUMBUF  0032   122    1543
NUMCO$  00C8   334    1178 1609 1856
OFFSET  0060   227     228  229  664  667  677  695  731  914 1439
ON$     009B   287     434 1742
OPEN$   009F   291     486
OPENB   005B   206
OPTIO$  009E   290     496 1599
OUTL$1  2F9D  1425    1430
OUTLN   2F9A  1423    1150 1248
OUTPU$  00F7   370     499
OUTREC  401A    33    1148 1469
OV      0005   235
OVUN    0004   234
PABPTR  0004    70    1015
PERC    0025   216
PERMA$  00FB   374     516
PGMCH   001B   254     959 1209 1212 1244 1246 1368 1457 1493 1619 1675
                      1691 1731 1761 1780 1854 2063 2112 2120
PGMERR  3481  2112    1603 1629 1767 1877
PGMUPT  31DB  1758    1725
PLUS    002B   222     423
PLUS$   00C1   325     423
POP     0005     1
POPS    001E    40
POS$    00D9   351     458
PRA2    2FFF  1485
PRA3    301C  1514    1488
PRA4    3028  1527    1510
PRGLST  2FFF  1484     382
PRINT$  009C   288     492
PROP    3450  2091     397  939
PTLN$1  2F14  1329    1341
PTLNE   2F12  1326     391 1312
PTLNER  2F43  1348     383 1334 1339 1344
PTLNR1  2F4D  1357     902
PUSH    001C    39
PUSHCH  2CA0   904     403  942
PWR     0024    43
QUEST   003F   202
QUOTE   0022   213    1223 1463
```

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

RAM     0001     1      574  596  647  652  652  657  657  661  664  667
                        676  677  695  714  721  721  722  724  724  732
                        740  762  772  775  788  802  894  905  912  926
                        946  948 1001 1011 1015 1028 1048 1068 1104 1108
                       1149 1153 1396 1397 1434 1438 1439 1452 1472 1473
                       1644 1655 1655 1657 1664 1665 1685 1685 1686 1693
                       1828 19   2017 2021 2032 2034 2049 2054 2055 2055
                       2180 2180 2188 2192 2194 2195
RAMPTR  0038   125     892  893  894  925  926  929  946  947  949
RANDO$  0095   281     515
RANDOM  0078    57
RBACK   2BC3   759     623
READ$   0097   283     476
READ$1  2A56   569     588  609  691  737  746  752
READ$2  2A78   592     578
READ$3  2A7A   594     584
READ$4  2B54   717     713
READ00  2A4F   561     407
READ01  2A51   566     664  668  766
READ05  2B86   734     758
READL1  2A49   553     406
READL2  2A4C   555     683
READLN  2A42   546     387
REC$    00DE   356     461
RECLEN  0007    76    1468
RELAT$  00F4   367     513
REM$    009A   286     443 1696
RESTO$  0094   280     506
RETUR$  0088   268     497 1741
RFORW   2BBF   756     627
RKEY    0075    56     580  598  602  623  627  631  637  673  689  690
                        691  731  732
RND$    00D7   349     452
ROM     000A     1    1001 1009 1011 1261 1264 1288 1455
ROUND$  0001   231
ROUNU$  0002   232
RPAR    0029   220     416
RPAR$   00B6   314     416 1652 1871 1879
RPL     0012    36
RSTK    0088   146    1053
RTNRST  3480  2107    2104
RTNSET  2C80   883     876  911  974  977  979 1095 2093 2096 2105
RUNCM   0001   192
SAVECM  0004   191
SCANER  2016    12     849
SCHSYM  0016   251    1817 1912
SCR010  4D86    10
SCROLL  4D00     9     747 1000 1004 1025 1042 1067 1348 1502
SEARCH  2FC4  1449     389 1586
SEE1    2F64  1394
SEETWO  2F5D  1391     393
SEG$$   00D8   350     481
SEMIC   003B   198     428
SEMIC$  00B4   312     428
SEQUE$  00F6   369     518
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
SGN$	0075	135	
SGN$$	00D1	343	453
SIN	002E	48	
SIN$	00D2	344	454
SKPSTR	3488	2120	1690 1730 1891
SMB	0014	250	
SPACE	0020	211	878 946
SPRITE	0006	1	
SQR	0026	44	
SQR$	00D3	345	455
SREF	001C	108	
STACK	0072	53	1849 1867 1888 1904 1910 1918 1924 1942 2003 2026 2028 2029 2030 2032 2033 2034
START	0012	86	87
STEP$	00B2	310	477
STK1	2C4D	849	841 864 866 898 1362
STKCHR	2C84	891	402 906
STKERR	2C91	897	892
STKMAX	00BD	148	1867 1888
STKMIN	00B7	147	1813 1814 1849 1905 1942 2026 2028
STLN	0030	119	120 804 816 819 1401 1402 1555 1571
STOP$	0098	284	478
STPT	0002	69	905 948 949 1434
STR$$	00DB	353	482
STREND	001A	107	1563 2159 2165 2175 2176 2177 2180 2189
STRIN$	00C7	331	1202 1673 1723 1881
STRSP	0018	106	1561 1562 1563 2154 2174 2182 2186
STVSPT	0024	112	1103 1131
SUB$	00A1	293	444
SUBSTK	0073	54	1053
SYM	0013	249	
SYMBOL	03E0	154	2021
SYMPT	0800	161	
SYMPTR	0040	127	1555 1557 1560 1561 1665 1667 1978 1986 2014 2015 2017 2018 2022 2032 2034 2043 2049 2054 2055 2055 2056 2057 2153
SYMTAB	003E	126	1047 1048 1545 1644 1655 1662 1665 1666 1683 1685 1686 1687 1693 2012 2018 2021 2056
SYNCHK	30B9	1628	1606 1608 1610 1651 1653 1714 1855
SYNTER	30C0	1630	1616 1621 1768 1801 1868 1872 1880 1889 1908 2114
TAB$	00FC	375	445
TABLE	000B	1	
TAN	0030	49	
TAN$	00D4	346	456
THEN$	00B0	308	479
TIMER	0079	58	575 587
TO$	00B1	309	435
TONE2	0036	51	736 1007
TOP	0003	1	
TOPSTK	0010	85	1814 1832 1910 1918 1955 2003 2030
TRACE	0005	185	
TRACE$	0090	276	493
UNBRE$	008F	275	505
UNLN	005F	210	2098
UNQ$1	2CEB	945	931 932 943 949

```
 /MBOL VALUE  DEF    REFERENCE TABLE

UNQST$ 00C8   332      333  334  926
UNQSTR 2CC0   924      404
UNTRA$ 0091   277      504
UPDAT$ 00F8   371      501
VAL$   00DA   352      459
VAR0   0000    66      779  7 5  787  788  802  803  804  811  812  815
                       841  853  866 1308 1317 1330 1331 1332 1336 1337
                      1657 1662 1663 1664 1828 1831 1832 1875 1884 1894
                      1899 1904 1905 1906 1907 1909 1974 1975 1976 1978
                      1988
VAR1   000D    82      562  579  581  583  593
VAR4   000E    83       84  856
VAR8   0013    88      932  944
VAR81  002C   116      117 1153 1154 1452 1453 1455 1759 1790
VAR9   0016   100
VARA   002A   115      548  639  647  648  650  656  660  661  662  664
                       665  667  676  677  679  680  681  694  695  696
                       713  714  715  718  727  728  743  744  746  748
                       911
VARC   0008    77      775  776  778  802  813  819  820  822  834 1315
                      1928 1929 1930 1936 2121 2122 2123
VARD   0009    79      856
VARIA$ 00F3   366      511
 'ARV  0001    67      568  574  595  596
 ARW   0020   110      548  556  680  749  760  836  911  912  915  919
                      1008 1011 1012 1014 1024 1026 1028 1029 1151 1152
                      1438 1439 1440 1470
VARY   0004    71      788  794  795  796  810  811  813  814
VARZ   0005    72
VDP    000C     1
VDPSTS 007B    59
VEL    0007     1
VPOP   0018   253     1105
VPUSH  0017   252
VRAMVS 06F8   160
VSPTR  006E   134     1103 1104 1108 1131 2160 2166
WARN$$ 2D24   991      400
WARNTP 03EB   155
X1     0002   163
X2     0003   164
XFLAG  0016    91     1503 1541 1587 1604 1618 1622 1640 1646 1678 1727
                      1751 1793 1798 1800 1826 1829 1834 1839 1851 1880
                      1916 1944 1969 1997 2000 2008 2010 2023 2058 2060
XN     001C   169
XPOINT 007E    61
XPT    000E     1
Y1     0017   165
YPOINT 007F    62
YPT    000D     1
7      005A   205
 ERO   0030   195     2092

TOTAL NUMBER OF SYMBOLS =   482
TOTAL NUMBER OF REFERENCES =  1931
```

```
                    1         TITLE FLMGR
                    2  ************ R E L O C A T I O N   I N F O **************
4D00                3  M$EXEC  EQU  >4D00          Module EXEC branch table address
2010                4  M$EDIT  EQU  >2010          Module EDIT branch table address
2828                5  M$PSCN  EQU  >2828          Module PSCAN branch table addres
2022                6  M$MSG   EQU  M$EDIT+>12     Message area start
                    7
                    8
4D88                9  ERR     EQU  M$EXEC+>88        'CAN'T DO THAT'
                   10
                   11
215A               12  KILSYM  EQU  M$EDIT+>14A       Kill symbol table routine
                   13  ********************************************************
001C               14  SREF    EQU  >1C            String ref. bytes
0020               15  VARW    EQU  >20            Usually for cursor position
0022               16  ERRCOD  EQU  >22            2 BYTES FOR EACH EXCEPT
0024               17  STVSPT  EQU  >24            'CHAT', 'BASE', AND 'FLAG'
0026               18  RTNG    EQU  >26
0028               19  NUDTAB  EQU  >28            Start of NUD table
002A               20  VARA    EQU  >2A            End of input string
002C               21  VAR81   EQU  >2C            Program token pointer
002E               22  EXTRAM  EQU  >2E
0030               23  STLN    EQU  >30            Line number table limit pointers
0032               24  ENLN    EQU  >32             "      "     "     "      "
0034               25  DATA    EQU  >34            Pointer to current DATA stmt
0036               26  LNBUF   EQU  >36
0038               27  RAMPTR  EQU  >38            During crunch loc. in crunch buf
003C               28  IOSTRT  EQU  >3C            Start of I/O chain in memory
003E               29  SYMTAB  EQU  >3E            Start of symbol table
0040               30  SYMPTR  EQU  >40            First free data-byte in memory
0040               31  FREWRD  EQU  SYMPTR         MULTI-USE OF SAME VARIABLE
0042               32  CHAT    EQU  >42            Current program character
0043               33  BASE    EQU  >43
0044               34  BUFFY   EQU  >44            Who knows ????????
004A               35  FAC     EQU  >4A            Floating point accumulator #1
005C               36  ARG     EQU  >5C            Floating point accumulator #2
0066               37  TEMP5   EQU  >66            TEMP FOR LITS05
006E               38  VSPTR   EQU  >6E            Value stack pointer
0073               39  SUBSTK  EQU  >73            Subroutine stack pointer
0075               40  RKEY    EQU  >75            Key code of last scan
0075               41  SIGN$   EQU  >75
0076               42  EXP$    EQU  >76
0088               43  FLAG    EQU  >88            General flag byte
00 9               44  GROMFL  EQU  >89            Indication of GROM execution
00                 45  RSTK    EQU  >88            STARTS AT >8A
00CE               46  PRTNFN  EQU  >CE            SOUND - previous tone finished
                   47  *
                   48  *          T E M P O R A R Y   V A R I A B L E S
                   49  *
0000               50  VAR0    EQU  >00
0001               51  VARV    EQU  >01
0002               52  MNUM    EQU  >02            Usually a counter
0004               53  PABPTR  EQU  >04            Pointer to current PAB
                   54  *       P A B   o f f s e t s
0002               55  FIL     EQU  2              File number within BASIC (0-255)
```

```
0003    56  OFS     EQU  3          Offset within record
0004    57  COD     EQU  4          I/O opcode
0005    58  FLG     EQU  5          I/O mode flag byte
0006    59  BUF     EQU  6          Start of data buffer
0008    60  LEN     EQU  8          Record length
0009    61  CNT     EQU  9          Character count
000A    62  RNM     EQU  10         Record number
000C    63  SCR     EQU  12         Screen base offset
000D    64  NLEN    EQU  13         Length of file descriptor
        65  *
0004    66  VARY    EQU  >04
0005    67  VARZ    EQU  >05
0006    68  CCPPTR  EQU  >06        Pointer to current column (base
0007    69  RECLEN  EQU  >07        Length of current record (maxim
0008    70  CCPADR  EQU  >08        Actual buffer address of column
0008    71  VARC    EQU  >08
0009    72  VARD    EQU  >09
000A    73  STADDR  EQU  >0A        Start address - usually for cop
000B    74  VAR2    EQU  >0B
000C    75  BYTE    EQU  >0C        String length for GETSTR
0017    76  FNUM    EQU  >17        Current file number for search
0017    77  TEMP    EQU  FNUM       Temporary...misc
0017    78  DSRFLG  EQU  FNUM       Internal = 60, External = 0
0017    79  OPTFLG  EQU  FNUM       Option flag byte during OPEN
        80  *       SUBDIVIDED INTO :
0000    81  MFLAG   EQU  0          Bit 0 - Mode flag (I/O mode)
0001    82  DFLAG   EQU  1          Bit 1 - DISPLAY/INTERNAL
0002    83  PFLAG   EQU  2          Bit 2 - PERMANENT
0003    84  SFLAG   EQU  3          Bit 3 - SEQUENTIAL/RELATIVE
0004    85  RFLAG   EQU  4          Bit 4 - FIXED/VARIABLE length
        86  *
000D    87  VAR1    EQU  >0D
000E    88  VAR4    EQU  >0E
0010    89  VAR5    EQU  >10
0011    90  VAR6    EQU  >11
0012    91  BFF     EQU  >12        2 BYTES
0013    92  VAR8    EQU  >13
0014    93  NBC     EQU  >14        2 BYTES
0016    94  VAR9    EQU  >16        2 BYTES
0018    95  STRSP   EQU  >18        2 BYTES
001A    96  STREND  EQU  >1A        2 BYTES
001E    97  NBD     EQU  >1E        2 BYTES
001E    98  BEE     EQU  >1E        2 BYTES
```

```
                    100  *
                    101  *              B A S I C   T O K E N   T A B L E
                    102  *
                    103  *                                      REVISED 8/23/78
                    104  *       EQU   >80          SPARE
0081                105  ELSE$    EQU   >81
                    106  *       EQU   >82          "::"              (62)
                    107  *       EQU   >83          "!"               (62)
0084                108  IF$      EQU   >84          "IF"
0085                109  GO$      EQU   >85          "GO"
0086                110  GOTO$    EQU   >86          "GOTO"
0087                111  GOSUB$   EQU   >87          "GOSUB"
0088                112  RETUR$   EQU   >88          "RETURN"
0089                113  DEF$     EQU   >89          "DEF"
008A                114  DIM$     EQU   >8A          "DIM"
008B                115  END$     EQU   >8B          "END"
008C                116  FOR$     EQU   >8C          "FOR"
008D                117  LET$     EQU   >8D          "LET"
008E                118  BREAK$   EQU   >8E          "BREAK"
008F                119  UNBRE$   EQU   >8F          "UNBREAK"
0090                120  TRACE$   EQU   >90          "TRACE"
0091                121  UNTRA$   EQU   >91          "UNTRACE"
0092                122  INPUT$   EQU   >92          "INPUT"
0093                123  DATA$    EQU   >93          "DATA"
0094                124  RESTO$   EQU   >94          "RESTORE"
0095                125  RANDO$   EQU   >95          "RANDOMIZE"
0096                126  NEXT$    EQU   >96          "NEXT"
0097                127  READ$    EQU   >97          "READ"
0098                128  STOP$    EQU   >98          "STOP"
0099                129  DELET$   EQU   >99          "DELETE"
009A                130  REM$     EQU   >9A          "REM"
009B                131  ON$      EQU   >9B          "ON"
009C                132  PRINT$   EQU   >9C          "PRINT"
009D                133  CALL$    EQU   >9D          "CALL"
009E                134  OPTIO$   EQU   >9E          "OPTION"
009F                135  OPEN$    EQU   >9F          "OPEN"
00A0                136  CLOSE$   EQU   >A0          "CLOSE"
00A1                137  SUB$     EQU   >A1          "SUB"
00A2                138  DISPL$   EQU   >A2          "DISPLAY"
                    139  *       EQU   >A3          "SUBEXIT"         (62)
                    140  *       EQU   >A4          "SUBEND"          (62)
                    141  *       EQU   >A5          "REAL"            (62)
                    142  *       EQU   >A6          "INTEGER"         (62)
                    143  *       EQU   >A7          "SCRATCH"         (62)
                    144  *       EQU   >A8          "ACCEPT"          FUTURE
                    145  *       EQU   >A9          "IMAGE"           FUTURE
                    146  *       EQU   >AA          SPARES
                    147  *       EQU   >AB
                    148  *       EQU   >AC
                    149  *       EQU   >AD
                    150  *       EQU   >AE
                    151  *       EQU   >AF
00B0                152  THEN$    EQU   >B0          "THEN"
00B1                153  TO$      EQU   >B1          "TO"
00B2                154  STEP$    EQU   >B2          "STEP"
```

```
00B3    155  COMMA$ EQU  >B3          ","
00B4    156  SEMIC$ EQU  >B4          ";"
00B5    157  COLON$ EQU  >B5          ":"
00B6    158  RPAR$  EQU  >B6          ")"
00B7    159  LPAR$  EQU  >B7          "("
        160  *      EQU  >B8          "&"                (62)
        161  *      EQU  >B9          SPARE
        162  *      EQU  >BA          "OR"               (62)
        163  *      EQU  >BB          "AND"              (62)
        164  *      EQU  >BC          SPARE FOR "XOR"    (62)
        165  *      EQU  >BD          "NOT"              (62)
00BE    166  EQUAL$ EQU  >BE          "="
00BF    167  LESS$  EQU  >BF          "<"
00C0    168  GREAT$ EQU  >C0          ">"
00C1    169  PLUS$  EQU  >C1          "+"
00C2    170  MINUS$ EQU  >C2          "-"
00C3    171  MULT$  EQU  >C3          "*"
00C4    172  DIVI$  EQU  >C4          "/"
00C5    173  CIRCU$ EQU  >C5          "^"
        174  *      EQU  >C6          SPARE
00C7    175  STRIN$ EQU  >C7          QUOTED STRING
00C8    176  UNQST$ EQU  >C8          UNQUOTED STRING
00C8    177  NUM$   EQU  UNQST$       ALSO NUMERICAL STRING
00C9    178  LN$    EQU  >C9          LINE NUMBER
        179  *      EQU  >CA          SPARE
00CB    180  ABS$   EQU  >CB          "ABS"
00CC    181  ATN$   EQU  >CC          "ATN"
00CD    182  COS$   EQU  >CD          "COS"
00CE    183  EXP$$  EQU  >CE          "EXP"
00CF    184  INT$   EQU  >CF          "INT"
00D0    185  LOG$   EQU  >D0          "LOG"
00D1    186  SGN$$  EQU  >D1          "SGN"
00D2    187  SIN$   EQU  >D2          "SIN"
00D3    188  SQR$   EQU  >D3          "SQR"
00D4    189  TAN$   EQU  >D4          "TAN"
00D5    190  LEN$   EQU  >D5          "LEN"
00D6    191  CHR$$  EQU  >D6          "CHR$"
00D7    192  RND$   EQU  >D7          "RND"
        193  *      EQU  >D8          "SEG$"
        194  *      EQU  >D9          "POS"
        195  *      EQU  >DA          "VAL"
        196  *      EQU  >DB          "STR$"
        197  *      EQU  >DC          "ASC"
        198  *      EQU  >DD          "PI"
00DE    199  REC$   EQU  >DE          "REC"
        200  ********************************************************
        201  *      EQU  >EC          "ALL"              (62)
        202  *      EQU  >ED          "USING"            (62)
        203  *      EQU  >EE          "BEEP"             (62)
        204  *      EQU  >EF          "ERASE"            (62)
        205  *      EQU  >F0          "AT"               (62)
00F1    206  BASE$  EQU  >F1          "BASE"
        207  *      EQU  >F2          "TEMPORARY"        (62)
00F3    208  VARIA$ EQU  >F3          "VARIABLE"         (62)
00F4    209  RELAT$ EQU  >F4          "RELATIVE"         (62)
```

```
00F5    210  INTER$ EQU  >F5        "INTERNAL"       (62)
00F6    211  SEQUE$ EQU  >F6        "SEQUENTIAL"
00F7    212  OUTPU$ EQU  >F7        "OUTPUT"
00F8    213  UPDAT$ EQU  >F8        "UPDATE"
00F9    214  APPEN$ EQU  >F9        "APPEND"
00FA    215  FIXED$ EQU  >FA        "FIXED"
00FB    216  PERMA$ EQU  >FB        "PERMANENT"
00FC    217  TAB$   EQU  >FC        "TAB"
00FD    218  NUMBE$ EQU  >FD        "#"
        219  *      EQU  >FE        GROM LINE POINTER
        220  *      EQU  >FF        SPARE
```

```
      *
      *            A S C I I   D E F I N I T I O N S
      *
0002  BREAK  EQU  >02            Keyboard break key
0020  SPACE  EQU  : :            " " - SPACE
002C  COMMA  EQU  :,:            "," - COMMA
002D  MINUS  EQU  :-:            "-" - MINUS
      *
      *        P R O P E R T Y   D E F I N I T I O N S
      *
000E  PABLEN EQU  14             PAB LENGTH
0320  CRNBUF EQU  >320           LOCATION OF CRUNCH BUFFER
      *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
      *   REPLACE    VRAMVS EQU >5F8   TO ADD LOWERCASE CHAR.
      *   FOR 99/4A BY CAROLYN LEE 3/3/81
      *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
06F8  VRAMVS EQU  >6F8           DEFAULT BASE OF VALUE STACK
      *                          V-STACK START ADDRESS
03EC  SAVEPC EQU  >3EC           PC SAVE ADDRESS
0065  STRVAL EQU  >65            VALUE IN ACCU IS STRING VALUE
001C  VWIDTH EQU  28             VIDEO SCREEN WIDTH
0060  OFFSET EQU  >60            OFFSET FOR VIDEO TABLES
02E0  SCRNBS EQU  >2E0           SCREEN BASE ADDRESS FOR LAST LI
      *
      *       E X T E R N A L   S U B R O U T I N E S
      *
0015  ASSGNV EQU  21             ASSIGN VARIABLE
2012  MAO    EQU  M$EDIT+>02     RETURN FROM "OLD" OR "SAVE"
4D16  CONVER EQU  M$EXEC+>16     CONVERT WITH WARNING
284E  ERR$$  EQU  M$PSCN+>26     ERROR MESSAGE ROUTINE
284C  WARN$$ EQU  M$PSCN+>24     WARNING MESSAGE ROUTINE
2844  MEMCHK EQU  M$PSCN+>1C     CHECK MEMORY FULL&MOVE STRING S
4D06  LINE   EQU  M$EXEC+>06     GET LINE NUMBER ROUTINE
4D08  DATAST EQU  M$EXEC+>08     SEARCH FOR NEXT "DATA" STATEMEN
0010  CALDSR EQU  >10            CALL DEVICE SERVICE ROUTINE
0014  CNS    EQU  >14            CONVERT FROM NUMBER TO STRING
0034  TONE1  EQU  >34            ATTRACTIVE BEEP
0036  TONE2  EQU  >36            DISTRACTIVE BEEP
0017  VPUSH  EQU  23             PUSH TO V-STACK
0018  VPOP   EQU  24             POP FROM V-STACK
001B  PGMCH  EQU  27             GET PROGRAM CHARACTER
0013  SYM    EQU  19             FIND SYMBOL ENTRY
0014  SMB    EQU  20             ALSO FOR ARRAYS
4D00  SCROL  EQU  M$EXEC+>00     SCROLL SCREEN UP ONE LINE
282E  LLIST  EQU  M$PSCN+>06     List a line
2832  READLN EQU  M$PSCN+>0A     Read a line from keyboard
2834  AUTO1  EQU  M$PSCN+>0C     Get args. for LIST command
2 3E  SEETWO EQU  M$PSCN+>16     Get program line
2014  SCDATA EQU  M$EDIT+>04     SCAN DATA AND MOVE TO CRUNCH BU
201A  MAIN1  EQU  M$EDIT+>0A     Return to main and re-init.
4D02  LITS05 EQU  M$EXEC+>02     Literal string common code
4D12  GETSTR EQU  M$EXEC+>12     Get string space
      *
      *     I / O   C O D E   D E F I N I T I O N S
0000  C$OPEN EQU  0              OPEN code
```

```
                277  C$CLOS EQU  1            CLOSE code
                278  C$READ EQU  2            READ code
                279  C$WRIT EQU  3            WRITE code
                280  C$REST EQU  4            RESTORE/REWIND code
                281  C$LOAD EQU  5            LOAD code
                282  C$SAVE EQU  6            SAVE code
                283  C$DELE EQU  7            DELETE code
                284  C$SCR  EQU  8            SCRATCH code
                285  C$STAT EQU  9            STATUS code
                286  *
                287  *       M E S S A G E S
                288  *
                289  MSG1   EQU  M$MSG+>0A    SYNTAX ERROR
                290  ERR$X  EQU  M$EXEC+>81   STRING/NUMBER MISMATCH
                291  ERR$4  EQU  M$EXEC+>7C   BAD VALUE
                292  MSG17  EQU  M$MSG+>9B    CAN'T DO THAT
                293  MSG20  EQU  M$MSG+>E6    TRY AGAIN
                294  MSG21  EQU  M$MSG+>F1    I/O ERROR
                295  MSG22  EQU  M$MSG+>FB    FILE ERROR
                296  MSG23  EQU  M$MSG+>106   INPUT ERROR
                297  MSG24  EQU  M$MSG+>112   DATA ERROR
                298  *
                299  *       C O N N E C T I O N   T O   R E S T
                300  *
                301          GROM 2
                302          ORG  0
  `00 426C      303          BR   DISPL1      "DISPLAY" ROUTINE
  ,02 4160      304          BR   DELET       "DELETE" ROUTINE
4004 4227       305          BR   PRINT       PRINT ROUTINE
4006 4344       306          BR   INPUT       INPUT ROUTINE (NOT YET IMPLEMENT
4008 401E       307          BR   OPEN        OPEN ROUTINE
400A 4174       308          BR   CLOSE       CLOSE ROUTINE
400C 41D7       309          BR   RESTOR      RESTORE ROUTINE
400E 45E3       310          BR   READ        READ ROUTINE
4010 4956       311          BR   GETDAT      GET DATA FROM GRAM/GROM
4012 41CF       312          BR   CLSALL      CLOSE ALL OPEN FILES (SUBROUTINE
4014 46FC       313          BR   SAVE        PROGRAM SAVE ROUTINE
4016 4641       314          BR   OLD         PROGRAM LOAD ROUTINE
4018 474C       315          BR   LIST        LIST routine
401A 4BFC       316          BR   OUTREC      Output record routine
401C 482B       317          BR   EOF         End of file routine
```

```
                    319  *
                    320  *       " O P E N "   S U B R O U T I N E
                    321  *
                    322  *       Handle the BASIC OPEN statement.  A legal syntax
                    323  *       only be something like
                    324  *
                    325  *            OPEN #{exp}:{string-exp}[,{open-option}]*
                    326  *
                    327  *       in which {open-option} is any of the following
                    328  *
                    329  *            DISPLAY, INPUT, VARIABLE, RELATIVE, INTERNAL
                    330  *            SEQUENTIAL, OUTPUT, UPDATE, APPEND, FIXED or
                    331  *            PERMANENT
                    332  *
                    333  *       Each keyword can only be used once, which is bein
                    334  *       checked with an OPTFLG-bit.  For each specific
                    335  *       option please refer to the related routine.
                    336  *
                    337  *       Scanning stops as soon as no next field starting
                    338  *       a comma can be found.
                    339  *
                    340  *       NOTE :  After the actual DSR OPEN has been perform
                    341  *               the length of the record, whether VARIABLE
                    342  *               FIXED, has to be non-zero.  A zero length
                    343  *               cause an INCORRECT STATEMENT error.
                    344  *
                    345  OPEN
 401E 064993        346    CALL CHKFN              See if we specified any file
 4021 77DE          347    BS   ERR$3              Definitely not...no # or #0 or
 4023 0649B1        348    CALL CHKCON             Check and search given filenumb
 4026 77DE          349    BS   ERR$3              *** FILE NUMBER EXISTS ***
                    350    $
                    351    $     *** ERROR IF NOT STOPPED ON COLON
                    352    $
 4028 D642B5        353    $IF @CHAT .NE. COLON$ GOTO ERR$1 Check for colon and g
 402B 40F5
 402D 0F1B          354    XML  PGMCH              Skip the colon...next token
 402F 064BA1        355    CALL PARFN              Parse filename and create PAB
 4032 932C          356    DDEC @VAR81             Backup pgm pointer for next tok
                    357  OPTION
 4034 0F1B          358    XML  PGMCH              Get next program character
                    359    $
                    360    $     Next field should start with a comma
                    361    $
                    362  OPTI$0
 4036 D642B3        363    $IF @CHAT .NE. COMMA$ GOTO CHECK
 4039 40FD
                    364    $
                    365    $     Enter HERE after comma exit in "SEQUENTIAL"
                    366    $
                    367  OPTI$1
 403B 0F1B          368    XML  PGMCH              Next token please.....
                    369    $
                    370    $     Treat DISPLAY and INPUT as special cases
                    371    $
```

```
403D D642A2   372     $IF @CHAT .EQ. DISPL$ GOTO OPT$6
4040 60D6
 042 D64292   373     $IF @CHAT .EQ. INPUT$ GOTO OPT$7
4045 60E0
4047 A642F3   374     SUB  VARIA$,@CHAT       Reduce keyword offset to 0
404A CA4209   375     $IF @CHAT .HE. 9 GOTO OPERR Keyword too high
404D 60F2
404F 8A42     376     CASE @CHAT              JUST IN CASE
4051 40AB     377       BR   OPT$01           Option VARIABLE
4053 406B     378       BR   OPT$02                  RELATIVE
4055 40D1     379       BR   OPT$03                  INTERNAL
4057 4070     380       BR   OPT$1                   SEQUENTIAL
4059 4095     381       BR   OPT$2                   OUTPUT
405B 409A     382       BR   OPT$3                   UPDATE
405D 40A4     383       BR   OPT$4                   APPEND
405F 40B0     384       BR   OPT$5                   FIXED
              385   *   BR   OPT$0                   PERMANENT
              386     $
              387     $          C A S E   0   -   " P E R M A N E N T "
              388     $
              389     $     Only check for multiple usage.  Since PERMANENT
              390     $     is the default, we might as well ignore it.....
              391     $
              392   OPT$0
4061 DA1704   393     $IF .BIT(PFLAG) @OPTFLG .NE. 0 GOTO OPERR
4064 40F2
4066 B61704   394     SB   @OPTFLG,PFLAG      Not used....use now
 069 4034     395     BR   OPTION             Treat as simple default
              396     $
              397     $          C A S E   2   -   " R E L A T I V E "
              398     $
              399     $     Select relative record file in PAB and fall throug
              400     $     in SEQUENTIAL code for multiple usage check.  Also
              401     $     handle initial file-size there.
              402     $
              403   OPT$02
406B B6E005   404     SB   RAM(FLG(PABPTR)),0     Indicate RELATIVE RECORD
406E 0401
              405     $
              406     $          C A S E   4   -   " S E Q U E N T I A L "
              407     $
              408     $     Check for multiple usage.  Remainder of syntax
              409     $     demands that we have something like
              410     $
              411     $          [{numeric expression}],.......
              412     $
              413     $     In case only a comma is found, we use the default.
              414     $     Everything else has to be evaluated as a numeric
              415     $     expression, convertable to a 16-bit integer value.
              416     $
              417   OPT$1
4070 DA1708   418     $IF .BIT(SFLAG) @OPTFLG .NE. 0 GOTO OPERR
  73 40F2
 075 B61708   419     SB   @OPTFLG,SFLAG      First time usage -> o.k.
4078 0F1B     420     XML  PGMCH              Check next token for default
```

```
                    421     $
                    422     $     Comma means default has been used
                    423     $
407A D642B3         424     $IF @CHAT .EQ.  COMMA$ GOTO OPTI$1
407D 603B
407F 8E4260         425     $IF @CHAT .EQ.  0 GOTO CHECK Check for end of statement
4082 FD
4083 06408D         426     CALL CHKPAR                 Perform combined checking & par
4086 BDE00A         427     DST  @FAC,RAM(RNM(PABPTR))  Non-zero result
4089 044A
408B 4036           428     BR   OPTI$0                 Scan other options
                    429     $
                    430     $     Parse and check a numeric argument in here....
                    431     $
                    432   CHKPAR
408D 0EB3           433     PARSE COMMA$                If not...parse up to comma
408F 06499C         434     CALL CHKCNV                 Check and convert to integer
4092 60F2           435     BS   OPERR                  Oops....someone made a mistake
4094 00             436     RTN                         Return to caller
                    437     $
                    438     $          C A S E   5   -   " O U T P U T "
                    439     $
                    440     $     Select mode code "01" and check for multiple
                    441     $     usage.  Use MFLAG bit in OPTFLG for checking.
                    442     $
                    443   OPT$2
4095 B6E005         444     OR   >2,RAM(FLG(PABPTR)) Mode code = 01
4098 0402
                    445     $
                    446     $          C A S E   6   -   " U P D A T E "
                    447     $
                    448     $     Default....Check for multiple usage only...
                    449     $
                    450   OPT$3
                    451     $
                    452     $     Test for previous usage of any mode setting
                    453     $
409A DA1701         454     $IF .BIT(MFLAG) @OPTFLG .NE. 0 GOTO OPERR
409D 40F2
409F B61701         455     SB   @OPTFLG,MFLAG          If not...set "MODE USED" bit
40A2 4034           456     BR   OPTION                 Continue option scan
                    457     $
                    458     $          C A S E   7   -   " A P P E N D "
                    459     $
                    460     $     Mode code "11" indicates APPEND mode.
                    461     $
                    462   OPT$4
40A4 B6E005         463     OR   >6,RAM(FLG(PABPTR)) Mode code = 11
40A7 0406
40A9 409A           464     BR   OPT$3
                    465     $
                    466     $          C A S E   1   -   " V A R I A B L E "
                    467     $
                    468     $     Change record type to VARIABLE and continue as ..
                    469     $
```

```
                    470  OPT$01
   B B6E005         471    SB   RAM(FLG(PABPTR)),4     Indicate variable length mo
  AE 0410
                    472    $
                    473    $          C A S E   8   -   " F I X E D "
                    474    $
                    475    $    FIXED is default.  Don't change anything, unless
                    476    $    argument is given.  In this case evaluate as numer
                    477    $    expression and check for 8-bit integer range.....
                    478    $    This routine is also used for VARIABLE !!!!!
                    479    $
                    480  OPT$5
40B0 0F1B           481    XML  PGMCH              Get next character
40B2 D642B3         482    $IF @CHAT .NE. COMMA$ THEN Could be some argument
40B5 60C7
40B7 8E4260         483      $IF @CHAT .NE. 0 THEN  Has to be....no end of line
40BA C7
40BB 06408D         484        CALL CHKPAR         Check & parse expression
                    485        $
                    486        $ Check for byte overflow (records can only be up
                    487        $ to 255 bytes in length).....
                    488        $
40BE 8E4A40         489        $IF @FAC .NE. 0 GOTO OPERR
40C1 F2
40C2 BCE008         490        ST   @FAC+1,RAM(LEN(PABPTR))  Select non-zero rec.-
40C5 044B
                    491      $END IF
                    492    $END IF
40C7 DA1710         493    $IF .BIT(RFLAG) @OPTFLG .NE. 0 GOTO OPERR
40CA 40F2
40CC B61710         494    SB   @OPTFLG,RFLAG      Prevent too much usage of modes
40CF 4036           495    BR   OPTI$0             Continue option scan
                    496    $
                    497    $          C A S E   3   -   " I N T E R N A L "
                    498    $
                    499    $    Select INTERNAL file type and continue in DISPLAY.
                    500    $
                    501  OPT$03
40D1 B6E005         502    SB   RAM(FLG(PABPTR)),3     Select INTERNAL type
40D4 0408
                    503    $
                    504    $          C A S E   9   -   " D I S P L A Y "
                    505    $
                    506    $    Default.  Only check for mutiple usage of either
                    507    $    DISPLAY or INTERNAL.......
                    508    $
                    509  OPT$6
40D6 DA1702         510    $IF .BIT(DFLAG) @OPTFLG .NE. 0 GOTO OPERR
40D9 40F2
40DB B61702         511    SB   @OPTFLG,DFLAG      Else set "DISPLAY/INTERNAL" flag
40DE 4034           512    BR   OPTION             Continue....DISPLAY is default
                    513    $
                    514    $          C A S E  1 0  -   " I N P U T "
                    515    $
                    516    $    Same as any other I/O type definition.  Mode code
```

```
                    517     $     "10"....continue in OPT$3
                    518     $
                    519   OPT$7
40E0 B6E005         520     OR    >4,RAM(FLG(PABPTR)) Mode code = 10
40E3 0404
40E5 409A           521     BR    OPT$3
                    522     $
                    523     $     CLRFRE deallocates previously allocated (parts of
                    524     $     PAB's and returns with an error message
                    525     $
                    526   CLRFRE
40E7 8602           527     CLR   @MNUM                 Undo any allocation
40E9 BC03E0         528     ST    RAM(OFS(PABPTR)),@MNUM+1 We need the length for t
40EC 0304
40EE A14002         529     DADD  @MNUM,@FREWRD         Update the first free word
40F1 00             530     RTN                         And return
                    531     $
                    532   OPERR
40F2 0640E7         533     CALL  CLRFRE                First undo the allocation
                    534   ERR$1
40F5 06284E         535     CALL  ERR$$                 Then give an error
40F8 202C           536     DATA  #MSG1                 "* INCORRECT STATEMENT"
40FA 052012         537     B     MAO
                    538     $
                    539     $     Continue with CHECK to complete the actual OPEN
                    540     $
                    541   CHECK
                    542     $
                    543     $     If the user hasn't specified VARIABLE or FIXED,
                    544     $     the default specification depends on the file
                    545     $     type.  Change current default (=VARIABLE) to
                    546     $     FIXED for RELATIVE files.
                    547     $
40FD DAE005         548     $IF .BITO RAM(FLG(PABPTR)) .EQ. 1 THEN RELATIVE RECORD
4100 040161
4103 0D
4104 DAE005         549       $IF .BIT4 RAM(FLG(PABPTR)) .EQ. 1 GOTO OPERR
4107 041040
410A F2
410B 4117           550     $SELSE                      Sequential file...check rec. mo
410D DA1710         551       $IF .BIT(RFLAG) @OPTFLG .EQ. 0 THEN No definition ye
4110 4117
4112 B6E005         552         SB    RAM(FLG(PABPTR)),4 Force VARIABLE mode
4115 0410
                    553       $END IF
                    554     $END IF
4117 064CC6         555     CALL  CDSR                  Call the DSR...return with erro
411A 57C0           556     BR    ERR$2B                  indication in COND.....
411C 87E00A         557     DCLR  RAM(RNM(PABPTR))      Make sure we start with record
411F 04
                    558     $
                    559     $     Check for undefined record length...The record
                    560     $     length for any type might be defined by the DSR
                    561     $
4120 8EE008         562     $IF RAM(LEN(PABPTR)) .EQ. 0 GOTO OPERR
```

```
4123 0460F2
4126 BC03E0   563    ST   RAM(LEN(PABPTR)),@MNUM+1 Get record length
4129 0804
412B 8602     564    CLR  @MNUM                 Create two byte result and alloc
412D 86E003   565    CLR  RAM(OFS(PABPTR))     - remove offset for later use
4130 04
4131 BD4A02   566    DST  @MNUM,@FAC           - prepare for space claim
              567    $
              568    $    Check for special case : no PAB's yet
              569    $
4134 8F3C41   570    $IF @IOSTRT .DEQ. 0 THEN
4137 3D
4138 BD3C04   571      DST  @PABPTR,@IOSTRT Simply enter the first pointer
413B 414F     572    $SELSE
413D BD0A3C   573      DST  @IOSTRT,@STADDR Search for the end of the chain
              574      $
4140 8FB00A   575      $WHILE RAM(@STADDR) .DNE. 0
4143 614B
4145 BD0AB0   576        DST  RAM(@STADDR),@STADDR Keep on deferring
4148 0A
4149 4140     577      $END WHILE
414B BDB00A   578      DST  @PABPTR,RAM(@STADDR) Update last chain link
414E 04
              579    $END IF
414F BDE006   580    DST  @PABPTR,RAM(BUF(PABPTR)) Set empty buffer first
4152 0404
4154 062844   581    CALL MEMCHK               Check memory overflow&string spa
4157 A54002   582    DSUB @MNUM,@FREWRD        Compute buffer entry address
415A A5E006   583    DSUB @MNUM,RAM(BUF(PABPTR)) Correct buffer address in P
415D 0402
415F 10       584    CONT                      Return to the parser
```

```
                    586  *
                    587  *          " D E L E T E "   R O U T I N E
                    588  *
                    589  *       Use file # 0 for this operation...  Parse the fi
                    590  *       name string-expression as usual, and delete the F
                    591  *       before actually calling the DSR.....
                    592  *
                    593  DELET
4160 8617           594    CLR  @FNUM                  Create file #0 - non-existing
4162 064BA1         595    CALL PARFN                  Handle as normal PAB OPEN
4165 8602           596    CLR  @MNUM                  Delete PAB again before calling
4167 BC03E0         597    ST   RAM(OFS(PABPTR)),@MNUM+1 Create double byte PAB 1
416A 0304
416C A14002         598    DADD @MNUM,@FREWRD          Update free word pointer
416F 064C39         599    CALL IOCALL                 Perform I/O call for actual del
4172 07             600    DATA C$DELE
4173 10             601    CONT                        Check for Junk on End in CONT
```

```
                    603  *
                    604  *              " C L O S E "   R O U T I N E
                    605  *
                    606  *       Syntax could be
                    607  *
                    608  *       CLOSE #<num exp>  or  CLOSE #<num exp>:DELETE
                    609  *
                    610  *       Possibly output pending records before
                    611  *       closing or deleting the file.....
                    612  *
                    613  CLOSE
4174 064993         614    CALL CHKFN                 Check for "no #" / "#0" cases
4177 77DE           615    BS   ERR$3                 Not for "CLOSE" you don't
4179 0649B1         616    CALL CHKCON                Check file number etc...
417C 57DE           617    BR   ERR$3                 *** FILE NUMBER NOT IN SYSTEM
417E 0649D0         618    CALL OUTEOF                Output pending records
4181 BEE004         619    ST   C$CLOS,RAM(COD(PABPTR)) Default to CLOSE I/O code
4184 0401
4186 D642B5         620    $IF @CHAT .EQ. COLON$ THEN  Check for ":DELETE" spec.
4189 4199
418B 0F13           621      XML  PGMCH               Request next input token
418D D64299         622      $IF @CHAT .NE. DELET$ GOTO ERR$1 Has to be DELETE
4190 40F5
4192 BEE004         623      ST   C$DELE,RAM(COD(PABPTR)) Change CLOSE to DELETE
4195 0407
4197 0F13           624      XML  PGMCH               Skip DELETE keyword
                    625    $END IF
 .99 064CC6         626    CALL CDSR                  Call DSR with whatever we have
419C 41A2           627    BR   CLOS$1               Reset means error.....
419E 0649E6         628    CALL DELPAB                Delete PAB and data-buffer
41A1 10             629    CONT                       Return to parser routine
                    630  CLOS$1
41A2 BD5CE0         631    DST  RAM(4(PABPTR)),@ARG    Save error code for message
41A5 0404
41A7 0649E6         632    CALL DELPAB                Now delete the PAB
41AA BD0440         633    DST  @FREWRD,@PABPTR       Store error-code in free memory
41AD A70400         634    DSUB 6,@PABPTR             Create standard size PAB
41B0 06
41B1 BDE004         635    DST  @ARG,RAM(4(PABPTR))   Copy error-code
41B4 045C
41B6 57D6           636    BR   ERR$2A                Exit to error-routine
```

```
                    638  *
                    639  *        " C L O S E   A L L "   R O U T I N E
                    640  *
                    641  *       CLOSE all the existing PABs...ignore errors
                    642  *
                    643  *       NOTE:  "CLSLBL" is used in the I/O error routine
                    644  *              to determine if a warning should be given
                    645  *              rather than an error.......
                    646  *
                    647    $REPEAT
 41B8 BD04B0        648      DST  RAM(@PABPTR),@PABPTR
 41BB 04
                    649  CLSA$0
 41BC 8FB004        650    $UNTIL RAM(@PABPTR) .DEQ. 0 Find last PAB in chain
 41BF 41B8
 41C1 0649D0        651    CALL OUTEOF               Take care of pending records
                    652  CLSLBL
 41C4 BEE004        653    ST   C$CLOS,RAM(COD(PABPTR)) Select CLOSE code
 41C7 0401
 41C9 064CC6        654    CALL CDSR                 CLOSE to DSR routine
 41CC 0649E6        655    CALL DELPAB               Delete PAB - ignore CLOSE erro
                    656  CLSALL
 41CF BD043C        657    DST  @IOSTRT,@PABPTR      Start at beginning of chain
 41D2 8F3C41        658    $IF @IOSTRT .DNE. 0 GOTO CLSA$0  Continue until done
 41D5 BC
 41D6 00            659    RTN                       And return
```

```
                 *
                 *            " R E S T O R E "   R O U T I N E
                 *
                 *      RESTORE can have any of four forms :
                 *
                 *      RESTORE                 Restore to first DATA
                 *      RESTORE 20              Restore DATA pointer
                 *      RESTORE #1              Rewind file number 1
                 *      RESTORE #1, REC 2       Position file 1 at rec 2
                 *
                 *      The code is simple and self explanatory...
                 *
                 RESTOR
41D7 874A          DCLR @FAC                Assume simple RESTORE
41D9 D642FD        $IF @CHAT .NE. NUMBE$ GOTO OLDCD
41DC 41F9
41DE 064993        CALL CHKFN               Check for #<filenumber>
41E1 8F4A62        $IF @FAC .DEQ. 0 GOTO OLDC$0 Found equivalent of #0
41E4 02
41E5 0649B1        CALL CHKCON              Check and decode file #
41E8 57DE          BR   ERR$3               Give error if file not there
41EA 0649D0        CALL OUTEOF              Output pending record
41ED 87E00A        DCLR RAM(RNM(PABPTR))    Initialize to record 0
41F0 04
41F1 064B03        CALL PARREC              Parse possible record clause
41F4 064CB9        CALL IOCALL              Call DSR routine with
 1F7 04            DATA C$REST                RESTORE I/O code
  F8 10            CONT                     Return if no error found
                   $
                   $    Following code is for handling RESTORE to
                   $    line number within program
                   $
                 OLDCD
41F9 8E4262        $IF @CHAT .NE. 0 THEN  Some line reference is available
41FC 02
41FD 8611            CLR @VAR6              Decode into line number
41FF 064D06          CALL LINE                in FAC (double)
                   $END IF
                 OLDC$0
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                 *      ERR is located in EXEC and is directly referenced
                 *      here.  If EXEC is moved the equate for ERR must be
                 *      changed and this must be reassembled.
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
4202 D53032        $IF @STLN .DEQ. @ENLN GOTO ERR      "* INCORRECT STATEME
4205 6D88
4207 BD3632        DST  @ENLN,@LNBUF        Start at beginning of program
420A A73600        DSUB 3,@LNBUF            Backup for first line number
420D 03
                   $
                   $    Check against given line number
                   $
  0E C54AB0        $WHILE @FAC .DH. RAM(@LNBUF) Try to get something highe
4211 36421F
4214 D53630          $IF @LNBUF .DEQ. @STLN GOTO ERR$7 Last line in progra
```

```
4217 77E8
4219 A73600    709      DSUB 4,@LNBUF          Get next entry in line # table
421C 04
421D 420E      710    $SEND WHILE              Try again with next line
421F A33600    711    DADD 3,@LNBUF            Undo subtraction
4222 03
4223 064D08    712    CALL DATAST              Setup pointers for READ
4226 10        713    CONT                     Continue PARSE
```

```
                    715  *
                    716  *                P R I N T   R O U T I N E
                    717  *
                    718  *       Handler for BASIC PRINT and DISPLAY statements....
                    719  *
                    720  *       First sort out if a file is used or the screen....
                    721  *       For the remainder of the PRINT routines, this
                    722  *       difference is indicated with the DSRFLG byte......
                    723  *       This byte is 0 for file I/O, <>0 for screen I/O...
                    724  *
                    725  PRINT
4227 D642FD         726    $IF @CHAT .EQ. NUMBE$ THEN Could still be anything
422A 426C
422C 064C9B         727      CALL INITKB            Initialize keyboard I/O
422F 064993         728      CALL CHKFN             Check if default or open channel
4232 8F4A62         729      $IF @FAC .DEQ. 0 GOTO PRN$10 Default intended
4235 5F
4236 0649B1         730      CALL CHKCON            Check and convert expression
4239 57DE           731      BR   ERR$3             Error if PAB not in system
                    732      $
                    733      $  PRINT allowed in output, append or update modes
                    734      $
423B DAE005         735      $IF .BIT2 RAM(FLG(PABPTR)) .EQ. 1 THEN
423E 040462
4241 49
4242 DAE005         736        $IF .BIT1 RAM(FLG(PABPTR)) .EQ. 0 GOTO ERR$3
   15 040277
   48 DE
                    737      $END IF
4249 D6E004         738      $IF RAM(COD(PABPTR)) .EQ. C$READ THEN
424C 040242
424F 54
4250 86E003         739        CLR  RAM(OFS(PABPTR))   Unpend pending INPUTs
4253 04
                    740      $END IF                Next WRITE selection is for
4254 BEE004         741      ST   C$WRITE,RAM(COD(PABPTR))  uncomplete PRINTs
4257 0403
4259 064C2A         742      CALL PRINIT            Initialize some variables
425C 064B03         743      CALL PARREC            Parse possible record clause
                    744  PRN$10
425F 8E4263         745      $IF @CHAT .EQ. 0 GOTO EOLEX End Of Line Exit
4262 25
4263 D642B5         746      $IF @CHAT .NE. COLON$ GOTO ERR$1  Where's that ":" ??
4266 40F5
4268 0F1B           747      XML  PGMCH
426A 426F           748    $SELSE                   Short branch over keyboard init
                    749  DISPL1
426C 064C9B         750      CALL INITKB            Init keyboard (DISPLAY entry)
                    751    $END IF
                    752    $
                    753    $REPEAT
  6F 064B6F         754      CALL TSTSEP            Test separator character
  72 D642FC         755      $IF @CHAT .EQ. TAB$ GOTO PRTAB  Handle TABs
4275 62D3
                    756      $
```

```
                    757       $  At this point we've checked TAB and ";", ",", ":"
                    758       $  The only remaining print items have to be expre
                    759       $       All expressions are being handled below.
                    760       $  If the result of the expression is a numeric,
                    761       $  the string is transformed into a string and
                    762       $  printed.  Strings are printed "as is".
                    763       $       The code for strings and converted numerics
                    764       $  cannot be made common, since numerics may require
                    765       $  an extra space behind the item, depending upon th
                    766       $  current position in the record.
                    767       $       Either way, the string is chunked up into li
                    768       $  pieces if it won't fit in an empty record.
                    769       $
                    770  OTHER
 4277 0EB5          771       PARSE COLON$          Evaluate the expression
                    772       $
                    773       $  Special code for INTERNAL file handling
                    774       $       Translate numeric datums into string
                    775       $  format and indicate length 8.  Then
                    776       $  check to see if the item fits within the
                    777       $  current record.  If not, it is an error,
                    778       $  since each item has to fit.
                    779       $
 4279 06433A        780       CALL TSTINT           Test for internal files
 427C 62A8          781       BS   OTHE$1           Nope...something different
 427E D64C65        782       $IF @FAC+2 .NE. STRVAL THEN Change numerics
 4281 6291
 4283 BE5608        783         ST   8,@FAC+12          to string length 8
 4286 350008        784         MOVE 8 FROM @FAC TO @ARG  save in ARG
 4289 5C4A
 428B BE555C        785         ST   ARG,@FAC+11        and use this as source
 428E 064B53        786         CALL RSTRING        Reserve some string space
                    787       $END IF
 4291 BC5C07        788       ST   @RECLEN,@ARG     Compute remaining space to EOR
 4294 A45C06        789       SUB  @CCPPTR,@ARC       for space checking
 4297 905C          790       INC  @ARG             Make it real space
 4299 C8515C        791       $IF @FAC+7 .HE. @ARG GOTO ERR$3 Not enough !!!!!!
 429C 77DE
                    792       $                     The = check includes length by
 429E BCB008        793       ST   @FAC+7,RAM(@CCPADR) Prestore string length
 42A1 51
 42A2 9108          794       DINC @CCPADR          Update actual RAM address
 42A4 9006          795       INC  @CCPPTR            and internal column pointer
 42A6 42AD          796       BR   OTHE$0
                    797  OTHE$1
 42A8 D64C65        798       $IF @FAC+2 .EQ. STRVAL THEN Print the string resul
 42AB 42B2
                    799  OTHE$0
 42AD 064C6E        800         CALL OSTRNG         Output the string to the reco
 42B0 42CE          801       $SELSE                      Numeric datum
 42B2 8655          802         CLR  @FAC+11        Select standard BASIC format
 42B4 060014        803         CALL CNS            Convert number to string
 42B7 064B53        804         CALL RSTRING        Reserve and copy string
 42BA 064C6E        805         CALL OSTRNG         Output the string
                    806         $
```

```
                    807         $ Possibly add an extra space if we're not at the e
                    808         $ of the current record...
                    809         $
42BD C80706         810         $IF @RECLEN .HE. @CCPPTR THEN Enough space left
42C0 42CE
42C2 BE8008         811           ST   SPACE,RAM(@CCPADR)   Add trailing space
42C5 20
42C6 A08008         812           ADD  @DSRFLG,RAM(@CCPADR) Take care of screen I/C
42C9 17
42CA 9108           813           DINC @CCPADR      Update current column address
42CC 9006           814           INC  @CCPPTR        and base 1 pointer
                    815         $END IF
                    816       $END IF
                    817   CHKSEP
42CE 064B6F         818       CALL TSTSEP           Check for legal delimiter
42D1 40F5           819       BR   ERR$1            Illegal delimiter
                    820       $
                    821       $  PRTAB - Print TAB as part of PRINT command
                    822       $
                    823   PRTAB
42D3 06433A         824       CALL TSTINT           Watch out for INTERNAL file type
42D6 57DE           825       BR   ERR$3            They can't handle TABs
42D8 0F1B           826       XML  PGMCH            Skip TAB keyword
42DA D642B7         827       $IF @CHAT .NE. LPAR$ GOTO ERR$1
42DD 40F5
42DF 0EB6           828       PARSE RPAR$           Parse TAB expression
42E1 064AF9         829       CALL CNVDEF           Check and convert to integer
 2E4 BC4C07         830       ST   @RECLEN,@FAC+2   Set modulo number
+2E7 064B62         831       CALL COMMOD           Compute remainder
42EA C4064B         832       $IF @CCPPTR .H. @FAC+1 THEN Position on next output r
42ED 42F2
42EF 064BFC         833         CALL OUTREC         Output current record - no pendi
                    834       $END IF
42F2 D4064B         835       $IF @CCPPTR .EQ. @FAC+1 GOTO CHKSEP Stay here
42F5 62CE
42F7 BC034B         836       ST   @FAC+1,@MNUM+1   Fill with spaces
42FA 064C43         837       CALL FILSPC           O.K...go ahead...fill'r up
42FD 42CE           838       BR   CHKSEP           And check separator again
                    839       $
                    840       $  Comma is simular to TAB, except that it generates
                    841       $  at least one space.  The exact number of spaces
                    842       $  generated depends upon the current position within
                    843       $  the record.  If the next fixed tab-position is
                    844       $  outside the record, the current record is output
                    845       $  and the column pointer is reset to column 1 of the
                    846       $  next record.
                    847       $
                    848   PRTCOM
42FF BC0306         849       ST   @CCPPTR,@MNUM+1  Compute initial # of spaces
4302 9203           850       DEC  @MNUM+1          Decrement for 0 origin
4304 8602           851       CLR  @MNUM            Clear high byte for double
4306 AE020E         852       DIV  14,@MNUM         TABs are 14 spaces apart
  09 9002           853       INC  @MNUM            Compute next TAB-stop
 30B AA020E         854       MUL  14,@MNUM           and actual position
430E C40703         855       $IF  @RECLEN .H. @MNUM+1 THEN Within this record
```

```
4311 431A
4313 9003      856          INC  @MNUM+1          Convert to real position
4315 064C43    857          CALL FILSPC           Fill spaces to new location
4318 431D      858        $SELSE                  Outside current record
               859          $
               860          $ The ":" (colon) separator is used to output the
               861          $ current record, and proceed to position 1 of the
               862          $ next record.
               863          $
               864  PRCOL
431A 064BFC    865          CALL OUTREC           Output the current record...
               866        $END IF
               867        $
               868        $  The ";" generates the null string.  Since all pri
               869        $  items should be separated by a separator, this or
               870        $  has been introduced to separate without moving to
               871        $  another position.  Notice that all separators joi
               872        $  up here.
               873        $
               874  PRSEM
431D 0F1B      875        XML  PGMCH            Skip the separator
431F 8E4242    876      $UNTIL @CHAT .EQ. 0     Exit on end of line
4322 6F
4323 4328      877      BR   PREXIT            End of PRINT statement
               878      $
               879      $    End of line without pending output
               880      $
               881  EOLEX
4325 064BFC    882      CALL OUTREC            Output current record
               883      $
               884      $    Print list ends with legal separator
               885      $
               886  PREXIT
4328 8E1743    887      $IF @DSRFLG .EQ. 0 THEN Regular file/device I/O
432B 34
432C 9206      888        DEC  @CCPPTR         Back to actual offset
432E BCE003    889        ST   @CCPPTR,RAM(OFS(PABPTR)) Save for next time
4331 0406
4333 10        890        CONT                 Continue with next stmt
               891      $END IF
4334 BC7F06    892      ST   @CCPPTR,XPT       Save actual value of screen ^
4337 947F      893      INCT XPT                 and update for base 3
               894  PARCON
4339 10        895      CONT                   Continue in PARSE routine
               896      $
               897      $    TSTINT - test for INTERNAL type file...set COND
               898      $               if file is NOT INTERNAL
               899      $
               900  TSTINT
433A 8E1749    901      $IF @DSRFLG .NE. 0 GOTO RTC Couldn't possibly be INTE
433D CC
433E DAE005    902      CLOG >08,RAM(FLG(PABPTR))   Set COND according to bit
4341 0408
4343 01        903      RTNC                   Return without changing COND
```

```
                    905  *
                    906  *                  I N P U T   R O U T I N E
                    907  *
                    908  *       First check for file or screen I/O.  If file I/O
                    909  *       then check for pending output and print that.
                    910  *       If screen I/O then check for input prompt.
                    911  *
                    912  *       Next collect the INPUT variable list on the V-stac
                    913  *       Get enough input from either file or keyboard, and
                    914  *       compare types with entries on V-stack.
                    915  *
                    916  *       After verification and approval, assign the values
                    917  *
                    918  INPUT
4344 064C9B         919    CALL INITKB              Assume keyboard INPUT
4347 D642FD         920    $IF @CHAT .EQ. NUMBE$ THEN Might be #0 or #1-255
434A 44B3
434C 064993         921      CALL CHKFN             Check for default #0
434F 8F4A64         922      $IF @FAC .DEQ. 0 GOTO INPU$2 #0 is equivalent to no #
4352 DC
4353 0649B1         923      CALL CHKCON            Check and convert and search and
4356 57DE           924      BR   ERR$3             Give error if required
                    925  *
                    926  *       INPUT allowed for input and update modes
                    927  *
4358 DAE005         928      $IF .BIT1 RAM(FLG(PABPTR)) .NE. 0 GOTO ERR$3
435B 040257
435E DE
435F 0649D0         929      CALL OUTEOF            Output any pending PRINT stuff
4362 BEE004         930      ST   C$READ,RAM(COD(PABPTR))  Ensure read operation
4365 0402
4367 064B03         931      CALL PARREC            Parse REC clause
436A D642B5         932      $IF @CHAT .NE. COLON$ GOTO ERR$1
436D 40F5
436F 0F1B           933      XML  PGMCH             Get next character
4371 8617           934      CLR  @DSRFLG           CLEAR KEYBOARD INPUT FLAG
                    935      $
                    936      $  INTERNAL files get special treatment
                    937      $
4373 DAE005         938      $IF .BIT3 RAM(FLG(PABPTR)) .EQ. 1 THEN INTERNAL file
4376 040864
4379 10
437A 8EE003         939        $IF RAM(OFS(PABPTR)) .EQ. 0 THEN Fresh start....
437D 044383
                    940  INTR$0
4380 064CC0         941          CALL IOCL$1        Get a new record through the DSR
                    942        $END IF
4383 BC28E0         943        ST   RAM(OFS(PABPTR)),@VARA+1  Regain possible offs
4386 0304
4388 862A           944        CLR  @VARA           Make that a two byte constant
438A BD66E0         945        DST  RAM(BUF(PABPTR)),@TEMP5  Get first address
438D 0604
438F A1662A         946        DADD @VARA,@TEMP5    Compute actual address within re
                    947  INTR$1
                    948  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
```

```
                    949  *      CHANGED FROM:
                    950  *      XML  SYM            Get the symbol table entry
                    951  *      XML  SMB              of the given variable
                    952  *      BY CAROLYN LEE 2/25/81 TO FIX THE PROBLEM WITH A
                    953  *      LONG CONSTANT IN A VARIABLE FIELD IN INPUT, ACCEPT,
                    954  *      READ ETC. STATEMENT FOR 99/4A
                    955  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
4392 064CEA         956          CALL BUGQ1
                    957  ********************************************************
4395 0F17           958          XML  VPUSH          And save it on the stack
4397 870C           959          DCLR @BYTE          Assume no data available
4399 C82BE0         960          $IF @VARA+1 .L. RAM(CNT(PABPTR)) THEN Pick up data
439C 090463
439F A8
43A0 BC0DB0         961            ST   RAM(@TEMP5),@BYTE+1 Length byte first
43A3 66
43A4 9166           962            DINC @TEMP5       Update both actual address
43A6 902B           963            INC  @VARA+1        and offset
                    964          $END IF
43A8 C64C63         965          $IF @FAC+2 .H. 99 THEN Has to be string variable
43AB 43B5
43AD BD500C         966            DST  @BYTE,@FAC+6 Set length of string
43B0 06492D         967            CALL CTMPST       Create temporary string
43B3 43E3           968          $SELSE
43B5 D60D08         969            $IF @BYTE+1 .NE. 8 GOTO ERR$3 "* FILE ERROR"
43B8 57DE
43BA 340C4A         970            MOVE @BYTE FROM RAM(@TEMP5) TO @FAC Copy value
43BD B066
43BF 8F4A63         971            $IF @FAC .DNE. 0 THEN Watch out for non-scaled s
43C2 E1
43C3 BE5C51         972              ST   FAC+7,@ARG     Test for legal numeric
                    973              $REPEAT
43C6 C6905C         974                $IF *ARG .H. 99 GOTO ERR$3 "* FILE ERROR"
43C9 6377DE
43CC 925C           975                DEC  @ARG     Next digit for test
43CE D65C4B         976              $UNTIL @ARG .EQ. FAC+1
43D1 43C6
43D3 BD5C4A         977              DST  @FAC,@ARG Copy in ARG for some testing
43D6 815C           978              DABS @ARG       Be sure we're positive
43D8 925D           979              DEC  @ARG+1     0 would cause underflow here
43DA C65D62         980              $IF @ARG+1 .H. 98 GOTO ERR$3 0 or >99
43DD 77DE
43DF 43E3           981            $SELSE
43E1 874C           982              DCLR @FAC+2     Be sure FAC+2 = 0 (no strings)
                    983            $SEND IF
                    984          $SEND IF
43E3 A1660C         985          DADD @BYTE,@TEMP5  Update address and
43E6 A02B0D         986          ADD  @BYTE+1,@VARA+1    offset again
43E9 0F15           987          XML  ASSGNV         Assign value to variable
43EB 86E003         988          CLR  RAM(OFS(PABPTR))   Undo allocated offsets
43EE 04
43EF D642B3         989          $IF @CHAT .EQ. COMMA$ THEN
43F2 440F
43F4 0F1B           990            XML  PGMCH        Get next text character
43F6 8E4264         991            $IF @CHAT .NE. 0 THEN Still some variables left
```

```
 43F9 03
  3FA C82BE0   992          $IF @VARA+1 .HE. RAM(CNT(PABPTR)) GOTO INTR$0
  3FD 090463
 4400 80
 4401 4392     993            BR   INTR$1     Still something left
               994          $END IF
 4403 C82BE0   995          $IF @VARA+1 .L. RAM(CNT(PABPTR)) THEN
 4406 090464
 4409 0F
 440A BCE003   996            ST   @VARA+1,RAM(OFS(PABPTR)) Save value of off
 440D 042B
               997          $END IF
               998        $END IF
 440F 10       999        CONT                   And CONTINUE
              1000      $END IF
 4410 0648CC  1001      CALL GETVAR            Collect variable list on stack
 4413 BF3803  1002      DST  CRNBUF,@RAMPTR   Initialize crunch buffer pointer
 4416 20
 4417 8607    1003      CLR  @RECLEN           Initialize field counter
 4419 BEE004  1004      ST   C$READ,RAM(COD(PABPTR)) Select READ operation
 441C 0402
 441E 8EE003  1005      $IF RAM(OFS(PABPTR)) .NE. 0 GOTO INP$31
 4421 044449
 4424 9138    1006      DINC @RAMPTR           Adjust for used record usage
              1007      $REPEAT
 4426 9338    1008        DDEC @RAMPTR         Back to end of last crunch
  428 BEB038  1009        ST   COMMA$,RAM(@RAMPTR) Fake legal separator
  42B B3
 442C 064CC0  1010        CALL IOCL$1          Get next input record
 442F 86E003  1011        CLR  RAM(OFS(PABPTR)) Reset offset within record
 4432 04
 4433 0645C6  1012        CALL RECENT          Compute record entry
 4436 BC2AE0  1013        ST   RAM(CNT(PABPTR)),@VARA Get record length
 4439 0904
 443B 8E2A64  1014        $WHILE @VARA .NE. 0
 443E 49
 443F A2B020  1015          ADD  OFFSET,RAM(@VARW)  Add video offset for norm
 4442 60
 4443 9120    1016          DINC @VARW          screen-type crunch - proceed f
 4445 922A    1017          DEC  @VARA          entire record.
 4447 443B    1018        $SEND WHILE
              1019   INP$31
              1020   *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
              1021   *
              1022   *      Correction for "null" inputs with files...
              1023   *                                   7/26/79 - Herman Schuurman
              1024   *
              1025   *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
              1026   *      $IF RAM(OFS(PABPTR)) .EQ. RAM(CNT(PABPTR)) THEN
              1027   *        DINCT @RAMPTR    Only empty field left - don't
              1028   *        CLR  @VAR6          crunch but count as one data-i
              1029   *      $SELSE
   49 0645C6  1030        CALL RECENT          Compute actual record entry
 444C BC2BE0  1031        ST   RAM(CNT(PABPTR)),@VARA+1 Compute end of record
 444F 0904
```

```
4451 862A    1032          CLR  @VARA           Make that a double byte
4453 A12AE0  1033          DADD RAM(BUF(PABPTR)),@VARA  Add buffer start addr
4456 0604
4458 932A    1034          DDEC @VARA           Point to last position in reco
445A 062014  1035          CALL SCDATA          Scan data fields as in DATA stm
445D 77E3    1036          BS   ERR$6           Crunch buffer overflowed
             1037  *       $SEND IF
445F 9011    1038          INC  @VAR6           Get correct # of fields (one of
4461 A00711  1039          ADD  @VAR6,@RECLEN Update # of fields up to now
4464 C80710  1040        $UNTIL @RECLEN .HE. @VAR5 O.K...THAT'S ENOUGH !!!!!!
4467 4426
4469 972C    1041        DDECT @VAR81           Backup program pointer
446B 0F1B    1042        XML  PGMCH             Re-inspect last token before EC
446D 0645C6  1043        CALL RECENT            Precompute record entry
4470 86E003  1044        CLR  RAM(OFS(PABPTR))  Assume no pending record
4473 04
4474 D642B3  1045        $IF @CHAT .EQ. COMMA$ THEN Make record pending
4477 44AE
4479 D40710  1046          $IF @RECLEN .NE. @VAR5 THEN Enough left for pendi
447C 64AE
447E A40710  1047            SUB  @VAR5,@RECLEN Compute remaining # of field
4481 A41107  1048            SUB  @RECLEN,@VAR6 # of fields used in last rec
             1049  INP$32
4484 D6B020  1050            $WHILE RAM(@VARW) .EQ. :":+OFFSET
4487 824496
             1051              $REPEAT          Skip quoted strings
448A 9120    1052                DINC @VARW
448C D6B020  1053              $UNTIL RAM(@VARW) .EQ. :":+OFFSET
448F 82448A
4492 9120    1054              DINC @VARW
4494 4484    1055            $SEND WHILE
             1056            $REPEAT            Search for Nth data item
4496 9120    1057              DINC @VARW       Update pointer
4498 D6EFFF  1058            $UNTIL RAM(-1(VARW)) .EQ. :,:+OFFSET
449B FF208C
449E 4496
44A0 9211    1059            DEC  @VAR6         Commas denote end of field
44A2 4484    1060            BR   INP$32        Continue until done
44A4 A520E0  1061            DSUB RAM(BUF(PABPTR)),@VARW  Compute current of
44A7 0604
44A9 BCE003  1062            ST   @VARW+1,RAM(OFS(PABPTR)) Store for next re
44AC 0421
             1063          $END IF
             1064        $END IF
44AE BC1110  1065        ST   @VAR5,@VAR6       Copy # of variables for check
44B1 452D    1066      $SELSE                   Now we're in for screen I/O
44B3 064C9B  1067        CALL INITKB            Initialize some variables for
44B6 BD0A2C  1068        DST  @VAR81,@STADDR  Save tokenpointer for prompt (
44B9 930A    1069        DDEC @STADDR           Backup to previous token
             1070        $REPEAT                Go into a tight loop
44BB 064B2F  1071          CALL NXTCHR          Get next program character
44BE 64D6    1072          BS   INP$37          Detected end of stmt
44C0 D642B5  1073        $UNTIL @CHAT .EQ. COLON$   Stop if we find a color
44C3 44BB
44C5 BD2C0A  1074        DST  @STADDR,@VAR81  Backup for actual prompt scan
```

```
44C8 0F1B    1075      XML  PGMCH            Jump into first character of prom
44CA 0E35    1076      PARSE COLON$          And try to decode a string expr.
44CC D64C65  1077      $IF @FAC+2 .NE. STRVAL GOTO ERR$1 Numeric prompts ille
44CF 40F5
44D1 064C6E  1078      CALL OSTRNG           Output the given prompt
44D4 44E5    1079      BR   INP$39           Exit without prompt backup
             1080  INP$37
44D6 BD2C0A  1081      DST  @STADDR,@VAR81   Backup to beginning of line
44D9 BE42B5  1082      ST   COLON$,@CHAT     Fake prompt with ":"
             1083  INPU$2
44DC 0645D3  1084      CALL CHKRM            Check for room for ?
44DF BE8008  1085      ST   :?:+OFFSET,RAM(@CCPADR) Display ?
44E2 9F
44E3 9508    1086      DINCT @CCPADR         Count it too
             1087  INP$39
44E5 D642B5  1088      $IF @CHAT .NE. COLON$ GOTO ERR$1 Join up with prompt
44E8 40F5
44EA 0F1B    1089      XML  PGMCH            Get next character
44EC 0648CC  1090      CALL GETVAR           Get variable list on V-stack
             1091  INPU$3
44EF 0645D3  1092      CALL CHKRM            Check for room for answer
44F2 BD2008  1093      DST  @CCPADR,@VARW    Copy current cursor position
             1094      $REPEAT
44F5 BE8008  1095        ST   : :+OFFSET,RAM(@CCPADR) Clear the remainder
44F8 80
44F9 9108    1096        DINC @CCPADR          of the current line
44FB CB0802  1097      $UNTIL @CCPADR .DHE. >2FE Stop if we're there
44FE FE44F5
4501 BFA2FE  1098      DST  >7F7F,RAM(>2FE) Replace edgechars in cur. line
4504 7F7F
4506 8E80CE  1099      $IF @PRTNFN .EQ. 0 THEN PREVIOUS TONE FINISHED
4509 450E
450B 060034  1100        CALL TONE1          ------ "BEEP" ------
             1101      $END IF
450E C16E0E  1102      DEX  @VAR4,@VSPTR     Don't destroy V-stack on BREAK
4511 062832  1103      CALL READLN           Input a line from the keyboard
4514 C16E0E  1104      DEX  @VAR4,@VSPTR     Restore V-stack pointer
4517 BF3803  1105      DST  CRNBUF,@RAMPTR   Initialize crunch buffer pointer
451A 20
451B 062014  1106      CALL SCDATA           Scan and crunch input line
451E 655F    1107      BS   WRN$5            Crunch buffer too small !!!!!!
4520 064D00  1108      CALL SCROL            Scroll up after crunching
4523 BE7F03  1109      ST   3,XPT            Reset XPT too - pending records
4526 9011    1110      INC  @VAR6            # fields = # commas + 1
4528 D41011  1111      $IF @VAR5 .NE. @VAR6 GOTO WRN$5 # of variables wrong
452B 455F
             1112    $END IF
             1113    $
             1114    $    Once we're here, all information should be availabl
             1115    $    After type verification for inptu and variables, we
             1116    $    push all value entries on the V-stack.
             1117    $    VAR6 = VAR5 = number of variables
             1118    $
452D BD1E34  1119    DST  @DATA,@BEE         Save current DATA pointer
4530 BF3403  1120    DST  CRNBUF+1,@DATA     Get crunch entry...skip length by
```

```
4533 21
4534 BD020E  1121     DST  @VAR4,@MNUM        Get entry in V-stack before PUSH
             1122   INPU$4
4537 A30200  1123     DADD 8,@MNUM            Point to first symbol table ent
453A 08
453B BD0630  1124     DST  RAM(@MNUM),@CCPPTR Get intermediate result
453E 02
453F 06495A  1125     CALL GETRAM             Get value descriptor from RAM
4542 DA8006  1126     $IF .BIT7 RAM(@CCPPTR) .EQ. 0 THEN Numerical value
4545 804571
4548 06493B  1127       CALL CHKNUM           Check entered value against numb
454B 4558    1128       BR   INPU$5           Found error
454D 8E5465  1129       $IF @FAC+10 .EQ. 0 GOTO INPU$6 Watch out for overflo
4550 76
4551 BD341E  1130       DST  @BEE,@DATA       Restore DATA pointer
4554 8E1745  1131       $IF @DSRFLG .NE. 0 GOTO WR$$5  Ask for input reente
4557 64
             1132   INPU$5
4558 8E1777  1133       $IF @DSRFLG .EQ. 0 GOTO ERR$6  FILE I/O IS FATAL
455B E3
455C BD341E  1134       DST  @BEE,@DATA       Restore DATA pointer on error
             1135   WRN$5
455F 06284C  1136       CALL WARN$$           Go here for simple warnings too
4562 2128    1137       DATA #MSG23           INPUT ERROR - TRY AGAIN
             1138   WR$$5
4564 31000B  1139       MOVE 11 FROM ROM(MSG20) TO RAM(SCRNBS+2)
4567 A2E221
456A 08
456B BF0802  1140       DST  SCRNBS+13,@CCPADR Reset CCPADR after warning s
456E ED
456F 44EF    1141       BR   INPU$3
             1142     $END IF
4571 06496C  1143     CALL CHKSTR             Check string input
4574 6558    1144     BS   INPU$5             ERROR...CHECK I/O TYPE
             1145   INPU$6
4576 06495A  1146     CALL GETRAM             Get separation character (RAM)
4579 D601B3  1147     $IF @VAR0+1 .NE. COMMA$ THEN
457C 6588
457E 9211    1148       DEC  @VAR6            Has to be end of data
4580 4558    1149       BR   INPU$5           If not...ERROR
4582 8E0145  1150       $IF @VAR0+1 .NE. 0 GOTO INPU$5
4585 58
4586 458C    1151     $SELSE
4588 9211    1152       DEC  @VAR6            Count number of value entries
458A 4537    1153       BR   INPU$4           Continue
             1154     $END IF
             1155     $
             1156     $    Assign cycle - assign values to variables
             1157     $    Because it rescans the program line, this code
             1158     $    can not be used for imperative statements, since
             1159     $    the crunch buffer get's destroyed on input.
             1160     $         The rescan is necessary because subscripts
             1161     $    should be evaluated AFTER all previous values
             1162     $    have been assigned, i.e.
             1163     $
```

```
                 $         INPUT I,A(I)   with values 2,3
                 $
                 $   Should assign value 3 to A(2) !!!!!!
                 $
                 $         No error-checking is done here, since types
                 $   are already validated.  We might get subscripts
                 $   out of range though !!!!
                 $
458C BF3403      DST  CRNBUF+1,@DATA      Prepare for input rescan
458F 21
4590 BD2C0A      DST  @STADDR,@VAR81      Restore token pointer for rescan
4593 932C        DDEC @VAR81              Backup one token
4595 BD6E0E      DST  @VAR4,@VSPTR        Restore original stack pointer
                 $REPEAT
4598 0F1B          XML  PGMCH             Get next program characters
459A 8E4265        $IF @CHAT .EQ. 0 GOTO INPU$7 Check ",EOL" combinations
459D C2
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><
                *      CHANGED FROM:
                *      XML  SYM           Rescan variable name
                *      XML  SMB           Get correct entry for arrays
                *      BY CAROLYN LEE 2/25/81 TO FIX THE PROBLEM WITH A
                *      LONG CONSTANT IN A VARIABLE FIELD IN INPUT, ACCEPT,
                *      READ ETC. STATEMENT FOR 99/4A
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><
459E 064CEA            CALL BUG01
                ****************************************************************
  A1 0F17          XML  VPUSH             Save on stack for ASSGNV
45A3 06495A        CALL GETRAM            Get first token of input value
45A6 D64C65        $IF @FAC+2 .NE. STRVAL THEN Numerical case
45A9 65B0
45AB 06493B          CALL CHKNUM          Check for numerical value
45AE 65B9            BS   INP$67          COND should be set (valid numeri
                   $END IF
45B0 06496C        CALL CHKSTR            Get the correct string value
45B3 BD0C50        DST  @FAC+6,@BYTE      Length for temporary string
45B6 06492D        CALL CTMPST            Create temporary string
                INP$67
45B9 0F15          XML  ASSGNV            Assign value to variable
45BB 06495A        CALL GETRAM            Skip separator (already checked)
45BE 8E4245      $UNTIL @CHAT .EQ. 0      Denotes end of input statement
45C1 98
                INPU$7
45C2 BD341E      DST  @BEE,@DATA          Restore DATA pointer
45C5 10          CONT                     Continue in PARSE
                 $
                RECENT
45C6 BC21E0      ST   RAM(OFS(PABPTR)),@VARW+1 Get record offset
45C9 0304
45CB 8620        CLR  @VARW               Double byte value reqr'd......
45CD A120E0      DADD RAM(BUF(PABPTR)),@VARW Got it.................
45D0 0604
  D2 00          RTN                      AND NOW, THE END IS NEAR.......
                 $
                CHKRM
```

```
45D3 C70802  1213    $IF @CCPADR .DH. SCRNBS+29 THEN not enough room for "?
45D6 FD45E0
45D9 064D00  1214      CALL SCROL              Scroll one line for "?"
45DC BF0802  1215      DST  SCRNBS+2,@CCPADR   and update CCPADR accordingly
45DF E2
             1216    $END IF
45E0 00      1217    RTN
```

```
                *
                *          R E A D   S T A T E M E N T
                *
                *      Assign DATA values to variables in READ-list
                *      one at a time.  Possibly search for new DATA
                *      statements in the program if the current DATA
                *      statement has been used.  Be careful with null
                *      entries............!
                *
                  $REPEAT
45E1 0F1B           XML  PGMCH           Get character following ","
                READ
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                *     CHANGED FROM:
                *     XML  SYM             Get pointers and correct entries
                *     XML  SMB             also allow for array-variables
                *     BY CAROLYN LEE 2/25/81 TO FIX THE PROBLEM WITH A
                *     LONG CONSTANT IN A VARIABLE FIELD IN INPUT, ACCEPT,
                *     READ ETC. STATEMENT FOR 99/4A
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
45E3 064CEA           CALL BUG01
                ****************************************************************
45E6 0F17           XML  VPUSH           Push on Vstack for assignment
45E8 D634FF         $IF @DATA .EQ. >FF GOTO ERR$7
45EB 77E8
45ED 064952         CALL GETGFL          Get next data item (RAM/GROM)
45F0 D64C65         $IF @FAC+2 .NE. STRVAL THEN
 5F3 6614
+5F5 D601C8           $IF @VAR0+1 .NE. NUM$ GOTO ERR$7 Not a numeric
45F8 57E8
45FA 064978           CALL CHKS$0          Build up string info
45FD 9150             DINC @FAC+6          Force legal delimiter on end
45FF 064D02           CALL LITS05          Copy numeric into string space
4602 BD561C           DST  @SREF,@FAC+12 Copy string start address
4605 A11C50           DADD @FAC+6,@SREF    Compute end address of string
4608 931C             DDEC @SREF           Back up over delimiter
460A 064D16           CALL CONVER          Convert string to number
460D D5561C           $IF @FAC+12 .DNE. @SREF GOTO ERR$7 WRONG !!!!!!!!
4610 57E8
4612 461C           $SELSE
4614 06496C           CALL CHKSTR          Check string input
4617 77E8             BS   ERR$7           Give error on error
4619 064D02           CALL LITS05          Allocate string in string space
                    $SEND IF
461C 0F15           XML  ASSGNV          Assign variable
461E 064952         CALL GETGFL          Get next datum from DATA stmt
4621 8E0146         $IF @VAR0+1 .EQ. 0 THEN   Only if not end of DATA stmt
4624 36
4625 9736             DDECT @LNBUF         Pointer to line # of DATA stmt
4627 BE34FF           ST   >FF,@DATA       Assume the worst - no more DATAs
                      $  WATCH OUT FOR "DATA" STMT AT END OF PROGRAM
462A D53630           $IF @LNBUF .DNE. @STLN THEN
  2D 6634
.J2F 9336               DDEC @LNBUF          Next line's 1st token address
4631 064D08             CALL DATAST          Get next DATA statement
```

```
                1268          $END IF
4634 463B       1269        $SELSE                    Within same DATA stmt
4636 D6018B     1270          $IF @VAR0+1 .NE. COMMA$ GOTO ERR$7  Has to be com
4639 57E8
                1271        $END IF
463B D64283     1272      $UNTIL @CHAT .NE. COMMA$ Worry about junk in CONT
463E 65E1
4640 10         1273      CONT
```

```
                    1275  *
                    1276  *                O L D   S T A T E M E N T
                    1277  *
                    1278  *       Get a program from an external device and
                    1279  *       reinitialize the program pointers.  Also
                    1280  *       update the line pointer table, since the
                    1281  *       memory size of the machine on which the program
                    1282  *       was created doesn't have to be the same as on the
                    1283  *       current system !!!!!!
                    1284  *
                    1285  OLD
  4641 064888       1286    CALL GPNAME               Get program name & reinitialize
  4644 BD0A04       1287    DST  @PABPTR,@STADDR      Compute memory start address
  4647 A10AE0       1288    DADD RAM(NLEN-1(PABPTR)),@STADDR Add PAB-name length
  464A 0C04
  464C A30A00       1289    DADD PABLEN-4,@STADDR                and PAB length
  464F 0A
  4650 BDE00A       1290    DST  @>70,RAM(RNM(PABPTR))  Compute # of available byte
  4653 0470
  4655 A5E00A       1291    DSUB @STADDR,RAM(RNM(PABPTR))
  4658 040A
  465A 91E00A       1292    DINC RAM(RNM(PABPTR))  Inlcude current address.
  465D 04
  465E BDE006       1293    DST  @STADDR,RAM(BUF(PABPTR)) for copy start
  4661 040A
  4663 BEE004       1294    ST   C$LOAD,RAM(COD(PABPTR)) Select LOAD I/O code
  4666 0405
  >68 064CC6        1295    CALL CDSR                 Call device service routine
 +66B 46DA          1296    BR   OLDER                OLD error detected
                    1297    $    STADDR still points to the info bytes
  466D BD02E0       1298    DST  RAM(2(STADDR)),@MNUM   First test checksum
  4670 020A
  4672 B902E0       1299    DXOR RAM(4(STADDR)),@MNUM        which is a simple XOR
  4675 040A
  4677 D5B00A       1300    $IF  RAM(@STADDR) .DNE. @MNUM GOTO OLDER
  467A 0246DA
  467D BD32E0       1301    DST  RAM(2(STADDR)),@ENLN   Copy new ENLN,
  4680 020A
  4682 BD30E0       1302    DST  RAM(4(STADDR)),@STLN             STLN and
  4685 040A
  4687 BD02E0       1303    DST  RAM(6(STADDR)),@MNUM               top of memory in
  468A 060A
  468C A30A00       1304    DADD 8,@STADDR            Point to program data
  468F 08
  4690 A53202       1305    DSUB @MNUM,@ENLN          Compute ENLN relative to top (-)
  4693 A53002       1306    DSUB @MNUM,@STLN             "    STLN     "    "   "  (-)
  4696 A50A30       1307    DSUB @STLN,@STADDR        Highest memory address used
  4699 8702         1308    DCLR @MNUM                Total # of bytes to be moved
  469B A50230       1309    DSUB @STLN,@MNUM          STLN = - (# of bytes - 1)
  469E 9102         1310    DINC @MNUM                Take care of that one
  46A0 BD0470       1311    DST  @>70,@PABPTR         PABPTR=destination - STADDR=sour
  46A3 A13270       1312    DADD @>70,@ENLN           Compute new address of ENLN
    A6 A13070       1313    DADD @>70,@STLN                and STLN
                    1314  OLD$1
  46A9 BCB004       1315    ST   RAM(@STADDR),RAM(@PABPTR)  Move the data bytes up
```

```
 46AC B00A
 46AE 930A     1315    DDEC @STADDR              Update the source
 46B0 9304     1317    DDEC @PABPTR                   and destination pointers
 46B2 9302     1318    DDEC @MNUM                Count the # of bytes moved
 46B4 46A9     1319    BR   OLD$1                And continue while not 0
 46B6 06215A   1320    CALL KILSYM               Release string space/symbol tab
               1321    $
               1322    $     Update line # links according to new size
               1323    $
 46B9 930A     1324    DDEC @STADDR              Get old memory size and compute
 46BB BD02B0   1325    DST  RAM(@STADDR),@MNUM        difference with new size
 46BE 0A
 46BF A50270   1326    DSUB @>70,@MNUM           Stop if sizes are same
 46C2 6745     1327    BS   LRMA0
 46C4 BD0A30   1328    DST  @STLN,@STADDR        Start relocations at STLN
               1329  OLD$2
 46C7 C9320A   1330    $IF @ENLN .DL. @STADDR GOTO LRMA0 AND CONTINUE UP TO E
 46CA 4745
 46CC 950A     1331    DINCT @STADDR             Skip the line #
 46CE A5B00A   1332    DSUB @MNUM,RAM(@STADDR)        UPDATE THE LINK
 46D1 02
 46D2 B2B00A   1333    AND  >7F,RAM(@STADDR)     Remove any breakpoints set
 46D5 7F
 46D6 950A     1334    DINCT @STADDR             Skip the link...next line #
 46D8 46C7     1335    BR   OLD$2                And continue until done
               1336    $
               1337    $     OLD error exit code...don't kill machine....
               1338    $
               1339  OLDER
               1340  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><
               1341  * In 99/4A:  During OLD process, do not destroys the exi
               1342  *            ING PROGRAM IN VDP WHEN ERRORS COMING BACK
               1343  *            FROM CALLING DSR.
               1344  * DELETE FOLLOWING TWO LINES
               1345  * DST  @>70,@ENLN           RE-INITIALIZE ENLN
               1346  * DST  @>70,@STLN                         STLN
               1347  * ADD FOLLOWING TWO LINES AND THE WARNING MESSAGE ITSELF
               1348  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><
 46DA 06284C   1349    CALL WARN$$
 46DD 46E4     1350    DATA #CHKMSG
 46DF 06215A   1351    CALL KILSYM               Kill symbol table and strings
 46E2 57C3     1352    BR   ERR$2                AND TAKE ERROR EXIT
               1353   BASE 0,0,>300,>300,0,0,>60
 46E4 17A3A8   1354  CHKMSG DATA 23,:CHECK PROGRAM IN MEMORY:
 46E7 A5A3AB
 46EA 80B0B2
 46ED AFA7B2
 46F0 A1AD80
 46F3 A9AE80
 46F6 ADA5AD
 46F9 AFB2B9
               1355   BASE 0,0,>300,>300,0,0,0
```

```
                    1357  *
                    1358  *              S A V E   S T A T E M E N T
                    1359  *
                    1360  *       THE SAVE STATEMENT SAVES A PROGRAM IMAGE TO AN
                    1361  *       external device, includinh all the information
                    1362  *       the system needs for rebuilding the program image
                    1363  *       a machine with a different memory size..also inclu
                    1364  *       is a checksum for rudimetary error checking.....
                    1365  *
                    1366  SAVE
 46FC 064888        1367    CALL @PNAME             This will also close all files
                    1368  *
                    1369  *           Clear all BREAKPOINTS first
                    1370  *
 46FF BD0A30        1371    DST  @STLN,@STADDR      Point to first line
 4702 950A          1372    DINCT @STADDR           Skip line #
                    1373    $REPEAT
 4704 B2B00A        1374      RB   RAM(@STADDR),7   Be sure there's no breakpoint
 4707 7F
 4708 A30A00        1375      DADD 4,@STADDR        Move to next entry
 470B 04
 470C C50A32        1376    $UNTIL @STADDR .DH. @ENLN
 470F 4704
 4711 BD0A40        1377    DST  @FREWRD,@STADDR    Store additional control info
 4714 930A          1378    DDEC @STADDR            Back up some more for 2 byte sav
 4716 BDB00A        1379    DST  @>70,RAM(@STADDR)  First current top of memory
 4719 70
  '1A 970A          1380    DDECT @STADDR                     then
 471C BDB00A        1381    DST  @STLN,RAM(@STADDR)       STLN line # table ^
 471F 30
 4720 970A          1382    DDECT @STADDR                     then
 4722 BDB00A        1383    DST  @ENLN,RAM(@STADDR)       ENLN line # table ^
 4725 32
 4726 970A          1384    DDECT @STADDR                     then
 4728 BDB00A        1385    DST  @STLN,RAM(@STADDR)       STLN
 472B 30
 472C B9B00A        1386    DXOR @ENLN,RAM(@STADDR)           XORed with ENLN
 472F 32
 4730 BDE006        1387    DST  @STADDR,RAM(BUF(PABPTR)) Save start address in PAB
 4733 040A
 4735 930A          1388    DDEC @STADDR
 4737 BDE00A        1389    DST  @>70,RAM(RNM(PABPTR))  Compute # of bytes used
 473A 0470
 473C A5E00A        1390    DSUB @STADDR,RAM(RNM(PABPTR))   and store that in PAB t
 473F 040A
 4741 064CB9        1391    CALL IOCALL             Call Device Service Routine for
 4744 06            1392    DATA C$SAVE               SAVE operation
                    1393  LRMA0
 4745 BE8088        1394    ST   >20,@FLAG          Reset EDIT mode flag
 4748 20
                    1395    $                       "Ours is not to wonder why"..sti
 4749 052012        1396    B    MA0                And continue
```

```
                    1398  *
                    1399  *                L I S T   R O U T I N E
                    1400  *
                    1401  *       LIST lists a readable copy of the current program
                    1402  *       image to the specified device.  In case no device
                    1403  *       is specified, the listing is copied to the screen
                    1404  *
                    1405  *            This routine uses the fact that ERR$$ returns
                    1406  *       to the caller if the call has been issued in EDIT
                    1407  *       mode.  Therefore, ERR$2 will return to MAIN1, whi
                    1408  *       will reinitiate the variable stuff
                    1409  *
                    1410  LIST
474C 8714           1411    DCLR @NBC                 Create some kind of control for
474E 871E           1412    DCLR @NBD                   defaults
4750 BE082D         1413    ST   MINUS,@VARC          Select "-" as separator
4753 062834         1414    CALL AUTO1                Pick up any available arguments
                    1415    $
                    1416    $     If either NBC or NBD is non-zero, use it.
                    1417    $     For zero values replace the default (ENLN-3, STLN
                    1418    $
4756 8F1447         1419    $IF @NBC .DEQ. 0 THEN
4759 68
475A BD14EF         1420      DST  RAM(-3(ENLN)),@NBC Use standard default
475D FFFD32
4760 8F1E47         1421      $IF @NBD .DEQ. 0 THEN
4763 68
                    1422  LIST$0
4764 BD1E30         1423        DST  RAM(@STLN),@NBD  Also default for end line
4767 30
                    1424      $END IF
                    1425    $END IF
                    1426    $
                    1427    $     Now first evaluate what we've got in NBC
                    1428    $
4768 8F1E47         1429    $IF @NBD .DEQ. 0 THEN  Check for combination xxx-
476B 7A
                    1430      $REPEAT
476C 9320           1431        DDEC @VARW            Backup to the separation mark
476E D6B020         1432      $UNTIL RAM(@VARW) .NE. : :+OFFSET
4771 80676C
4774 D6B020         1433      $IF RAM(@VARW) .EQ. :-:+OFFSET GOTO LIST$0 Select la
4777 8D6764
                    1434    $END IF
477A C91E14         1435    $IF @NBD .DL. @NBC THEN Found something like LIST 15-1
477D 6782
477F BD1E14         1436      DST  @NBC,@NBD          Replace by LIST 15-15
                    1437    $END IF
4782 BD4414         1438    DST  @NBC,@BUFFY          Prepare for line # search
4785 06283E         1439    CALL SEETWO
4788 BD142E         1440    DST  @EXTRAM,@NBC         Get first real line # in NBC
478B BD441E         1441    DST  @NBD,@BUFFY
478E 06283E         1442    CALL SEETWO               Evaluate second line #
4791 C5B02E         1443    $IF  RAM(@EXTRAM) .DH. @NBD THEN
4794 1E479B
```

```
4797 A32E00  1444        DADD 4,@EXTRAM          Else take next lower line
479A 04
             1445    $END IF
479B BD1E2E  1446    DST  @EXTRAM,@NBD       Which could be equal to NBC
479E 932C    1447    DDEC @VAR81             Backup to last CHAT
47A0 0F1B    1448    XML  PGMCH              Retrieve last CHAT
47A2 8E4267  1449    $IF @CHAT .NE. 0 THEN   Devicename available
47A5 F3
47A6 0641CF  1450      CALL CLSALL             Close all files that are open
47A9 BF6E06  1451      DST  VRAMVS,@VSPTR      Re-initialize the V-stack
47AC F8
47AD BD246E  1452      DST  @VSPTR,@STVSPT     And it's base
47B0 0F1B    1453      XML  PGMCH              Get name length in CHAT
47B2 BF0407  1454      DST  VRAMVS+16,@PABPTR Get entrypoint in PAB
47B5 08
47B6 8617    1455      CLR  @DSRFLG            Indicate device I/O
47B8 31000D  1456      MOVE NLEN FROM ROM(PAB) TO RAM(@PABPTR)
47BB B00448
47BE 1E
47BF BF0807  1457      DST  VRAMVS+16+NLEN,@CCPADR Selsct start address for
47C2 15
47C3 BC4C42  1458      ST   @CHAT,@FAC+2       Number of characters to copy
47C6 904C    1459      INC  @FAC+2             Plus length byte
             1460  LIST$1
47C8 BCB008  1461      ST   @CHAT,RAM(@CCPADR) Copy the bytes one by one
47CB 42
47CC 0F1B    1462      XML  PGMCH              Get next character
47CE 9108    1463      DINC @CCPADR            CCPADR ends up with highest addr
47D0 924C    1464      DEC  @FAC+2             Count total # of characters
47D2 47C8    1465      BR   LIST$1
47D4 064CC0  1466      CALL IOCL$1             Perform OPEN on DSR
47D7 864A    1467      CLR  @FAC               Create double byte PAB length
47D9 BC07E0  1468      ST   RAM(LEN(PABPTR)),@RECLEN Get record length
47DC 0804
47DE BC4B07  1469      ST   @RECLEN,@FAC+1     Compute record length
47E1 A14A08  1470      DADD @CCPADR,@FAC       Get highest address used
47E4 BDE006  1471      DST  @CCPADR,RAM(BUF(PABPTR)) Store it
47E7 0408
47E9 C54A30  1472      $IF @FAC .DH. @STLN GOTO ERR$2A  Not enough room
47EC 77D6
47EE BE0601  1473      ST   1,@CCPPTR          Clear first line in output
47F1 47F9    1474    $SELSE
47F3 BE7F1F  1475      ST   VWIDTH+3,XPT       For common code usage
47F6 064C9B  1476      CALL INITKB             Reset current record length
             1477    $SEND IF
             1478    $REPEAT                 Display at least one line
47F9 06282E  1479      CALL LLIST              List the current line
47FC 03      1480      SCAN                    Test for a break key
47FD 4804    1481      BR   LIST$3             No key
47FF D67502  1482      $IF @RKEY .EQ. BREAK GOTO LIST$4
4802 680D
             1483  LIST$3
04 A71400    1484      DSUB 4,@NBC             Pointer to next line
4807 04
4808 C51E14  1485    $UNTIL @NBD .DH. @NBC   Displayed all lines in range
```

```
 480B 47F9
                  1486  LIST$4
 480D 8E1748      1487    $IF @DSRFLG .EQ. 0 THEN Device I/O -> output last rec
 4810 1B
 4811 0648FC      1488      CALL OUTREC           Output the last record
 4814 064CB9      1489      CALL IOCALL           Close the device properly
 4817 01          1490      DATA C$CLOS
 4818 05201A      1491      B    MAIN1
                  1492    $END IF
 481B 052012      1493    B     MAO               Restart the variables too
                  1494  *
                  1495  *       PAB image used in LIST function
                  1496  *
                  1497  PAB
 481E 000000      1498    DATA #0,#0,0,>12,#0,0,0,#0,OFFSET
 4821 000012
 4824 000000
 4827 000000
 482A 60
```

```
                    1500  *
                    1501  *              E O F   R O U T I N E
                    1502  *
                    1503  *       EOF(X) returns status codes on file X.  The meaning
                    1504  *       of the resultcodes is :
                    1505  *
                    1506  *            -1              Physical End Of File
                    1507  *            0               Not at End Of File yet
                    1508  *            +1              Logical End Of File
                    1509  *
                    1510  EOF
 482B D642B7        1511    $IF @CHAT .NE. LPAR$ GOTO ERR$1 "* INCORRECT STATEMENT"
 482E 40F5
 4830 0EFF          1512    PARSE >FF                 Parse up to matching ")"
 4832 06499C        1513    CALL CHKCNV               Convert and search for PAB
 4835 6D7C          1514    BS   ERR$4                Avoid 0's and negatives
 4837 8E4A4D        1515    $IF @FAC .NE. 0 GOTO ERR$4 >255 is also illegal
 483A 7C
 483B BD5C3C        1516    DST  @IOSTRT,@ARG         Search PAB chain
                    1517  EOF$0
 483E 8E5C77        1518    $IF @ARG .EQ. 0 GOTO ERR$3 Avoid end of PAB chain
 4841 DE
 4842 D4E002        1519    $IF RAM(FIL(ARG)) .EQ. @FAC+1 GOTO EOF$1
 4845 5C4B68
 4848 4F
 4849 BD5CB0        1520    DST  RAM(@ARG),@ARG       Defer to next PAB
 484C 5C
 484D 483E          1521    BR   EOF$0
                    1522  EOF$1
 484F C1045C        1523    DEX  @ARG,@PABPTR         Save current PAB ^ and set new o
 4852 BE5E09        1524    ST   C$STAT,@ARG+2        Select status code without
 4855 C0E004        1525    EX   @ARG+2,RAM(COD(PABPTR))  destroying original code
 4858 045E
 485A 064CC0        1526    CALL IOCL$1               Get the information from the DSR
 485D C1045C        1527    DEX  @ARG,@PABPTR         Restore original PAB ^ and origi
 4860 BCE004        1528    ST   @ARG+2,RAM(COD(ARG))   I/O code
 4863 5C5E
 4865 BC5EE0        1529    ST   RAM(SCR(ARG)),@ARG+2  And pick up STATUS
 4868 0C5C
 486A 310008        1530    MOVE 8 FROM ROM(FLOAT1) TO @FAC Get floating 1
 486D 4A4880
 4870 DA5E03        1531    CLOG 3,@ARG+2             Test EOF bits
 4873 687D          1532    BS   EOF$2                No EOF indication
 4875 DA5E02        1533    $IF .BIT1 @ARG+2 .EQ. 1 THEN Physical EOF
 4878 687C
 487A 834A          1534      DNEG @FAC               Make result -1
                    1535    $END IF
 487C 10            1536    CONT
                    1537  EOF$2
 487D 874A          1538    DCLR @FAC                 Create result 0
 487F 10            1539    CONT
                    1540  *
                    1541  FLOAT1
 4880 400100        1542    DATA >40,1,0,0,0,0,0,0  Floating point +1
 4883 000000
```

- 4886 0000

```
                    1544  *
                    1545  *     L O A D / S A V E   U T I L I T Y   R O U T I N E S
                    1546  *
                    1547  *       GPNAME gets program name for OLD and SAVE
                    1548  *       Can also be used for future implementation of
                    1549  *       REPLACE statement.
                    1550  *            Also gives valuable contribution to updating
                    1551  *       of program pointers (VSPTR, STVSPT, FLAG, etc.)
                    1552  *       and creation of LOAD/SAVE PAB.
                    1553  *
                    1554  GPNAME
4888 868088         1555    CLR  @FLAG                Avoid returns from ERR$$ routine
488B D642C7         1556    $IF @CHAT .NE. STRIN$ THEN
488E 6895
4890 D642C8         1557      $IF @CHAT .NE. NUM$ GOTO ERR$1 "* SYNTAX ERROR"
4893 40F5
                    1558    $END IF
4895 0641CF         1559    CALL CLSALL               First close all open files
4898 BF6E06         1560    DST  VRAMVS,@VSPTR        Re-initialize V-stack
489B F8
489C BD246E         1561    DST  @VSPTR,@STVSPT          and stack-bottom
489F 06215A         1562    CALL KILSYM               Kill the symbol table
48A2 BF0407         1563    DST  VRAMVS+8,@PABPTR     Create PAB as low as possible
48A5 00
48A6 86B004         1564    CLR  RAM(@PABPTR)         Clear PAB with ripple-move
48A9 350009         1565    MOVE PABLEN-5 FROM RAM(@PABPTR) TO RAM(1(PABPTR))
48AC E00104
 8AF B004
48B1 0F1B           1566    XML  PGMCH                Get length of file-spec.
48B3 A70400         1567    DSUB 4,@PABPTR            Make it a regular PAB ^
48B6 04
48B7 BCE00D         1568    ST   @CHAT,RAM(NLEN(PABPTR)) Copy name length to PAB
48BA 0442
48BC BD0AE0         1569    DST  RAM(NLEN-1(PABPTR)),@STADDR Avoid problems (bugs!)
48BF 0C04
48C1 340AE0         1570    MOVE @STADDR FROM RAM(@VAR81) TO RAM(NLEN+1(PABPTR))
48C4 0E04B0
48C7 2C
                    1571    $
                    1572    $     OLD and SAVE can only be imperative
                    1573    $
48C8 BE34FF         1574    ST   >FF,@DATA            Clear DATA line
48CB 00             1575    RTN                       That's all folks
```

```
                1577  *
                1578  *  R E A D / I N P U T   U T I L I T Y   R O U T I N E S
                1579  *
                1580  GETVAR
 48CC BD0A2C    1581    DST  @VAR81,@STADDR    Save token pointer to first chr
 48CF 8610      1582    CLR  @VAR5             Clear # of parsed variables
 48D1 BD0E6E    1583    DST  @VSPTR,@VAR4      Save first entry in V-stack
                1584    *
                1585    $    Start parse cycle for INPUT statement
                1586    *
                1587  GETV$0
                1588  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                1589  *      CHANGED FROM:
                1590  *      XML  SYM          Get correct symbol table entry
                1591  *      BY CAROLYN LEE 2/25/81 TO FIX THE PROBLEM WITH A
                1592  *      LONG CONSTANT IN A VARIABLE FIELD IN INPUT, ACCEPT
                1593  *      READ ETC. STATEMENT FOR 99/4A
                1594  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
 48D4 CA4280    1595    $IF @CHAT .HE. >80 GOTO ERR$1
 48D7 60F5
 48D9 0F13      1596    XML  SYM               Get correct symbol table entry
                1597  ************************************************************
 48DB 8611      1598    CLR  @VAR6             Start with zero paren nesting
                1599  GETV$1
 48DD D642B7    1600    $IF @CHAT .EQ. LPAR$ THEN Increment counter for "("
 48E0 48E4
 48E2 9011      1601      INC  @VAR6
                1602    $END IF
 48E4 8E1168    1603    $IF @VAR6 .NE. 0 THEN  Watch out for final balance
 48E7 F7
 48E8 8E4260    1604      $IF @CHAT .EQ. 0 GOTO ERR$1  EOL without balance
 48EB F5
 48EC D642B6    1605      $IF @CHAT .EQ. RPAR$ THEN    Decrement for ")"
 48EF 48F3
 48F1 9211      1606        DEC  @VAR6
                1607      $END IF
 48F3 0F1B      1608      XML  PGMCH           Get character following last ")
 48F5 48DD      1609      BR   GETV$1
                1610    $END IF
 48F7 0F17      1611    XML  VPUSH             Push entry to V-stack
 48F9 9010      1612    INC  @VAR5             Count all pushed variables
 48FB 8E4269    1613    $IF @CHAT .NE. 0 THEN  Next should either be EOL or ",
 48FE 0E
 48FF D642B3    1614      $IF @CHAT .NE. COMMA$ GOTO ERR$1  Not EOL - try ","
 4902 40F5
 4904 0F1B      1615      XML  PGMCH           Watch out for ",<EOL>" combinat
 4906 8E4248    1616      $IF @CHAT .NE. 0 GOTO GETV$0  Just continue
 4909 D4
 490A 8E1740    1617      $IF @DSRFLG .NE. 0 GOTO ERR$1 Error for keyboard I/O
 490D F5
                1618    $END IF
 490E 00        1619    RTN
                1620    *
                1621    $    Create a temporary string in memory.  BYTE
                1622    $    contains the length
```

```
                    1623    $
                    1624  CTSTR
 ?0F BF4C65         1625    DST  >6500,@FAC+2       Indicate string in FAC
+912 00
                    1626  CTSTR0
4913 BD500C         1627    DST  @BYTE,@FAC+6       Copy string length in FAC+6
4916 35001A         1628    MOVE 26 FROM @FAC+8 TO RAM(>3C00) Save FAC+8 and up
4919 A3C052
491C 064D12         1629    CALL GETSTR             Reserve the string
491F 35001A         1630    MOVE 26 FROM RAM(>3C00) TO @FAC+8 Restore FAC+8 area
4922 52A3C0
4925 BD4E1C         1631    DST  @SREF,@FAC+4       Copy start address of string
4928 BF4A00         1632    DST  SREF,@FAC          And indicate temp. string
492B 1C
492C 00             1633    RTN
                    1634    $
                    1635    $     Create a temporary string from TEMP5.  Length
                    1636    $     is given in BYTE.
                    1637    $
                    1638  CTMPST
492D 06490F         1639    CALL CTSTR              Create the temporary string
4930 8E5169         1640    $IF @FAC+7 .NE. 0 THEN
4933 3A
4934 340CB0         1641      MOVE @BYTE FROM RAM(@TEMP5) TO RAM(@SREF)
4937 1CB066
                    1642    $END IF                 non-empty
493A 00             1643    RTN
                    1644  *
                    1645  *       CHKNUM - Check for numeric argument
                    1646  *
                    1647  CHKNUM
493B D601C8         1648    $IF @VAR0+1 .EQ. NUM$ THEN
493E 4951
4940 06495A         1649      CALL GETRAM           Get string length
4943 BD5634         1650      DST  @DATA,@FAC+12    STore entry for conversion
4946 8600           1651      CLR  @VAR0            Prepare for double action
4948 A13400         1652      DADD @VAR0,@DATA      Get end of data field
494B 064D16         1653      CALL CONVER           Convert data to FAC #
                    1654      $
                    1655      $  Conversion should also end at end of field
                    1656      $
494E D55634         1657      DCEQ @DATA,@FAC+12    Set COND according to equality
                    1658    $END IF
4951 01             1659    RTNC                    Back to the caller
                    1660  *
                    1661  *       Get data from GRAM or GROM
                    1662  *
                    1663  GETGFL
4952 BC4D80         1664    ST   @GROMFL,@FAC+3     Select target memory = GROM
4955 89
                    1665  GETDAT
4956 8E4D49         1666    $IF @FAC+3 .EQ. 0 THEN Get everything from RAM
 '59 62
                    1667  GETRAM
495A BC01B0         1668      ST   RAM(@DATA),@VAR0+1   Get data in VAR0+1
```

```
 495D 34
 495E 864D     1669       CLR  @FAC+3           Be sure FAC+3 = 0 !!!!
 4960 4969     1670     $SELSE
 4962 330001   1671       MOVE 1 FROM ROM(@DATA) TO @VAR0+1
 4965 010000
 4968 34
               1672     $END IF
 4969 9134     1673     DINC @DATA              Go to next datum for next time
 496B 00       1674     RTN
               1675   *
               1676   *
               1677   *
               1678   CHKSTR
 496C 8750     1679     DCLR @FAC+6             Assume we'll have an empty stri
 496E D601C7   1680     $IF @VAR0+1 .NE. STRIN$ THEN
 4971 6978
 4973 D601C8   1681       $IF @VAR0+1 .NE. NUM$ GOTO EMPSTR  See.......
 4976 4987
               1682     $END IF
               1683   CHKS$0
 4978 064956   1684     CALL GETDAT             Next datum is length byte
 497B 8650     1685     CLR  @FAC+6             Be sure high byte = 0 !!!!!!!
 497D BC5101   1686     ST   @VAR0+1,@FAC+7     Prepare FAC for string assignme
 4980 BD6634   1687     DST  @DATA,@TEMP5       Save string address for assignm
 4983 A13450   1688     DADD @FAC+6,@DATA       Update DATA ^ for end of field
 4986 00       1689     RTN
               1690     $
               1691     $     Empty strings are handled below
               1692     $
               1693   EMPSTR
 4987 D601B3   1694     $IF @VAR0+1 .NE. COMMA$ THEN
 498A 6990
 498C 8E0149   1695       $IF @VAR0+1 .NE. 0 GOTO RTC
 498F CC
               1696     $END IF
 4990 9334     1697     DDEC @DATA              Backup data pointer for empties
 4992 00       1698     RTN
               1699
```

```
                    1701  *
                    1702  *          O P E N / C L O S E / R E S T O R E
                    1703  *            U T I L I T Y   R O U T I N E S
                    1704  *
                    1705  *       CHKFN - Check for token = "#" and collect and check
                    1706  *       filenumber.  Also convert filenumber to (two byte)
                    1707  *       integer and check for range 0<X<256.
                    1708  *
                    1709  CHKFN
 4993 D642FD        1710    $IF @CHAT .NE. NUMBE$ GOTO ERR$1
 4996 40F5
 4998 0F1B          1711    XML  PGMCH                Skip "#" token
 499A 0E35          1712    PARSE COLON$              Parse argument up to ":"
                    1713    $
                    1714    $     Code to check for negative or zero result in
                    1715    $     floating point accu.  If not...convert to
                    1716    $     integer and return two byte integer in FAC
                    1717    $
                    1718  CHKCNV
 499C D64C65        1719    $IF @FAC+2 .EQ. STRVAL GOTO ERR$X  String/number mismatc
 499F 6D81
 49A1 8654          1720    CLR  @FAC+10              Clear error-code byte
 49A3 0F12          1721    XML  18                   Convert to two byte integer
 49A5 8E544D        1722    $IF @FAC+10 .NE. 0 GOTO ERR$4
 49A8 7C
 49A9 DA4A80        1723    $IF .BIT7 @FAC .NE. 0 GOTO RTC Negative result
 49AC 49CC
 49AE 8F4A          1724    DCZ  @FAC                 And return with COND set/reset
 49B0 01            1725    RTNC
                    1726    $
                    1727  CHKCON
 49B1 BC174B        1728    ST   @FAC+1,@FNUM         Move result into FNUM
                    1729    $
                    1730    $     Check for high byte not zero ( >255 )
                    1731    $
 49B4 8E4A4D        1732    $IF @FAC .NE. 0 GOTO ERR$4
 49B7 7C
                    1733    $
                    1734    $     Search routine - Search for a given file number
                    1735    $     in the chain of allocated PABs.
                    1736    $     IOSTRT contains the start of the PAB-chain
                    1737    $
 49B8 BD043C        1738    DST  @IOSTRT,@PABPTR      Get first link in the chain
                    1739  CHKF$1
                    1740    $
                    1741    $     Check for last PAB in the chain and exit if found
                    1742    $
 49BB 8F0469        1743    $IF @PABPTR .DNE. 0 THEN  Check if file # is correct
 49BE CF
 49BF D4E002        1744      $IF RAM(FIL(PABPTR)) .EQ. @FNUM GOTO RTC
 49C2 041769
 49C5 CC
 49C6 BD04B0        1745      DST  RAM(@PABPTR),@PABPTR    Try the next PAB
 49C9 04
 49CA 49BB          1746      BR   CHKF$1
```

```
                1747  RTC
 49CC D40000    1748      CEQ  @0,@0              Force COND to "SET"
                1749    $END IF
 49CF 01        1750    RTNC                      Exit with no COND change
                1751  *
                1752  *       OUTEOF outputs the last record if this
                1753  *       record is non-empty, and if the PAB is
                1754  *       open for non-input mode (UPDATE, APPEND
                1755  *       or OUTPUT).
                1756  *
                1757  OUTEOF
 49D0 8617      1758    CLR  @DSRFLG
 49D2 D6E004    1759    $IF RAM(COD(PABPTR)) .EQ. C$WRITE THEN Non-input mode
 49D5 040349
 49D8 E5
 49D9 8EE003    1760      $IF RAM(OFS(PABPTR)) .NE. 0 THEN Non-empty record
 49DC 0469E5
 49DF 064C2A    1761        CALL PRINIT           Initiate for output
 49E2 064BFC    1762        CALL OUTREC           Output and remove pending cond.
                1763      $END IF
                1764    $END IF
 49E5 00        1765    RTN                       Return to whoever called
```

```
                    1767  *
                    1768  *      DELPAB routine - delete a given PAB from the chain
                    1769  *      under the assumption that the PAB exists
                    1770  *
                    1771  DELPAB
                    1772    $
                    1773    $     First compute start and end address for block move
                    1774    $
 49E6 BD0AE0        1775    DST   RAM(BUF(PABPTR)),@STADDR Get lowest used address
 49E9 0604
 49EB 930A          1776    DDEC @STADDR              Make that an address following F
 49ED 8608          1777    CLR  @CCPADR              Get highest address in CCPADR(2)
 49EF BC09E0        1778    ST   RAM(NLEN(PABPTR)),@CCPADR+1  complete the two byte
 49F2 0D04
 49F4 A2090D        1779    ADD  PABLEN-1,@CCPADR+1   Add PAB length - 1
 49F7 A10804        1780    DADD @PABPTR,@CCPADR   Compute actual address within RA
 49FA D53C04        1781    $IF @IOSTRT .DNE. @PABPTR THEN Watch out for first PAB
 49FD 6A26
 49FF BD023C        1782      DST  @IOSTRT,@MNUM   Figure out where link to PAB is
 4A02 D5B002        1783      $WHILE RAM(@MNUM) .DNE. @PABPTR Continue while not fo
 4A05 046A0E
 4A08 BD02B0        1784        DST  RAM(@MNUM),@MNUM   Defer to next link in chair
 4A0B 02
 4A0C 4A02          1785      $END WHILE              Short end for code-savings
 4A0E BDB002        1786      DST  RAM(@PABPTR),RAM(@MNUM) Copy link over deleted P
 4A11 B004
 4A13 8FB002        1787      $IF RAM(@MNUM) .DNE. 0 THEN Adjust link only if not s
   16 6A20
 4A18 A1B002        1788        DADD @CCPADR,RAM(@MNUM) Add deleted # of bytes for
 4A1B 08
 4A1C A5B002        1789        DSUB @STADDR,RAM(@MNUM)    link correction
 4A1F 0A
                    1790      $END IF
 4A20 BD04B0        1791      DST  RAM(@MNUM),@PABPTR   Get new PABPTR
 4A23 02
 4A24 4A37          1792    $SELSE
 4A26 BD3CB0        1793      DST  RAM(@PABPTR),@IOSTRT Update first link
 4A29 04
 4A2A 8F3C6A        1794      $IF @IOSTRT .DNE. 0 THEN  Only adjust if not last lir
 4A2D 34
 4A2E A13C08        1795        DADD @CCPADR,@IOSTRT    Add deleted # of bytes
 4A31 A53C0A        1796        DSUB @STADDR,@IOSTRT
                    1797      $END IF
 4A34 BD043C        1798      DST  @IOSTRT,@PABPTR      Get new PABPTR
                    1799    $END IF
                    1800    $
                    1801    $     Move the bytes below the deleted block up in
                    1802    $     memory. This includes both variables and PABs
                    1803    $
 4A37 BD020A        1804    DST  @STADDR,@MNUM      Get # of bytes to move
 4A3A A50240        1805    DSUB @FREWRD,@MNUM
 4A3D BD0608        1806    DST  @CCPADR,@CCPPTR    Save destination address
   40 8F026A        1807    $WHILE @MNUM .DNE. 0
 4A43 51
 4A44 BCB008        1808      ST   RAM(@STADDR),RAM(@CCPADR) Move byte-by-byte
```

```
4A47 B00A
4A49 930A     1809      DDEC @STADDR          Update source
4A4B 9308     1810      DDEC @CCPADR                  and destination pointe
4A4D 9302     1811      DDEC @MNUM            Also update counter value
4A4F 4A40     1812    $SEND WHILE             End WHILE loop (Also ends on 0)
4A51 A5080A   1813    DSUB @STADDR,@CCPADR    Compute # of bytes of old PAB
4A54 8F046A   1814    $IF @PABPTR .DNE. 0 THEN    Avoid trouble with last PA
4A57 71
4A58 8FB004   1815      $WHILE RAM(@PABPTR) .DNE. 0 Ad infinitum (or fundum)
4A5B 6A6C
4A5D A1B004   1816        DADD @CCPADR,RAM(@PABPTR)  Adjust link to next PAB
4A60 08
4A61 A1E006   1817        DADD @CCPADR,RAM(BUF(PABPTR)) Update the buffer li
4A64 0408
4A66 BD04B0   1818        DST  RAM(@PABPTR),@PABPTR Get next link in chain
4A69 04
4A6A 4A58     1819      $SEND WHILE
4A6C A1E006   1820      DADD @CCPADR,RAM(BUF(PABPTR))  Update buffer link
4A6F 0408
              1821    $END IF
              1822    $    Adjust symbol table links
4A71 8F3E6A   1823    $IF @SYMTAB .DNE. 0 THEN
4A74 F5
4A75 D13E06   1824      $IF @SYMTAB .DLT. @CCPPTR THEN Only update lower lin
4A78 6AF5
4A7A A13E08   1825        DADD @CCPADR,@SYMTAB Get symbol table pointer back
4A7D BD043E   1826        DST  @SYMTAB,@PABPTR Get pointer for update
              1827  DELP$1
4A80 8E8089   1828        $IF @GROMFL .NE. 0 GOTO DELP$2
4A83 4A8C
4A85 D1E004   1829        $IF RAM(4(PABPTR)) .DLT. @STLN THEN If imperative
4A88 04306A
4A8B 91
4A8C A1E004   1830  DELP$2  DADD @CCPADR,RAM(4(PABPTR))  Adjust name pointer
4A8F 0408
              1831        $END IF
4A91 D2B004   1832        $IF RAM(@PABPTR) .LT. 0 THEN If string-fix bkptrs
4A94 006ADC
4A97 BE4A07   1833          ST   >07,@FAC          Mask to get # of dims
4A9A B04AB0   1834          AND  RAM(@PABPTR),@FAC Get # of dims
4A9D 04
4A9E BD4C04   1835          DST  @PABPTR,@FAC+2    Ptr to 1st dim max
4AA1 A34C00   1836          DADD 6,@FAC+2             or string pointer
4AA4 06
4AA5 BF5000   1837          DST  1,@FAC+6          Number of pointers to char
4AA8 01
4AA9 864E     1838          CLR  @FAC+4            For 2-byte use of option t
4AAB 8E4A6A   1839          $WHILE @FAC .NE. 0     While more dimensions
4AAE C3
4AAF BE4F01   1840              ST   1,@FAC+5      Assume option base 0
4AB2 A44F43   1841              SUB  @BASE,@FAC+5 But correct if base 1
4AB5 A14EB0   1842              DADD RAM(@FAC+2),@FAC+4   Get dim maximum
4AB8 4C
4AB9 A94E50   1843              DMUL @FAC+6,@FAC+4   Mpy it in
4ABC 924A     1844              DEC  @FAC          Next dim
```

```
4ABE 954C     1845              DINCT @FAC+2
4AC0 054AAB   1846            $END WHILE
              1847  *
              1848  *       FAC+2 now points at the 1st string pointer
              1849  *       FAC+6 contains the # of ptrs that need to be change
              1850  *
4AC3 8F506A   1851          $WHILE @FAC+6 .DNE. 0  While pointers to change
4AC6 DC
4AC7 BD4AB0   1852             DST   RAM(@FAC+2),@FAC   Get ptr to string
4ACA 4C
4ACB 8F4A6A   1853             $IF @FAC .DNE. 0 THEN   If string in non-null
4ACE D5
4ACF BDEFFF   1854               DST  @FAC+2,RAM(-3(FAC))  Fix backpointer
4AD2 FD4A4C
              1855             $END IF
4AD5 9350     1856             DDEC  @FAC+6          One less ptr to change
4AD7 954C     1857             DINCT @FAC+2          Point to next pointer
4AD9 054AC3   1858           $END WHILE
              1859         $END IF
4ADC 8FE002   1860         $IF RAM(2(PABPTR)) .DNE. 0 THEN
4ADF 046AF5
4AE2 D1E002   1861           $IF RAM(2(PABPTR)) .DLT. @CCPPTR THEN
4AE5 04066A
4AE8 F5
4AE9 A1E002   1862             DADD @CCPADR,RAM(2(PABPTR)) Adjust next value li
4AEC 0408
4AEE BD04E0   1863             DST  RAM(2(PABPTR)),@PABPTR    Next entry
4AF1 0204
4AF3 4A80     1864             BR    DELP$1
              1865           $END IF
              1866         $END IF
              1867       $END IF
              1868     $END IF
4AF5 A14008   1869     DADD @CCPADR,@FREWRD   Update free word pointer
4AF8 00       1870     RTN
```

```
                    1872  *
                    1873  *       CNVDEF - Convert to 2-byte integer and default to
                    1874  *                1 on negative or 0.....
                    1875  *
                    1876  CNVDEF
4AF9 06499C         1877    CALL CHKCNV             Check and convert
4AFC 4B02           1878    BR   CNVD$0
4AFE BF4A00         1879    DST  1,@FAC             Default to 1 for minus and 0
4B01 01
                    1880  CNVD$0
4B02 00             1881    RTN                     And return without COND set
                    1882  *
                    1883  *       PARREC parses a possible REC clause in INPUT,
                    1884  *       PRINT or RESTORE.  In case a comma is detected
                    1885  *       without a REC clause following it, the COND
                    1886  *       is set upon return.  In case a REC clause is
                    1887  *       specified for a file opened for SEQUENTIAL access
                    1888  *       a * FILE ERROR is given.
                    1889  *
                    1890  PARREC
4B03 D642B3         1891    $IF @CHAT .EQ. COMMA$ THEN Only check if we have a ","
4B06 4B2E
4B08 0F1B           1892      XML  PGMCH            Check next token for REC
4B0A D642DE         1893      $IF @CHAT .NE. REC$ GOTO ERR$1 Error off if no REC
4B0D 40F5
4B0F DAE005         1894      $IF .BIT0 RAM(FLG(PABPTR)) .NE. 1 GOTO ERR$3
4B12 040177
4B15 DE
4B16 0F1B           1895      XML  PGMCH            Get first character of express
4B18 0649D0         1896      CALL OUTEOF           Output possible pending output
4B1B 86E003         1897      CLR  RAM(OFS(PABPTR))  Clear record offset
4B1E 04
4B1F 0EB5           1898      PARSE COLON$          Translate the expression in RE
4B21 06499C         1899      CALL CHKCNV           Check numeric and convert to
4B24 DA4A80         1900      $IF .BIT7 @FAC .EQ. 1 GOTO ERR$4  2 byte integer
4B27 4D7C
4B29 BDE00A         1901      DST  @FAC,RAM(RNM(PABPTR)) Store actual record numb
4B2C 044A
                    1902    $END IF
4B2E 00             1903    RTN
                    1904    $
                    1905    $     NXTCHR - Get next program character - skip
                    1906    $            all strings, numerics and line refs...
                    1907    $
                    1908  NXTCHR
4B2F 8E4269         1909    $IF @CHAT .EQ. 0 GOTO RTC   Stop on EOS
4B32 CC
4B33 D642C7         1910    $IF @CHAT .EQ. STRIN$ GOTO NXTC$0 Skip all strings
4B36 6B3D
4B38 D642C8         1911    $IF @CHAT .EQ. NUM$ THEN   and numerics/unquoted stri
4B3B 4B49
                    1912  NXTC$0
4B3D 0F1B           1913      XML  PGMCH            Get string length
4B3F BC4B42         1914      ST   @CHAT,@FAC+1     Make that a double please....
4B42 864A           1915      CLR  @FAC             Hic.....Oops, sorry
```

```
4B44 A12C4A  1916      DADD @FAC,@VAR81      Back to the serious stuff
4B47 4B50    1917    $SELSE
 ?49 D642C9  1918      $IF @CHAT .EQ. LN$ THEN Line # = skip 2 tokens
 .84C 4B50
4B4E 952C    1919        DINCT @VAR81        <----- That's the skip
             1920      $END IF
             1921    $SEND IF
4B50 0F1B    1922    XML  PGMCH              Get the next token
4B52 00      1923    RTN
```

```
                    1925  *
                    1926  *       P R I N T   U T I L I T Y   R O U T I N E S
                    1927  *
                    1928  *       Copy (converted) numerical datum in string
                    1929  *
                    1930  RSTRING
4B53 BC0D56         1931    ST    @FAC+12,@BYTE+1    Get actual string length
4B56 860C           1932    CLR   @BYTE              Create double byte value
4B58 06490F         1933    CALL CTSTR               Create a temporary string
4B5B 340CB0         1934    MOVE @BYTE FROM *FAC+11 TO RAM(@SREF) Copy value strin
4B5E 1C9055
4B61 00             1935    RTN
                    1936    $
                    1937    $     COMMOD - Compute FAC module FAC+2
                    1938    $
                    1939  COMMOD
4B62 AC4A4C         1940    DIV   @FAC+2,@FAC        Compute remainder
4B65 8E4B4B         1941    $IF @FAC+1 .EQ. 0 THEN Avoid zero remainders
4B68 6C
4B69 BC4B4C         1942      ST    @FAC+2,@FAC+1    Assume maximum remainder
                    1943    $END IF
4B6C 864A           1944    CLR   @FAC               Clear upper byte
4B6E 00             1945    RTN
                    1946    $
                    1947    $     TSTSEP tests for separator in print and
                    1948    $     branches to the correct evaluation routine.
                    1949    $     if no separator is found, simple return.
                    1950    $
                    1951  TSTSEP
                    1952    $     Test case end of line
4B6F 8E424B         1953    $IF @CHAT .EQ. 0 THEN
4B72 78
4B73 BF9073         1954      DST   #EOLEX,*SUBSTK  Replace returnaddress with EOL
4B76 4325
                    1955    $END IF
4B78 CA42B3         1956    $IF @CHAT .L. COMMA$ GOTO TSTS$1
4B7B 4BA0
4B7D C642B5         1957    $IF @CHAT .H. COLON$ GOTO TSTS$1
4B80 6BA0
4B82 BF9073         1958    DST   #PRSEM,*SUBSTK     Expect it to be a ";"
4B85 431D
4B87 06433A         1959    CALL TSTINT              Test for INTERNAL files
4B8A 4BA0           1960    BR    TSTS$1             Treat all separators as ";"
4B8C D642B3         1961    $IF @CHAT .EQ. COMMA$ THEN
4B8F 4B96
4B91 BF9073         1962      DST   #PRTCOM,*SUBSTK
4B94 42FF
                    1963    $END IF
4B96 D642B5         1964    $IF @CHAT .EQ. COLON$ THEN
4B99 4BA0
4B9B BF9073         1965      DST   #PRCOL,*SUBSTK
4B9E 431A
                    1966    $END IF
                    1967  TSTS$1
4BA0 00             1968    RTN
```

```
                    1969  *
                    1970  *       PARFN - Parse string expression and create PAB
                    1971  *       automatically continue in CSTRIN for copy
                    1972  *       string to PAB
                    1973  *            Exit on non-string values
                    1974  *
                    1975  PARFN
                    1976    $
                    1977    $     First evaluate string expression
                    1978    $
4BA1 0EB3           1979    PARSE COMMA$                Parse up to next comma
4BA3 D64C65         1980    $IF @FAC+2 .NE. STRVAL GOTO ERR#X Check for "STRING"
4BA6 4D81
4BA8 BD0250         1981    DST  @FAC+6,@MNUM           Copy length byte in MNUM
4BAB A2030E         1982    ADD  PABLEN,@MNUM+1         Account for PAB length + control
4BAE 0F17           1983    XML  VPUSH                  Save start of string somewhere
4BB0 BD4A02         1984    DST  @MNUM,@FAC             Setup for MEMCHK - check for memor
4BB3 062844         1985    CALL MEMCHK                                    overflow
4BB6 0F18           1986    XML  VPOP                   Restore all FAC information again
4BB8 A54002         1987    DSUB @MNUM,@FREWRD          Update free word pointer
4BBB BD0440         1988    DST  @FREWRD,@PABPTR        Assign PAB entry address
4BBE 9104           1989    DINC @PABPTR                Correct for byte within PAB
4BC0 868004         1990    CLR  RAM(@PABPTR)           Clear PAB plus control info
4BC3 35000D         1991    MOVE PABLEN-1 FROM RAM(@PABPTR) TO RAM(1(PABPTR)) Ripple
4BC6 E00104
4BC9 B004
4BCB BCE003         1992    ST   @MNUM+1,RAM(OFS(PABPTR)) Save length of PAB
 ?CE 0403
 ?D0 BC0251         1993    ST   @FAC+7,@MNUM           Compute # of bytes in name
4BD3 BCE00D         1994    ST   @FAC+7,RAM(NLEN(PABPTR)) Store name length
4BD6 0451
4BD8 BCE002         1995    ST   @FNUM,RAM(FIL(PABPTR)) Copy file number in PAB
4BDB 0417
4BDD BD0804         1996    DST  @PABPTR,@CCPADR        Get start address for string dest.
4BE0 A30800         1997    DADD NLEN+1,@CCPADR         Add offset to actual start addres
4BE3 0E
                    1998    $
                    1999    $     TRICKY - OPTFLG also resets offset added in CSTRIN
                    2000    $
4BE4 9617           2001    CLR  @OPTFLG                Clear all option flags
                    2002  CSTRIN
4BE6 8E026B         2003    $WHILE @MNUM .NE. 0 THEN Watch out for empty names
4BE9 FB
4BEA BC8008         2004      ST   @DSRFLG,RAM(@CCPADR) Store offset
4BED 17
4BEE A0B008         2005      ADD  RAM(@FAC+4),RAM(@CCPADR) Add in character
4BF1 B04E
4BF3 914E           2006      DINC @FAC+4               Get next address for input
4BF5 9108           2007      DINC @CCPADR                and for output
4BF7 9202           2008      DEC  @MNUM                Decrement counter
4BF9 4BE6           2009    $SEND WHILE
4BFB 00             2010    RTN
```

```
                    2012  *
                    2013  *       OUTREC and INITRC are used to output a record to
                    2014  *       either screen or external I/O devices, and to
                    2015  *       initiate pointers for further I/O
                    2016  *
                    2017  OUTREC
                    2018    $     Sort out INTERNAL or EXTERNAL I/O (O)
 4BFC 8E176C        2019    $IF @DSRFLG .NE. 0 THEN INTERNAL I/O
 4BFF 08
 4C00 064D00        2020      CALL SCROL              Just scroll up one line
 4C03 BE0601        2021      ST   1,@CCPPTR          Reset record pointer
 4C06 4CAF          2022      BR   INTKBO             And re-initialize KB
                    2023    $END IF                   Else do EXTERNAL I/O stuff
                    2024    $  This is also entry for last record output
 4C08 DAE005        2025    $IF .BIT4 RAM(FLG(PABPTR)) .EQ. 0 THEN FIXED records
 4C0B 04104C
 4C0E 17
 4C0F BC0307        2026      ST   @RECLEN,@MNUM+1    Ready for space filling
 4C12 9003          2027      INC  @MNUM+1            Move to first position outside
 4C14 064C43        2028      CALL FILSPC             And do it up to end of record
                    2029    $END IF
 4C17 9206          2030    DEC  @CCPPTR              Update last character position
 4C19 BCE009        2031    ST   @CCPPTR,RAM(CNT(PABPTR)) Store # of characters
 4C1C 0406
 4C1E 86E003        2032    CLR  RAM(OFS(PABPTR))     Undo pending record offsets
 4C21 04
 4C22 064CB9        2033    CALL IOCALL               Call DSR
 4C25 03            2034    DATA C$WRIT                     for WRITE mode
                    2035    $
                    2036  INITRC
 4C26 8609          2037    CLR  @CCPADR+1            Get address at bufferstart
 4C28 4C36          2038    BR   PR$$O
                    2039  *
                    2040  *       PRINIT initializes the variables CCPADR, CCPPTR
                    2041  *       RECLEN and DSRFLG, for a given PABPTR.
                    2042  *
                    2043  PRINIT
 4C2A 8617          2044    CLR  @DSRFLG              Indicate external I/O in DSRFL
 4C2C BC07E0        2045    ST   RAM(LEN(PABPTR)),@RECLEN  Pick up record length
 4C2F 0804
 4C31 BC09E0        2046    ST   RAM(OFS(PABPTR)),@CCPADR+1 Get offset in record
 4C34 0304
                    2047  PR$$O
 4C36 BC0609        2048    ST   @CCPADR+1,@CCPPTR Compute columnar position
 4C39 9006          2049    INC  @CCPPTR              And convert from offset
 4C3B 8608          2050    CLR  @CCPADR              Clear upper byte
 4C3D A108E0        2051    DADD RAM(BUF(PABPTR)),@CCPADR Compute actual address
 4C40 0604
 4C42 00            2052    RTN
```

```
                2054  *
                2055  *     FILSPC - Fill a record with spaces, starting from
                2056  *     the current position at CCPADR to the position
                2057  *     indicated in @MNUM+1 (1 byte)
                2058  *
                2059  FILSPC
4C43 8E034C     2060    $IF @MNUM+1 .EQ. 0 THEN In case record length = 255
4C46 4D
                2061  *                              (255+1, overflow to be 0)
4C47 8E064C     2062      $IF @CCPPTR .NE. 0 GOTO FILS$2
4C4A 52
                2063  *
4C4B 4C6D       2064      BR FILS$3                 Just RTN (Unconditional branch)
                2065    $END IF
4C4D C40306     2066    $IF @MNUM+1 .H. @CCPPTR THEN Make sure difference >=1
4C50 4C6D
                2067  FILS$2
4C52 A40306     2068      SUB  @CCPPTR,@MNUM+1 Compute the # of bytes to move
4C55 A00603     2069      ADD  @MNUM+1,@CCPPTR Actually move pointer ahead
4C58 BC0217     2070      ST   @DSRFLG,@MNUM   Assume zero filling
4C5B 06433A     2071      CALL TSTINT          Which would be correct for
4C5E 4C63       2072      BR   FILS$1            INTERNAL type files
4C60 A20220     2073      ADD  SPACE,@MNUM     Make that space filling
                2074  FILS$1
4C63 BCB008     2075      ST   @MNUM,RAM(@CCPADR) Fill with fillers
4C66 02
4C67 9108       2076      DINC @CCPADR         Next address to be treated
4C69 9203       2077      DEC  @MNUM+1         Count that space too
4C6B 4C63       2078      BR   FILS$1         One space is enough !!
                2079    $END IF
                2080  FILS$3
4C6D 00         2081    RTN                    Return after completing move
                2082  *
                2083  *     OSTRNG - Copy the value of the string expression
                2084  *     to the screen
                2085  *
                2086  OSTRNG
4C6E BC0C51     2087    ST   @FAC+7,@BYTE      Pick up the string length
4C71 8E0C6C     2088    $WHILE @BYTE .NE. 0    Output as many records as require
4C74 9A
                2089      $
                2090      $  CHKREC check available space in current record.
                2091      $  If the string to be output is too long, it
                2092      $  is chunked up into digestable pieces.  If the
                2093      $  current record is partly filled up, it is
                2094      $  output before any chunking is done.
                2095      $
                2096  CHKREC
4C75 BC0207     2097      ST   @RECLEN,@MNUM1   Compute remaining area
4C78 A40206     2098      SUB  @CCPPTR,@MNUM1    between column and end
4C7B 9002       2099      INC  @MNUM           Also count current column
4C7D C8020C     2100      $IF @MNUM .L. @BYTE THEN Won't fit in current record
4C80 6C8C
  32 D60601     2101        $IF @CCPPTR .EQ. 1 GOTO CHKR$1 Unused record
4C85 6C8F
```

```
4C87 064BFC  2102          CALL OUTREC          Output whatever we have
4C8A 4C75    2103          BR   CHKREC         And try again
             2104        $END IF
4C8C BC020C  2105        ST   @BYTE,@MNUM       Use actual count if fit
             2106    CHKR$1
4C8F A4CC02  2107        SUB  @MNUM,@BYTE       Update remaining chars count
4C92 A00602  2108        ADD  @MNUM,@CCPPTR     Also new column pointer
4C95 064BE6  2109        CALL CSTRIN            COPY STRING TO OUTPUT
4C98 4C71    2110      $END WHILE               CONTINUE AS LONG AS NEEDED
4C9A 00      2111      RTN
             2112    *
             2113    *       INITKB - Initialize the variables needed
             2114    *       for keyboard output
             2115    *
             2116    INITKB
4C9B 8604    2117      CLR  @PABPTR           Don't use any DISPLAY options
4C9D BE1760  2118      ST   OFFSET,@DSRFLG    Load for correction of screen c
4CA0 BE0601  2119      ST   1,@CCPPTR         Assume un-initialized XPT
4CA3 8E7F6C  2120      $IF XPT .NE. 0 THEN    *** Patch for un-initialized XP
4CA6 AC
4CA7 BC067F  2121        ST   XPT,@CCPPTR     Initialize CCPPTR
4CAA 9606    2122        DECT @CCPPTR         Correct for incorrect XPT offse
             2123      $END IF
4CAC BE071C  2124      ST   VWIDTH,@RECLEN    Get video screen width
             2125    INTKB0
4CAF BC0906  2126      ST   @CCPPTR,@CCPADR+1 Initialize screen address
4CB2 8608    2127      CLR  @CCPADR           Clear upper byte CCPADR
4CB4 A30802  2128      DADD SCRNBS+1,@CCPADR  Add start-address plus compens
4CB7 E1
4CB8 00      2129      RTN
             2130    *
             2131    IOCALL
4CB9 8856    2132      FETCH @FAC+12          I/O code to FAC+12 (BUG!!!!)
4CBB BCE004  2133      ST   @FAC+12,RAM(COD(PABPTR)) Pick up the I/O code
4CBE 0456
             2134    IOCL$1
4CC0 064CC6  2135      CALL CDSR              Call the DSR routine
4CC3 57C3    2136      BR   ERR$2             Give I/O error on error
4CC5 00      2137      RTN                    Or else return
             2138    *
             2139    *       DSR CALL ROUTINE - NORMAL ENTRY
             2140    *
             2141    CDSR
4CC6 BEE00C  2142      ST   OFFSET,RAM(SCR(PABPTR)) Always set screen offset
4CC9 0460
4CCB 35001E  2143      MOVE 30 FROM @FAC TO RAM(>3C0) Save FAC area
4CCE A3C04A
4CD1 BD5604  2144      DST  @PABPTR,@FAC+12   Get PAB pointer in FAC
4CD4 A35600  2145      DADD NLEN,@FAC+12      Compute name length entry
4CD7 0D
4CD8 060010  2146      CALL CALDSR            Call actual DSR link routine
4CDB 08      2147      DATA 8
4CDC 35001E  2148      MOVE 30 FROM RAM(>3C0) TO @FAC
4CDF 4AA3C0
             2149    *           MOVE does not affect status
```

```
- 4CE2 6CE9         BS   CDSR$0              ERROR = ERROR = ERROR = .....
  4CE4 DAE005       CLOG >E0,RAM(FLG(PABPTR)) Set COND if no error
   'E7 04E0
                   CDSR$0
  4CE9 01            RTNC
                   *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                   *      THE FOLLOWING SECTION HAS BEEN ADDED
                   *      BY CAROLYN LEE 2/25/81 TO FIX THE PROBLEM WITH A
                   *      LONG CONSTANT IN A VARIABLE FIELD IN INPUT, ACCEPT,
                   *      READ ETC. STATEMENT FOR 99/4A
                   *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                   BUG01
  4CEA CA4280           $IF @CHAT .HE. >80 GOTO ERR$1 'Make sure of var. name
  4CED 60F5
  4CEF 0F13             XML  SYM             Get the information of the variabl
  4CF1 0F14             XML  SMB
  4CF3 00               RTN
                   ***************************************************************
```

```
                **** Split FLMGR to avoid moving EXEC for 99/4A
                *       GROM 2
                        ORG  >17C0
                *
                *                 E R R O R   M E S S A G E S
                *
                ERR$2B
57C0 0640E7       CALL CLRFRE                Undo allocation of PAB
                ERR$2
                * <><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *
                *       Next code is to avoid recursion of errors in
                *       CLSALL routine.  If this entry is taken from
                *       CLSALL, the stack will contain CLSLBL as a
                *       returnaddress in the third level.    8/30/79 - HS
                *
                * <><><><><><><><><><><><><><><><><><><><><><><><><><><><>
57C3 A67304       SUB  4,@SUBSTK             Back down two levels
57C6 D79073       $IF *SUBSTK .DEQ. CLSLBL THEN
57C9 41C457
57CC D3
57CD 06284C         CALL WARN$$              Give warning to the user
57D0 2113           DATA #MSG21              * I/O ERROR
57D2 00             RTN                      And return to close routine
                  $END IF
57D3 A27304       ADD  4,@SUBSTK             Back up two levels for OLD/SAVE
                ERR$2A
57D6 06284E       CALL ERR$$
57D9 2113         DATA #MSG21                * I/O ERROR
57DB 05201A       B    MAIN1
                *
                ERR$3
57DE 06284E       CALL ERR$$
57E1 211D         DATA #MSG22                * FILE ERROR
                *
                ERR$6
57E3 06284E       CALL ERR$$
57E6 2128         DATA #MSG23                * INPUT ERROR
                *
                ERR$7
57E8 06284E       CALL ERR$$                 GIVE ERROR MESSAGE
57EB 2134         DATA #MSG24                "* DATA ERROR"
                        END
```

ERRORS=   0

LENGTH= 3361 (>0D21)

372 SYMBOLS USED

SYMBOL	VALUE	DEF	REFERENCE TABLE
ABS$	00CB	180	
APPEN$	00F9	214	
ARG	005C	36	631 635 784 785 788 789 790 791 972 974 975 976 977 978 979 980 1516 1518 1519 1520 1520 1523 1524 1525 1527 1528 1528 1529 1529 1531 1533
ASSGNV	0015	248	987 1199 1259
ATN$	00CC	181	
AUTO1	2834	268	1414
BASE	0043	33	1841
BASE$	00F1	206	
BEE	001E	98	1119 1130 1134 1203
BFF	0012	91	
BREAK	0002	225	1482
BREAK$	008E	118	
BUF	0006	59	580 583 945 1033 1061 1209 1293 1387 1471 1775 1817 1820 2051
BUFFY	0044	34	1438 1441
BUG01	4CEA	2160	956 1187 1239
BYTE	000C	75	959 961 966 969 970 983 986 1196 1627 1641 1931 1932 1934 2087 2088 2100 2105 2107
C$CLOS	0001	277	619 653 1490
C$DELE	0007	283	600 623
C$LOAD	0005	281	1294
C$OPEN	0000	276	
C$READ	0002	278	738 930 1004
C$REST	0004	280	684
C$SAVE	0006	282	1392
C$SCR	0008	284	
C$STAT	0009	285	1524
C$WRIT	0003	279	741 1759 2034
CALDSR	0010	256	2146
CALL$	009D	133	
CB	0008	1	
CCPADR	0008	70	793 794 811 812 813 1085 1086 1093 1095 1096 1097 1140 1213 1215 1457 1461 1463 1470 1471 1777 1778 1779 1780 1783 1795 1806 1808 1810 1813 1816 1817 1820 1825 1830 1862 1869 1996 1997 2004 2005 2007 2037 2046 2048 2050 2051 2075 2076 2126 2127 2128
CCPPTR	0006	68	789 795 810 814 832 835 849 888 889 892 1124 1126 1473 1806 1824 1861 2021 2030 2031 2048 2049 2062 2066 2069 2069 2098 2101 2108 2119 2121 2122 2126
CDSR	4CC6	2141	555 626 654 1295 2135
CDSR$0	4CE9	2152	2150
CHAR	0004	1	
CHAT	0042	32	353 363 372 373 374 375 376 424 425 482 483 620 622 675 691 726 745 746 755 827 876 920 932 989 991 1045 1073 1082 1088 1178 1201 1272 1449 1458 1461 1511 1556 1557 1568 1595 1600 1604 1605 1613 1614 1616 1710 1891 1893 1909 1910 1911 1914 1918 1953 1956 1957 1961 1964 2161
CHECK	40FD	541	363 425
CHKCNV	499C	1718	434 1513 1877 1899

```
SYMBOL VALUE  DEF    REFERENCE TABLE

 4KCON 49B1  1727      348  616  678  730  923
 AKF$1 49BB  1739     1746
CHKFN  4993  1709      346  614  676  728  921
CHKMSG 46E4  1354     1350
CHKNUM 493B  1647     1127 1192
CHKPAR 408D   432      426  484
CHKR$1 4C8F  2106     2101
CHKREC 4C75  2096     2103
CHKRM  45D3  1212     1084 1092
CHKS$0 4978  1683     1246
CHKS   42CE   817      835  838
CHKSTR 496C  1678     1143 1195 1255
CHR$$  00D6   191
CIRCU$ 00C5   173
CLOS$1 41A2   630      627
CLOSE  4174   613      308
CLOSE$ 00A0   136
CLRFRE 40E7   526      533 2174
CLSA$0 41BC   649      658
CLSALL 41CF   656      312 1450 1559
CLSLBL 41C4   652     2185
CNS    0014   257      803
CNT    0009    61      960  992  995 1013 1031 2031
CNVD$0 4B02  1880     1878
CNVDEF 4AF9  1876      829
 OD    0004    57      619  623  653  738  741  930 1004 1294 1525 1528
                     1759 2133
COLON$ 00B5   157      353  620  746  771  932 1073 1076 1082 1088 1712
                     1898 1957 1964
COMMA  002C   227
COMMA$ 00B3   155      363  424  433  482  989 1009 1045 1147 1270 1272
                     1614 1694 1891 1956 1961 1979
COMMOD 4B62  1939      831
CONVER 4D16   250     1252 1653
COS$   00CD   182
CRNBUF 0320   233     1002 1105 1120 1172
CSTRIN 4BE6  2002     2109
CTMPST 492D  1638      967 1197
CTSTR  490F  1624     1639 1933
CTSTR0 4913  1626
DATA   0034    25     1119 1120 1130 1134 1172 1203 1242 1263 1574 1650
                     1652 1657 1668 1671 1673 1687 1688 1697
DATA$  0093   123
DATAST 4D08   255      712 1267
D F$   0089   113
D L T  4160   593      304
DELET$ 0099   129      622
DELP$1 4A80  1827     1864
DELP$2 4A8C  1830     1828
DELPAB 49E6  1771      628  632  655
 FLAG  0001    82      510  511
 .M$   008A   114
DISPL$ 00A2   138      372
DISPL1 426C   749      303
DISPLA 0009     1
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
DIVI$	00C4	172	
DSRFLG	0017	78	812 887 901 934 1131 1133 1455 1487 1617 1758 2004 2019 2044 2070 2118
ELSE$	0081	105	
EMPSTR	4987	1693	1681
END$	008B	115	
ENLN	0032	24	701 702 1301 1305 1312 1330 1376 1383 1386 1420
EOF	482B	1510	317
EOF$0	483E	1517	1521
EOF$1	484F	1522	1519
EOF$2	487D	1537	1532
EOLEX	4325	881	745 1954
EQUAL$	00BE	166	
ERR	4D88	9	701
ERR$$	284E	251	535 2192 2197 2201 2205
ERR$1	40F5	534	353 622 746 819 827 932 1077 1088 1511 1557 1595 1604 1614 1617 1710 1893 2161
ERR$2	57C3	2175	1352 2136
ERR$2A	57D6	2191	636 1472
ERR$2B	57C0	2173	556
ERR$3	57DE	2196	347 349 615 617 679 731 736 791 825 924 928 969 974 980 1518 1894
ERR$4	4D7C	291	1514 1515 1722 1732 1900
ERR$6	57E3	2200	1036 1133
ERR$7	57E8	2204	708 1242 1245 1253 1256 1270
ERR$X	4D81	290	1719 1980
ERRCOD	0022	16	
EXP$	0076	42	
EXP$$	00CE	183	
EXTRAM	002E	22	1440 1443 1444 1446
FAC	004A	35	427 489 490 566 674 677 707 729 782 783 784 785 791 793 798 802 830 832 835 836 922 965 966 970 971 972 976 977 982 1077 1129 1191 1196 1244 1247 1249 1250 1253 1458 1459 1464 1467 1469 1470 1472 1515 1519 1530 1534 1538 1625 1627 1628 1630 1631 1632 1640 1650 1657 1664 1666 1669 1679 1685 1686 1688 1719 1720 1722 1723 1724 1728 1732 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1842 1843 1843 1844 1845 1851 1852 1852 1853 1854 1854 1856 1857 1879 1900 1901 1914 1915 1916 1931 1934 1940 1940 1941 1942 1942 1944 1980 1981 1984 1993 1994 2005 2006 2087 2132 2133 2143 2144 2145 2148
FIGURE	0002	1	
FIL	0002	55	1519 1744 1995
FILS$1	4C63	2074	2072 2078
FILS$2	4C52	2067	2062
FILS$3	4C6D	2080	2064
FILSPC	4C43	2059	837 857 2028
FIXED$	00FA	215	
FLAG	0088	43	1394 1555
FLG	0005	58	404 444 463 471 502 520 548 549 552 735 736 902 928 938 1894 2025 2151
FLOAT1	4880	1541	1530
FNUM	0017	76	77 78 79 594 1728 1744 1995

SYMBOL	VALUE	DEF	REFERENCE TABLE
R$	008C	116	
REWRD	0040	31	529 582 598 633 1377 1805 1869 1987 1988
GETDAT	4956	1665	311 1684
GETGFL	4952	1663	1243 1260
GETRAM	495A	1667	1125 1146 1190 1200 1649
GETSTR	4D12	273	1629
GETV$0	48D4	1587	1616
GETV$1	48DD	1599	1609
GETVAR	48CC	1580	1001 1090
GO$	0085	109	
GOSUB$	0087	111	
GOTO$	0086	110	
GPNAME	4888	1554	1286 1367
GREAT$	00C0	168	
GROMFL	0089	44	1664 1828
IF$	0084	108	
INITKB	4C9B	2116	727 750 919 1067 1476
INITRC	4C26	2036	
INP$31	4449	1019	1005
INP$32	4484	1049	1060
INP$37	44D6	1080	1072
INP$39	44E5	1087	1079
INP$67	45B9	1198	1193
INPU$2	44DC	1083	922
INPU$3	44EF	1091	1141
INPU$4	4537	1122	1153
INPU$5	4558	1132	1128 1144 1149 1150
INPU$6	4576	1145	1129
INPU$7	45C2	1202	1178
INPUT	4344	918	306
INPUT$	0092	122	373
INT$	00CF	184	
INTER$	00F5	210	
INTKB0	4CAF	2125	2022
INTR$0	4380	940	992
INTR$1	4392	947	993
IOCALL	4CB9	2131	599 683 1391 1489 2033
IOCL$1	4CC0	2134	941 1010 1466 1526
IOSTRT	C03C	28	570 571 573 657 658 1516 1738 1781 1782 1793 1794 1795 1796 1798
KILSYM	215A	12	1320 1351 1562
LEN	0008	60	490 562 563 1468 2045
LEN$	00D5	190	
LESS$	00BF	167	
LET$	00 D	117	
LINE	4D06	254	693
LIST	474C	1410	315
LIST$0	4764	1422	1433
LIST$1	47C8	1460	1465
LIST$3	4804	1483	1481
LIST$4	480D	1486	1482
LITS05	4D02	272	1248 1257
LLIST	282E	266	1479
LN$	00C9	178	1918
LNBUF	0036	26	702 703 707 708 709 711 1262 1265 1266

SYMBOL	VALUE	DEF	REFERENCE TABLE
LOG$	00D0	185	
LPAR$	0087	159	827 1511 1600
LRMA0	4745	1393	1327 1330
M$EDIT	2010	4	6 12 249 270 271
M$EXEC	4D00	3	9 250 254 255 265 272 273 290 291
M$MSG	2022	6	289 292 293 294 295 296 297
M$PSCN	2828	5	251 252 253 266 267 268 269
MA0	2012	249	537 1396 1493
MAIN1	201A	271	1491 2194
MEMCHK	2844	253	581 1985
MFLAG	0000	81	454 455
MINUS	002D	228	1413
MINUS$	00C2	170	
MNUM	0002	52	527 528 529 563 564 566 582 583 596 597 598 836 849 850 851 852 853 854 855 856 1121 1123 1124 1298 1299 1300 1303 1305 1306 1308 1309 1310 1318 1325 1326 1332 1782 1783 1784 1784 1786 1787 1788 1789 1791 1804 1805 1807 1811 1981 1982 1984 1987 1992 1993 2003 2009 2026 2027 2060 2066 2068 2069 2070 2073 2075 2077 2097 2098 2099 2100 2105 2107 2108
MSG1	202C	289	536
MSG17	20BD	292	
MSG20	2108	293	1139
MSG21	2113	294	2187 2193
MSG22	211D	295	2198
MSG23	2128	296	1137 2202
MSG24	2134	297	2206
MULT$	00C3	171	
NBC	0014	93	1411 1419 1420 1435 1436 1438 1440 1484 1485
NBD	001E	97	1412 1421 1423 1429 1435 1436 1441 1443 1446 1485
NEXT$	0096	126	
NLEN	000D	64	1288 1456 1457 1568 1569 1570 1778 1994 1997 2145
NUDTAB	0028	19	
NUM$	00C8	177	1245 1557 1648 1681 1911
NUMBE$	00FD	218	675 726 920 1710
NXTC$0	4B3D	1912	1910
NXTCHR	4B2F	1908	1071
OFFSET	0060	243	1015 1050 1053 1058 1085 1095 1432 1433 1498 2119 2142
OFS	0003	56	528 565 597 739 889 939 943 988 996 1005 1011 1044 1062 1207 1760 1897 1992 2032 2046
OLD	4641	1285	314
OLD$1	46A9	1314	1319
OLD$2	46C7	1329	1335
OLDC$0	4202	695	677
OLDCD	41F9	690	675
OLDER	46DA	1339	1296 1300
ON$	009B	131	
OPEN	401E	345	307
OPEN$	009F	135	
OPERR	40F2	532	375 393 418 435 454 489 493 510 549 562
OPT$0	4061	392	
OPT$01	40AB	470	377
OPT$02	406B	403	378

```
YMBOL VALUE  DEF    REFERENCE TABLE

PT$03 40D1   501      379
PT$1  4070   417      380
PT$2  4095   443      381
PT$3  409A   450      382  464  521
PT$4  40A4   462      383
PT$5  40B0   480      384
PT$6  40D6   509      372
PT$7  40E0   519      373
PTFLG 0017    79      393  394  418  419  454  455  493  494  510  511
                      551 2001
PTI$0 4036   362      428  495
PTI$1 403B   367      424
PTIO$ 009E   134
PTION 4034   357      395  456  512
STRNG 4C6E  2086      800  805 1078
THE$0 42AD   799      796
THE$1 42A8   797      781
THER  4277   770
UTEOF 49D0  1757      618  651  680  929 1896
UTPU$ 00F7   212
UTREC 4BFC  2017      316  833  865  882 1488 1762 2102
AB    481E  1497     1456
ABLEN 000E   232     1289 1565 1779 1982 1991
ABPTR 0004    53      404  427  444  463  471  490  502  520  528  548
                      549  552  557  562  563  565  571  578  580  580
                      583  597  619  623  631  633  634  635  648  648
                      650  653  657  681  735  736  738  739  741  889
                      902  928  930  938  939  943  945  960  988  992
                      995  996 1004 1005 1011 1013 1031 1033 1044 1061
                     1062 1207 1209 1287 1288 1290 1291 1292 1293 1294
                     1311 1315 1317 1387 1389 1390 1454 1456 1468 1471
                     1523 1525 1527 1563 1564 1565 1565 1567 1568 1569
                     1570 1738 1743 1744 1745 1745 1759 1760 1775 1778
                     1780 1781 1783 1786 1791 1793 1798 1814 1815 1816
                     1817 1818 1818 1820 1826 1829 1830 1832 1834 1835
                     1860 1861 1862 1863 1863 1894 1897 1901 1988 1989
                     1990 1991 1991 1992 1994 1995 1996 2025 2031 2032
                     2045 2046 2051 2117 2133 2142 2144 2151
ARCON 4339   894
ARFN  4BA1  1975      355  595
ARREC 4B03  1890      682  743  931
ERMA$ 00FB   216
FLAG  0002    83      393  394
GMCH  001B   262      354  358  368  420  481  621  624  747  826  875
                      933  990 1042 1075 1089 1177 1229 1448 1453 1462
                     1566 160  1615 1711 1892 1895 1913 1922
LUS$  00C1   169
OP    0005     1
R$$0  4C36  2047     2038
RCOL  431A   864     1965
`EXIT 4328   886      877
 INIT 4C2A  2043      742 1761
RINT  4227   725      305
RINT$ 009C   132
RN$10 425F   744      729
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
PRSEM	431D	874	1958
PRTAB	42D3	823	755
PRTCOM	42FF	848	1962
PRTNFN	00CE	46	1099
RAM	0001	1	404 427 444 463 471 490 502 520 528 548 549 552 557 562 563 565 575 576 578 580 583 597 619 623 631 635 648 650 653 681 707 735 736 738 739 741 793 811 812 889 902 928 930 938 939 943 945 960 961 970 988 992 995 996 1004 1005 1009 1011 1013 1015 1031 1033 1044 1050 1053 1058 1061 1062 1085 1095 1098 1124 1126 1139 1207 1209 1288 1290 1291 1292 1293 1294 1298 1299 1300 1301 1302 1303 1315 1315 1325 1332 1333 1374 1379 1381 1383 1385 1386 1387 1389 1390 1420 1423 1432 1433 1443 1456 1461 1468 1471 1519 1520 1525 1528 1529 1564 1565 1565 1568 1569 1570 1570 1628 1630 1641 1641 1668 1744 1745 1759 1760 1775 1778 1783 1784 1786 1786 1787 1788 1789 1791 1793 1808 1808 1815 1816 1817 1818 1820 1829 1830 1832 1834 1842 1852 1854 1860 1861 1862 1863 1894 1897 1901 1934 1990 1991 1991 1992 1994 1995 2004 2005 2005 2025 2031 2032 2045 2046 2051 2075 2133 2142 2143 2148 2151
RAMPTR	0038	27	1002 1006 1008 1009 1105
RANDO$	0095	125	
READ	45E3	1230	310
READ$	0097	127	
READLN	2832	267	1103
REC$	00DE	199	1893
RECENT	45C6	1206	1012 1030 1043
RECLEN	0007	69	78 810 830 855 1003 1039 1040 1046 1047 1048 1468 1469 2026 2045 2097 2124
RELAT$	00F4	209	
REM$	009A	130	
RESTO$	0094	124	
RESTOR	41D7	673	309
RETUR$	0088	112	
RFLAG	0004	85	493 494 551
RKEY	0075	40	1482
RND$	00D7	192	
RNM	000A	62	427 557 681 1290 1291 1292 1389 1390 1901
ROM	000A	1	1139 1456 1530 1671
RPAR$	00B6	158	828 1605
RSTK	0088	45	
RSTRIN	4B53	1930	786 804
RTC	49CC	1747	901 1695 1723 1744 1909
RTNG	0026	18	
SAVE	46FC	1366	313
SAVEPC	03EC	240	
SCDATA	2014	270	1035 1106
SCR	000C	63	1529 2142
SCRNBS	02E0	244	1139 1140 1213 1215 2128
SCROL	4D00	265	1108 1214 2020
SEETWO	283E	269	1439 1442
SEMIC$	00B4	156	

SYMBOL	VALUE	DEF	REFERENCE TABLE
EQUE$	00F6	211	
FLAG	0003	84	418 419
SGN$$	00D1	186	
SIGN$	0075	41	
SIN$	00D2	187	
SMB	0014	264	2163
SPACE	0020	226	811 2073
SPRITE	0006	1	
SQR$	00D3	188	
SREF	001C	14	1249 1250 1251 1253 1631 1632 1641 1934
STADDR	000A	73	573 575 576 576 578 1068 1069 1074 1081 1173 1287 1288 1289 1291 1293 1298 1299 1300 1301 1302 1303 1304 1307 1315 1316 1324 1325 1328 1330 1331 1332 1333 1334 1371 1372 1374 1375 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1390 1569 1570 1581 1775 1776 1789 1796 1804 1808 1809 1813
STEP$	00B2	154	
STLN	0030	23	701 708 1265 1302 1306 1307 1309 1313 1328 1371 1381 1385 1423 1472 1829
STOP$	0098	128	
STREND	001A	96	
STRIN$	00C7	175	1556 1680 1910
STRSP	0018	95	
STRVAL	0065	241	782 798 1077 1191 1244 1719 1980
TVSPT	0024	17	1452 1561
JB$	00A1	137	
SUBSTK	0073	39	1954 1958 1962 1965 2184 2185 2190
SYM	0013	263	1596 2162
SYMPTR	0040	30	31
SYMTAB	003E	29	1823 1824 1825 1826
TAB$	00FC	217	755
TABLE	000B	1	
TAN$	00D4	189	
TEMP	0017	77	
TEMP5	0066	37	945 946 961 962 970 985 1641 1687
THEN$	00B0	152	
TO$	00B1	153	
TONE1	0034	258	1100
TONE2	0036	259	
TOP	0003	1	
TRACE$	0090	120	
TSTINT	433A	900	780 824 1959 2071
TSTS$1	4BA0	1967	1956 1957 1960
TSTSEP	4B6F	1951	754 818
UNBRE$	008F	119	
UNQST$	00C8	176	177
UNTRA$	0091	121	
UPDAT$	00F8	213	
VAR0	0000	50	1147 1150 1245 1261 1270 1648 1651 1652 1668 1671 1680 1681 1686 1694 1695
R1	000D	87	
VAR2	000B	74	
VAR4	000E	88	1102 1104 1121 1175 1583
VAR5	0010	89	1040 1046 1047 1065 1111 1582 1612

SYMBOL	VALUE	DEF	REFERENCE TABLE
VAR6	0011	90	692 1038 1039 1048 1059 1065 1110 1111 1148 1152 1598 1601 1603 1606
VAR8	0013	92	
VAR81	002C	21	356 1041 1068 1074 1081 1173 1174 1447 1570 1581 1916 1919
VAR9	0016	94	
VARA	002A	20	943 944 946 960 963 986 992 995 996 1013 1014 1017 1031 1032 1033 1034
VARC	0008	71	1413
VARD	0009	72	
VARIA$	00F3	208	374
VARV	0001	51	
VARW	0020	15	1015 1016 1050 1052 1053 1054 1057 1058 1061 1062 1093 1207 1208 1209 1431 1432 1433
VARY	0004	66	
VARZ	0005	67	
VDP	000C	1	
VEL	0007	1	
VPOP	0018	261	1986
VPUSH	0017	260	958 1189 1241 1611 1983
VRAMVS	06F8	238	1451 1454 1457 1560 1563
VSPTR	006E	38	1102 1104 1175 1451 1452 1560 1561 1563
VWIDTH	001C	242	1475 2124
WARN$$	284C	252	1136 1349 2186
WR$$5	4564	1138	1131
WRN$5	455F	1135	1107 1111
XPT	000E	1	892 893 1109 1475 2120 2121
YPT	000D	1	

TOTAL NUMBER OF SYMBOLS = 372
TOTAL NUMBER OF REFERENCES = 2077

```
                1         TITLE EXEC
                2 ********************************************************
                3 *       SEE DUMMY SPACE BEFORE PGMCTR
                4 ********************************************************
2828            5 M$PSCN  EQU   >2828          Module PSCAN branch table address
2010            6 M$EDIT  EQU   >2010          Module EDIT branch table address
4000            7 M$FLMG  EQU   >4000          Module FLMGR branch table address
2022            8 M$MSG   EQU   M$EDIT+>12     Start of message area
                9 ********************************************************
00CE           10 PRTNFN  EQU   >CE
               11 ********************************************************
2836           12 SEARCH  EQU   M$PSCN+>0E     EQUATES FOR ROUTINES FROM OTHERS
2842           13 DISO    EQU   M$PSCN+>1A
283E           14 SEETWO  EQU   M$PSCN+>16
283C           15 GETLN   EQU   M$PSCN+>14
2012           16 MAO     EQU   M$EDIT+>02          RESTART IN NEDIT "MAIN"
282C           17 GETNB   EQU   M$PSCN+>04
4012           18 CLSALL  EQU   M$FLMG+>12
284C           19 WARN$$  EQU   M$PSCN+>24
284E           20 ERR$$   EQU   M$PSCN+>26
201C           21 CHRTAB  EQU   M$EDIT+>0C
201E           22 MVDN    EQU   M$EDIT+>0E
2848           23 ENTER   EQU   M$PSCN+>20
284A           24 ENT09   EQU   M$PSCN+>22
4004           25 PRINT   EQU   M$FLMG+>04
4006           26 INPUT   EQU   M$FLMG+>06
4008           27 OPEN    EQU   M$FLMG+>08
400A           28 CLOSE   EQU   M$FLMG+>0A
400C           29 RESTOR  EQU   M$FLMG+>0C
400E           30 NREAD   EQU   M$FLMG+>0E
4010           31 GETDAT  EQU   M$FLMG+>10
4000           32 DISPL1  EQU   M$FLMG+>00
4002           33 DELET   EQU   M$FLMG+>02
401C           34 EOF     EQU   M$FLMG+>1C
20E9           35 MSGBRK  EQU   M$MSG+>C7
202C           36 MSG1    EQU   M$MSG+>0A
20BD           37 MSG2    EQU   M$MSG+>9B
2040           38 MSG4    EQU   M$MSG+>1E
2049           39 MSG5    EQU   M$MSG+>27
2064           40 MSG7    EQU   M$MSG+>42
206E           41 MSG8    EQU   M$MSG+>4C
207D           42 MSG11   EQU   M$MSG+>5B
2094           43 MSG12   EQU   M$MSG+>72
20A1           44 MSG13   EQU   M$MSG+>7F
20D9           45 MSGBLN  EQU   M$MSG+>B7
               46 ********************************************************
0010           47 CPL     EQU   >10             CALL LINK    FLOATING-POINT PACKAGE
0012           48 RPL     EQU   >12             RETURN LINK
0014           49 CNS     EQU   >14             NUMBER TO STRING
0022           50 GRINT   EQU   >22             GREATEST INTEGER
0024           51 PWR     EQU   >24             EXPONENTIATION
0026           52 SQR     EQU   >26             SQUARE ROOT
0028           53 EXP     EQU   >28             EXPONENTIAL
002A           54 LOG     EQU   >2A             NATURAL LOG
002C           55 COS     EQU   >2C             COSINE
```

```
002E      56  SIN     EQU  >2E          SINE
0030      57  TAN     EQU  >30          TANGENT
0032      58  ATN     EQU  >32          ARCTANGENT
0036      59  TONE2   EQU  >36          BAD BEEP
          60  ********************************************************
0072      61  STACK   EQU  >72          STACK FOR DATA          STATUS BLO
0073      62  SUBSTK  EQU  >73          SUBROUTINE STACK
0074      63  KEYBD   EQU  >74          KEYBOARD SELECTION
0075      64  RKEY    EQU  >75          KEY CODE
0076      65  JOYY    EQU  >76          JOYSTICK Y POSITION
0077      66  JOYX    EQU  >77          JOYSTICK X POSITION
0078      67  RANDOM  EQU  >78          RANDOM NUMBER GENERATOR
0079      68  TIMER   EQU  >79          TIMING REGISTER
          69  ********************************************************
0000      70  VAR0    EQU  >00          TEMPORARY WORKING SPACES IN
0001      71  VARV    EQU  >01          EXECUTION
0002      72  MNUM    EQU  >02          2 BYTES
0004      73  VARY    EQU  >04
0005      74  VARZ    EQU  >05
0006      75  CCPPTR  EQU  >06          2 BYTES
0008      76  VARC    EQU  >08
0009      77  VARD    EQU  >09
000A      78  STADDR  EQU  >0A          2 BYTES
000B      79  VAR2    EQU  >0B
000C      80  BYTE    EQU  >0C
000D      81  VAR1    EQU  >0D
000E      82  VAR4    EQU  >0E
0010      83  VAR5    EQU  >10
0011      84  VAR6    EQU  >11
0012      85  BEE     EQU  >12
0013      86  VAR8    EQU  >13
0014      87  NBC     EQU  >14          2 BYTES
0016      88  VAR9    EQU  >16          2 BYTES
          89  ********************************************************
0018      90  STRSP   EQU  >18              BEGINNING OF STRING SPACE
001A      91  STREND  EQU  >1A              END OF STRING SPACE
001C      92  SREF    EQU  >1C              TEMPORARY STRING POINTER
001E      93  NBD     EQU  >1E              UNUSED
0020      94  VARW    EQU  >20
0022      95  ERRCOD  EQU  >22              PERMANENT WORKING SPACES IN
0024      96  STVSPT  EQU  >24              EXECUTION (2 BYTES FOR EACH
0026      97  RTNG    EQU  >26              'CHAT', 'BASE', AND 'FLAG')
0028      98  NUDTAB  EQU  >28
002A      99  VARA    EQU  >2A
002C     100  VAR81   EQU  >2C
002E     101  EXTRAM  EQU  >2E
0030     102  STLN    EQU  >30
0032     103  ENLN    EQU  >32
0034     104  DATA    EQU  >34
0036     105  LNBUF   EQU  >36
0038     106  RAMPTR  EQU  >38
003A     107  FREWRD  EQU  >3A
003C     108  INSTRT  EQU  >3C
003E     109  SYMTAB  EQU  >3E
0040     110  SYMPTR  EQU  >40
```

```
0042     CHAT   EQU  >42
0043     BASE   EQU  >43
0044     BUFFY  EQU  >44
004A     FAC    EQU  >4A
         *
         *      NOTE:  These are used by the string routines.
         *             They are used to avoid any variable conflicts
         *             because the string routines can be invoked from
         *             prescan as well as from execution(EXEC and 9900
         *             code)
         *
0052     BKPTR  EQU  FAC+8               BACKPOINTER - USED IN STRINGS
0054     COPYTO EQU  FAC+10              COPY-TO ADDR IN COMPCT
0056     TEMP1  EQU  FAC+12              TEMPORARY 1
0058     TEMP2  EQU  FAC+14              TEMPORARY 2
005A     TEMP3  EQU  FAC+16              TEMPORARY 3
         *
005C     ARG    EQU  >5C
         *
0064     TEMP4  EQU  ARG+8               TEMPORARY 4
0066     TEMP5  EQU  ARG+10              TEMPORARY 5
0068     TEMP6  EQU  ARG+12              TEMPORARY 6
006A     TEMP7  EQU  ARG+14              TEMPORARY 7
         *
006C     FPERAD EQU  >6C
006E     VSPTR  EQU  >6E
0075     SIGN$  EQU  >75
0076     EXP$   EQU  >76
         *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
         *  IT WAS USED FOR CALCULATOR FLAG REMOVED FROM 99/4A                  *
         *CALCFL EQU  >82                                                       *
         * REPLACE BY ONE BYTE FLAG ALLUP FOR ADDING LOWERCASE CHAR.            *
         * SET IN 99/4A 3/3/81                                                  *
         *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
0082     ALLUP  EQU  >82
0088     FLAG   EQU  >88
0089     GROMFL EQU  >89
0088     RSTK   EQU  >88           STARTS AT >8A
00BE     END    EQU  >BE           END OF SUBROUTINE STACK
         ************************************************************************
02E2     NLNADD EQU  >2E2          LOCATIONS ON VDP
02FE     ENDSCR EQU  >2FE
0320     CRNBUF EQU  >320
03C0     VROA$  EQU  >3C0          TEMPORARY MEMORY ROLL-OUT AREA
03DD     ONECHR EQU  >3DD          1 BYTE (USED FOR CHR$)
03EC     SAVEPC EQU  >3EC          2 BYTES ('BREAK' OR 'CONTINUE' ON)
03E0     SYMBOL EQU  >3E0          2 BYTES
03E2     VRMSND EQU  >3E2          9 BYTES (SOUND BLOCKS)
         *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
         *  REPLACE  VRAMVS EQU >5F8   FOR ADDING LOWERCASE CHAR. SET
         *  IN 99/4A     3/3/81                                                 *
         *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
06F8     VRAMVS EQU  >6F8
0800     SYMPT  EQU  >800
         ************************************************************************
```

```
0002    166  X1      EQU  >02          IMMEDIATE VALUES
0003    167  X2      EQU  >03
0017    168  Y1      EQU  >17
0020    169  BKGD    EQU  >20
0060    170  OFFSET  EQU  >60
```

```
                    172  *
                    173  *              B A S I C   T O K E N   T A B L E
                    174  *
                    175  *                                      REVISED 8/23/78
                    176  *       EQU  >80          SPARE
0081                177  ELSE$    EQU  >81
                    17   *       EQU  > 2          "::"                (62)
                    179  *       EQU  >83          "!"                 (62)
0084                1 0  IF$     EQU  >84          "IF"
0085                1 1  GO$     EQU  >85          "GO"
0086                182  GOTO$    EQU  >86          "GOTO"
0087                183  GOSUB$   EQU  >87          "GOSUB"
0088                184  RETUR$   EQU  >88          "RETURN"
0089                185  DEF$     EQU  > 9          "DEF"
008A                186  DIM$     EQU  >8A          "DIM"
008B                187  END$     EQU  >8B          "END"
008C                188  FOR$     EQU  >8C          "FOR"
008D                189  LET$     EQU  >8D          "LET"
008E                190  BREAK$   EQU  >8E          "BREAK"
008F                191  UNBRE$   EQU  > F          "UNBREAK"
0090                192  TRACE$   EQU  >90          "TRACE"
0091                193  UNTRA$   EQU  >91          "UNTRACE"
0092                194  INPUT$   EQU  >92          "INPUT"
0093                195  DATA$    EQU  >93          "DATA"
0094                196  RESTO$   EQU  >94          "RESTORE"
0095                197  RANDO$   EQU  >95          "RANDOMIZE"
0096                198  NEXT$    EQU  >96          "NEXT"
0097                199  READ$    EQU  >97          "READ"
0098                200  STOP$    EQU  >98          "STOP"
0099                201  DELET$   EQU  >99          "DELETE"
009A                202  REM$     EQU  >9A          "REM"
009B                203  ON$      EQU  >9B          "ON"
009C                204  PRINT$   EQU  >9C          "PRINT"
009D                205  CALL$    EQU  >9D          "CALL"
009E                206  OPTIO$   EQU  >9E          "OPTION"
009F                207  OPEN$    EQU  >9F          "OPEN"
00A0                208  CLOSE$   EQU  >A0          "CLOSE"
00A1                209  SUB$     EQU  >A1          "SUB"
00A2                210  DISPL$   EQU  >A2          "DISPLAY"
                    211  *       EQU  >A3          "SUBEXIT"           (62)
                    212  *       EQU  >A4          "SUBEND"            (62)
                    213  *       EQU  >A5          "REAL"              (62)
                    214  *       EQU  >A6          "INTEGER"           (62)
                    215  *       EQU  >A7          "SCRATCH"           (62)
                    216  *       EQU  >A8          "ACCEPT"            FUTURE
                    217  *       EQU  >A9          "IMAGE"             FUTURE
                    218  *       EQU  >AA          SPARES
                    219  *       EQU  >AB
                    220  *       EQU  >AC
                    221  *       EQU  >AD
                    222  *       EQU  >AE
                    223  *       EQU  >AF
00B0                224  THEN$    EQU  >B0          "THEN"
00B1                225  TO$      EQU  >B1          "TO"
00B2                226  STEP$    EQU  >B2          "STEP"
```

```
00B3    227  COMMA$  EQU  >B3        ","
00B4    228  SEMIC$  EQU  >B4        ";"
00B5    229  COLON$  EQU  >B5        ":"
00B6    230  RPAR$   EQU  >B6        ")"
00B7    231  LPAR$   EQU  >B7        "("
00B     232  CONC$   EQU  >B8        "&"  CONCATENATE STRINGS
        233  *       EQU  >B9        SPARE
        234  *       EQU  >BA        "OR"                (62)
        235  *       EQU  >BB        "AND"               (62)
        236  *       EQU  >BC        SPARE FOR "XOR"     (62)
        237  *       EQU  >BD        "NOT"               (62)
00BE    238  EQUAL$  EQU  >BE        "="
00BF    239  LESS$   EQU  >BF        "<"
00C0    240  GREAT$  EQU  >C0        ">"
00C1    241  PLUS$   EQU  >C1        "+"
00C2    242  MINUS$  EQU  >C2        "-"
00C3    243  MULT$   EQU  >C3        "*"
00C4    244  DIVI$   EQU  >C4        "/"
00C5    245  CIRCU$  EQU  >C5        "^"
        246  *       EQU  >C6        SPARE
00C7    247  STRIN$  EQU  >C7        QUOTED STRING
00C8    248  UNQST$  EQU  >C8        UNQUOTED STRING
00C8    249  NUM$    EQU  UNQST$     ALSO NUMERICAL STRING
00C9    250  LN$     EQU  >C9        GROM NUMERIC CONSTANT
        251  *       EQU  >CA        SPARE
00CB    252  ABS$    EQU  >CB        "ABS"
00CC    253  ATN$    EQU  >CC        "ATN"
00CD    254  COS$    EQU  >CD        "COS"
00CE    255  EXP$$   EQU  >CE        "EXP"
00CF    256  INT$    EQU  >CF        "INT"
00D0    257  LOG$    EQU  >D0        "LOG"
00D1    258  SGN$$   EQU  >D1        "SGN"
00D2    259  SIN$    EQU  >D2        "SIN"
00D3    260  SQR$    EQU  >D3        "SQR"
00D4    261  TAN$    EQU  >D4        "TAN"
00D5    262  LEN$    EQU  >D5        "LEN"
00D6    263  CHR$$   EQU  >D6        "CHR$"
00D7    264  RND$    EQU  >D7        "RND"
00D8    265  SEG$$   EQU  >D8        "SEG$"
00D9    266  POS$    EQU  >D9        "POS"
00DA    267  VAL$    EQU  >DA        "VAL"
00DB    268  STR$$   EQU  >DB        "STR$"
        269  *
        270  *
        271  *       EQU  >EC        "ALL"               (62)
        272  *       EQU  >ED        "USING"             (62)
        273  *       EQU  >EE        "BEEP"              (62)
        274  *       EQU  >EF        "ERASE"             (62)
        275  *       EQU  >F0        "AT"                (62)
00F1    276  BASE$   EQU  >F1        "BASE"
        277  *       EQU  >F2        "VARIABLE"          (62)
        278  *       EQU  >F3        "TEMPORARY"         (62)
        279  *       EQU  >F4        "RELATIVE"          (62)
        280  *       EQU  >F5        "INTERNAL"          (62)
00F6    281  SEQUE$  EQU  >F6        "SEQUENTIAL"
```

```
00F7    282  OUTPU$ EQU  >F7          "OUTPUT"
00F8    283  UPDAT$ EQU  >F8          "UPDATE"
00F9    284  APPEN$ EQU  >F9          "APPEND"
00FA    285  FIXED$ EQU  >FA          "FIXED"
00FB    286  PERMA$ EQU  >FB          "PERMANENT"
00FC    287  TAB$   EQU  >FC          "TAB"
00FD    288  NUMBE$ EQU  >FD          "#"
        289  *      EQU  >FE          GROM LINE POINTER
        290  *      EQU  >FF          SPARE
        291  ********************************************************************
0002    292  BREAK  EQU  >02
        293  ********************************************************************
        294  *            ASCII CODES FOR EACH CHARACTER
0030    295  ZERO   EQU  >30          0
0039    296  NINE   EQU  >39          9
003C    297  LESS   EQU  >3C          <
003E    298  GREAT  EQU  >3E          >
0041    299  A      EQU  >41          A
0046    300  F      EQU  >46          F
005F    301  UNLN   EQU  >5F
0020    302  SPACE  EQU  >20
007F    303  EDGECH EQU  >1F+OFFSET   SPECIAL EDGE OF SCREEN CHAR.
        304  ********************************************************************
0001    305  ROUND$ EQU  1             ROUND
0002    306  ROUNU$ EQU  2             ROUND UP
0009    307  FDIV   EQU  9             DIVISION
000A    308  FCOMP  EQU  10            COMPARE
0010    309  CSNUM  EQU  16            STRING TO NUMBER
0012    310  FLTINT EQU  18            FLOATING TO INTEGER
0013    311  SYM    EQU  19            GET SYMBOL TABLE ENTRY
0014    312  SMB    EQU  20            GET SYMBOL VALUE
0015    313  ASSGNV EQU  21            ASSIGN VALUE TO VARIABLE
0017    314  VPUSH  EQU  23            PUSH VALUE ON VALUE STACK
0018    315  VPOP   EQU  24            POP VALUE OFF VALUE STACK
001B    316  PGMCH  EQU  27            GET PROGRAM CHARACTER
        317  ********************************************************************
```

```
                319         GROM 2
                320         ORG  >0D00
4D00 56CD       321         BR   SCROLL
4D02 5120       322         BR   LITS05        WAS SCROL
4D04 4DB0       323         BR   EXEC
4D06 56BB       324         BR   LINE          WAS CATALG
4D08 5613       325         BR   DATAST        WAS REPLAC
4D0A 5645       326         BR   ASC           WAS UNSAVE,VPOP
4D0C 4DBF       327         BR   EXEC1
4D0E 4E38       328         BR   EXEC6C        RESUME OPERATION AFTER "BREAK"
4D10 4D8A       329         BR   RUN
4D12 515C       330         BR   GETSTR
4D14 55BB       331         BR   DELINK
4D16 56E1       332         BR   CONV1         CONVERT WITH WARNING
4D18 51A9       333         BR   COMPCT        GARBAGE COLLECTION
```

```
4D1A 4D2435   LINK1   DATA #LINK2,#SOUND,5,:SOUND:
 D1D 380553
 D20 4F554E
4D23 44
4D24 4D2E35   LINK2   DATA #LINK3,#GRAPH,5,:CLEAR:
4D27 1C0543
4D2A 4C4541
4D2D 52
4D2E 4D3 57   LINK3   DATA #LINK4,#COLOR,5,:COLOR:
4D31 130543
4D34 4F4C4F
4D37 52
4D38 4D4256   LINK4   DATA #LINK5,#GCHAR,5,:GCHAR:
4D3B EF0547
4D3E 434841
4D41 52
4D42 4D4C36   LINK5   DATA #LINK6,#HCHAR,5,:HCHAR:
4D45 0E0548
4D48 434841
4D4B 52
4D4C 4D5636   LINK6   DATA #LINK7,#VCHAR,5,:VCHAR:
4D4F 2A0556
4D52 434841
4D55 52
4D56 4D5F36   LINK7   DATA #LINKA,#CHARLY,4,:CHAR:
4D59 430443
 D5C 484152
 D5F 4D6737   LINKA   DATA #LINKB,#KEY,3,:KEY:
4D62 08034B
4D65 4559
4D67 4D7137   LINKB   DATA #LINKC,#JOYST,5,:JOYST:
4D6A 48054A
4D6D 4F5953
4D70 54
4D71 000037   LINKC   DATA #0,#BORDER,6,:SCREEN:
4D74 BF0653
4D77 435245
4D7A 454E
              *
              *       USED BY FLMGR
              *
              ERR14
4D7C 06284E   ERR5    CALL ERR$$          ERROR
4D7F 2064             DATA #MSG7          'BAD VALUE'
              *
4D81 06284E   ERR6    CALL ERR$$          ERROR
4D84 207D             DATA #MSG11         'STRING-NUMBER MISMATCH'
              *
              *       USED BY ERROR ROUTINE IN PSCAN
              *
4D86 56D4             BR   SCR010
              *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
              *       LINE ADDED BY STAN HUME 06/23/80 TO TAKE AWAY
              *       THE DIRECT REFERENCE BY FLMGR TO ERR
4D88 566C             BR   ERR
```

```
362  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><
```

```
                    364  ********************************************************
                    365  **      SUBROUTINE TO START TO RUN THE PROGRAM
                    366  *            RETURN WITH POINTER INTO LINE #BUFFER
                    367  *            IN @DATA AND LINE NUMBER IN @BUFFY
                    368  ********************************************************
4D8A 9320           369  RUN     DDEC  VARW          SO NOT TO MISS A CHARACTER
4D8C 06282C         370           CALL GETNB         GET A NON-BLANK CHARACTER IN
4D F 6DA3           371           BS   RUN2          END OF LINE
4D91 06283C         372           CALL G TLN         GET THE LINE # IN
4D94 C9202A         373           $IF  @VARW .DL. @VARA GOTO RUN4  ILLEGAL STMT
4D97 5671
4D99 06283E         374           CALL SEETWO        FIND THE LINE IN V. RAM
4D9C 5682           375           BR   RUN3          LINE NOT FOUND
4D9E BD342E         376           DST  @EXTRAM,@DATA     PROGRAM RUN STARTS HERE
4DA1 4DAA           377           BR   RUN2A             LINE NOT FOUND
4DA3 BD3432         378  RUN2     DST  @ENLN,@DATA       DEFAULT TO FIRST LINE
4DA6 A73400         379           DSUB >03,@DATA         POINTER TO LINE #
4DA9 03
4DAA BD44B0         380  RUN2A    DST  RAM(@DATA),@BUFFY LINE TO START THE EXECUTION
4DAD 34
4DAE 56CD           381           BR   SCROLL        SCROLL UP ONE LINE ***************
```

```
                    383  ********************************************************
                    384  **      SUBROUTINE TO START TO EXECUTE THE PROGRAM
                    385  *
                    386  *       DATA:
                    387  *            @DATA POINTS TO FIRST LINE NUMBER BUFFER
                    388  *                  ENTRY TO EXECUTE
                    389  *            @BUFFY CONTAINS THE FIRST LINE NUMBER
                    390  *                   OR ZERO IF IMPERATIVE STATEMENT
                    391  ********************************************************
4DB0 8F44           392  EXEC    DCZ   @BUFFY        STATEMENT OR WHOLE PROGRAM?
4DB2 6DCD           393          BS    EXEC2         NO LINE # STATEMENT
4DB4 BD2E34         394          DST   @DATA,@EXTRAM 1ST OF THE LINE # WHERE EXEC. ST
4DB7 952E           395          DINCT @EXTRAM       WHERE LENGTH ADDRESS STORED
4DB9 BD3632         396          DST   @ENLN,@LNBUF
4DBC 065613         397          CALL  DATAST
4DBF B28088         398  EXEC1   RB    @FLAG,7       IN 'RUN' MODE
4DC2 7F
4DC3 BE7F03         399          ST    X2,XPT        DISPLAY STARTS AT 1ST COLUMN.
                    400  *
                    401  *       TEST FOR END OF PROGRAM FOR CONTINUE
                    402  *
4DC6 C52E30         403  EXEC1B  DCH   @STLN,@EXTRAM
4DC9 4E5B           404          BR    EXEC7                 PAST PROGRAM - TERMINATE
4DCB 4DD1           405          BR    EXEC1A
                    406  *
4DCD BF2C03         407  EXEC2   DST   CRNBUF,@VAR81
4DD0 20
4DD1 BF264D         408  EXEC1A  DST   #EXEC20,@RTNG       RETURN FROM EXEC
4DD4 EB
4DD5 BF284E         409          DST   #NUDTB,@NUDTAB      NUD TABLE ADDRESS
4DD8 84
4DD9 8F446D         410        $IF     @BUFFY .DNE. 0 THEN
4DDC EA
4DDD 0403           411          BACK  >03                 Load backdrop couleur
4DDF BEA30F         412          ST    >10,RAM(>30F)       Load first color register
4DE2 10
4DE3 350010         413          MOVE 16 FROM RAM(>30F) TO RAM(>310) Copy to the res
4DE6 A310A3
4DE9 0F
                    414        $END IF
4DEA 11             415  EXEC1Z  EXEC
                    416  *
4DEB 8A23           417  EXEC20  CASE  @ERRCOD+1           WHAT TYPE OF RETURN
4DED 4E5B           418          BR    EXEC7               NORMAL END
4DEF 4E2E           419          BR    EXEC6               BREAKPOINT
4DF1 4E01           420          BR    EXEC5               TRACE
4DF3 565C           421          BR    ERROR$              ERROR
4DF5 5689           422          BR    WARNG$              WARNING
4DF7 5696           423          BR    EXPON               '^'
4DF9 54CF           424          BR    UDF                 FUNCTION
4DFB 5156           425          BR    GETST9              SYSTEM GET STRING
4DFD 51F2           426          BR    CONCAT              CONCATENATE STRINGS '&'
4DFF 51A3           427          BR    COMPC9              COMPACTIFY STRING SPACE
                    428  *
                    429  *
```

```
                    430  **      'TRACE ON' IS SET ON THIS LINE.
                    431  *
                    432  EXEC5
4E01 8620           433    CLR  @VARW                CLEAR UPPER ADDRESS BYTE
4E03 BC217F         434    ST   XPT,@VARW+1          COLLECT LOWER BYTE XPT
4E06 A32002         435    DADD NLNADD-3,@VARW       MAKE THAT A VALID ADDRESS ON SCREE
4E09 DF
4E0A C72002         436    $IF @VARW .DH. NLNADD+22 THEN ADDRESS MIGHT GO OFF SCREEN
4E0D F 4E17
4E10 0656CD         437      CALL SCROL              SCROLL TO NEXT LINE
4E13 BF2002         438      DST  NLNADD,@VARW       AND RE-INITIATE VARW
4E16 E2
                    439    $END IF
4E17 BEB020         440    ST   LESS+OFFSET,RAM(@VARW) OUTPUT OPEN BRACKET ("<")
4E1A 9C
4E1B 9120           441    DINC @VARW                INCREMENT ADDRESS POINTER
4E1D 065645         442    CALL ASC                  CONVERT LINE # INTO ASCII CODES.
4E20 BEB020         443    ST   GREAT+OFFSET,RAM(@VARW) CLOSE OFF WITH CLOSE BRACKET
4E23 9E
4E24 A72002         444    DSUB NLNADD-4,@VARW       DON'T FORGET TO UPDATE THE XPT
4E27 DE
4E28 BC7F21         445    ST   @VARW+1,XPT          <----- WE DO THAT HERE
4E2B 8722           446    DCLR @ERRCOD
4E2D 12             447    DATA >12                  RTNB INSTRUCTION
                    448  *
                    449  **      BREAKPOINT IS ON THIS LINE.
                    450  *
                    451  ************************************************************
                    452  *       CHANGED "GOTO EXEC1B" TO BE "GOTO EXEC1Z" TO ELIMINA
                    453  *          CHANGING COLOR TABLES ON GROM BASIC SHIFT-C
                    454  *            TOM FERRIO OCTOBER 2,1979
                    455  ************************************************************
                    456  *
                    457  EXEC6
4E2E 8E8089         458          $IF  @GROMFL .NE. 0 GOTO EXEC1Z  IGNORE BREAK IN GRO
4E31 4DEA
4E33 B2B02E         459          RB   RAM(@EXTRAM),7     MUST RESET B.P. SO CONTINUE
4E36 7F
4E37 03             460          SCAN                        WORKS.
4E38 BDA3EC         461  EXEC6C  DST  @EXTRAM,RAM(SAVEPC)    SAVE CONTINUE PC
4E3B 2E
                    462  EXEC6B
4E3C 8E8089         463          $IF  @GROMFL .NE. 0 GOTO EXEC1B  IGNORE BREAK IN GRO
4E3F 4DC6
4E41 0656CD         464          CALL SCROLL
4E44 060036         465          CALL TONE2
4E47 310010         466          MOVE 16 FROM ROM(#MSGBRK) TO RAM(NLNADD)
4E4A A2E220
4E4D E9
4E4E BF2002         467          DST  NLNADD+16,@VARW  STARTING LOCATION FOR LINE # D
4E51 F2
4E52 065645         468          CALL ASC             DISPLAY THE LINE #
                    469  EXEC7C
 :55 06201C         470          CALL CHRTAB
4E58 052012         471  EXEC6D  B    MAO
```

```
537  ********************************************************
538  ********************************************************
539  *  FLAGS USED IN RUN MODE:
540  *
541  ** @FLAG  BIT  RESET                SET
542  **          0   INTEGER             *FLOATING-POINT
543  **          1   OPTION BASE = 1     *OPTION BASE = 0
544  **          2
545  **
546  **          3   NON-FUNCTION        *EXECUTING A FUNCTION
547  **          4   'TRACE'             *'UNTRACE'
548  **          5   'PRESCAN' OR 'RUN'*'EDIT'
549  **          6   'UNBREAK'           *'BREAK
550  **          7   IN 'RUN' MODE       *IN 'PRESCAN' MODE
551  ********************************************************
552  ********************************************************
```

```
                    554  ********************************************************
                    555  **      SUBROUTINES TO EVALUATE FUNCTION VALUES
                    556  ********************************************************
4ED1 0657A6         557  NABS    CALL COMB           'ABS(X)'
4ED4 0ECB           558          PARSE ABS$
4ED6 064F79         559          CALL CKSTNM
4ED9 814A           560          DABS @FAC           NUMERIC
4EDB 10             561          CONT
                    562  *
                    563  *
                    564  *
4EDC BF5C00         565  NATN    DST  ATN,@ARG    'ATN'
4EDF 32
4EE0 4F50           566          BR   COMMON
                    567  *
                    568  *
                    569  *
4EE2 BF5C00         570  NCOS    DST  COS,@ARG      'COS'
4EE5 2C
4EE6 4F50           571          BR   COMMON
                    572  *
                    573  *
                    574  *
4EE8 BF5C00         575  NEXP    DST  EXP,@ARG      'EXP'
4EEB 28
4EEC 4F50           576          BR   COMMON
                    577  *
                    578  *
                    579  *
4EEE 0657A6         580  NINT    CALL COMB           'INT(X)'
4EF1 0ECF           581          PARSE INT$
4EF3 064F79         582          CALL CKSTNM
4EF6 060022         583          CALL GRINT
4EF9 10             584          CONT
                    585  *
                    586  *
                    587  *
4EFA BF5C00         588  NLOG    DST  LOG,@ARG      'LOG'
4EFD 2A
4EFE 4F50           589          BR   COMMON
                    590  *
                    591  *
                    592  *
4F00 BE4A3F         593  NRND    ST   >3F,@FAC      DECIMAL POINT
4F03 BE104B         594          ST   FAC+1,@VAR5   'RND'
                    595  *
4F06 0263           596  RNDO    RAND 99            GET 1 DIGIT
4F08 8E78           597          CZ   @RANDOM       0?
4F0A 4F16           598          BR   RND2          NO
4F0C 924A           599          DEC  @FAC          NORMALIZE IT
4F0E 8E4A           600          CZ   @FAC          0?
4F10 6F23           601          BS   RND3          YES
4F12 4F06           602          BR   RNDO          NO
                    603  *
4F14 0263           604  RND1    RAND 99
```

```
4F16 BC9010   605  RND2   ST   @RANDOM,*VAR5 PUT IT ON VALUE STACK.
4F19 78
4F1A D61051   606         CEQ  >51,@VAR5     7 DIGITS?
4F1D 6F25     607         BS   RND4          YES
4F1F 9010     608         INC  @VAR5         GET ANOTHER DIGIT.
4F21 4F14     609         BR   RND1
              610  *
4F23 8648     611  RND3   CLR  @FAC+1        RANDOM VALUE 0
4F25 10       612  RND4   CONT
              613  *
              614  *
              615  *
4F26 0657A6   616  NSGN   CALL COMB          'SGN(X)'
4F29 0ED1     617         PARSE SGN$$
4F2B 064F79   618         CALL CKSTNM
4F2E 8F4A     619         DCZ  @FAC          ZERO ?
4F30 6F3F     620         BS   SGN2          YES, LEAVE AS IS
4F32 DA4A80   621         TBR  @FAC,7        TEST POSITIVE OR NEGATIVE
4F35 310008   622         MOVE 8 FROM ROM(#FLT1) TO @FAC  WILL BE + OR -1
4F38 4A50C0
              623  *                         MOVE DOES NOT AFFECT STATUS
4F3B 6F3F     624         BS   SGN2          BRANCH IF POSITIVE (+1)
4F3D 834A     625         DNEG @FAC          -1
4F3F 10       626  SGN2   CONT
              627  *
              628  *
              629  *
4F40 BF5C00   630  NSIN   DST  SIN,@ARG      'SIN'
4F43 2E
4F44 4F50     631         BR   COMMON
              632  *
              633  *
              634  *
4F46 BF5C00   635  NSQR   DST  SQR,@ARG      'SQR'
4F49 26
4F4A 4F50     636         BR   COMMON
              637  *
              638  *
              639  *
4F4C BF5C00   640  NTAN   DST  TAN,@ARG      'TAN'
4F4F 30
              641  *
              642  COMMON
4F50 064F7F   643         CALL STKCHK
4F53 0657A6   644         CALL COMB          CHECK FOR '('
              645  *
4F56 9473     646         INCT @SUBSTK
4F58 BF9073   647         DST  #CA1,*SUBSTK WHERE TO RETURN TO AFTER FUNC. CA
4F5B 4F6B
              648  *
4F5D 9473     649         INCT @SUBSTK
4F5F BD9073   650         DST  @ARG,*SUBSTK WHICH FUNCTION TO CALL
4F62 5C
              651  *
4F63 0EFF     652         PARSE >FF
```

```
 F65 064F79   653          CALL CKSTNM
 F68 8654     654          CLR  @FAC+10       ASSUME NO ERROR OR WARNING
4F6A 00       655          RTN
              656  *
4F6B 8E54     657  CA1     CZ   @FAC+10       ANY WARNING OR ERROR?
4F6D 6FA7     658          BS   CA3           NO
4F6F C65401   659          CH   1,@FAC+10     OVFL OR BAD ARG?
4F72 56B5     660          BR   CA2           OVERFLOW
4F74 06284E   661  ILLARG  CALL ERR$$         BAD ARGUMENT
4F77 2094     662          DATA #MSG12        '* BAD ARGUMENT TO FUNCTIONS'
              663  *
              664  *
4F79 D64C65   665  CKSTNM  $IF @FAC+2 .EQ. >65 GOTO ERR6
4F7C 6D81
4F7E 00       666          RTN
              667  *
              668  *       CHECK STACKS FOR ENOUGH ROOM TO CALL MONITOR
              669  *
4F7F CA73B2   670  STKCHK  CH   END-12,@SUBSTK CHECK SUBROUTINE STACK
4F82 767D     671          BS   ERR3          '* MEMORY FULL'
4F84 BD5E6E   672          DST  @VSPTR,@ARG+2 VALUE STACK POINTER
4F87 A35E00   673          DADD 40,@ARG+2     POINTER TO LAST BYTE ON VALUE STAC
4F8A 28
4F8B C95E1A   674          $IF  @ARG+2 .DL. @STREND GOTO STKCH2
4F8E 4F98
4F90 0651A9   675          CALL COMPCT
 F93 C95E1A   676          $IF  @ARG+2 .DHE. @STREND GOTO ERR3
 F96 767D
4F98 00       677  STKCH2  RTN
```

```
                    679  ********************************************************
                    680  **      SUBROUTINE FOR '('
                    681  ********************************************************
                    682  NLPR
4F99 D642B6         683          CEQ   RPAR$,@CHAT
4F9C 7671           684          BS    ERR1
4F9E 0EB7           685          PARS  LPAR$         '('
4FA0 D642B6         686          CEQ   RPAR$,@CHAT
4FA3 5671           687          BR    ERR1
4FA5 0F1B           688          XML   PGMCH
4FA7 10             689  CA3     CONT
                    690  *
                    691  *
                    692  ********************************************************
                    693  **      SUBROUTINE FOR UNARY MINUS
                    694  ********************************************************
                    695  *
                    696  *       NUD FOR MINUS (UNARY MINUS)
                    697  *
                    698  NMINUS
4FA8 0EC2           699          PARSE MINUS$        '-'
4FAA 834A           700          DNEG @FAC
4FAC C64C63         701  NMIN10  CH    >63,@FAC+2          MUST HAVE NUMERIC OPERATOR
4FAF 7671           702          BS    ERR1
4FB1 10             703          CONT
                    704  *
                    705  *
                    706  *
                    707  ********************************************************
                    708  *       NUD FOR PLUS (UNARY PLUS)
                    709  ********************************************************
                    710  NPLUS
4FB2 0EC1           711          PARSE PLUS$         '+'
4FB4 4FAC           712          BR    NMIN10
                    713  *
```

```
                    715  *
                    716  *
                    717  ********************************************************
                    718  **      SUBROUTINE FOR 'FOR'
                    719  **      PUT THE INITIAL VALUE OF THE VARIABLE IN ITS
                    720  **      VARIBALE SPACE OR SYMBOL FORMAT.
                    721  ********************************************************
                    722  *
                    723  *         A STACK ENTRY FOR A 'FOR' STMT LOOKS LIKE:
                    724  ********************************************************
                    725  *PTR TO S.T. ENTRY[2];>67; 00 ; VALUE PTR[2] ; FOR LINE#[2]
                    726  *             STEP VALUE[8]
                    727  *              LIMIT[8]
                    728  ********************************************************
                    729  *
4FB6 CE4200         730  NFOR    $IF   @CHAT .NOT. .GT. 0 GOTO ERR1
4FB9 5671
                    731  *                                MUST NOT BE TOKEN OR END OF LINE
4FBB 0F13           732          XML   SYM
4FBD 8EB04A         733          $IF   RAM(@FAC) .NE. 0 GOTO ERR1  CAN'T BE STRING,ARR
4FC0 5671
                    734  *                                                  OR FUNCTION
4FC2 D642BE         735          $IF   @CHAT .NE. EQUAL$ GOTO ERR1  MUST HAVE '='
4FC5 5671
4FC7 0F14           736          XML   SMB                  GET INDEX'S VALUE SPACE
4FC9 0F1B           737          XML   PGMCH                GET INITIAL VALUE
4FCB BE4C67         738          ST    >67,@FAC+2
4FCE BD502E         739          DST   @EXTRAM,@FAC+6
                    740  *
                    741  ***     SEARCH STACK FOR ANOTHER FOR ENTRY WITH THE SAME
                    742  ***        LOOP VARIABLE.   IF ONE IS FOUND REMOVE IT.
                    743  *
4FD1 BD5C6E         744          DST   @VSPTR,@ARG          COPY STACK POINTER
                    745  *
                    746  *       SEE IF END OF STACK
                    747  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><*
                    748  *       CHANGED FROM:                                   *
                    749  *       $IF  @ARG .NOT. .H. @STVSPT GOTO NFOR1E         *
                    750  *       ON 9/6/79 BY STAN HUME                          *
                    751  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><*
4FD4 CD5C24         752  NFOR1A $IF   @ARG .DLE. @STVSPT GOTO NFOR1E
4FD7 5007
                    753  *
                    754  *       SEE IF FOR ENTRY
4FD9 D6E002         755          $IF   RAM(2(ARG)) .NE. >67 GOTO NFOR1B
4FDC 5C674F
4FDF EC
                    756  *
                    757  *       COMPARE LOOP VARIABLES
4FE0 D5B05C         758          $IF   RAM(@ARG) .DEQ. @FAC GOTO NFOR1C
4FE3 4A6FF2
4FE6 A75C00         759          DSUB 24,@ARG               SKIP THIS FOR ENTRY
4FE9 18
4FEA 4FD4           760          BR    NFOR1A               LOOP
                    761  ***
```

```
4FEC A75C00   762  NFOR1B DSUB 8,@ARG            SKIP 8 BYTE STACK ENTRY
4FEF 08
4FF0 4FD4     763        BR   NFOR1A            LOOP
              764  ***
              765  **    FOUND MATCHING LOOP VARIABLE.
              766  **    MOVE STACK DOWN 24 BYTES.
              767  *
4FF2 BD5E6E   768  NFOR1C DST  @VSPTR,@ARG+2     COPY STACK POINTER
4FF5 A55E5C   769        DSUB @ARG,@ARG+2       CALC. NUMBER OF BYTES TO M V
4FF8 7003     770        BS   NFOR1D            0 BYTES, SKIP MOVE.
4FFA 345EEF   771        MOVE @ARG+2 FROM RAM(8(ARG)) TO RAM(-16(ARG))
4FFD FFF05C
5000 E0085C
5003 A76E00   772  NFOR1D DSUB 24,@VSPTR         ADJUST TOP OF STACK
5006 18
              773  NFOR1E
              774  *
              775  **    NOW PUT NEW FOR ENTRY ON STACK
              776  *
5007 0F17     777        XML  VPUSH             RESERVE SPACE FOR LIMIT
5009 0F17     778        XML  VPUSH             RESERVE SPACE FOR STEP
500B 0F17     779        XML  VPUSH             PUSH SYMBOL I.D. ENTRY
500D 0EB1     780        PARSE TO$              PARSE THE INITIAL VALUE
500F 0F17     781        XML  VPUSH             PUSH THE LIMIT ABOVE THE ENT
5011 D642B1   782        $IF  @CHAT .NE. TO$ GOTO ERR1
5014 5671
5016 0F1B     783        XML  PGMCH
5018 0EB2     784        PARSE STEP$            GET THE LIMIT
501A C64C63   785        $IF @FAC+2 .H. >63 GOTO ERR6
501D 6D81
501F A76E00   786        DSUB 32,@VSPTR
5022 20
5023 0F17     787        XML  VPUSH             PUSH THE LIMIT
5025 8E4270   788        $IF  @CHAT .NE. 0 THEN
5028 45
5029 D642B2   789           $IF @CHAT .NE. STEP$ GOTO ERR1
502C 5671
502E A36E00   790           DADD 24,@VSPTR      CORRECT STACK PTR
5031 18
5032 0F1B     791           XML  PGMCH
5034 0E00     792           PARSE 00            GET THE INCREMENT
5036 A76E00   793           DSUB 24,@VSPTR      GET STACK TO NEEDED PLACE
5039 18
503A 8F4A6D   794           $IF  @FAC .DEQ. 0 GOTO ERR5   CAN'T HAVE ZERO IN
503D 7C
503E C64C63   795           $IF  @FAC+2 .H. >63 GOTO ERR6 CAN'T HAVE STRING
5041 6D81
5043 504B     796        $SELSE
5045 310008   797           MOVE 8 FROM ROM(#FLT1) TO @FAC
5048 4A50C0
              798        $SEND IF
504B 0F17     799        XML  VPUSH             PUSH THE STEP
504D A36E00   800        DADD 16,@VSPTR         GET TO INITIAL VALUE
5050 10
5051 0F18     801        XML  VPOP              GET INITIAL VALUE BACK
```

```
5053 0F15      802          XML  ASSGNV              ASSIGN THE INITIAL VALUE
 055 A36E00    803          DADD 8,@VSPTR            RESTORE TO TOP OF ENTRY
5058 08
               804  *
               805  *       NOW CHECK TO SEE IF EXECUTE THE LOOP AT ALL
               806  *
5059 BD00E0    807          DST  RAM(4(VSPTR)),@VARO  PREPARE TO GET RESULT
505C 046E
505E 350008    808          MOV      FROM RAM(@VARO) TO @ARG      GET RESULT
5061 5CB000
5064 350008    809          MOVE  8 FROM RAM(-16(VSPTR)) TO @FAC   GET LIMIT
5067 4AEFFF
506A F06E
506C 0F0A      810          FLTPT FCOMP              COMPARE THEM
506E 707B      811          BS   NFOR03
5070 DAEFFF    812          TBR  RAM(-8(VSPTR)),7    CHECK NEGATIVE
5073 F86E80
5076 507C      813          BR   NFOR05              IF A DECREMENT
5078 0A        814          GT
5079 707F      815          BS   NFOR07
507B 10        816  NFOR03 CONT                     RESULT IS W/IN LIMIT
               817  *
507C 0A        818  NFOR05 GT
507D 707B      819          BS   NFOR03              RESULT IS W/IN LIMIT
               820  ************************************************************
507F BE0001    821  NFOR07 ST   1,@VARO
 082 BD022E    822          DST  @EXTRAM,@MNUM
 085 BD0430    823          DST  @STLN,@VARY
5088 9504      824          DINCT @VARY
508A D52E04    825  NFOR09 $IF  @EXTRAM .DEQ. @VARY THEN        END OF PROGRAM-E
508D 5094
508F BD2E02    826             DST  @MNUM,@EXTRAM
5092 566C      827             BR   ERR
               828          $END IF
5094 062836    829          CALL SEARCH
5097 D6428C    830          $IF  @CHAT .EQ. FOR$ THEN     LOOK FOR
509A 50A0
509C 9000      831             INC  @VARO                MATCHING FOR
509E 50A9      832          $ELSE
50A0 D64296    833             $IF @CHAT .EQ. NEXT$ THEN
50A3 50A9
50A5 9200      834               DEC  @VARO
50A7 70AB      835               BS  NFOR10
               836             $END IF
               837          $END IF
50A9 508A      838             BR NFOR09
50AB 0F1B      839  NFOR10 XML  PGMCH               PICK UP NAME TO COMPARE
50AD 8E4276    840          $IF  @CHAT .EQ. 0 GOTO ERR1   MUST HAVE NAME
50B0 71
50B1 0F13      841          XML  SYM                 GET SYMBOL TABLE ENTRY
50B3 D54AB0    842          $IF  @FAC .DNE. RAM(@VSPTR) GOTO ERR  MUST BE SAME A
50B6 6E566C
 0B9 A76E00    843          DSUB  24,@VSPTR          THROW AWAY THE ENTRY
 0BC 18
50BD 8642      844          CLR  @CHAT
```

```
50BF 10          845           CONT
                 846   *
50C0 400100      847   FLT1    DATA >40,1,0,0,0,0,0,0
50C3 000000
50C6 0000
```

```
                849  ********************************************************
                850  *       SUBROUTINE FOR 'RANDOMIZE'
                851  ********************************************************
50C8 8E42       852  NRNDMZ CZ    @CHAT          'RANDOMIZE', SEED PROVIDED?
50CA 50D1       853         BR    RNDMZ1         YES
50CC BC80C1     854         ST    @TIMER,@>C1    NO, USE VALUE IN @TIMER AS NEW SEED
50CF 79
50D0 10         855         CONT                 CONTINUE PARSE
                856
50D1 0E00       857  RNDMZ1 PARSE 0
50D3 064F79     858         CALL  CKSTNM         STRING OR NUMERIC?
50D6 BD80C0     859         DST   @FAC,@>C0      NEW SEED
50D9 4A
50DA 10         860         CONT                 CONTINUE PARSE
```

```
                    862  ********************************************************
                    863  *       SUBROUTINE FOR 'CALL'
                    864  ********************************************************
50DB D642C8         865  NCALL   CEQ   UNQST$,@CHAT      Unquoted string for name?
50DE 5671           866          BR    ERR1              No-so error
50E0 860C           867          CLR   @BYTE
                    868  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    869  *       CHANGED FROM:
                    870  *       $IF @GROMFL .EQ. 0 THEN
                    871  *         DST  @VAR81,@FAC+12 POINTER TO LENGTH OF THE NAME
                    872  *       $SELSE
                    873  *       BY STAN HUME 06/23/80 TO SAVE 2 BYTES
                    874  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
50E2 BD562C         875          DST   @VAR81,@FAC+12    Pointer to name in VDP
50E5 8E8089         876          $IF @GROMFL .NE. 0 THEN If GROM program-move to VDP
50E8 7101
50EA 330001         877            MOVE 1 FROM ROM(@VAR81) TO @VAR1 Fetch name lengt
50ED 0D0000
50F0 2C
50F1 900D           878            INC  @VAR1            Going to copy name + length
                    879  *     COPY LENGTH AND NAME INTO CRUNCH BUFFER
50F3 320CA3         880            MOVE @BYTE FROM ROM(@VAR81) TO RAM(CRNBUF)
50F6 200000
50F9 2C
50FA BF5603         881            DST  CRNBUF,@FAC+12   Pointer to length
50FD 20
50FE A12C0C         882            DADD @BYTE,@VAR81     Advance ptr to after name
                    883          $END IF
5101 060010         884          CALL CPL                Attempt to call the subprogr
5104 0A             885          DATA 10                 Search GPL subprogram links
5105 710C           886          BS   ERRC               If name not found-error
5107 8E4256         887          $IF  @CHAT .NE. 0 GOTO ERR1 If not EOL- error
510A 71
510B 10             888          CONT                    Otherwise-A OK-continue on
                    889
                    890  ERRC
510C 06284E         891          CALL ERR$$
510F 2040           892          DATA #MSG4         BAD NAME
```

```
                    894  ********************************************************
                    895  *            THE FOLLOWING SEVERAL PAGES CONTAIN THE   *
                    896  *                  EXTENDED STRING PACKAGE             *
                    897  *       THE ROUTINES ARE:                              *
                    898  *         LITSTR - Move a string literal from the      *
                    899  *                  program to the string space         *
                    900  *         GETSTR - Get space for a string in the string*
                    901  *                  space                               *
                    902  *         COMPCT - Garbage collection routine          *
                    903  *         COPYST - COPY STRING(OPTIMAL  COPY)          *
                    904  *         INTARG - CH CKS THAT AN ARGUMENT IS A NUMERIC*
                    905  *                  and converts it from floating point *
                    906  *                  to an integer                       *
                    907  *         PUSSTR - Checks that an argument is a string *
                    908  *                  and pushes it on the stack fixing up*
                    909  *                  the back pointer if it is a temporary*
                    910  *         CONCAT - Concatenates 2 strings together     *
                    911  *         SEG$   - Segments a string                   *
                    912  *         LEN    - Puts the length of a string in the FAC*
                    913  *         CHR$   - Converts an integer into its ASCII  *
                    914  *                  character                           *
                    915  *         STR$   - Converts a number into its string   *
                    916  *                  equivalent                          *
                    917  *         VAL    - Converts a string into its numeric  *
                    918  *                  equivalent                          *
                    919  *         POS    - Gives the position of one string within*
                    920  *                  another                             *
                    921  *                                                      *
                    922  *       AN ENTRY IN THE FAC LOOKS LIKE:                *
                    923  * +-----------+-----+---+---- --------+---------------+*
                    924  * |addr of ptr| >65 | 00 |addr of string|length of string|*
                    925  * +-----------+-----+----+ -----------+---------------+*
                    926  * ^           ^     ^    ^            ^               ^*
                    927  * FAC      FAC+2 FAC+3 FAC+4         FAC+6           FAC+8*
                    928  *                                                      *
                    929  ********************************************************
                    930  *                                                      *
                    931  *       SUBROUTINE FOR 'STRCON' (QUOTED STRING CONSTANT)*
                    932  ********************************************************
5111 BD662C         933  NSTRCN DST  @VAR81,@TEMP5      ADDR OF STRING
5114 06511D         934         CALL LITSTR
5117 A12C50         935         DA   @FAC+6,@VAR81      SKIP OVER STRING
511A 0F13           936         XML  PGMCH              GET THE NEXT CHARACTER
511C 10             937         CONT
                    938  *
                    939  LITSTR
511D BC5142         940         ST   @CHAT,@FAC+7       SAVE LENGTH OF STRING
5120 8650           941  LITS05 CLR  @FAC+6
5122 BD0C50         942         DST  @FAC+6,@BYTE       LENGTH FOR GETSTR
5125 06515C         943         CALL GETSTR             ALLOCATE STRING SPACE
5128 BC5280         944         ST   @GROMFL,@FAC+8     COPY GROM FLAG FOR LATER
512B 89
 12C BF4A00         945  LITS08 DST  SREF,@FAC          SAVE ADDR OF STRING PTR
 12F 1C
5130 BD4E1C         946         DST  @SREF,@FAC+4       SAVE ADDR OF STRING
```

```
                    947  *
                    948  *     NOTE: THIS CAUSES EXTRA LEVEL OF INDIRECT
                    949  *     CAN EASILY CHECK FOR TEMPORARY STRINGS NOW.
                    950  *
5133 BF4C65         951          DST  >6500,@FAC+2       INDICATES A STRING CONSTANT.
5136 00
                    952  ******COPY STRING INTO STRING SPACE
5137 8F0C           953  LITS09 DCZ  @BYTE               NULL STRING?
5139 714E           954         BS   LITS10              YEP-NOTHING TO DO
513B 8E5251         955         $IF @FAC+8 .EQ. 0 THEN   SAVED GROM FLAG OR CLEARED
513E 47
513F 340CB0         956         MOVE @BYTE FROM RAM(@TEMP5) TO RAM(@SREF)
5142 1CB066
5145 514E           957         $SELSE
5147 320CB0         958         MOVE @BYTE FROM ROM(@TEMP5) TO RAM(@SREF)
514A 1C0000
514D 66
                    959         $SEND IF
514E 00             960  LITS10 RTN
                    961  *
514F 06515C         962  LITS06 CALL GETSTR              ALLOCATE STRING SPACE
5152 8652           963         CLR  @FAC+8             SET FLAG TO VDP
5154 512C           964         BR   LITS08             JUMP INTO CODE
```

```
                    966  *
                    967  ********************************************************
                    968  *
                    969  *       GETSTR - checks to see if there is enough
                    970  *            space in the string area to allocate
                    971  *            a string.  it there is it allocates it.
                    972  *            IF THERE IS NOT IT DOES A GARBAGE
                    973  *            collection and once again checks to see
                    974  *            if there is enough room.  If so it
                    975  *            allocates it,  if not it issues a
                    976  *            * MEMORY FULL   message.
                    977  *
                    978  *       INPUT:  #OF BYTES NEEDED IN @BYTE
                    979  *       OUTPUT:  PTR TO NEW STRING IN @SREF
                    980  *                 BOTH LENGTH BYTES IN PLACE & ZEROED BKPTR
                    981  *                 @STREND POINTS 1ST FREE BYTE(NEW)
                    982  *       USES:   TEMP1 AND TEMP2 AS TEMPORARIES
                    983  *       Note:   See comment in COMPCT.
                    984  *
                    985  *       Note:   COMPCT allows a buffer zone of 8 stack entries
                    986  *               above what is there when COMPCT is called.
                    987  *               This should allow enough space to avoid a
                    988  *               collision between the string space and the
                    989  *               stack.  If garbage begins to appear in the str
                    990  *               space that can't be accounted for, the buffer
                    991  *               zone will need to be increased.
                    992  *
                    993  ********************************************************
                    994  *       ENTRY POINT FOR ASSEMBLY LANGUAGE
5156 06515C         995  GETST9 CALL GETSTR
5159 8623           996         CLR  @ERRCOD+1
515B 12             997         DATA >12                    RETURN TO ASSEMBLY LANGUAGE
                    998  *
                    999  GETSTR
515C A30C00        1000         DADD 4,@BYTE                ADJUST FOR BACKPTR & 2 LENGTHS
515F 04
5160 BD561A        1001         DST  @STREND,@TEMP1         CHECK IF ENOUGH ROOM
5163 A5560C        1002         DSUB @BYTE,@TEMP1           BY ADVANCING THE FREE PTR
5166 BD586E        1003         DST  @VSPTR,@TEMP2          GET VALUE STACK PTR
5169 A35800        1004         DADD 64,@TEMP2              ALLOW BUFFER ZONE
516C 40
516D C55658        1005         DCH  @TEMP2,@TEMP1          ENOUGH SPACE?
5170 7187          1006         BS   GETS18                 YES-ALL IS WELL
5172 0651A9        1007         CALL COMPCT                 NO-COMPACTIFY
5175 BD586E        1008         DST  @VSPTR,@TEMP2          GET VALUE STACK POINTER
5178 A35800        1009         DADD 64,@TEMP2              ALLOW BUFFER ZONE
517B 40
517C BD561A        1010         DST  @STREND,@TEMP1         GET NEW FREE PTR
517F A5560C        1011         DSUB @BYTE,@TEMP1           ADVANCE IT
5182 C55658        1012         DCH  @TEMP2,@TEMP1          ENOUGH SPACE NOW?
5185 567D          1013         BR   ERR3                   NO *MEMORY FULL
5187 A70C00        1014  GETS18 DSUB 4,@BYTE                EXACT # OF BYTES NEEDED
  518A 04
518B BCB01A        1015         ST   @BYTE+1,RAM(@STREND)   STORE ENTRY LENGTH(-4)
518E 0D
```

```
518F A51A0C  1016          DSUB @BYTE,@STREND      PT AT FIRST BYTE OF STRING
5192 BD1C1A  1017          DST  @STREND,@SREF      POINT SREF AT THE STRING
5195 A71A00  1018          DSUB 4,@STREND          PT AT FIRST FREE BYTE
5198 04
5199 87E001  1019          DCLR RAM(1(STREND))     ZERO THE BACKPTR
519C 1A
519D BCE003  1020          ST   @BYTE+1,RAM(3(STREND))   PUT IN STRING LENGTH
51A0 1A0D
51A2 00      1021          RTN                     ALL DONE
```

```
                1023  *
                1024  ********************************************************
                1025  *
                1026  *       COMPCT - IS THE GARBAGE COLLECTION ROUTINE.    IT
                1027  *                CAN BE INVOKED BY GETSTR OR MEMCHK.   IT
                1028  *                COPIES ALL USED STRINGS TO THE TOP OF THE
                1029  *                STRING SPACE SUPPRESSING OUT ALL OF THE
                1030  *                UNUS D STRINGS.
                1031  *
                1032  *       INPUT: NONE
                1033  *       OUTPUT:UPDATED @STRSP AND @STREND
                1034  *       USES:
                1035  *            BKPTR: EQUATED TO FAC+8        >52
                1036  *            COPYTO: EQUATED TO FAC+10      >54
                1037  *            TEMP1:  EQUATED TO FAC+12      >56
                1038  *            TEMP2:  EQUATED TO FAC+14      >58
                1039  *            TEMP3:  EQUATED TO FAC+16      >5A
                1040  *            TEMP4:  EQUATED TO ARG+8       >64
                1041  *            TEMP5:  EQUATED TO ARG+10      >66
                1042  *            TEMP6:  EQUATED TO ARG+12      >68
                1043  *=============================================================*
                1044  *>>>   NOTE:  THESE ARE USED TO AVOID ANY POSSIBLE       <<<
                1045  *>>>          variable conflicts because COMPCT and      <<<
                1046  *>>>          COPYST(CALLED BY COMPCT) CAN BE CALLED     <<<
                1047  *>>>          DURING PRESCAN AS WELL AS DURING EXECUTION<<<
                1048  *=============================================================*
                1049  ********************************************************
51A3 8623       1050  COMPC9 CLR   @ERRCOD+1          ENTRY FOR 9900
51A5 0651A9     1051         CALL  COMPCT             DO THE GARBAGE COLLECTION
51A8 12         1052         DATA  >12                RETURN
                1053  COMPCT
51A9 BD5440     1054         DST   @SYMPTR,@COPYTO    GET COPY-TO ADDRESS
51AC BD5218     1055         DST   @STRSP,@BKPTR      GET BEGINNING OF STRING SPACE
51AF BD1840     1056         DST   @SYMPTR,@STRSP     SET NEW STRING SPACE PTR
51B2 9152       1057         DINC  @BKPTR             POOR WAY TO DO IT,BUT HOW ELSE
51B4 9352       1058  COMP03 DDEC  @BKPTR             POINT AT LENGTH OF ENTRY
51B6 8656       1059         CLR   @TEMP1
51B8 BC57B0     1060         ST    RAM(@BKPTR),@TEMP1+1  SAVE LENGTH JUST IN CASE
51BB 52
                1061  *                              WE NEED TO MOVE THIS ENTRY
51BC C91A52     1062         DCHE  @BKPTR,@STREND     ARE WE FINISHED?
51BF 51C5       1063         BR    COMP05             NOPE-TOO BAD
51C1 BD1A54     1064         DST   @COPYTO,@STREND    SET END OF FREE SPACE
51C4 00         1065         RTN
51C5 A55256     1066  COMP05 DSUB  @TEMP1,@BKPTR      POINT AT THE STRING
51C8 A75200     1067         DSUB  3,@BKPTR           POINT AT THE BACK POINTER
51CB 03
51CC 8FB052     1068         DCZ   RAM(@BKPTR)        GARBAGE?
51CF 71B4       1069         BS    COMP03             YES-SKIP IT
                1070  *
                1071  *       WE NOW HAVE AN ENTRY TO BE COPIED
                1072  *       PERTINENT VARIABLES AT THIS POINT:
                1073  *            COPYTO-IS WHERE THE ENTRY WILL END
                1074  *            TEMP1- # OF BYTES TO BE MOVED(DOES NOT
                1075  *                 INCLUDE LENGTHS AND BACKPTR)
```

```
                    1076  *          BKPTR-TELLS WHERE TO MOVE FROM(PTS AT BKPTR)
                    1077  *      IN GENERAL : MOVE @TEMP1 BYTES FROM RAM(@BKPTR TO
                    1078  *                   RAM(@COPYTO-@TEMP1)
                    1079  *
51D1 065423         1080        CALL COPYST             DO THE COPY
51D4 A35400         1081        DADD 4,@COPYTO          POINT AT STRING
51D7 04
51D8 BD5 EF         1082        DST  RAM(-3(COPYTO)),@TEMP2   PICK UP BACK POINTER
51DB FFFD54
51DE BDB058         1083        DST  @COPYTO,RAM(@TEMP2)   ADJUST FORWARD POINTER
51E1 54
51E2 A75400         1084        DSUB 4,@COPYTO          POINT BACK AT FREE SPACE
51E5 04
51E6 51B4           1085        BR   COMP03             CONTINUE ON(JUMP ALWAYS)
                    1086  *
```

```
                1088  *
                1089  ********************************************************
                1090  *
                1091  *       PUSSTR - Insures that the entry in the FAC
                1092  *            IS A STRING AND PUSHES IT ONTO THE STACK,
                1093  *            fixing up the back pointer if the string
                1094  *            is a temp.
                1095  *
                1096  ********************************************************
                1097  *
                109   PUSSTR
51E8 D64C65     1099          $IF @FAC+2 .NE. >65 GOTO ERR6
51EB 4D81
51ED 0F17       1100          XML  VPUSH              PUSH THE ARGUMENT
51EF 0F1B       1101          XML  PGMCH              GET NEXT CHARACTER
51F1 00         1102          RTN
```

```
                 *
                 ****************************************************************
                 *
                 *       CONCAT - CONCATENATES TWO STRINGS TOGETHER
                 *
                 *       INPUT: FLOATING POINT ACCUMULATOR ENTRIES
                 *       OUTPUT: CONCATENATED STRING AND (POSSIBLE)
                 *               ZEROED BACK-POINTERS FOR THE OLD STRINGS
                 *       US S: TEMP1,TEMP4 AND TEMP5 AS TEMPORARIES
                 *         Note:  See comment in COMPCT
                 ****************************************************************
                 CONCAT
51F2 8623                CLR   @ERRCOD+1           CLEAR THE ERROR CODE
51F4 0651E8              CALL  PUSSTR              PUSH THE STRING AND GET NEXT
51F7 0EB8                PARSE CONC$               GET THE R.H. ARGUMENT
51F9 D64C65              CEQ   >65,@FAC+2          R.H. ARG A STRING?
51FC 4D81                BR    ERR6                NO-ERROR-STRING EXPECTED
51FE BD0C50              DST   @FAC+6,@BYTE        GET R.H. LENGTH
5201 A10CE0              DADD  RAM(6(VSPTR)),@BYTE  ADD IN L.H. LENGTH
5204 066E
5206 C70C00              DCH   255,@BYTE           >255 CHARACTERS?
5209 FF
520A 5210                BR    CONC04              NOPE-OK
520C BF0C00              DST   255,@BYTE           YEP-TRUNCATE IT
520F FF
5210 BD680C      CONC04  DST   @BYTE,@TEMP6        KEEP LENGTH FOR LATER
5213 0F17                XML   VPUSH
5215 06515C              CALL  GETSTR              ALLOCATE THE RESULT STRING
5218 0F18                XML   VPOP                RETRIEVE R.H.
521A 350008              MOVE 8 FROM @FAC TO @ARG
521D 5C4A
521F 0F18                XML   VPOP                RETREIEVE L.H.
5221 BD664E              DST   @FAC+4,@TEMP5       SET PTR TO L.H. ARG(FOR FRES
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                 *       THE FOLLOWING LINE REMOVED BY STAN HUME ON 06/23/8C
                 *       BECAUSE FRESTR NOT NEEDED BECAUSE VPOP TAKES CARE C
                 *       FREEING STRINGS.
                 *       CALL  FRESTR              FREE IT IF NECESSARY
                 *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
5224 BD0C50              DST   @FAC+6,@BYTE        LENGTH OF L.H. ARG.
5227 8652                CLR   @FAC+8              FORCE VDP MODE
5229 06512C              CALL  LITS08              SET UP FAC&COPY L.H. ARG IN
522C 8F62                DCZ   @ARG+6              R.H. LENGTH = 0?
522E 7246                BS    CONC06              YES - NO NEED TO COPY IT
5230 BD641C              DST   @SREF,@TEMP4        GET PTR TO NEW STRING
5233 A16450              DADD  @FAC+6,@TEMP4       PTR TO WHERE 2nd STRING BEGI
5236 A56850              DSUB  @FAC+6,@TEMP6       LENGTH OF 2nd STRING(POSSIBL
                 *                                 TRUNCATED)
5239 8F68                DCZ   @TEMP6             WHOLE 2nd STRING TRUNCATED OU
523B 7246                BS    CONC06              YES-DON'T TRY TO COPY
523D 3468B0              MOVE @TEMP6 FROM RAM(@ARG+4) TO RAM(@TEMP4) COPY T
5240 64B060
5243 A15068              DADD  @TEMP6,@FAC+6       ADD IN LENGTH OF 2nd ARG
                 *       NOTE: FAC+6 ALREADY CONTAINED LENGTH OF 1ST ARG FRC
                 *             THE PARSE THAT WAS DONE ON IT
```

```
- 5246 BD6660  CONC06 DST  @ARG+4,@TEMP5      PT TO R.H. ARG TO FREE IT
               *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
               *        THE FOLLOWING LINE REMOVED BY STAN HUME ON 06/23/80 *
               *        BECAUSE VPOP TAKES CARE OF FREEING STRINGS          *
               *        CALL FRESTR               FREE IT IF NECESSARY      *
               *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*
  5249 10                CONT                     DONE
```

```
                    1162  ***************************************************
                    1163  *
                    1164  *       SEG$(A$,X,Y) - EXTRACTS THE DESIGNATED STRING FROM
                    1165  *            A$.  X specifies the character position within
                    1166  *            A$ at which the extraction begins.  Y specifie
                    1167  *            the length of the string to be extracted.  If
                    1168  *            X or Y is negative an error occurs.  If X=0 an
                    1169  *            error occurs.  If Y=0 or X > length of A$ then
                    1170  *            the null string is returned.  If the remaining
                    1171  *            length in A$ starting at the position specifie
                    1172  *            by X is less than the length specified by Y, t
                    1173  *            the remainder of A$ starting at position X is
                    1174  *            returned.
                    1175  *       INPUT - Control is turned over to SEG$ from PARSE.
                    1176  *            THE ONLY REQUIREMENT IS THAT A SEG$ WAS ENCOU
                    1177  *       OUTPUT - The Floating Point Accumulator is set up
                    1178  *            with the header for the segmented string.
                    1179  *       USES - TEMP1(Others in calls to GETSTR,LITSO8)
                    1180  *
                    1181  ***************************************************
                    1182  *
524A 0657A6         1183  SEG$01 CALL COMB                  LEFT PAREN?
524D 0F1B           1184         XML  PGMCH                 GET NEXT TOKEN
524F 0EB3           1185         PARSE COMMA$               GET SOURCE STRING
5251 D642B3         1186         CEQ  COMMA$,@CHAT          END ON A COMMA?
5254 5671           1187         BR   ERR1                  NOPE-SYNTAX ERROR
5256 0651E8         1188         CALL PUSSTR                PUSH THE STRING AND GET NEXT
5259 0EB3           1189         PARSE COMMA$               GET POSITION W/IN SOURCE
525B D642B3         1190         CEQ  COMMA$,@CHAT          END ON A COMMA?
525E 5671           1191         BR   ERR1                  * SYNTAX ERROR
5260 065740         1192         CALL INTARG                CHECK & CONVERT ARG TO INTE
5263 8F4A           1193         DCZ  @FAC                  ARG=0?
5265 6D7C           1194         BS   ERR5                  * VALUE OUT OF RANGE
5267 0F17           1195         XML  VPUSH                 PUSH THE ARG
5269 0F1B           1196         XML  PGMCH                 GET NEXT TOKEN
526B 0EB6           1197         PARSE RPAR$                GET EXTRACTION LENGTH
526D D642B6         1198         CEQ  RPAR$,@CHAT           END ON A RPAR?
5270 5671           1199         BR   ERR1                  * SYNTAX ERROR
5272 065740         1200         CALL INTARG                CHECK & CONVERT ARG TO INTE
5275 BD5C4A         1201         DST  @FAC,@ARG             MOVE EXTRACTION LENGTH
5278 0F18           1202         XML  VPOP                  GET POSITION BACK
527A BD5E4A         1203         DST  @FAC,@ARG+2           MOVE POSITION
527D 0F18           1204         XML  VPOP                  RETRIEVE SOURCE STRING
527F BD565E         1205         DST  @ARG+2,@TEMP1         GET POSITION W/IN STRING
5282 C55650         1206         DCH  @FAC+6,@TEMP1         POSITION > LENGTH?
5285 72BA           1207         BS   SEG$08                YES-EXTRACT NULL
5287 A1565C         1208         DADD @ARG,@TEMP1           COMPUTE END OF SUBSTRING
528A A55650         1209         DSUB @FAC+6,@TEMP1         COMPUTE LENGTH BEYOND END
528D 9356           1210         DDEC @TEMP1                  STRING
528F D35600         1211         DCGE 0,@TEMP1
5292 00
5293 529D           1212         BR   SEG$06                FINE IF SUBSTRING IS SHORT
5295 BD5C50         1213         DST  @FAC+6,@ARG           ELSE, TRUNCATE LENGTH OF SU
5298 A55C5E         1214         DSUB @ARG+2,@ARG           SUBT POS FROM SOURCE LENGTH
529B 915C           1215         DINC @ARG                  INC TO INCLUDE LAST CHAR
```

```
529D BD0C5C  1216  SEG$06 DST  @ARG,@BYTE        # OF BYTES NEEDED FOR SUBSTRIN
 2A0 0F17    1217         XML  VPUSH
52A2 06515C  1218         CALL GETSTR           ALLOCATE RESULT STRING
52A5 0F18    1219         XML  VPOP
52A7 BD664E  1220         DST  @FAC+4,@TEMP5    PTR TO SOURCE FOR FRESTR & LIT
             1221  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><*
             1222  *       TH  FOLLOWING LIN  REMOVED BY STAN HUME ON 06/23/80 *
             1223  *       BECAUSE VPOP TAKES CARE OF FREEING STRINGS          *
             1224  *       CALL FRESTR           FREE SOURCE IF NECESSARY      *
             1225  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><*
52AA A1665E  1226         DADD @ARG+2,@TEMP5    PTR TO START OF SUBSTRING
52AD 9366    1227         DDEC @TEMP5
52AF BD500C  1228         DST  @BYTE,@FAC+6     SET LENGTH OF STRING.
52B2 8652    1229         CLR  @FAC+8           FORCE VDP MODE
52B4 06512C  1230         CALL LITS08           COPY IN & SET UP FAC
52B7 0F1B    1231         XML  PGMCH            GET NEXT TOKEN
52B9 10      1232         CONT
52BA 875C    1233  SEG$08 DCLR @ARG             EXTRACT A NULL STRING
52BC 529D    1234         BR   SEG$06           >>JUMP ALWAYS<<
```

```
                1236  ************************************************************
                1237  *
                1238  *       LEN(A$) - Calculate the length of a string and
                1239  *            leave the result in the FAC.
                1240  *       CONTROL - Turned over to NLEN from the parser
                1241  *       US S - No temporaries.
                1242  *
                1243  ************************************************************
                1244  *
52BE 0657A6     1245  LEN01   CALL COMB           'LEN(X$)' CHECK FOR '('
52C1 0EFF       1246          PARSE >FF                 Get tge string characters
52C3 D64C65     1247          $IF @FAC+2 .NE. >65 GOTO ERR6  If not string value
52C6 4D81
52C8 BC5C51     1248          ST    @FAC+7,@ARG         LENGTH
52CB 865D       1249  LEN02   CLR   @ARG+1            Second byte 0 for proper stor_
52CD 310008     1250          MOVE 8 FROM ROM(#FLT1) TO @FAC
52D0 4A50C0
52D3 C65C63     1251    $IF @ARG .H. 99 THEN      Overflow and compute for proper
52D6 52E0
                1252      $  range between 100 and 9999
52D8 C05D5C     1253      EX    @ARG,@ARG+1               Prepare for divide
52DB AE5C64     1254      DIV   100,@ARG          Get result in ARG, REM. in ARG+1
52DE 904A       1255      INC   @FAC              Update exponent
                1256    $END IF                   And continue normal
52E0 BD4B5C     1257    DST   @ARG,@FAC+1
52E3 8E4B52     1258    $IF @FAC+1 .EQ. 0 THEN
52E6 E9
52E7 864A       1259      CLR @FAC
                1260    $END IF
52E9 10         1261          CONT
```

```
                    1263  ********************************************************
                    1264  *
                    1265  *       CHR$(X) - Takes integer value X and converts
                    1266  *            the number into the ASCII representation for th
                    1267  *            number.
                    126   *       CONTROL - Turned over to NCHR by the parser.
                    1269  *       OUPUT - FAC is set up with the string entry
                    1270  *       USES - Uses temporaries when invoking LITSO6(LITSTR)
                    1271  *
                    1272  ********************************************************
                    1273  *
52EA 0657A6         1274  CHR$01 CALL COMB                 'CHR$' CHECK FOR '('
52ED 0EFF           1275         PARSE >FF                 GET THE VALUE
52EF 065740         1276         CALL INTARG               CONVERT INTO INTEGER
52F2 BF0C00         1277         DST  1,@BYTE              CREATE A LENGTH 1 STRING
52F5 01
52F6 BCA3DD         1278         ST   @FAC+1,RAM(ONECHR)   MOVE THE VALUE TO VDP(FOR LI
52F9 4B
52FA BF6603         1279         DST  ONECHR,@TEMP5        ADDRESS OF CHARACTER.
52FD DD
52FE 06514F         1280         CALL LITSO6               CREATE STRING AND SET UP FAC
5301 BF5000         1281         DST  1,@FAC+6             LENGTH OF STRING
5304 01
5305 10             1282         CONT
                    1283  *
                    1284  ********************************************************
                    1285  *
                    1286  *       ASC(A$) - Takes the numeric value of the first
                    1287  *            character in A$.
                    1288  *
                    1289  ********************************************************
                    1290  *
5306 0657A6         1291  ASC01  CALL COMB                 'ASC' CHECK FOR "("
5309 0EFF           1292         PARSE >FF                 GET THE VALUE
530B D64C65         1293         $IF  @FAC+2 .NE. >65 GOTO ERR6
530E 4D81
5310 8E516F         1294         $IF  @FAC+7 .EQ. 0 GOTO ILLARG
5313 74
5314 BC5C80         1295         ST   RAM(@FAC+4),@ARG     GET THE FIRST CHARACTER
5317 4E
5318 52CB           1296         BR   LENO2                USE COMMON CODE
                    1297  *      Jump always
```

```
                    1299  *
                    1300  ***************************************************************
                    1301  *
                    1302  *       STR$(X) - Takes as its input an integer X and
                    1303  *            converts it to its string representation.
                    1304  *            e.g.    3.1415 -> "3.1415"
                    1305  *       CONTROL - Turned over to STR$ by the parser
                    1306  *       USES - The usual temporaries used by string functio
                    1307  *            when it calls LITS06.  Uses the Roll-out
                    1308  *            AREA FOR A TEMPORARY STORAGE AREA WHEN
                    1309  *            allocating the result string.
                    1310  *       OUTPUT - FAC is set up in the usual manner for a
                    1311  *            string.
                    1312  ***************************************************************
                    1313  *
                    1314  STR$01
531A 0657A6         1315          CALL COMB                 Left paren?
531D 0EFF           1316          PARSE >FF                 Get the numeric
531F CA4C64         1317          CHE   >64,@FAC+2          Get a numeric?
5322 6D81           1318          BS    ERR6                * STRING-NUMBER MISMATCH
5324 8655           1319          CLR   @FAC+11             Select basic floating type
5326 060014         1320          CALL CNS                  CONVERT THE NUMBER TO STRIN
5329 D69055         1321          $IF *FAC+11 .EQ. SPACE THEN  If leading space
532C 205333
532F 9055           1322               INC @FAC+11          Suppress it out
5331 9256           1323               DEC @FAC+12          Shorten the length
                    1324          $END IF
5333 860C           1325          CLR   @BYTE               Prepare for 2-byte value
5335 BC0D56         1326          ST    @FAC+12,@BYTE+1     Get length of string
5338 340CA3         1327          MOVE @BYTE FROM *FAC+11 TO RAM(VROA$) Put the
533B C09055
                    1328  *                                 string in VDP
533E BF6603         1329          DST   VROA$,@TEMP5        Copy-from addr(For LITSTR).
5341 C0
5342 06514F         1330          CALL LITS06               Allocate and set up FAC
5345 BD500C         1331          DST   @BYTE,@FAC+6        Put in the length
5348 10             1332          CONT
                    1333  *
                    1334  ***************************************************************
                    1335  *
                    1336  *       VAL(A$) - Takes as its input a string, A$, and
                    1337  *            converts the string into a number if the strin
                    1338  *            is a valid representation of a number.
                    1339  *            e.g. - "3.1415" -> 3.1415
                    1340  *            VAL suppresses leading and trailing blanks fro
                    1341  *            the source string.
                    1342  *            VAL is effectively the inverse of STR$.
                    1343  *       CONTROL - from the parser
                    1344  *       OUTPUT - FAC CONTAINS THE FLOATION POINT NUMBER
                    1345  *
                    1346  ***************************************************************
                    1347  *
5349 0657A6         1348  VAL01   CALL COMB                 Left paren?
534C 0EFF           1349          PARSE >FF                 Get the string
534E D64C65         1350          CEQ   >65,@FAC+2          Get string back?
```

```
5351 4D81    1351        BR   ERR6                No - = STRING-NUMERIC MISMATCH
 353 8E516F  1352        $IF @FAC+7 .EQ. 0 GOTO ILLARG  Can't have null strin
5356 74
5357 BD664E  1353        DST  @FAC+4,@TEMP5       Ptr to string
535A A16650  1354        DADD @FAC+6,@TEMP5       Ptr to trailing length byte
535D 9366    1355        DDEC @TEMP5              Ptr to last byte of string
535F BD0C50  1356        DST  @FAC+6,@BYTE        COPY STRING LENGTH
5362 D6B066  1357        $WHIL  RAM(@TEMP5) .EQ. SPACE   While trailing blanks
5365 205371
5368 930C    135              DD C  @BYTE         Decrease length of string
536A 6F74    1359             BS    ILLARG        If all blanks
536C 9366    1360             DDEC  @TEMP5        Go back a character
536E 055362  1361        $END WHILE
5371 910C    1362        DINC @BYTE               Allow for terminator
5373 0F17    1363        XML  VPUSH               Save the ptr to the string
5375 06515C  1364        CALL GETSTR              Get a new string
5378 0F18    1365        XML  VPOP                Retrieve the ptr to the strin
537A BD664E  1366        DST  @FAC+4,@TEMP5       Get the ptr to the string
537D 8652    1367        CLR  @FAC+8              Force VDP mode
537F 065137  1368        CALL LITS09              COPY THE STRING AND SET UP FA
5382 A10C1C  1369        DADD @SREF,@BYTE         Point to the trailing length
5385 930C    1370        DD C @BYTE               Point at the last character
5387 BEB00C  1371        ST   SPACE,RAM(@BYTE)    Put in the terminator
538A 20
538B BD561C  1372        DST  @SREF,@FAC+12       Addr for the conversion
538E D6B056  1373        $WHILE  RAM(@FAC+12) .EQ. SPACE  While leading space
 391 205399
5394 9156    1374          DINC  @FAC+12          Skip leading blank
5396 05538E  1375        $END WHILE
5399 864C    1376        CLR  @FAC+2              Get rid of string (in case=0)
539B 8654    1377        CLR  @FAC+10             Assume no error
539D 0F10    1378        XML  CSNUM               Convert it
539F D5560C  1379        DCEQ @BYTE,@FAC+12       Convert all of it?
53A2 4F74    1380        BR   ILLARG              NO - ILLEGAL CHARACTER
53A4 8E5456  1381        $IF @FAC+10 .NE. 0 GOTO EXP005 If overflow warning
53A7 B5
53A8 10      1382        CONT
```

```
                    1384  *
                    1385  ***************************************************************
                    1386  *
                    1387  *       POS(A$,B$,X) - Attempts to match the string, B$, in
                    1388  *            in A$ beginning at character # X in A$.  If X
                    1389  *            is > len(A$), a match is not found or A$ is th
                    1390  *            null string then the returned value is 0.
                    1391  *            If B$ is the null string then the returned val
                    1392  *            IS 1.  OTHERWISE, THE RETURNED VALUE IS THE
                    1393  *            column # of the 1st character matched in A$.
                    1394  *       CONTROL - From the parser.  Returns through common
                    1395  *            code in LEN.
                    1396  *       USES - No temps - Utilizes FAC and ARG.
                    1397  *
                    1398  ***************************************************************
                    1399  *
                    1400  POS01
 53A9 0657A6        1401          CALL COMB                   MUST BE "("
 53AC 0F1B          1402          XML  PGMCH                  GET NEXT TOKEN
 53AE 0EB3          1403          PARSE COMMA$                GET SOURCE STRING
 53B0 D642B3        1404          $IF @CHAT .NE. COMMA$ GOTO ERR1
 53B3 5671
 53B5 0651E8        1405          CALL PUSSTR                 STACK THE STRING AND GET TOK
 53B8 0EB3          1406          PARSE COMMA$                GET MATCH STRING
 53BA D642B3        1407          $IF @CHAT .NE. COMMA$ GOTO ERR1
 53BD 5671
 53BF 0651E8        1408          CALL PUSSTR                 STACK THE STRING AND GET TOK
 53C2 0EB6          1409          PARSE RPAR$                 GET POSITION
 53C4 D642B6        1410          $IF @CHAT .NE. RPAR$ GOTO ERR1
 53C7 5671
 53C9 065740        1411          CALL INTARG                 CHECK AND CONVERT IT
 53CC 8F4A6D        1412          $IF @FAC .DEQ. 0 GOTO ERR5  VALUE OUT OF RANGE
 53CF 7C
 53D0 BD0C4A        1413          DST  @FAC,@BYTE             KEEP THE OFFSET
 53D3 930C          1414          DDEC @BYTE                  CORRECT FOR POSITION 0
 53D5 0F18          1415          XML VPOP                    GET MATCH STRING BACK
 53D7 350008        1416          MOVE 8 FROM @FAC TO @ARG    PUT MATCH IN ARG
 53DA 5C4A
 53DC 0F18          1417          XML  VPOP                   GET SOURCE BACK
 53DE 8E51          1418          CZ   @FAC+7                 SOURCE STRING NULL?
 53E0 741F          1419          BS   POS12                  NO MATCH POSSIBLE
                    1420  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    1421  *       CHANGED FROM:
                    1422  *       CGT  @BYTE+1,@FAC+7         OFFSET > LENGTH?
                    1423  *       BY STAN HUME ON 06/19/80 TO FIX POS OF STRINGS WITH
                    1424  *       LENGTH > 127 NOT WORKING PROPERLY
                    1425  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
 53E2 C4510D        1426          CH   @BYTE+1,@FAC+7          OFFSET > LENGTH?
 53E5 541F          1427          BR   POS12                  YES - NO MATCH POSSIBLE
 53E7 8E63          1428          CZ   @ARG+7                  MATCH STRING NULL?
 53E9 740E          1429          BS   POS06                  MATCH AT POSITION+1
 53EB A14E0C        1430          DADD @BYTE,@FAC+4           ADJUST PTR FOR OFFSET
 53EE A4510D        1431          SUB  @BYTE+1,@FAC+7         ADJUST LENGTH
                    1432  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    1433  *       CHANGED FROM:
```

```
                1434  *POS02  CGE  @ARG+7,@FAC+7     ENOUGH SPACE LEFT FOR A MATCH
                1435  *       BY STAN HUME ON 06/19/80 TO FIX POS OF STRINGS WITH +
                1436  *       A DIFFERENCE IN LENGTH > 127 NOT WORKING PROPERLY   +
                1437  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><+
53F1 C85163     1438  POS02   CHE  @ARG+7,@FAC+7     ENOUGH SPACE LEFT FOR A MATCH
53F4 541F       1439          BR   POS12             NO - NO MATCH POSSIBLE
53F6 BD4A4E     1440          DST  @FAC+4,@FAC       GET FIRST ARG
53F9 BD5C60     1441          DST  @ARG+4,@ARG       GET SECOND ARG
53FC BC6463     1442          ST    ARG+7,@ARG+8     AND LENGTH OF SECOND
53FF D4B05C     1443  POS04   CEQ  RAM(@FAC),RAM(@ARG)  COMPARE THE CHARACTERS
5402 B04A
5404 5417       1444          BR   POS10             DIDN'T MATCH
5406 914A       1445          DINC @FAC              NEXT IN SOURCE
5408 915C       1446          DINC @ARG              NEXT IN MATCH
540A 9264       1447          DEC  @ARG+             REACHED END OF MATCH?
540C 53FF       1448          BR   POS04             NOT YET-SO LOOP
540E 900D       1449  POS06   INC  @BYTE+1          MATCHED! CORRECT FOR 1 INDEXIN
5410 BC5C0D     1450  POS08   ST   @BYTE+1,@ARG     CHARACTER POSITION OF MATCH
5413 0F1B       1451          XML  PGMCH             GET NEXT TOKEN
5415 52CB       1452          BR   LEN02             CONVERT TO FLOATING
                1453  *       NOTE: UTILIZES THE SAME CODE AS LEN
                1454  *       LEN WILL FINISH UP AND DO A 'CONT'
                1455  *
5417 900D       1456  POS10   INC  @BYTE+1          STEP INDEX OF MATCH CHARACTER
5419 9251       1457          DEC  @FAC+7            MOVE 1 POSITION DOWN 1ST
541B 914E       1458          DINC @FAC+4               ARGUMENT
541D 53F1       1459          BR   POS02             TRY TO MATCH AGAIN
                1460  *
541F 860D       1461  POS12   CLR  @BYTE+1          NO MATCH POSSIBLE
5421 5410       1462          BR   POS08
```

```
                    1464  *
                    1465  ************************************************************
                    1466  *
                    1467  *       COPYST - IS AN OPTIMAL COPY PROGRAM THAT IS
                    1468  *                USED BY COMPCT.  IT COPIES STRINGS
                    1469  *                FROM LOWER MEMORY TO HIGHER MEMORY
                    1470  *                IN THE LARGEST HUNKS POSSIBLE W/O
                    1471  *                WIPING OUT ANYTHING.
                    1472  *       INPUT: TEMP1- # OF BYTES TO BE MOVED(# IN STRING)
                    1473  *       OUTPUT: MOVED STRING-UPDATED POINTERS
                    1474  *       USES:
                    1475  *            BKPTR  EQUATED TO FAC+8
                    1476  *            COPYTO EQUATED TO FAC+10
                    1477  *            TEMP1  EQUATED TO FAC+12
                    1478  *            TEMP2  EQUATED TO FAC+14
                    1479  *            TEMP3  EQUATED TO FAC+16
                    1480  *       Note:  See comment in COMPCT.
                    1481  *
                    1482  ************************************************************
5423 A35600         1483  COPYST DADD 4,@TEMP1           NEED TO MOVE LENGTHS & BKPTR
5426 04
5427 BD5854         1484  COPYS2 DST  @COPYTO,@TEMP2     GET FIRST FREE BYTE
542A A55852         1485         DSUB @BKPTR,@TEMP2      CALCULATE SIZE OF FREE BLOCK
542D A55856         1486         DSUB @TEMP1,@TEMP2      SIZE OF FREE BLOCK
                    1487  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    1488  *  INSERT FOLLOWING TWO LINES FOR 99/4A TO FIX GARBAGE
                    1489  *  COLLECTION LOCKUP BY ALLEN ACREE 2/28/81
                    1490  *  DINC @TEMP2
                    1491  *  BS COPYS3
                    1492  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
5430 9158           1493         DINC @TEMP2              OFFSET ONE
                    1494  *  NO NEED TO MOVE STRING TO ITSELF IN THE FOLLOWING CASE
5432 7455           1495         BS  COPYS3               ZERO SIZE FREE BLOCK
5434 C95856         1496         DCHE @TEMP1,@TEMP2       ENOUGH SPACE TO MOVE THE RES
5437 543C           1497         BR   COPYS1              NO-MOVE A HUNK AT A TIME
5439 BD5856         1498         DST  @TEMP1,@TEMP2       TOTAL TO BE MOVED
543C BD5A52         1499  COPYS1 DST  @BKPTR,@TEMP3       PTR TO BEGINNING OF STRING
543F A15A56         1500         DADD @TEMP1,@TEMP3       PTR TO END OF STRING
5442 A55A58         1501         DSUB @TEMP2,@TEMP3       PTR TO FIRST BYTE TO BE MOVE
5445 A55458         1502         DSUB @TEMP2,@COPYTO      PLACE TO COPY TO
5448 3458E0         1503         MOVE @TEMP2 FROM RAM(@TEMP3) TO RAM(1(COPYTO))
544B 0154B0
544E 5A
544F A55658         1504         DSUB @TEMP2,@TEMP1       REMAINING TO BE MOVED
                    1505  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    1506  *       CHANGED FROM:
                    1507  *       DCZ  @TEMP1              ALL MOVED?
                    1508  *       BY STAN HUME ON 06/19/80 TO NOTHING TO SAVE 2 BYTES
                    1509  *       SINCE THE DSUB ABOVE SETS CONDITION ON ZERO
                    1510  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
5452 5427           1511         BR   COPYS2            NOPE-DO THE REST TOO
5454 00             1512         RTN                     YEP-ALL DONE
                    1513  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    1514  *  INSERT FOLLOWING TWO LINES FOR 99/4A TO FIX GARBAGE
                    1515  *  COLLECTION LOCKUP BY ALLEN ACREE 2/28/81
```

```
                *COPYS3 DSUB @TEMP1,@COPYTO
                *       RTN
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
5455 A55456     COPYS3  DSUB @TEMP1,@COPYTO
5458 00                 RTN
```

```
                    1522  *
                    1523  *
                    1524  ***************************************************
                    1525  **      SUBROUTINE FOR 'TRACE'
                    1526  ***************************************************
5459 B68088         1527  NTRACE SB    @FLAG,4        'TRACE'
545C 10
545D 10             1528         CONT                 CONTINUE PARSE
                    1529  *
                    1530  *
                    1531  *
                    1532  ***************************************************
                    1533  **      SUBROUTINE FOR 'UNTRACE'
                    1534  ***************************************************
545E B28088         1535  NUNTRC RB    @FLAG,4        'UNTRACE'
5461 EF
5462 10             1536         CONT                 CONTINUE PARSE
                    1537  *
                    1538  *
                    1539  *
                    1540  ***************************************************
                    1541  **      SUBROUTINE FOR 'BREAK' AND 'UNBREAK'
                    1542  ***************************************************
5463 BE00FF         1543  NBREAK ST    >FF,@VAR0      'BREAK' FLAG
5466 8E42           1544         CZ    @CHAT          LINE NUMBER?
5468 547F           1545         BR    LINEGP         YES-LOOK FOR IT(THEM)
546A 8F44           1546         DCZ   @BUFFY         IMPERATIVE W/NO LINE #?
546C 766C           1547         BS    ERR            YES-THATS A NO-NO
546E BDA3EC         1548         DST   @EXTRAM,RAM(SAVEPC)   SAVE PC
5471 2E
5472 A7A3EC         1549         DSUB  4,RAM(SAVEPC) MUST LEAVE EXTRAM INTACT FOR L.N
5475 0004
                    1550  *                          BUT ADVANCE SAVEPC TO SKIP 'BREAK
5477 4E3C           1551         BR    EXEC6B         NO-BREAK ON THIS LINE
                    1552  *
5479 8600           1553  NUNBRK CLR   @VAR0          'UNBREAK' FLAG
547B 8E42           1554         CZ    @CHAT          LINE NUMBER?
547D 74BC           1555         BS    UNBK01         NO-CLEAR ALL BREAKPTS
                    1556  LINEGP
547F 0656BB         1557         CALL  LINE           GET LINE #
5482 BD1232         1558         DST   @ENLN,@BEE
5485 A71200         1559         DSUB  >03,@BEE       1ST LINE #
5488 03
5489 C91230         1560  LNGP1  $IF @BEE .DL. @STLN GOTO LNGP3  IF LINE NOT FOUND
548C 54B5
548E D5B012         1561         $IF RAM(@BEE) .DEQ. @FAC GOTO LNGP2  IF LINE FOUND
5491 4A749A
5494 A71200         1562         DSUB  >04,@BEE       NEXT LINE ON RAM
5497 04
5498 5489           1563         BR    LNGP1
                    1564  *      Jump always
549A 9512           1565  LNGP2  DINCT @BEE           WHERE 1ST TOKEN'ADD. IS STORED
                    1566  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    1567  *      CHANGED FROM:
                    1568  *      $IF @VAR0 .EQ. 0 THEN
```

```
                    1569  *         RB    RAM(@BEE),7   'UNBREAK'
                    1570  *       $SELSE
                    1571  *         SB    RAM(@BEE),7   'BREAK'
                    1572  *       $END IF
                    1573  *       BY STAN HUME ON 06/23/80 TO SAVE 2 BYTES
                    1574  *oooooooooooooooooooooooooooooooooooooooooooooooo
549C B2B012         1575          RB    RAM(@BEE),7         Assume removing breakpoint
549F 7F
54A0   0074         1576          $IF @VAR0 .NE. 0 THEN     But if setting a breakpoint
54A3 A
54A4 B6B012         1577            SB    RAM(@BEE),7       Set the breakpoint bit
54A7 80
                    1578          $END IF
54A8 8E4274         1579  LNGP2B  $IF   @CHAT .EQ.  0 GOTO LNGP4
54AB CE
54AC D642B3         1580          $IF   @CHAT .NE.  COMMA$ GOTO ERR1
54AF 5671
54B1 0F1B           1581          XML   PGMCH           GET NEXT CHARACTER
54B3 547F           1582          BR    LINEGP
                    1583  *       Jump always
54B5 06284C         1584  LNGP3   CALL  WARN$$
54B8 20D9           1585          DATA  #MSGBLN         'LINE NOT FOUND'
54BA 54A8           1586          BR    LNGP2B
                    1587  *       Jump always
                    1588  *
                    1589  ********CLEAR ALL BREAKPOINTS******
 54BC BD1230        1590  UNBK01  DST   @STLN,@BEE          END OF LINE # BUFFER
 54BF 9512          1591          DINCT @BEE                PT AT THE LINE PTR
54C1 B2B012         1592  UNBK02  RB    RAM(@BEE),7         CLR BKPT ON THIS LINE
54C4 7F
54C5 A31200         1593          DADD  4,@BEE              GOTO NEXT LINE
54C8 04
54C9 C51232         1594          DCH   @ENLN,@BEE          FINISHED?
54CC 54C1           1595          BR    UNBK02              NOPE-GO ON
54CE 10             1596  LNGP4   CONT                      YES-CONTINUE
```

```
                    1598  ********************************************************
                    1599  *
                    1600  *       USER-DEFINED FUNCTIONS
                    1601  *
                    1602  *       Subroutine to store away the information of the
                    1603  *       tokens in a function reference, go into the  'DEF'
                    1604  *       statement, calculate the value of the expression
                    1605  *       and then resume execution of the user's program
                    1606  *       after the reference.
                    1607  *
                    1608  *       An entry in the FAC and on the stack for a functior
                    1609  *       reference looks like:
                    1610  *
                    1611  *       ! old VAR81 ! >68 ! string/numeric flag !
                    1612  *       ^             ^     ^                   ^
                    1613  *       FAC       FAC+2  FAC+3                FAC+4
                    1614  *
                    1615  *                                 ! old SYMTAB ! old SYMPTR
                    1616  *                                 ^            ^
                    1617  *                                 FAC+4        FAC+6
                    1618  *       The 'old VAR81' is where execution resumes after
                    1619  *       evaluating the function.   String(80)/numeric(00)
                    1620  *       flag is function type.  Old SYMTAB is the old symbc
                    1621  *       table pointer and old SYMPTR is the old free space
                    1622  *       pointer.  These are restored after the function is
                    1623  *       evaluated.
                    1624  ********************************************************
54CF 8651           1625  UDF     CLR  @FAC+7             Assume no args
54D1 8722           1626          DCLR @ERRCOD            Clear the error code for cor
54D3 865E           1627          CLR  @ARG+2             Safety for VPUSH
54D5 864C           1628          CLR  @FAC+2             Safety for VPUSH
54D7 D642B7         1629          $IF  @CHAT .EQ. LPAR$   THEN
54DA 54ED
54DC 0F17           1630            XML  VPUSH            Save ptr to func definition
54DE 0F1B           1631            XML  PGMCH            Skip the lpar
54E0 0EB6           1632            PARSE RPAR$           PARSE to get arg value
54E2 0F1B           1633            XML  PGMCH            Skip the rpar
54E4 350008         1634            MOVE 8 BYTES FROM @FAC TO @ARG  Save PARSE resul
54E7 5C4A
54E9 0F18           1635            XML   VPOP            Get S.T. ptr to func def
54EB 9051           1636            INC   @FAC+7          Indicate that we have an ar
                    1637          $END IF
                    1638  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    1639  *       CHANGED FROM:
                    1640  *          ST @FAC+7,@FAC+10
                    1641  *          ST @FAC,@FAC+8
                    1642  *       BY STAN HUME 9/6/79     TO FIX CONFLICT W/COMPCT
                    1643  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
54ED BC6651         1644          ST   @FAC+7,@TEMP5      Move the parm count
54F0 BD644A         1645          DST  @FAC,@TEMP4        S.T. ptr to definition
54F3 0F17           1646          XML  VPUSH              Allow room for UDF entry
54F5 350008         1647          MOVE 8 FROM @ARG TO @FAC  Retrieve parse result
54F8 4A5C
54FA 0F17           1648          XML  VPUSH              Save parse result
                    1649  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
```

```
                1650  *      CHANG D FROM:
                1651  *         ST  RAM(@FAC+ ),@FAC+2
                1652  *      BY STAN HUME 9/6/79    TO FIX CONFLICT W/COMPCT
                1653  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
54FC BC4CB0     1654         ST   RAM(@TEMP4),@FAC+2  Get S.T. declarations
54FF 64
5500 BC4D4C     1655         ST   @FAC+2,@FAC+3      Do this to save string bit
                1656  *
                1657  *      NOT : THIS IS TO ALLOW THE CHECKING AFTER THE
                1658  *      FUNCTION HAS BEEN EVALUATED TO MAKE SURE THE
                1659  *      FUNCTION TYPE(STRING/NUM) MATCHES THE RESULT IT
                1660  *      PRODUCES.
                1661  *
5503 B24C07     1662         AND  >07,@FAC+2         Mask all but # of parms
                1663  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                1664  *      CHANGED FROM:
                1665  *         $IF @FAC+2 .NE. @FAC+10 GOTO ERR1
                1666  *      BY STAN HUME 9/6/79    TO FIX CONFLICT W/COMPCT
                1667  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
5506 D44C66     1668         $IF  @FAC+2 .NE. @TEMP5 GOTO ERR1
5509 5671
550B BD4A2C     1669         DST  @VAR81,@FAC        Will resume execution here
550E 864C       1670         CLR  @FAC+2             Dummy ID on the stack
                1671  *                              Until S.T. entry is formed
5510 B24D80     1672         AND  >80,@FAC+3         Mask all but string bit
5513 A76E00     1673         DSUB 16,@VSPTR          Get below parse result
5516 10
5517 BD4E3E     1674         DST  @SYMTAB,@FAC+4     Save current symbol table ptr
551A BD5040     1675         DST  @SYMPTR,@FAC+6     Save current free space ptr
551D 0F17       1676         XML  VPUSH              Save the return info
551F A36E00     1677         DADD 8,@VSPTR           Get back to parse result
5522 08
                1678  *
                1679  *******SHIFT EXECUTION TO FUNCTION DEFINITION
                1680  *
                1681  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                1682  *      CHANGED FROM:
                1683  *         DST  RAM(6(FAC+8)),@VAR81
                1684  *      BY STAN HUME 9/6/79    TO FIX CONFLICT W/COMPCT
                1685  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
5523 BD2CE0     1686         DST  RAM(6(TEMP4)),@VAR81   Set text ptr to def
5526 0664
5528 0F1B       1687         XML  PGMCH              Get 1st char in the def
552A C673AC     1688         CH   >AC,@SUBSTK        Enough room on subr stack?
552D 767D       1689         BS   ERR3               Nope-Is recursive eg.F(X)=F(X
552F 350018     1690         MOVE 24 FROM @00 TO RAM(VROA$)  Roll out temporaries
5532 A3C000
5535 B68088     1691         SB   @FLAG,3            Set function flag for ENTER
5538 08
5539 BE1680     1692         ST   >80,@VAR9          Make calls look like 'ENTERX'
553C D642BE     1693         $IF  @CHAT .EQ. EQUAL$  THEN
553F 554E
                1694  *
                1695  *      Note: This is to keep the global/local variables
                1696  *      correct in the event that a function uses another
```

```
                1697  *       function in its evaluation.
                1698  *
 5541 8659      1699            CLR  @FAC+15            Create a dummy entry in tabl
 5543 06284A    1700            CALL ENT09              for no-parm function
 5546 972C      1701            DDECT @VAR81            Back up to equal sign
 5548 86E002    1702            CLR  RAM(2(VSPTR))      This is to keep ASSGNV(calle
 554B 6E
                1703  *             below) not to screw up in case FAC+2 happens t
                1704  *             have a value .gt. >65
 554C 5551      1705          $SELSE
 554E 062848    1706            CALL ENTER              Enter the parameter
                1707          $SEND
 5551 0F1B      1708          XML  PGMCH                Get the '=' (Checked in PSCAI
 5553 B28088    1709          RB   @FLAG,3              Reset to normal ENTERs
 5556 F7
 5557 350018    1710          MOVE 24 FROM RAM(VROA$) TO @00  Roll in temporaries
 555A 00A3C0
 555D BEEFFF    1711          ST   >68,RAM(-6(VSPTR))     Correct stack entry ID
 5560 FA6E68
 5563 BDE002    1712          DST  RAM(SYMBOL),RAM(2(SYMTAB))     Fudge link to
 5566 3EA3E0
                1713  *                                   get global values
 5569 BD4A3E    1714          DST  @SYMTAB,@FAC         Set up for SMB
 556C 0F14      1715          XML  SMB                  Get value space
 556E 350008    1716          MOVE 8 FROM @FAC TO @FAC+8     Destination
 5571 524A
 5573 0F18      1717          XML  VPOP                 Get arg back
 5575 350008    1718          MOVE 8 FROM @FAC TO @ARG  Argument value
 5578 5C4A
 557A 350008    1719          MOVE 8 FROM @FAC+8 TO @FAC  Destination
 557D 4A52
 557F 0F17      1720          XML  VPUSH                Push the destination
 5581 350008    1721          MOVE 8 FROM @ARG TO @FAC  Argument value
 5584 4A5C
 5586 D64C65    1722          $IF @FAC+2 .EQ. >65 THEN  If a string
 5589 5595
 558B D74A00    1723            $IF @FAC .NOT. .DEQ. SREF  THEN  If not temp
 558E 1C7595
 5591 BD4EB0    1724               DST  RAM(@FAC),@FAC+4  Get new loc. of string
 5594 4A
                1725            $END IF             String probably moved when
                1726          $END IF               Parameter was allocated in S.T.
 5595 0F1B      1727          XML  PGMCH              Skip the '='
 5597 0F15      1728          XML  ASSGNV             Assign the value to the parameter
                1729  *
 5599 0E00      1730          PARSE 00                PARSE to end of function definiti
                1731  *
                1732  *******CHECK FOR TYPE MATCH (STRING/STRING OR NUM/NUM)
                1733  *******BETWEEN THE RESULT AND THE FUNCTION TYPE
                1734  *
 559B D64C65    1735          $IF  @FAC+2 .EQ. >65 THEN  If result string
 559E 55A8
 55A0 8EE003    1736            $IF RAM(3(VSPTR)) .EQ. 0 GOTO ERR6   If functio.
 55A3 6E6D81
 55A6 55AE      1737          $SELSE                      if result not string
```

```
 5A8 8EE003  1738          $IF RAM(3(VSPTR)) .NE. 0 GOTO ERR6   If function i
 5AB 6E4D81
             1739          $SEND
             1740  *
             1741  *******NOW RESTORE SYMBOL TABLE AND
             1742  *******RESUME EXECUTION AT THE ORIGINAL LINE
             1743  *
 55AE 0655BB 1744          CALL D LINK                    Delink the parameter entry
 55B1 BD2CE0 1745          DST  RAM(8(VSPTR)),@VAR81      Manual pop to get ptr back
 55B4 086E
 55B6 932C   1746          DDEC @VAR81                    Back up text pointer
 55B8 0F1B   1747          XML  PGMCH                     Get next token
 55BA 10     1748          CONT                           And continue
```

```
                DELINK
55BB BD663E              DST   @SYMTAB,@TEMP5     Save addr of S.T. entry just
                *             in case entry is a string(Must free the string
55BE 350004              MOVE 4 FROM RAM(4(VSPTR)) TO @SYMTAB  Restore old
55C1 3EE004
55C4 6E
                *        symbol table ptr and free space pointer
                *        This handles the freeing of the string value
                *        which was assigned to the parameter.
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                *        CHANGED FROM:
                *        $IF RAM(@TEMP5) .NE. 0 THEN    If string parm
                *        BY STAN HUME ON 06/19/80 TO INSURE THAT ONLY THE
                *        STRING BIT IN THE SYMBOL TABLE HEADER IS CHECKED
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
55C5 D23066              $IF RAM(@TEMP5) .LT. 0 THEN  If string parameter
55C8 0075FB
55CB BD66E0                DST  RAM(6(TEMP5)),@TEMP5   Where the string is
55CE 0666
55D0 8F6675                $IF @TEMP5 .DNE. 0 THEN      If non-null string
55D3 E4
55D4 BD56EF                  DST  RAM(-3(TEMP5)),@TEMP1  Get backpointer
55D7 FFFD66
55DA C9563E                  $IF @TEMP1 .DL. @SYMTAB THEN  If not used
55DD 75E4
55DF 87EFFF                    DCLR RAM(-3(TEMP5))    Free up the string
55E2 FD66
                             $END IF
                           $END IF
                *        This handles the special case of F$(X$)=X$
                *        The result, which was permanent, must be made a tem
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
                *        THE FOLLOWING LINE (WITH SUPPORTING $END IF BELOW)
                *        ADDED BY STAN HUME ON 06/19/80 TO INSURE THAT THE
                *        RESULT IS REALLY A STRING SO THAT 'DEF F(X$)=0'
                *        WILL RETURN 0 AS THE RESULT INSTEAD OF SOME OTHER
                *        VALUE BECAUSE THIS CODE ASSUMED A STRING RESULT
                *        WHEN A STRING PARAMETER WAS USED.
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
55E4 D64C65               $IF  @FAC+2 .EQ. >65 THEN   If string result
55E7 55FB
55E9 C94A3E                 $IF @FAC .DL. @SYMTAB THEN  If came from argument
55EC 75FB
55EE 8F4E75                  $IF @FAC+4 .DNE. 0 THEN If non-null
55F1 F7
55F2 87EFFF                   DCLR RAM(-3(FAC+4))   Clear the backpointer
55F5 FD4E
                             $END IF
55F7 BF4A00                  DST  SREF,@FAC        Make it a temp
55FA 1C
                            $END IF
                          $END IF
                         $END IF
55FB A76E00              DSUB 8,@VSPTR            Trash the stack entry
55FE 08
```

```
;FF 00      1791          RTN                    And return
```

```
                    ATTNUT
5600 0EB6                  PARSE RPAR$
5602 064F79                CALL  CKSTNM          CHECK FOR NUMERIC OR STRING
5605 BF1000                DST   30,@VAR5
5608 1E
5609 063785                CALL  RANGE           0-30
560C E64B01                SRL   @FAC+1,1        0,1:0000      ATTENUATION/2
                    *                            2,3:0001
                    *                            4,5:0010
                    *                            6,7:0011      ETC.
560F B64BF0                OR    >F0,@FAC+1      REGISTER BITS
5612 00                    RTN
                    *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                    *      THE FOLLOWING ROUTINE REMOVED BY STAN HUME ON
                    *      06/23/80 BECAUSE VPOP TAKES CARE OF FREEING STRINGS
                    *      WHICH HAVE BEEN POPPED OFF OF THE STACK
                    ***************************************************************
                    *
                    *      FRESTR - FREES A TEMPORARY STRING THAT IS NO LONGER
                    *               NEEDED.  IF IT IS CALLED WITH A LINK
                    *               THROUGH THE SYMBOL TABLE NOTHING
                    *               IS DONE.
                    *
                    *      INPUT: PTR TO THE STRING IN @TEMP4
                    *      OUTPUT: PTR TO THE STRING AND/OR A ZEROED BACKPTR
                    *      USES: TEMP1 AS A TEMPORARY
                    *           Note:  See comment in COMPCT
                    ***************************************************************
                    *FRESTR
                    *      DCZ   @TEMP5               NULL STRING?
                    *      BS    FRES04               YEP-NOTHING TO DO
                    *      DST   RAM(-3(TEMP5)),@TEMP1   PICK UP THE BACK PTR
                    *      DCH   @TEMP1,@SYMTAB       POINT INTO THE SYMBOL TABLE?
                    *      BR    FRES04               YES - NOTHING TO DO
                    *      DCLR  RAM(-3(TEMP5))       ZERO THE BACKPTR
                    *FRES04 RTN                        DONE
                    *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
```

```
                1830  ********************************************************
                1831  *       SUBROUTINE TO S T POINTER TO EACH DATUM
                1832  ********************************************************
                1833  DATAST
5613 9336       1834    DDEC @LNBUF             POINT TO LENGTH ADDRESS
5615 8E8089     1835    $IF @GROMFL .EQ. 0 THEN
5618 5623
561A BD34B0     1 36     DST  RAM(@LNBUF),@DATA 1ST TOKEN ADD.
561D 36
561E B2347F     1837     RB @DATA,7
5621 562A       1838    $SELSE
5623 330002     1839     MOVE 2 FROM ROM(@LNBUF) TO @DATA
5626 340000
5629 36
                1840    $SEND IF
562A BC4D80     1841    ST   @GROMFL,@FAC+3
562D 89
562E 064010     1842    CALL GETDAT             GET A GRAM/GROM BYTE
5631 D60193     1843    $IF @VAR0+1 .NE. DATA$ THEN
5634 7644
5636 9736       1844      DDECT @LNBUF          NO
5638 D53630     1845      $IF @LNBUF .DNE. @STLN THEN
563B 7641
563D 9336       1846        DDEC @LNBUF       POINT TO 1ST TOKEN ADD.
563F 5613       1847        BR   DATAST
                1848      $END IF
 ,41 BE34FF     1849      ST   >FF,@DATA        INDICATE NO DATA
                1850    $END IF
5644 00         1851    RTN
                1852  *
                1853  *
                1854  *
                1855  ********************************************************
                1856  **      SUBROUTINE TO CONVERT LINE # INTO ASCII CODE
                1857  ********************************************************
                185   ASC
5645 8E8089     1859    $IF @GROMFL .EQ. 0 THEN
5648 5652
564A BD5EEF     1860     DST  RAM(-2(EXTRAM)),@ARG+2 GET LINE # IN BE
564D FFFE2E
5650 5659       1861    $SELSE
5652 330002     1862     MOVE 2 FROM ROM(-2(EXTRAM)) TO @ARG+2
5655 5EFFFE
5658 2E
                1863    $SEND IF
5659 052842     1864    B    DISO
                1865  *
                1866  *
                1867  *
                1868  ********************************************************
                1869  **      SUBROUTINE TO PRINT OUT ERROR MESSAGE
                1870  ********************************************************
  5C 8A22       1871  ERROR$ CASE @ERRCOD       DECODE ERROR FROM INTERPRETER
565E 5671       1872          BR   ERR1         SYNTAX
5660 567D       1873          BR   ERR3         OUT OF MEMORY
```

```
5662 4D7C  1874          BR    ERR5          BAD INDEX
5664 5682  1875          BR    ERR4          LINE NOT FOUND
5666 566C  1876          BR    ERR           EXECUTION ERROR
5668 5678  1877          BR    ERR2          BAD SUBSCRIPT
566A 4D81  1878          BR    ERR6          STRING-NUMBER MISMATCH
           1879  *
           1880  *
566C 06284E 1881  ERR     CALL  ERR$$         ERROR
566F 20BD  18 2          DATA  #MSG2         'CAN'T DO THAT'
           1883  *
           1884  RUN4
5671 06284E 1885  ERR1    CALL  ERR$$         ERROR
5674 202C  1886          DATA  #MSG1         'ILLEGAL STATEMENT'
5676 4E58  1887          BR    EXEC6D        EXIT FOR RUN4 ENTRY
           1888  *
5678 06284E 1889  ERR2    CALL  ERR$$         ERROR
567B 20A1  1890          DATA  #MSG13        'BAD SUBSCRIPT'
           1891  *
567D 06284E 1892  ERR3    CALL  ERR$$         ERROR
5680 2049  1893          DATA  #MSG5         'MEMORY FULL'
           1894  *
           1895  RUN3
5682 06284E 1896  ERR4    CALL  ERR$$         ERROR
5685 20D9  1897          DATA  #MSGBLN       'BAD LINE NUMBER'
5687 4E58  1898          BR    EXEC6D        EXIT FOR RUN3 ENTRY
           1899  *
5689 C673B0 1900  WARNG$  $IF   @SUBSTK .H. >3E-14 GOTO ERR3
568C 767D
           1901  *                           ALLOW ROOM ON STACK FOR WARNING C
568E 06284C 1902          CALL  WARN$$        ONLY ONE WARNING MSG FROM BASIC S
5691 206E  1903          DATA  #MSG8         'NUMERIC OVERFLOW'
5693 8722  1904          DCLR  @ERRCOD
5695 12    1905          DATA  >12           RTNB INSTRUCTION
```

```
                1907  ************************************************************
                1908  **      SUBROUTINES TO HANDLE 'LED'
                1909  ************************************************************
                1910  EXPON
5696 8722       1911          DCLR @ERRCOD
5698 064F7F     1912          CALL STKCHK         CHECK FOR ROOM ON STACKS
569B C64C63     1913          $IF  @FAC+2 .H. >63 GOTO ERR6     NEITHER OPERAND CAN
569E 6D81
56A0 C6E002     1914          $IF  RAM(2(V PTR)) .H. >63 GOTO ERR6      BE STRING!
56A3 6E636D
56A6 81
56A7 8654       1915          CLR  @FAC+10        CLEAR ERROR FLAG
56A9 060024     1916          CALL PWR
56AC 8E54       1917          CZ   @FAC+10        ANY ERROR OR WARNING?
56AE 76BA       1918          BS   EXP010         NO
56B0 D65401     1919          CEQ  1,@FAC+10      OVERFLOW WARNING?
56B3 4D7C       1920          BR   ERR5           NO, "BAD VALUE"
                1921  CA2
56B5 06284C     1922  EXP005  CALL WARN$$         OVERFLOW WARNING
56B8 206E       1923          DATA #MSG           '* WARNING-
                1924  *                              OVERFLOW'
56BA 10         1925  EXP010  CONT
                1926  *
                1927  *
                1928  ************************************************************
                1929  **      SUBROUTINE TO GET LINE # FOLLOWING
                1930  **      'BREAK', 'UNBREAK', 'RESTORE'.
                1931  ************************************************************
                1932  LINE
56BB D642C9     1933    $IF @CHAT .NE. LN$ GOTO ERR1   SHOULD BE LINE # REFERENCE
56BE 5671
56C0 0F1B       1934    XML  PGMCH              GET HIGH ORDER LINE #
56C2 BC4A42     1935    ST   @CHAT,@FAC         BUILD RESULT IN FAC,FAC+1
56C5 0F1B       1936    XML  PGMCH
56C7 BC4842     1937    ST   @CHAT,@FAC+1       LOW ORDER LINE #
56CA 0F1B       1938    XML  PGMCH              GET TOKEN FOLLOWING LINE #
56CC 00         1939    RTN
                1940  *
                1941  *
                1942  *
                1943  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                1944  *  Replace following lines for 99/4A without equation calc.
                1945  *  By CAROLYN LEE 2/26/81
                1946  *SCROLL
                1947  *SCROL  $IF @CALCFL .EQ. 0 THEN
                1948  *        MOVE 736 FROM DISPLAY(X=0,Y=1) TO DISPLAY(X=0,Y=0)
                1949  *       $SELSE
                1950  *        MOVE 160 FROM DISPLAY(X=0,Y=19) TO DISPLAY(X=0,Y=18
                1951  *       $END IF
                1952  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                1953  SCROLL
                1954  SCROL
 56CD 3502E0    1955          MOVE 736 FROM DISPLAY(X=0,Y=1) TO DISPLAY(X=0,Y=0)
 56D0 A000A0
 56D3 20
```

```
                1956  SCR010 FMT   XPT=0,YPT=Y1,2'EDGECH',28'SPACE+OFFSET';
 56D4 08FF00    1957                2'EDGECH'
 56D7 FE1741
 56DA 7F5B80
 56DD 417FFB
 56E0 00        1958         RTN
                1959  *
 56E1 8654      1960  CONV1  CLR   @FAC+10
 56E3 0F10      1961         FLTPT CSNUM            VALUE IN 8 BYTES
 56E5 8E54      1962         CZ    @FAC+10      ANY ERROR OR WARNING?
 56E7 76EE      1963         BS    CONV2        NO
 56E9 06284C    1964         CALL  WARN$$       WARNING
 56EC 206E      1965         DATA  #MSG8        '* WARNING-
 56EE 00        1966  CONV2  RTN                   STRING TO NUMBER OVERFLOW'
```

```
                1968  ********************************************************
                1969  **      SUBROUTINE FOR 'GET'
                1970  ********************************************************
56EF 06378E     1971  GCHAR   CALL GPHV           GET X,Y VALUES.
56F2 06578D     1972          CALL NUMVAR         GET A VARIABLE.
56F5 310008     1973          MOV  8 FROM RCM(#FLT1) TO @FAC CLEAR BUFFER
56F8 4A50C0
56FB BC4B7D     1974          ST   CB,@FAC+1      CHARACTER
56FE A64B60     1975          S    OFFS T,@FAC+1
5701 CA4B64     1976          $IF  @FAC+1 .HE. 100 THEN
5704 570E
5706 C04C4B     1977            EX   @FAC+1,@FAC+2
5709 AE4B64     1978            DIV  100,@FAC+1
570C 904A       1979            INC  @FAC
                1980          $END IF
570E 0F15       1981          XML  ASSGNV                    SYMBOL TABLE
5710 05361D     1982          B    XPTRTN
                1983  *
                1984  *
                1985  ********************************************************
                1986  **      SUBROUTINE FOR 'COLOR'
                1987  ********************************************************
5713 063767     1988  COLOR   CALL THREEC         GET SET VALUE
5716 063779     1989          CALL RAN16          ERROR IF > 16 OR < 1
5719 A34A03     1990          DADD >30F,@FAC      COLOR TABLE ADDRESS (>304 - >31F)
571C 0F
                1991  *
571D 0F17       1992          XML  VPUSH
571F 06376F     1993          CALL SUPY1          GET FORGROUOND COLOR
5722 063779     1994          CALL RAN16          OUT OF RANGE? (1 - 16)
5725 924B       1995          DEC  @FAC+1         INTERNAL RANGE: 0 - 15
5727 BC0E4B     1996          ST   @FAC+1,@VAR4
572A E20E04     1997          SLL  @VAR4,4        FORGROUND COLOR IN LEFT 4 BITS
                1998  *
572D 0EB6       1999          PARSE RPAR$         GET BACKGROUND COLOR
572F 063779     2000          CALL RAN16          OUT OF RANGE? (1 - 16)
5732 924B       2001          DEC  @FAC+1         INTERNAL RANGE: 0 - 15
5734 B40E4B     2002          OR   @FAC+1,@VAR4 BACKGROUND COLOR IN RIGHT 4 BITS
                2003  *
5737 0F18       2004          FLTPT VPOP          GET COLOR TABLE ADDRESS
5739 BCB04A     2005          ST   @VAR4,RAM(@FAC) LOAD THE COLOR.
573C 0E
573D 053620     2006          B    LNKRTN
```

```
                    2008  ********************************************************
                    2009  *
                    2010  *       INTARG - Insures that the value in FAC is a
                    2011  *            numeric, converts it to integer, issues
                    2012  *            error messages if necessary or returns.
                    2013  *
                    2014  ********************************************************
                    2015  FTINT
 5740 C64C63        2016  INTARG CH    >63,@FAC+2          IS ARG A NUMERIC?
 5743 6D81          2017         BS    ERR6                * STRING-NUMBER MISMATCH
 5745 8654          2018         CLR   @FAC+10       ASSUME NO ERROR OR WARNING
 5747 876C          2019         DCLR  @FPERAD
 5749 0F12          2020         FLTPT FLTINT
 574B 8E54          2021         CZ    @FAC+10       ANY ERROR OR WARNING?
 574D 4D7C          2022         BR    ERR5          ERROR, 'VALUE OUT OF RANGE'
 574F DA4A80        2023         $IF   .BIT7 @FAC .NE. 0 GOTO ERR5   CAN'T BE <0
 5752 4D7C
 5754 00            2024         RTN
                    2025  *
                    2026  *
                    2027  *       FAC IS SET UP WITH F.P. 1
                    2028  *
                    2029  JOYXY
 5755 DA0080        2030         TBR   @VAR0,7       POSITIVE OR NEGATIVE?
 5758 5767          2031         BR    JOYNEG        NEGATIVE
 575A 8E00          2032         CZ    @VAR0         0?
 575C 7763          2033         BS    JOY0          YES
 575E BC4B00        2034         ST    @VAR0,@FAC+1  (>40__000000000000)
 5761 576D          2035         BR    JOYXY1
 5763 874A          2036  JOY0   DCLR  @FAC          (>0000000000000000)
 5765 576D          2037         BR    JOYXY1
 5767 BE4ABF        2038  JOYNEG ST    >BF,@FAC
 576A BC4B00        2039         ST    @VAR0,@FAC+1  (>BF__000000000000)
 576D 0F15          2040  JOYXY1 XML   ASSGNV        ASSIGN VALUE TO VARIABLE
 576F 00            2041         RTN
                    2042  *
                    2043  *
                    2044  *
 5770 BC004B        2045  KEYJOY ST    @FAC+1,@VAR0  KEYBOARD SELECTION
                    2046  *
 5773 06578D        2047         CALL  NUMVAR        GET NUMERIC VARIABLE FOR KEY CODE
 5776 D642B3        2048         CEQ   COMMA$,@CHAT  OR JOYSTICK X. FOLLOWED BY ','?
 5779 5671          2049         BR    ERR1          NO
 577B 0F1B          2050         XML   PGMCH
                    2051  *
 577D 06578D        2052         CALL  NUMVAR        GET NUMERIC VARIABLE FOR KEY STA
 5780 BC7400        2053         ST    @VAR0,@KEYBD
 5783 310008        2054         MOVE  8 FROM ROM(#FLT1) TO @FAC  SET UP F.P. 1 FOR A
 5786 4A50C0
 5789 03            2055         SCAN                SCAN KEYBOARD
 578A 8674          2056         CLR   @KEYBD        CLEAR KEY CODE (DOES NOT AFFECT '
 578C 01            2057         RTNC                RETURN SCAN CONDITION CODE
                    2058  *
 578D 0F13          2059  NUMVAR XML   SYM
 578F DAB04A        2060         CLOG  >F8,RAM(@FAC)
```

```
 792 F8
 793 5671              BR    ERR1           CAN'T BE STRING OR FUNCTION
5795 0F14               XML   SMB
5797 0F17               XML   VPUSH          PUT ON STACK FOR ASSGNV
5799 00                 RTN
              *
              *
              *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
              *         CHANGED FROM:
              *ATTREG DATA #>8000,#>A000,#>C000,>9F,>BF,>DF,>FF,>00,>06,>
              *         BY STAN HUME ON 06/19/80 TO USE THE WASTED BYTES
              *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
579A 8000A0   ATTREG DATA #>8000,#>A000,#>C000,>9F,>BF,>DF,>FF,>00,06
579D 00C000
57A0 9FBFDF
57A3 FF0006

57A6 D642B7   COMB     CEQ   LPAR$,@CHAT    '('?
57A9 5671              BR    ERR1           NO
57AB 00                RTN
```

```
                    2078          GROM 1
                    2079          ORG  >1510
                    2080
3510 055671         2081  ERR1B   B    ERR1
3513 05567D         2082  ERR3B   B    ERR3
3516 054D7C         2083  ERR5B   B    ERR5
3519 054D81         2084  ERR6B   B    ERR6
                    2085
                    2086
                    2087  ********************************************************
                    2088  **      SUBROUTINE FOR 'CLEAR'
                    2089  ********************************************************
351C 063784         2090  GRAPH   CALL PGMCTR
351F 0780           2091          ALL  : :+OFFSET  CLEAR THE SCREEN
3521 BE7F03         2092          ST   3,@XPT
3524 060012         2093          CALL RPL          VALUE ON VALUE STACK
                    2094  *
                    2095  *
                    2096  *       INITIALIZATION DATA FOR SOUND
                    2097  *
3527 420B12         2098  FLTS    DATA >42,>0B,>12,>22,0,0,0,0
352A 220000
352D 0000
                    2099  *
352F 01FF01         2100  SNDREG  DATA >01,>FF,>01,>04,>9F,>BF,>DF,>FF,>00
3532 049FBF
3535 DFFF00
                    2101  *
```

```
                2103  ********************************************************
                2104  **      SUBROUTINE FOR 'SOUND'
                2105  ********************************************************
                2106  SOUND
                2107  *  2 BLOCKS IN VDP RAM
                2108  *
                2109  *  1ST BLOCK:  >01,<ATTENUATION FOR NOISE>,<INTERRUPT COUNT>
                2110  *  2ND BLOCK:  >04,>9F,>BF,>DF,>FF,>00
                2111  *
3538 310009     2112          MOVE 9 FROM ROM(#SNDREG) TO RAM(VRMSND)
353B A3E235
353E 2F
353F 063767     2113          CALL THREEC         DURATION IN MILLISECOND
3542 D24A00     2114          $IF  @FAC .LT. 0 THEN      DON'T WAIT FOR COMPLETION
3545 754C
3547 834A       2115            DNEG @FAC                   OF PREVIOUS SOUND
3549 8780CE     2116            DCLR @PRTNFN         MAKE INTERPRETER FORGET PREVIOU
                2117          $END
354C BF1010     2118          DST  4250,@VAR5     1 - 4250 (MILLISECOND)
354F 9A
3550 06377D     2119          CALL RANO1          OUT OF RANGE?
                2120  *
                2121  *  CONVERT IT IN 1/60 SECOND
                2122  *
3553 AB4A00     2123          DMUL 6,@FAC         DURATION * 6
3556 06
3557 AF4A00     2124          DDIV 100,@FAC       (DURATION * 6) / 100
355A 64
355B 8E4B       2125          CZ   @FAC+1         DURATION=0?
355D 5561       2126          BR   SOUND0         NO
355F 904B       2127          INC  @FAC+1         YES,SET IT TO 1(1/60 SEC)
                2128  *                           ( < 16 MILLISECOND)
3561 BCA3E4     2129  SOUND0 ST   @FAC+1,RAM(VRMSND+2) 3RD BYTE OF THE 1ST BLOCK
3564 4B
                2130  *                                    : INTERRUPT COUNT
                2131  ********************************************************
                2132  *  A TABLE OF 10 BYTES IN CPU RAM (>00 - >09)
                2133  *  >00 ->05 : FREQUENCY CONTROL
                2134  *  >06 ->08 : ATTENUATION CONTROL
                2135  *  >09 :      NOISE CONTROL(NON-0 MEANS NOISE ENCOUNTERED
                2136  *  >0A :      POINTER FOR CURRENT FREQ. CONTROL
                2137  *  >0B :      POINTER FOR CURRENT ATTEN. CONTROL
                2138  *
                2139  *                          >00 ,>01  FOR REG. 0;
                2140  *                          >02 ,>03  FOR REG. 1;
                2141  *                          >04 ,>05  FOR REG. 2
                2142  *
                2143  *  REG.0:>8000,  REG.1:>A000,  REG.2:>C000
                2144  *
                2145  *  INITIALIZE ATTENUATION CONTROL:
                2146  *  REG.0:>9F,  REG.1:>BF,  REG.2:>DF
                2147  ********************************************************
                2148  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                2149  *       CHANGED FROM:
                2150  *       MOVE 13 FROM ROM(#ATTREG) TO @>00
```

```
                2151  *       BY STAN HUME ON 06/19/80 TO ONLY MOVE THE CORRECT
                2152  *       NUMBER OF BYTES SINCE THE ATTREG DATA LINE CHANGED
                2153  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
3565 31000C     2154          MOVE 12 FROM RCM(#ATTREG) TO @>00
3568 00579A
356B 06376F     2155  SOUND1  CALL SUPY1          FREQUENCY
356E 064F79     2156          CALL CKSTNM         MUST BE NUMERIC
3571 D24A00     2157          $IF  @FAC .LT. 0 GOTO SOUND2   NOISE IF NEGATIVE
3574 55C3
3576 310008     2158          MOVE 8 FROM RCM(#FLTS) TO @ARG 111834
3579 5C3527
357C 0F09       2159          FLTPT FDIV          'P' = 111834/FREQUENCY
357E BF1003     2160          DST  1023,@VAR5     3 - 1023
3581 FF
3582 063785     2161          CALL RANGE          CHECK IF FREQUENCY OUT OF RANGE
3585 CB4A00     2162          DCHE 3,@FAC         < 3?
3588 03
3589 5516       2163          BR   ERR5B          YES, 'VALUE OUT OF RANGE'
                2164  *
                2165  *  GET THE L. S. 4 BITS AND M. S. 6 BITS OF 'P'
                2166  *
358B EB4A00     2167          DSRC @FAC,4
358E 04
358F E64A04     2168          SRL  @FAC,4
                2169  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                2170  *       CHANGED FROM:
                2171  *       DOR  @FAC,*BYTE     1ST BYTE OF FREQUENCY CONTROL BY
                2172  *       BY STAN HUME ON 06/19/80 TO USE STADDR INSTEAD OF
                2173  *       BYTE TO RESOLVE CONFLICT WITH THE STRING ROUTINES I
                2174  *       SOUND IS CALLED USING THE VAL STRING FUNCTION
                2175  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
3592 B5900A     2176          DOR  @FAC,*STADDR 1ST BYTE OF FREQUENCY CONTROL BYT
3595 4A
                2177  *                               BIT  7   6   5   4   3   2   1
                2178  *                                    1  <REG.>  0   < L.S. 4 OF
                2179  *
                2180  *                               2ND BYTE OF FREQUENCY CONTROL BYT
                2181  *                                    0   0   <   M.S. 6 OF 'P'
                2182  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
                2183  *       CHANGED FROM:
                2184  *       INCT @BYTE          ADVANCE POINTER FOR NEXT TIME
                2185  *       BY STAN HUME ON 06/19/80 TO USE STADDR INSTEAD OF
                2186  *       BYTE TO RESOLVE CONFLICT WITH THE STRING ROUTINES I
                2187  *       SOUND IS CALLED USING THE VAL STRING FUNCTION
                2188  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
3596 940A       2189          INCT @STADDR        Advance pointer for next time
3598 065600     2190          CALL ATTNUT         GET ATTENUATION
                2191  *                             BIT  7   6   5   4   3   2   1
                2192  *                                  1  <REG.>  1   0   0   0
359B B0900B     2193          AND  @FAC+1,*VAR2          1  <REG.>  1   < ATTN/2 D
359E 4B
359F 900B       2194          INC  @VAR2          ADVANCE POINTER FOR NEXT TIME
                2195  *
                2196  *  CHECK FOR END OF SOUND CALL
                2197  *
```

```
35A1 D642B6  SOUND3 $IF   @CHAT .EQ. RPAR$ GOTO SOUND5    END OF STMT ?
35A4 75E2
35A6 D642B3         $IF   @CHAT .NE. COMMA$ GOTO ERR1B    NO, COMMA ?
35A9 5510
35AB 0F1B           XML   PGMCH                            YES, SKIP IT
35AD E64A04         SRL   @FAC,4
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
             *      CHANGED FROM:
35B0 D60C06         CEQ   >06, BYTE      3 REG.S ALREADY?
             *      BY STAN HUME ON 06/19/80 TO USE STADDR INSTEAD OF
             *      BYTE TO RESOLVE CONFLICT WITH THE STRING ROUTINES IF
             *      SOUND IS CALLED USING THE VAL STRING FUNCTION
             *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
35B3 D60A06         CEQ   >06,@STADDR   3 REG.S ALREADY?
35B6 556B           BR    SOUND1        NO, GET NEXT REG.
             *                          YES, NEXT MUST BE NOISE
35B8 06376F         CALL  SUPY1         GET NEXT FREQUENCY(SHOULD BE NOISE
35BB 064F79         CALL  CKSTNM        MUST BE NUMERIC
35BE D24A00         $IF   @FAC .NOT. .LT. 0 GOTO ERR1B     IF NOT NOISE
35C1 7510
             *
             *  NOISE CONTROL
             *
35C3 D609FF  SOUND2 $IF   @>09 .NE. >FF GOTO ERR1B  IF NOISE ALREADY!
35C6 5510
35C8 834A           DNEG  @FAC          -(FREQUENCY)
35CA BF1000         DST   8,@VAR5       1-8
35CD 08
35CE 06377D         CALL  RANO1         CHECK IF OUT OF RANGE
35D1 924B           DEC   @FAC+1        0-7 (2ND BIT: 'T';
             *                                 OTH,1ST BITS: 'S')
35D3 BC094B         ST    @FAC+1,@>09
35D6 B609E0         OR    >E0,@>09      NOISE CONTROL BYTE:
             *                          BIT  7   6   5   4   3   2   1   0
             *                               1   1   1   0   0  <T>  < S >
             *
             *  PUT ATTENUATION IN THE 2ND BYTE OF 1ST BLOCK
             *
35D9 065600         CALL  ATTNUT        GET ATTENUATION
35DC BCA3E3         ST    @FAC+1,RAM(VRMSND+1)
35DF 4B
             *                                1   1   1   1   < ATTN/2  DB
35E0 55A1           BR    SOUND3        GO CHECK FOR END OF LIST
             *
             *
35E2 8610    SOUND5 CLR   @VAR5         PNTR TO SOUND TABLE
35E4 8E80CE  SND05  $IF   @PRTNFN .EQ. 0 GOTO SOUND6
35E7 75F9
35E9 03             SCAN
35EA 55E4           BR    SND05         NO KEY DOWN
35EC D67502         $IF   @RKEY .NE. BREAK GOTO SND05
35EF 55E4
35F1 8E8089         $IF   @GROMFL .NE. 0 GOTO SND05   IGNORE BREAK IF GRO
35F4 55E4
35F6 054E38         B     EXEC6C        IF BREAK KEY
```

```
               *
               *
               *  LOAD SOUND TABLE
               *
35F9 BC8100    SOUND6 ST   *VAR5,@>0100
35FC 9010
35FE 9010             INC  @VAR5          NEXT BYTE IN TABLE
3600 D6100A           CEQ  >0A,@VAR5      FINISHED?
3603 55F9             BR   SOUND6         NO, LOOP.
               *
3605 BF4A03    SOUND7 DST  VRMSND,@FAC    WHERE 2 BLOCKS ARE
3608 E2
3609 F64A01           I/O  @FAC,1         FROM VDP RAM
               *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
               *      CHANGED FROM:
               *      B    LNKRTN
               *      BY STAN HUME ON 06/19/80 TO SAVE 1 BYTE SINCE
               *      CONDITION WILL ALWAYS BE RESET
               *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
360C 5620             BR   LNKRTN
```

```
                2263  ************************************************************
                2264  **      SUBROUTINE FOR 'HCHAR'
                2265  ************************************************************
                2266  HCHAR
360E 0637D6     2267          CALL HVCHR          GET X,Y VALUES,CHARACTER,# OF CHRS
3611 8F4A76     2268          $IF  @FAC .DEQ. 0 GOTO XPTRTN    IF 0 CHARACTERS
3614 1D
                2269  HCHAR1
3615 0 E000     2270          FMT  '@VAR0'        PUT CHARACTERS HORIZONTALLY ON SCR
3618 FB
3619 934A       2271          DDEC @FAC           COUNT DOWN
361B 5615       2272          BR   HCHAR1         NO
361D BC7F02     2273  XPTRTN  ST   @MNUM,XPT
3620 D642B6     2274  LNKRTN  $IF  @CHAT .NE. RPAR$ GOTO ERR1B
3623 5510
3625 0F1B       2275          XML  PGMCH          GET LAST TOKEN
3627 060012     2276          CALL RPL
                2277
                2278
                2279
                2280  ************************************************************
                2281  **      SUBROUTINE FOR 'VCHAR'
                2282  ************************************************************
                22 3  VCHAR
362A 0637D6     2284          CALL HVCHR          GET X,Y VALUES,CHARACTER,# OF CHRS
362D 8F4A76     2285          $IF  @FAC .DEQ. 0 GOTO XPTRTN    IF 0 CHARACTERS
 630 1D
                2286  VCHAR1
3631 08E000     2287          FMT  '@VAR0',31<
3634 9EFB
3636 934A       2288          DDEC @FAC           END?
3638 761D       2289          BS   XPTRTN         YES
363A 8E7E56     2290          $IF  YPT .NE. 0 GOTO VCHAR1  IF NOT AT START OF COLU
363D 31
363E 907F       2291          INC  XPT            X POINTER MOVES TO RIGHT BY 1 SPAC
3640 053631     2292          B    VCHAR1
```

```
                2294  ********************************************************
                2295  **      SUBROUTINE FOR 'CHAR'
                2296  ********************************************************
                2297  CHARLY
3643 063767     2298          CALL THREEC         CHARACTER VALUE
3646 BF1000     2299          DST  159,@VAR5      32-159 LEGAL
3649 9F
364A 063785     2300          CALL RANGE
364D CB4A00     2301          $IF  @FAC .DL. 32 GOTO ERR5B
3650 205516
3653 E34A00     2302          DSLL @FAC,3         CONVERT CHAR. # TO ADDRESS
3656 03
3657 A34A03     2303          DADD >300,@FAC      CORRECT FOR OFFSET
365A 00
365B BD044A     2304          DST  @FAC,@VARY     SAVE IT
365E 0EB6       2305          PARSE RPAR$         GET STRING
3660 D64C65     2306          $IF  @FAC+2 .NE. >65 GOTO ERR6B  MUST BE STRING
3663 5519
                2307  *                           SEE IF THIS CHARACTER IS ALREADY
3665 CD0424     2308          $IF  @VARY .DLE. @STVSPT GOTO CHAR30
3668 56A3
366A BD5E04     2309          DST  @VARY,@ARG+2
366D A55E24     2310          DSUB @STVSPT,@ARG+2 DISTANCE TO MOVE
3670 BD066E     2311          DST  @VSPTR,@VARY+2
3673 A1065E     2312          DADD @ARG+2,@VARY+2 WHAT VSPTR WILL BE AFTER MOVE
3676 A30600     2313          DADD 15,@VARY+2     ALLOW FOR VSPTR OFFSET AND
3679 0F
                2314  *                                 SPARE ENTRY FOR VPUSH
367A C91A06     2315          $IF @STREND .DHE. @VARY+2 GOTO CHAR10  IF ROOM TO M
367D 7687
367F 0651A9     2316          CALL COMPCT         NO ROOM SQUISH STRINGS
3682 C91A06     2317          $IF @STREND .DHE. @VARY+2 GOTO ERR3B IF STILL NO R
3685 7513
                2318  CHAR10
3687 A70600     2319          DSUB 8,@VARY+2      DESTINATION FOR MOVE
368A 08
368B BD006E     2320          DST  @VSPTR,@VAR0
368E A30000     2321          DADD 7,@VAR0        SOURCE FOR MOVE
3691 07
3692 BD5C6E     2322          DST  @VSPTR,@ARG
3695 A55C24     2323          DSUB @STVSPT,@ARG NUMBER OF BYTES TO MOVE
3698 769D       2324          BS   CHAR20         IF ZERO BYTES
369A 06201E     2325          CALL MVDN
                2326  *             ARG = BYTE COUNT
                2327  *             VAR0 = SOURCE ADDRESS
                2328  *             VARY+2 = DESTINATION ADDRESS
369D BD2404     2329  CHAR20  DST  @VARY,@STVSPT      UPDATE STVSPT AND
36A0 A16E5E     2330          DADD @ARG+2,@VSPTR        VSPTR
36A3 BD5E4E     2331  CHAR30  DST  @FAC+4,@ARG+2 POINTER TO STRING VALUE
36A6 BD6050     2332          DST  @FAC+6,@ARG+4 LENGTH OF STRING VALUE
36A9 C75000     2333          DCH  16,@FAC+6      LENGTH > 16?
36AC 10
36AD 56B3       2334          BR   CHARL1         NO
36AF BF6000     2335          DST  16,@ARG+4      IGNORE THE EXCESSES
36B2 10
```

```
                *
36B3 BE4A30     CHARL1 ST   ZERO,@FAC      FLOATING POINT ACCUMULATOR (>30)
36B6 35000F            MOV  15 FROM @FAC TO @FAC+1
36B9 4B4A
36BB 8F60              DCZ  @ARG+4         NULL STRING?
36BD 76C4              BS   CHARL$         YES
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *      CHANGED FROM:
                *        $IF  GROMFL .EQ. 0 THEN
                *          MOV  @ARG+4 FROM RAM(@ARG+2) TO @FAC STRING CHARA
                *        $SELSE
                *          MOVE @ARG+4 FROM ROM(@ARG+2) TO @FAC
                *        $SEND IF
                *      BY STAN HUME 9/6/79     TO FIX GROM CALL CHAR
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
36BF 34604A            MOVE @ARG+4 FROM RAM(@ARG+2) TO @FAC STRING CHARACTE
36C2 B05E

36C4 BE114A     CHARL$ ST   FAC,@VAR6      START TO LOAD THE CHARACTERS
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *      CHANGED FROM:
                *      $FOR @VAR5 = 1 TO 8
                *      CLR  @BYTE          CLEAR DOT BUILDING BYTE
                *      BY STAN HUME 06/19/80 TO SAVE BYTES
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
36C7 BE1008            ST   8,@VAR5        Need to loop 8 times
36CA 860C       CHARL5 CLR  @BYTE          CLEAR DOT BUILDING BYTE
36CC E20C04     CHARL2 SLL  @BYTE,4        FOR LOOP (TWO CHARACTERS PER BYTE)
36CF BC5C90            ST   *VAR6,@ARG
36D2 11
36D3 CA5C30            CHE  ZERO,@ARG      CONVERT CHARACTER TO DOT EXPRESSIO
36D6 5516              BR   ERR5B          ILLEGAL CHARACTER
36D8 C65C39            CH   NINE,@ARG
36DB 56E7              BR   CHARL3         0-9
36DD CA5C41            CHE  A,@ARG
36E0 5516              BR   ERR5B          ILLEGAL CHARACTER
36E2 C65C46            CH   F,@ARG
36E5 7516              BS   ERR5B          ILLEGAL CHARACTER
36E7 A65C30     CHARL3 S    ZERO,@ARG      CHARACTER - >30
36EA C65C0A            CH   10,@ARG        0 - 9?
36ED 56F2              BR   CHARL4         YES
36EF A65C07            SUB  7,@ARG
36F2 B40C5C     CHARL4 OR   @ARG,@BYTE   DOT EXPRESSION
36F5 9011              INC  @VAR6
36F7 DA1101            TBR  @VAR6,0        1ST HALF OF THE ROW FINISHED?
36FA 56CC              BR   CHARL2         YES,DO 2ND HALF
                *                          (EACH TAKES HALF BYTE)
36FC BCB004            ST   @BYTE,RAM(@VARY) LOAD CHARACTERS
36FF 0C
3700 9104              DINC @VARY
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                *      CHANGED FROM:
                *      $END FOR            LOAD CHARACTERS ON NEXT ROW
                *      BY STAN HUME ON 06/19/80 TO SAVE BYTES IN
                *      CONJUNCTION WITH THE CHANGE MADE ABOVE
```

```
                *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
3702 9210                DEC  @VAR5             Decrement loop counter
3704 56CA                BR   CHARL5            Loop if all 8 not done yet
3706 5620                BR   LNKRTN
```

```
                2392  ********************************************************
                2393  **      SUBROUTINE FOR "KEY"
                2394  ********************************************************
3708 063767     2395  KEY     CALL THREEC         GET KEY UNIT
370B BF1000     2396          DST  5,@VAR5        0 - 5
370E 05
370F 063785     2397          CALL RANGE
3712 065770     2398          CALL KEYJOY         GET VARIABLES FOR CODE AND STATUS
                2399  *                           AND SCAN KEYBOARD
                2400  *                           KEYJOY RETURNS KEY STATUS
3715 7722       2401          BS   KEY1B          KEY STATUS = 1
3717 D675FF     2402          CEQ  >FF,@RKEY      ANY KEY DEPRESSED?
371A 5720       2403          BR   KEY1A          YES
371C 874A       2404          DCLR @FAC           KEY STATUS = 0
371E 5722       2405          BR   KEY1B
3720 834A       2406  KEY1A   DNEG @FAC           KEY STATUS = -1
3722 0F15       2407  KEY1B   XML  ASSGNV        ASSIGN VALUE TO VARIABLE
3724 BF4A40     2408          DST  >4001,@FAC     RE-STORE F.P. 1 IN FAC
3727 01
                2409  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                2410  *    Replace following lines for adding lowercase char. set
                2411  *    in 99/4A, 3/5/81
                2412  *      CZ   @RKEY          KEY CODE = 0?
                2413  *      BS   KEY2
                2414  *      CH   >63,@RKEY      KEY CODE > >63?
                2415  *    with
 728 D675FF     2416          CEQ  >FF,@RKEY      No key depressed?
                2417  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
372B 7742       2418          BS   KEY1C          YES - KEY CODE ASSIGNED TO -1
                2419  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                2420  * FORMAT FOR KEYCODES ABOVE 99 ADDED FOR 99/4A
                2421  * HIGHEST KEYCODE (OTHER THEN >FF) IS >C6=198
                2422  * BY ALLEN ACREE   03/07/81
                2423  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
372D CA7564     2424          $IF @RKEY .HE. 100 THEN
3730 573D
3732 904A       2425            INC @FAC
3734 A67564     2426            SUB 100,@RKEY
3737 BC4C75     2427            ST @RKEY,@FAC+2   FLOATING FORMAT(>4101__000000000
373A 053740     2428          $ELSE
373D BC4B75     2429            ST   @RKEY,@FAC+1 FLOATING FORMAT(>40__00000000000
                2430          $SEND IF
3740 5744       2431          BR  KEY2A
3742 834A       2432  KEY1C   DNEG @FAC           KEY CODE ASSIGNED TO -1
                2433  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                2434  *   Delete following lines for adding lowercase char. set
                2435  *   in 99/4A, 3/5/81
                2436  *      BR   KEY2A
                2437  *KEY2   DCLR @FAC           (>0000000000000000)
                2438  *<><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
3744 0F15       2439  KEY2A   XML  ASSGNV        ASSIGN VALUE TO VARIABLE
3746 5620       2440          BR   LNKRTN
                2441
                2442
                2443
```

```
                2444  ************************************************************
                2445  *       SUBROUTINE FOR 'JOYSTICK'
                2446  ************************************************************
3748 063767     2447  JOYST   CALL  THREEC          KEY UNIT
374B BF1000     2448          DST   4,@VAR5
374E 04
374F 06377D     2449          CALL  RAND1           1 - 4
3752 065770     2450          CALL  KEYJOY          GET VARIABLES FOR X, Y
                2451  *                               AND SCAN KEYBOARD
3755 BC0076     2452          ST    @JOYY,@VAR0     JOYSTICK Y POSITION
3758 065755     2453          CALL  JOYXY           -4 - +4
375B BF4A40     2454          DST   >4001,@FAC      RE-STORE F.P. 1 IN FAC
375E 01
375F BC0077     2455          ST    @JOYX,@VAR0     JOYSTICK X POSITION
3762 065755     2456          CALL  JOYXY           -4 - +4
3765 5620       2457          BR    LNKRTN
```

```
 767 0637B4  2459  THREEC CALL PGMCTR
376A 0657A6  2460        CALL COMB          CHECK FOR '('
376D 0F1B    2461        XML  PGMCH
             2462  ********************************************************
             2463  **      SUBROUTINE FOR 'SUPY' USED IN ALL GRAPHICS ROUTINES
             2464  ********************************************************
             2465  SUPY1
376F 0EB6    2466        PARS  RPAR$
3771 D642B3  2467        CEQ  COMMA$,@CHAT
3774 5510    2468        BR   ERR1B
3776 0F1B    2469  PGMRTN XML  PGMCH
3778 00      2470        RTN
             2471
             2472
             2473
             2474
3779 BF1000  2475  RAN16  DST  16,@VAR5      RANGE 1 TO 16
377C 10
377D 063785  2476  RAN01  CALL RANGE
3780 8F4A75  2477        $IF  @FAC .DEQ. 0 GOTO ERR5B  DON'T ALLOW ZERO
3783 16
3784 00      2478        RTN
             2479  ********************************************************
             2480  *       SUBROUTINE FOR 'RANGE' USED IN ALL SOUND AND GRAPHIC
             2481  ********************************************************
 785 065740  2482  RANGE  CALL FTINT         INTEGER?
3788 C54A10  2483        DCH  @VAR5,@FAC    OUT OF RANGE?
378B 7516    2484        BS   ERR5B         ERROR
378D 00      2485        RTN
             2486
             2487
             2488
             2489  ********************************************************
             2490  *       SUBROUTINE TO GET ROW,COLUMN VALUES
             2491  ********************************************************
378E 063767  2492  GPHV   CALL THREEC         ROW VALUE
3791 BF1000  2493        DST  24,@VAR5      ROW: 1-24
3794 18
3795 06377D  2494        CALL RAN01
3798 BC027F  2495        ST   XPT,@MNUM1
379B 924B    2496        DEC  @FAC+1        INTERNAL RANGE:  0 - 23
379D BC7E4B  2497        ST   @FAC+1,YPT    ROW POINTER
             2498  *
37A0 06376F  2499        CALL SUPY1
37A3 BF1000  2500        DST  32,@VAR5      COLUMN: 1-32
37A6 20
37A7 06377D  2501        CALL RAN01
37AA 924B    2502        DEC  @FAC+1        INTERNAL RANGE:  0 - 31
37AC BC7F4B  2503        ST   @FAC+1,XPT    COLUMN POINTER
37AF 00      2504        RTN
```

```
                *       NOTE: PGMCTR MUST RESIDE AT >37AC BECAUSE IT IS
                *             DIRECTLY REFERENCED BY THE MONITOR.  THE
                *             FOLLOWING DATA STATEMENT IS TO FILL SPACE
                *             TO GET PGMCTR AT >37AC.  IF BYTES ARE NEEDED
                *             ABOVE THE DATA MAY BE SHRUNK OUT.
                *               STAN HUME  06/24/80
37B0 202020             DATA :    :
37B3 20
                *
                *
                *!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                *  PGMCTR is EQU in MONITOR by a direct address.
                *  Therefore, if the address of PGMCTR got changed,
                *  Line 11: PGMCTR EQU >xxxx in MONITOR has to be changed.
                *!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
37B4 8E8089     PGMCTR $IF  @GROMFL .EQ. 0 THEN
37B7 57BC
37B9 BD2C56             DST  @FAC+12,@VAR81
                        $SEND IF
37BC 0F1B               XML  PGMCH
37BE 00                 RTN
```

```
                2526  ********************************************************
                2527  *             SUBPROGRAM TO CONTROL BORDER COLOR
                2528  *            CHARACTER BACKGROUND IS ALSO AFFECTED
                2529  *                SINCE TRANSPARENT IS USED
                2530  ********************************************************
37BF 0637B4     2531  BORDER CALL PGMCTR        SET UP PGMPTR
37C2 0657A6     2532         CALL COMB          TEST FOR "("
37C5 0F1B       2533         XML  PGMCH         SKIP IT
37C7 0EB6       2534         PARS  RPAR$        PARSE TO ")"
37C9 063779     2535         CALL RAN16         LEGAL 1-16
37CC 924B       2536         DEC  @FAC+1        CONVERT TO HARDWARE UNITS
37CE 3D0001     2537         MOVE 1 FROM @FAC+1 TO VDP(7)  LOAD TO VDP
37D1 074B
37D3 053620     2538         B    LNKRTN        RETURN TO BASIC PGM
                2539  *      Must be long branch because of DEC
                2540
                2541
                2542
                2543
                2544  *      GET ROW, COLUMN VALUES AND NUMBER OF CHARACTERS
37D6 06378E     2545  HVCHR  CALL GPHV          GET X,Y VALUES
37D9 0EB6       2546         PARSE RPAR$
37DB 065740     2547         CALL FTINT
37DE A24B60     2548         ADD  OFFSET,@FAC+1
37E1 BC004B     2549         ST   @FAC+1,@VARO SAVE THE CHARACTER
37E4 BF4A00     2550         DST  1,@FAC       ASSUME 1 CHARACTER
 7E7 01
37E8 D642B6     2551         $IF @CHAT .NE. RPAR$ THEN
37EB 77F9
37ED D642B3     2552           $IF @CHAT .NE. COMMA$ GOTO ERR1B
37F0 5510
37F2 0F1B       2553           XML  PGMCH
37F4 0EB6       2554           PARSE RPAR$       # OF CHARACTERS
37F6 065740     2555           CALL FTINT        FLOATING TO INTEGER
                2556         $END IF
37F9 00         2557         RTN
                2558
                2559         END
```

ERRORS= 0

LENGTH= 3478 (>0D96)

463 SYMBOLS USED

SYMBOL	VALUE	DEF	REFERENCE TABLE
A	0041	299	2367
ABS$	00CB	252	558
ALLUP	0082	145	
APPEN$	00F9	284	
ARG	005C	128	130 131 132 133 565 570 575 588 630 635 640 650 672 673 674 676 744 752 755 758 759 762 76 769 769 771 771 771 808 1130 1142 1150 1154 1201 1203 1205 1208 1213 1214 1214 1215 1216 1226 1233 1248 1249 1251 1253 1253 1254 1257 1295 1416 1428 1438 1441 1441 1442 1442 1443 1446 1447 1450 1627 1634 1647 1718 1721 1860 1862 2158 2309 2310 2312 2322 2323 2330 2331 2332 2335 2339 2350 2350 2362 2363 2365 2367 2369 2371 2372 2374 2375
ASC	5645	1858	326 442 468
ASC01	5306	1291	534
ASSGNV	0015	313	802 1728 1981 2040 2407 2439
ATN	0032	58	565
ATN$	00CC	253	
ATTNUT	5600	1793	2190 2231
ATTREG	579A	2072	2154
BASE	0043	112	
BASE$	00F1	276	
BEE	0012	85	1558 1559 1560 1561 1562 1565 1575 1577 1590 1591 1592 1593 1594
BKGD	0020	169	
BKPTR	0052	122	1055 1057 1058 1060 1062 1066 1067 1068 1485 1499
BORDER	37BF	2531	344
BREAK	0002	292	2241
BREAK$	008E	190	
BUFFY	0044	113	380 392 410 480 487 1546
BYTE	000C	80	867 880 882 942 953 956 958 1000 1002 1011 1014 1015 1016 1020 1121 1122 1123 1125 1126 1139 1216 1228 1277 1325 1326 1327 1331 1356 1358 1362 1369 1370 1371 1379 1413 1414 1426 1430 1431 1449 1450 1456 1461 2204 2360 2361 2375 2380
CA1	4F6B	657	647
CA2	56B5	1921	660
CA3	4FA7	689	658
CALL$	009D	205	
CB	0008	1	1974
CCPPTR	0006	75	
CHAR	0004	1	
CHAR10	3687	2318	2315
CHAR20	369D	2329	2324
CHAR30	36A3	2331	2308
CHARL$	36C4	2352	2340
CHARL1	36B3	2337	2334
CHARL2	36CC	2361	2378
CHARL3	36E7	2371	2366
CHARL4	36F2	2375	2373
CHARL5	36CA	2360	2389
CHARLY	3643	2297	341
CHAT	0042	111	683 686 730 735 782 788 789 830 833 840 844 852 865 887 940 1186 1190 1198 1404 1407

SYMBOL	VALUE	DEF	REFERENCE TABLE
			1410 1544 1554 1579 1580 1629 1693 1933 1935 1937 2048 2074 2198 2199 2274 2467 2551 2552
CHR$$	00D6	263	
CHR$01	52EA	1274	526
CHRTAB	201C	21	470
CIRCU$	00C5	245	
CKSTNM	4F79	665	559 582 61 653 858 1795 2156 2213
CLOSE	400A	2	511
CLOSE$	00A0	208	
CLSALL	4012	18	488
CNS	0014	49	1320
COLON$	00B5	229	
COLOR	5713	1988	337
COMB	57A6	2074	557 580 616 644 1183 1245 1274 1291 1315 1348 1401 2460 2532
COMMA$	00B3	227	1185 1186 1189 1190 1403 1404 1406 1407 1580 2048 2199 2467 2552
COMMON	4F50	642	566 571 576 589 631 636
COMP03	51B4	1058	1069 1085
COMP05	51C5	1066	1063
COMPC9	51A3	1050	427
COMPCT	51A9	1053	333 675 1007 1051 2316
CONC$	00B8	232	1118
CONC04	5210	1126	1124
CONC06	5246	1154	1143 1149
CONCAT	51F2	1115	426
CONV1	56E1	1960	332
CONV2	56EE	1966	1963
COPYS1	543C	1499	1497
COPYS2	5427	1484	1511
COPYS3	5455	1519	1495
COPYST	5423	1483	1080
COPYTO	0054	123	1054 1064 1081 1082 1083 1084 1484 1502 1503 1519
COS	002C	55	570
COS$	00CD	254	
CPL	0010	47	884
CRNBUF	0320	153	407 880 881
CSNUM	0010	309	1378 1961
DATA	0034	104	376 378 379 380 394 1836 1837 1839 1849
DATA$	0093	195	1843
DATAST	5613	1833	325 397 1847
DEF$	0089	185	
DELET	4002	33	529
DELET$	0099	201	
DELINK	55BB	1750	331 1744
DIM$	008A	186	
DISO	2842	13	1864
DISPL$	00A2	210	
DISPL1	4000	32	528
DISPLA	0009	1	1955 1955
DIVI$	00C4	244	
EDGECH	007F	303	1956 1956
ELSE$	0081	177	
END	00BE	149	670
END$	008B	187	

SYMBOL	VALUE	DEF	REFERENCE TABLE
ENDSCR	02FE	152	
ENLN	0032	103	378 396 1558 1594
ENT09	284A	24	1700
ENTER	2848	23	1706
EOF	401C	34	535
EQUAL$	00BE	238	735 1693
ERR	566C	1881	361 827 842 1547 1876
ERR$$	284E	20	349 352 661 891 1881 1885 1889 1892 1896
ERR1	5671	1885	684 687 702 730 733 735 782 789 840 866 887 1187 1191 1199 1404 1407 1410 1580 1668 1872 1933 2049 2061 2075 2081
ERR14	4D7C	348	
ERR1B	3510	2081	2199 2214 2218 2274 2468 2552
ERR2	5678	1889	1877
ERR3	567D	1892	671 676 1013 1689 1873 1900 2082
ERR3B	3513	2082	2317
ERR4	5682	1896	1875
ERR5	4D7C	349	794 1194 1412 1874 1920 2022 2023 2083
ERR5B	3516	2083	2163 2301 2364 2368 2370 2477 2484
ERR6	4D81	352	665 785 795 1099 1120 1247 1293 1318 1351 1736 1738 1878 1913 1914 2017 2084
ERR6B	3519	2084	2306
ERRC	510C	890	886
ERRCOD	0022	95	417 446 996 1050 1116 1626 1871 1904 1911
ERROR$	565C	1871	421
EXEC	4DB0	392	323
EXEC1	4DBF	398	327
EXEC1A	4DD1	408	405
EXEC1B	4DC6	403	463
EXEC1Z	4DEA	415	458
EXEC2	4DCD	407	393
EXEC20	4DEB	417	408
EXEC5	4E01	432	420
EXEC6	4E2E	457	419
EXEC6B	4E3C	462	1551
EXEC6C	4E38	461	328 2243
EXEC6D	4E58	471	1887 1898
EXEC7	4E5B	480	404 418
EXEC7C	4E55	469	481 490
EXEC7R	4E83	491	
EXP	0028	53	575
EXP$	0076	138	
EXP$$	00CE	255	
EXP005	56B5	1922	1381
EXP010	56BA	1925	1918
EXPON	5696	1910	423
EXTRAM	002E	101	376 394 395 403 459 461 739 822 825 826 1548 1860 1862
F	0046	300	2369
FAC	004A	114	122 123 124 125 126 560 593 594 599 600 611 619 621 622 625 654 657 659 665 700 701 733 738 739 758 785 794 795 797 809 842 859 875 881 935 940 941 942 944 945 946 951 955 963 1099 1119 1121 1130 1132 1139 1140 1145 1146 1151 1193 1201 1203 1206 1209 1213

- SYMBOL VALUE DEF REFERENCE TABLE

```
                                1220 122  1229 1247 1248 1250 1255 1257 1258 1259
                                1278 12 1 1293 1294 1295 1317 1319 1321 1322 1323
                                1326 1327 1331 1350 1352 1353 1354 1356 1366 1367
                                1372 1373 1374 1376 1377 1379 1381 1412 1413 1416
                                1418 1426 1430 1431 1438 1440 1440 1443 1445 1457
                                145  1561 1625 162  1634 1636 1644 1645 1647 1654
                                1655 1655 1662 1668 1669 1670 1672 1674 1675 1699
                                1714 1716 1716 171  1719 1719 1721 1722 1723 1724
                                1724 1735 1781 1782 1783 1784 1786 1798 1802 1841
                                1913 1915 1917 1919 1935 1937 1960 1962 1973 1974
                                1975 1976 1977 1977 1978 1979 1990 1995 1996 2001
                                2002 2005 2016 2018 2021 2023 2034 2036 2038 2039
                                2045 2054 2060 2114 2115 2123 2124 2125 2127 2129
                                2157 2162 2167 2168 2176 2193 2201 2214 2219 2222
                                2224 2232 2253 2254 2268 2271 2285 2288 2301 2302
                                2303 2304 2306 2331 2332 2333 2337 2338 2338 2350
                                2352 2404 2406 2408 2425 2427 2429 2432 2454 2477
                                24 3 2496 2497 2502 2503 2521 2536 2537 2548 2549
                                2550
FCOMP   000A   308       810
FDIV    0009   307      2159
FIGURE  0002     1
FIXED$  00FA   285
FLAG    0088   146       398 1527 1535 1691 1709
FLT1    50C0   847       622  797 1250 1973 2054
FLTINT  0012   310      2020
FLTS    3527  2098      2158
FOR$    008C   188       830
FPERAD  006C   135      2019
FREWRD  003A   107
FTINT   5740  2015      2482 2547 2555
GCHAR   56EF  1971       338
GETDAT  4010    31      1842
GETLN   283C    15       372
GETNB   282C    17       370
GETS18  5187  1014      1006
GETST9  5156   995       425
GETSTR  515C   999       330  943  962  995 1128 1218 1364
GO$     0085   181
GOSUB$  0087   183
GOTO$   0086   182
GPHV    378E  2492      1971 2545
GRAPH   351C  2090       336
GREAT   003E   298       443
GREAT$  00C0   240
GRINT   0022    50       583
GROMFL  0089   147       458  463  490  876  944 1835 1841 1859 2242 2520
HCHAR   360E  2266       339
HCHAR1  3615  2269      2272
HVCHR   37D6  2545      2267 2284
IF$     0084   180
ILLARG  4F74   661      1294 1352 1359 1380
INPUT   4006    26       509
INPUT$  0092   194
INSTRT  003C   108
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
INT$	00CF	256	581
INTARG	5740	2016	1192 1200 1276 1411
JOY0	5763	2036	2033
JOYNEG	5767	2038	2031
JOYST	3748	2447	343
JOYX	0077	66	2455
JOYXY	5755	2029	2453 2456
JOYXY1	576D	2040	2035 2037
JOYY	0076	65	2452
KEY	3708	2395	342
KEY1A	3720	2406	2403
KEY1B	3722	2407	2401 2405
KEY1C	3742	2432	2418
KEY2A	3744	2439	2431
KEYBD	0074	63	2053 2056
KEYJOY	5770	2045	2398 2450
LEN$	00D5	262	
LEN01	52BE	1245	525
LEN02	52CB	1249	1296 1452
LESS	003C	297	440
LESS$	00BF	239	
LET$	008D	189	
LINE	56BB	1932	324 1557
INEGP	547F	1556	1545 1582
INK1	4D1A	335	
LINK2	4D24	336	335
LINK3	4D2E	337	336
LINK4	4D38	338	337
LINK5	4D42	339	338
LINK6	4D4C	340	339
LINK7	4D56	341	340
LINKA	4D5F	342	341
LINKB	4D67	343	342
LINKC	4D71	344	343
LITS05	5120	941	322
LITS06	514F	962	1280 1330
LITS08	512C	945	964 1141 1230
LITS09	5137	953	1368
LITS10	514E	960	954
LITSTR	511D	939	934
LN$	00C9	250	1933
LNBUF	0036	105	396 1834 1836 1839 1844 1845 1846
LNGP1	5489	1560	1563
LNGP2	549A	1565	1561
LNGP2B	54A8	1579	1586
LNGP3	54B5	1584	1560
LNGP4	54CE	1596	1579
LNKRTN	3620	2274	2006 2261 2390 2440 2457 2538
LOG	002A	54	588
OG$	00D0	257	
AR$	00B7	231	685 1629 2074
M$EDIT	2010	6	8 16 21 22
M$FLMG	4000	7	18 25 26 27 28 29 30 31 32 33 34
M$MSG	2022	8	35 36 37 38 39 40 41 42 43 44

SYMBOL	VALUE	DEF	REFERENCE TABLE
			45
M$PSCN	2828	5	12 13 14 15 17 19 20 23 24
MAO	2012	16	471
MINUS$	00C2	242	699
MNUM	0002	72	822 826 2273 2495
MSG1	202C	36	18 6
MSG11	207D	42	353
MSG12	2094	43	662
MSG13	20A1	44	1890
MSG2	20BD	37	1882
MSG4	2040	38	892
MSG5	2049	39	1893
MSG7	2064	40	350
MSG8	206E	41	1903 1923 1965
MSGBLN	20D9	45	1585 1897
MSGBRK	20E9	35	466
MULT$	00C3	243	
MVDN	201E	22	2325
NABS	4ED1	557	515
NATN	4EDC	565	516
NBC	0014	87	
NBD	001E	93	
NBREAK	5463	1543	499
NCALL	50DB	865	505
NCOS	4EE2	570	517
NEXP	4EE8	575	518
NEXT$	0096	198	833
NFOR	4FB6	730	498
NFOR03	507B	816	811 819
NFOR05	507C	818	813
NFOR07	507F	821	815
NFOR09	508A	825	838
NFOR10	50AB	839	835
NFOR1A	4FD4	752	760 763
NFOR1B	4FEC	762	755
NFOR1C	4FF2	768	758
NFOR1D	5003	772	770
NFOR1E	5007	773	752
NINE	0039	296	2365
NINT	4EEE	580	519
NLNADD	02E2	151	435 436 438 444 466 467
NLOG	4EFA	588	520
NLPR	4F99	682	512
NMIN10	4FAC	701	712
NMINUS	4FA8	698	514
NPLUS	4FB2	710	513
NREAD	400E	30	503
NRND	4F00	593	527
NRNDMZ	50C8	852	508
NSGN	4F26	616	521
NSIN	4F40	630	522
NSQR	4F46	635	523
NSTRCN	5111	933	506
NTAN	4F4C	640	524
NTRACE	5459	1527	501

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

NUDTAB  0028    98     409
NUDTB   4E84   497     409
NUM$    00C8   249
NUMBE$  00FD   288
NUMVAR  578D  2059    1972 2047 2052
NUNBRK  5479  1553     500
NUNTRC  545E  1535     502
OFFSET  0060   170     303  440  443 1956 1975 2091 2548
ON$     009B   203
ONECHR  03DD   155    1278 1279
OPEN    4008    27     510
OPEN$   009F   207
OPTIO$  009E   206
OUTPU$  00F7   282
PERMA$  00FB   286
PGMCH   001B   316     688  737  783  791  839  936 1101 1184 1196 1231
                       1402 1451 1581 1631 1633 1687 1708 1727 1747 1934
                       1936 1938 2050 2200 2275 2461 2469 2523 2533 2553
PGMCTR  37B4  2520    2090 2459 2531
PGMRTN  3776  2469
PLUS$   00C1   241     711
POP     0005     1
POS$    00D9   266
POS01   53A9  1400     533
POS02   53F1  1438    1459
POS04   53FF  1443    1448
POS06   540E  1449    1429
POS08   5410  1450    1462
POS10   5417  1456    1444
POS12   541F  1461    1419 1427 1439
PRINT   4004    25     504
PRINT$  009C   204
PRTNFN  00CE    10    2116 2238
PUSSTR  51E8  1098    1117 1188 1405 1408
PWR     0024    51    1916
RAM     0001     1     380  412  413  413  440  443  459  461  466  733
                       755  758  771  771  807  808  809  812  842  880
                       956  956  958 1015 1019 1020 1060 1068 1082 1083
                      1122 1150 1150 1278 1295 1327 1357 1371 1373 1443
                      1443 1503 1503 1548 1549 1561 1575 1577 1592 1654
                      1686 1690 1702 1710 1711 1712 1712 1724 1736 1738
                      1745 1753 1763 1764 1766 1768 1784 1836 1860 1914
                      2005 2060 2112 2129 2232 2350 2350
RAMPTR  0038   106
RAN01   377D  2476    2119 2221 2449 2494 2501
RAN16   3779  2475    1989 1994 2000 2535
RAND0$  0095   197
RANDOM  0078    67     597  605
RANGE   3785  2482    1797 2161 2300 2397 2476
READ$   0097   199
TM$     009A   202
RESTO$  0094   196
RESTOR  400C    29     507
RETUR$  0088   184
RKEY    0075    64    2241 2402 2416 2424 2426 2427 2429
```

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

RND$    00D7   264
RND0    4F06   596      602
RND1    4F14   604      609
RND2    4F16   605      598
RND3    4F23   611      601
RND4    4F25   612      607
RNDMZ1  50D1   857      853
ROM     000A     1      466  622  797  877  880  958 1250 1839 1862 1973
                       2054 2112 2154 2158
ROUND$  0001   305
ROUNU$  0002   306
RPAR$   00B6   230      683  686 1197 1198 1409 1410 1632 1794 1999 2198
                       2274 2305 2466 2534 2546 2551 2554
RPL     0012    48      2093 2276
RSTK    0088   148
RTNG    0026    97      408
RUN     4D8A   369      329
RUN2    4DA3   378      371
RUN2A   4DAA   380      377
RUN3    5682  1895      375
RUN4    5671  1884      373
SAVEPC  03EC   156      461 1548 1549
SCR010  56D4  1956      357
SCROL   56CD  1954      437
SCROLL  56CD  1953      321  381  464  482  489
SEARCH  2836    12      829
SEETWO  283E    14      374
SEG$$   00D8   265
SEG$01  524A  1183      530
SEG$06  529D  1216     1212 1234
SEG$08  52BA  1233     1207
SEMIC$  00B4   228
SEQUE$  00F6   281
SGN$$   00D1   258      617
SGN2    4F3F   626      620  624
SIGN$   0075   137
SIN     002E    56      630
SIN$    00D2   259
SMB     0014   312      736 1715 2062
SND05   35E4  2238     2240 2241 2242
SNDREG  352F  2100     2112
SOUND   3538  2106      335
SOUND0  3561  2129     2126
SOUND1  356B  2155     2210
SOUND2  35C3  2218     2157
SOUND3  35A1  2198     2234
SOUND5  35E2  2237     2198
SOUND6  35F9  2248     2238 2251
SOUND7  3605  2253
SPACE   0020   302     1321 1357 1371 1373 1956
SPRITE  0006     1
SQR     0026    52      635
SQR$    00D3   260
SREF    001C    92      945  946  956  958 1017 1144 1369 1372 1723 1786
STACK   0072    61
```

SYMBOL	VALUE	DEF	REFERENCE TABLE
STADDR	000A	78	2176 2189 2209
STEP$	00B2	226	784 789
STKCH2	4F98	677	674
STKCHK	4F7F	670	643 1912
STLN	0030	102	403 823 1560 1590 1845
STOP$	0098	200	
STR$$	00DB	268	
STR$01	531A	1314	531
STREND	001A	91	674 676 1001 1010 1015 1016 1017 1018 1019 1020 1062 1064 2315 2317
STRIN$	00C7	247	
STRSP	0018	90	1055 1056
STVSPT	0024	96	486 752 2308 2310 2323 2329
SUB$	00A1	209	
SUBSTK	0073	62	646 647 649 650 670 1688 1900
SUPY1	376F	2465	1993 2155 2212 2499
SYM	0013	311	732 841 2059
SYMBOL	03E0	157	1712
SYMPT	0800	164	
SYMPTR	0040	110	1054 1056 1675
SYMTAB	003E	109	1674 1712 1714 1751 1753 1767 1782
TAB$	00FC	287	
TABLE	000B	1	
TAN	0030	57	640
AN$	00D4	261	
TEMP1	0056	124	1001 1002 1005 1010 1011 1012 1059 1060 1066 1205 1206 1208 1209 1210 1211 1483 1486 1496 1498 1500 1504 1519 1766 1767
TEMP2	0058	125	1003 1004 1005 1008 1009 1012 1082 1083 1484 1485 1486 1493 1496 1498 1501 1502 1503 1504
TEMP3	005A	126	1499 1500 1501 1503
TEMP4	0064	130	1144 1145 1150 1645 1654 1686
TEMP5	0066	131	933 956 958 1132 1154 1220 1226 1227 1279 1329 1353 1354 1355 1357 1360 1366 1644 1668 1751 1763 1764 1764 1765 1766 1768
TEMP6	0068	132	1126 1146 1148 1150 1151
TEMP7	006A	133	
THEN$	00B0	224	
THREEC	3767	2459	1988 2113 2298 2395 2447 2492
TIMER	0079	68	854
TO$	00B1	225	780 782
TONE2	0036	59	465
TOP	0003	1	
TRACE$	0090	192	
UDF	54CF	1625	424
UNBK01	54BC	1590	1555
UNBK02	54C1	1592	1595
UNBRE$	008F	191	
UNLN	005F	301	
UNQST$	00C8	248	249 865
ITRA$	0091	193	
PDAT$	00F8	283	
VAL$	00DA	267	
VAL01	5349	1348	532
VAR0	0000	70	807 808 821 831 834 1543 1553 1576 1843 2030

```
SYMBOL  VALUE  DEF    REFERENCE TABLE

                      2032 2034 2039 2045 2053 2270 2287 2320 2321 2452
                      2455 2549
VAR1    000D    81     877  878
VAR2    000B    79    2193 2194
VAR4    000E    82    1996 1997 2002 2005
VAR5    0010    83     594  605  606   60  1796 2118 2160 2220 2237 2248
                      2249 2250 2299 2359 2388 2396 2448 2475 2483 2493
                      2500
VAR6    0011    84    2352 2362  376 2377
VAR8    0013    86
VAR81   002C   100     407  875  877  880  882  933  935 1669 1686 1701
                      1745 1746 2521
VAR9    0016    88    1692
VARA    002A    99     373
VARC    0008    76
VARD    0009    77
VARV    0001    71
VARW    0020    94     369  373  433  434  435  436  438  440  441  443
                       444  445  467
VARY    0004    73     823  824  825 2304 2308 2309 2311 2312 2313 2315
                      2317 2319 2329 2380 2381
VARZ    0005    74
VCHAR   362A  2283     340
VCHAR1  3631  2286    2290 2292
VDP     000C     1    2537
VEL     0007     1
VPOP    0018   315     801 1129 1131 1202 1204 1219 1365 1415 1417 1635
                      1717 2004
VPUSH   0017   314     777  778  779  781  787  799 1100 1127 1195 1217
                      1363 1630 1646 1648 1676 1720 1992 2063
VRAMVS  06F8   163
VRMSND  03E2   158    2112 2129  232 2253
VROA$   03C0   154    1327 1329 1690 1710
VSPTR   006E   136     486  672  744  768  772  786  790  793  800  803
                       807  809  812  842  843 1003 1008 1122 1673 1677
                      1702 1711 1736 1738 1745 1753 1790 1914 2311 2320
                      2322 2330
WARN$$  284C    19    1584 1902 1922 1964
WARNG$  5689  1900     422
X1      0002   166     484
X2      0003   167     399
XPT     000E     1     399  434  445 2092 2273 2291 2495 2503
XPTRTN  361D  2273    1982 2268 2285 2289
Y1      0017   168     484 1956
YPT     000D     1    2290 2497
ZERO    0030   295    2337 2363 2371

TOTAL NUMBER OF SYMBOLS =   463
TOTAL NUMBER OF REFERENCES =  20 2
```