```
TTTTTTTTTTTTTTT     FFFFFFFFFFFFFFF       000000000         CCCCCCCCCCCC       AAAAAAAAA         LLL
TTTTTTTTTTTTTTT     FFFFFFFFFFFFFFF       000000000         CCCCCCCCCCCC       AAAAAAAAA         LLL
TTTTTTTTTTTTTTT     FFFFFFFFFFFFFFF       000000000         CCCCCCCCCCCC       AAAAAAAAA         LLL
      TTT           FFF                 000     000       CCC               AAA     AAA         LLL
      TTT           FFF                 000     000       CCC               AAA     AAA         LLL
      TTT           FFF                 000     000       CCC               AAA     AAA         LLL
      TTT           FFF                 000     000       CCC               AAA     AAA         LLL
      TTT           FFF                 000     000       CCC               AAA     AAA         LLL
      TTT           FFF                 000     000       CCC               AAA     AAA         LLL
      TTT           FFFFFFFFFFF         000     000       CCC               AAA     AAA         LLL
      TTT           FFFFFFFFFFF         000     000       CCC               AAA     AAA         LLL
      TTT           FFFFFFFFFFF         000     000       CCC               AAA     AAA         LLL
      TTT           FFF                 000     000       CCC               AAAAAAAAAAAAAAA     LLL
      TTT           FFF                 000     000       CCC               AAAAAAAAAAAAAAA     LLL
      TTT           FFF                 000     000       CCC               AAAAAAAAAAAAAAA     LLL
      TTT           FFF                 000     000       CCC               AAA     AAA         LLL
      TTT           FFF                 000     000       CCC               AAA     AAA         LLL
      TTT           FFF                 000     000       CCC               AAA     AAA         LLL
      TTT           FFF                   000000000         CCCCCCCCCCCC     AAA     AAA         LLLLLLLLLLLLLLL
      TTT           FFF                   000000000         CCCCCCCCCCCC     AAA     AAA         LLLLLLLLLLLLLLL
      TTT           FFF                   000000000         CCCCCCCCCCCC     AAA     AAA         LLLLLLLLLLLLLLL

PPPPPPPPPPP         RRRRRRRRRRR             IIIIIIIII
PPPPPPPPPPPP        RRRRRRRRRRRR            IIIIIIIII
PPPPPPPPPPPP        RRRRRRRRRRRR            IIIIIIIII
PPP         PPP     RRR         RRR           III
PPP         PPP     RRR         RRR           III
PPP         PPP     RRR         RRR           III
PPP         PPP     RRR         RRR           III
PPP         PPP     RRR         RRR           III
PPP         PPP     RRR         RRR           III
PPPPPPPPPPPP        RRRRRRRRRRRR              III
PPPPPPPPPPPP        RRRRRRRRRRRR              III
PPPPPPPPPPPP        RRRRRRRRRRRR              III
PPP                 RRR   RRR                 III
PPP                 RRR   RRR                 III
PPP                 RRR   RRR                 III
PPP                 RRR     RRR               III
PPP                 RRR     RRR               III
PPP                 RRR     RRR               III
PPP                 RRR       RRR         IIIIIIIII
PPP                 RRR       RRR         IIIIIIIII
PPP                 RRR       RRR         IIIIIIIII

LPTSPL VERSION 6(344)  RUNNING ON LPT000
 *START* USER 6502 GROUP [7066,322] JOB  TFOCAL SEQ. 4612 DATE 13-OCT-77 22:06:21 MONITOR CSM DECSYSTEM-10 603V1 *START*
REQUEST CREATED: 13-OCT-77  21:07:50
FILE: DSKB1:TFOCAL.PRI[7066,322,MARSH] CREATED: 13-OCT-77 13:29:00 <177>  PRINTED: 13-OCT-77 22:06:26
QUEUE SWITCHES:  /PRINT:ARROW /FILE:ASCII /COPIES:1 /SPACING:1 /LIMIT:1344 /FORMS:MINE
```

```
1                       ;
2                       ;                    ***** "FOCAL-65" INTERPRETER *****
3                       ;
4                       ;       ***** FORMULATING ONLINE CALCULATIONS IN ALGEBRAIC LANGUAGE *****
5                       ;
6                       ;                          AUTHOR: WAYNE WALL AND FRIENDS
7                       ;                                  CSM COMPUTING CENTER
8                       ;                                  GOLDEN, COLO,  80401
9                       ;
10                      ;                                  (303)-279-0302
11                      ;
12                      ;
13                      ;
14                      ;       "FOCAL" IS A REGISTERED TRADEMARK OF DIGITAL EQUIPMENT CORP.
15                      ;
16                      ;
17                      ;
18                      ;       COPYRIGHT 1976,1977 BY WAYNE WALL
19                      ;
20                      ;
21                      ;
22                      ;
23                      ;          THIS VERSION (3) IS NOT DIRECTLY COMPATABLE WITH OTHER FOCAL
24                      ;       IMPLEMENTATIONS, AS THE AUTHOR DECIDED TO STRIKE OUT IN SEVERAL
25                      ;       NEW DIRECTIONS.  HOWEVER, MOST FOCAL PROGRAMS ARE EASILY EXCHANGED.
26                      ;
27                      ;          THIS PROGRAM IS CODED IN THE ASSEMBLY LANGUAGE SYNTAX OF THE
28                      ;       6502 RESIDENT ASSEMBLER "ASM65", ALSO WRITTEN BY THE SAME AUTHOR.
29                      ;
30                      ;
31                      ;
```

```
                        .PAGE                   ;'PAGE ZERO STUFF'
                ;
                ;
                ;       ALL PAGE ZERO DEFINITIONS SHOULD APPEAR HERE.
                ;
                ;
                ;
        0000            .LOC    $0000
                ;
0000    53              .BYTE   %123            ;SO DOS ERCM THINKS SOMETHING IS LOADED
0001    53              .BYTE   %123            ;NEEDS BOTH 0 AND 1
0002    4C D21C         JMP     INTSRV          ;DOS ERCM COMES HERE ON AN IRQ
0005    4C D11C         JMP     NMISRV          ;DOS ERCM COMES HERE ON AN NMI
                ;
                ;
        0020            .LOC    $0020           ;ALLOWS FOR RESIDENCY OF DEBUG MONITOR
                ;
                ;
0020    00      DEBGSW: .BYTE   0               ;SWITCH USED FOR TRACE FEATURE
0021    00      DMPSW:  .BYTE   0               ;FLAG USED FOR TRACE
0022    00      ATSW:   .BYTE   0               ;FLAG USED IN "ASK-TYPE" COMMANDS
0023    00      IFONSW: .BYTE   0               ;FLAG USED IN "IF-ON" COMMANDS
0024    00      INSW:   .BYTE   0               ;FLAG USED TO CONTROL WHERE INPUT COMES FROM
0025    00      MSCHAR: .BYTE   0               ;HOLDS THE "MODIFY" SEARCH CHAR
                ;
                ;       THE TEXT POINTERS
                ;

                                                ;*** FOLLOWING MUST BE CONTIGUOUS ***

0026    FF 00   PC:     .WORD   $00FF           ;POINTS TO CURRENT LINE
0028    00 01   TXTADR: .WORD   $0100           ;HOLDS ADDRESS OF START OF TEXT LINE
002A    00      TEXTP:  .BYTE   0               ;POINTER TO CHAR OF TEXT LINE
002B    00      CHAR:   .BYTE   0               ;HOLDS CURRENT CHAR WE ARE PROCESSING
002C    00      GRPNO:  .BYTE   0               ;WHERE GROUP NUMBER IS STORED
002D    00      LINENO: .BYTE   0               ;WHERE LINE (STEP) NUMBER IS STORED
002E    00      PRILVL: .BYTE   0               ;HOLDS CURRENT SOFTWARE INTERRUPT PRIORITY

                                                ;*** END OF MUST BE CONTIGUOUS ***
                ;
002F    04 25   TXTBEG: .WORD   PROGREG         ;POINTER TO BEGINNING OF STORED PROGRAM
                ;
0031    F2 25   PBADR:  .WORD   PBEG            ;ADDR OF START OF USER'S PROGRAM TEXT
                ;
                ;
                ;       ALTERNATE TEXT POINTER FOR SOME OPERATIONS
                ;
0033    00 00   TXTAD2: .WORD   0
0035    00      TEXTP2: .BYTE   0
0036    00              .BYTE   0               ;COUNTERPART TO 'CHAR' ABOVE
```

```
                        .PAGE                   ;'MORE PAGE ZERO STUFF'


                        ;       POINTER TO VARIABLE LIST
                        ;
                        ;                       ;*** START OF MUST BE CONTIGUOUS ***
                        ;
0037    00 00   VARADR: .WORD   0               ;POINTER TO CURRENT VARIABLE IN LIST
                        ;
0039    00 00   VSUB:   .WORD   0               ;LOCATION TO HOLD SUBSCRIPT DURING
                                                ;VARIABLE SEARCH
                FSWIT:
003B    00      STRSWT: .BYTE   0               ;FLAGS IF CURRENT VARIABLE IS A STRING VARIABLE
                        ;
                        ;                       ;*** END OF MUST BE CONTIGUOUS ***
                        ;
003C    00      VSIZE:  .BYTE   0               ;HOLDS SIZE OF CURRENT STRING VARIABLE (GETVAR)
                        ;
                TCHAR:
003D    00      VCHAR:  .BYTE   0               ;PLACE TO HOLD VARIABLE NAME DURING SEARCH
                        ;
                        ;
003E    F3 25   VARBEG: .WORD   VEND            ;POINTS TO BEGINNING OF VARIABLE LIST
                        ;
0040    F3 25   VARST:  .WORD   VEND            ;POINTS TO WHERE TO END VARIABLES ON AN
                                                ;"ERASE ALL" COMMAND
                        ;
0042    F3 25   VAREND: .WORD   VEND            ;POINTS TO END OF VARIABLE LIST
                        ;
                        ;       POINTERS FOR STRING I-O
                        ;
0044    00 00   STIADR: .WORD   0               ;HOLDS ADDR POINTER TO INPUT STRING
0046    00      STIPNT: .BYTE   0               ;OFFSET POINTER
0047    00      STIMAX: .BYTE   0               ;HOLDS SIZE OF INPUT STRING
                        ;
                        ;
0048    00 00   STOADR: .WORD   0               ;HOLDS ADDR POINTER TO OUTPUT STRING
004A    00      STOPNT: .BYTE   0               ;OFFSET POINTER
004B    00      STOMAX: .BYTE   0               ;HOLDS SIZE OF OUTPUT STRING
                        ;
```

```
                        .PAGE                   ;'MORE PAGE ZERO STUFF'
                        ;
                        ;
                        ;       SOFTWARE PUSHDOWN LIST POINTERS
                        ;
024C  00 00   PDPADR: .WORD   0               ;ADDRESS TO LIST
024E  00      PDP:    .BYTE   0               ;OFFSET POINTER TO LIST
                        ;
024F  00 00   PDPTMP: .WORD   0               ;TEMPORARY LOC USED BY 'ROOMCK'
                        ;
                        ;
0251  00      TGRP:   .BYTE   0               ;HOLDS TEMPORARY GROUP NUMBER FOR SOME
0252  00      TLINE:  .BYTE   0               ;OPERATIONS.  TEMP LINE NUMBER HERE
                        ;
                        ;
0253  20 2F   PDLIST: .WORD   PDLBEG           ;STARTING ADDR OF SOFTWARE PDL
                        ;
              0255      .DEF    STRAD1=.        ;DEFINE POINTERS TO STRING 1 IN 'FSLK'
                        STRCNT:                 ;TEMP COUNTER FOR SOME STRING OPERATIONS
0255  00      ITMP1L: .BYTE   0               ;LOW ORDER BYTE OF TEMPORARY 1
0256  00      ITMP1H: .BYTE   0               ;HIGH ORDER BYTE OF TEMPORARY 1
0257  00      SBEG1:  .BYTE   0               ;HOLDS STARTING POSITION ON STRING 1
0258  00      SEND1:  .BYTE   0               ;HOLDS ENDING POSITION IN STRING 1
              0259      .DEF    STRAD2=.        ;DEFINE POINTERS TO STRING 2 IN 'FSLK'
                        STRMAX:                 ;TEMP FOR SOME STRING OPERATIONS
0259  00      ITMP2L: .BYTE   0               ;LOW ORDER BYTE OF TEMPORARY 2
025A  00      ITMP2H: .BYTE   0               ;HIGH ORDER BYTE OF TEMPORARY 2
025B  00      SBEG2:  .BYTE   0               ;START OF STRING 2
025C  00      SEND2:  .BYTE   0               ;END OF STRING 2
025D  00      STBSAV: .BYTE   0               ;PLACE TO SAVE BEGINNING SUBSCRIPT IN 'FSTI'-'FSTO'
025E  4C      JSRIND: .BYTE   54C             ;'JMP' OPCODE FOR 'JSR JSRIND'
                                              ;MUST BE PLACED JUST BEFORE 'TEMP1'
025F  00 00   TEMP1:  .WORD   0               ;*** VERY TEMPORARY STORAGE!
                                              ;*** NOT NECESSARILY PRESERVED ACCROSS
                                              ;*** SUBROUTINE CALLS!
```

```
                        .PAGE                   ;'MORE PAGE ZERO STUFF'
                        ;
                        ;
                        ;
                        ;
0061  6C        JMPIND: .BYTE   $6C             ;OPCODE FOR 'JMP INDIRECT' INSTRUCTION
0062  00        PJMPL:  .BYTE   0               ;LOW ORDER ADDR USED IN 'JMPIND'
0063  00        PJMPH:  .BYTE   0               ;HIGH ORDER ADDR USED IN 'JMPIND'
                        ;
0064  48        STRSIZ: .BYTE   STRLEN          ;HOLDS DEFAULT STRING LENGTH
                        ;
0065  00        ETEMP1: .BYTE   0               ;TEMPORARY USED BY 'EVAL'
                        ;
0066  00        IDEV:   .BYTE   0               ;SPECIFIES INPUT DEVICE
0067  00        ODEV:   .BYTE   0               ;SPECIFIES OUTPUT DEVICE
0068  00        IDVSAV: .BYTE   0               ;PLACE TO SAVE INPUT DEVICE NUMBER
0069  00        ODVSAV: .BYTE   0               ;PLACE TO SAVE OUTPUT DEVICE NUMBER
006A  00        CONDEV: .BYTE   0               ;SPECIFIES DEVICE NUMBER OF CONSOLE
                        ;
006B  00        ECHFLG: .BYTE   0               ;FLAG USED TO SUPPRESS ECHO ON INPUT
                        ;
006C  00        DELSPL: .BYTE   0               ;IF NON-ZERO, INDICATES SPECIAL CRT DELETE PROCESSING
                        ;
006D  00        ACTMSK: .BYTE   0               ;HOLDS MASK WHICH INDICATES WHICH SOFTWARE
                                                ;INTERRUPT CHANNELS ARE CURRENTLY ACTIVE
                        ;
006E  00        EVMASK: .BYTE   0               ;HOLDS EVENT BITS SET BY EXTERNAL ROUTINES
                                                ;WHEN THEY WANT FOCAL'S ATTENTION.
                        ;
006F  C0        BITV1:  .BYTE   %300            ;USED TO SET V=1, N=1
                        ;
0070  10        MSKBRK: .BYTE   $10             ;BIT TEST MASK TO TEST FOR BRK BIT
                        ;
                        ;
0071  00        ACSAV:  .BYTE   0               ;PLACE WHERE AC IS SAVED ON INTERRUPT
0072  00        STATUS: .BYTE   0               ;PLACE WHERE PROCESSOR STATUS SAVED ON INTERRUPT
0073  00 00     ITEMP1: .WORD   0               ;PLACE WHERE RETURN ADDR KEPT IN INTERRUPT
```

```
                        .PAGE                   ;'MORE PAGE ZERO STUFF'

                        ;
                        ;
                        ;
0075    00      LASTOP: .BYTE   0               ;USED BY 'EVAL'
                        ;
0076    00      HASH:   .BYTE   0               ;LOADED WITH A RANDOM HASH VALUE
                                                ;*** NOTE: SOME ROUTINE IN YOUR SPECIFIC
                                                ;***       COMPUTER SYSTEM SHOULD LOAD THIS
                                                ;***       LOC WITH A PSUEDO-RANDOM 8-BIT
                                                ;***       VALUE.
                        ;
0077    00      SEED:   .BYTE   0               ;HOLDS A 23 BIT RANDOM NUMBER SEED
0078    00              .BYTE   0               ;SEE 'FRAN' FUNCTION
0079    00              .BYTE   0
                        ;
                        ;
                        ;
```

```
                        .PAGE                   ;'MORE PAGE ZERO STUFF'
                        ;
                        ;       PAGE ZERO STUFF USED BY THE FLOATING POINT ROUTINES.


        007A                    .DEF    X2M1=.

007A    00      SIGN:   .BYTE   0

        0078            .DEF    FAC2=.

0078    00      X2:     .BYTE   0               ;EXPONENT FOR FAC2
007C    00      M2:     .BYTE   0               ;MANTISSA FOR FAC2
007D    00              .BYTE   0
007E    00              .BYTE   0               ;*
007F    00      X1M1:   .BYTE   0

        0080            .DEF    FAC1=.

0080    00      X1:     .BYTE   0               ;EXPONENT FOR FAC1
                FLCSGN:
0081    00      M1:     .BYTE   0               ;MANTISSA FOR FAC1
0082    00              .BYTE   0
0083    00              .BYTE   0               ;*
0084    00      EM1:    .BYTE   0

0085    00      E:      .BYTE   0               ;SCRATCH
0086    00              .BYTE   0
0087    00              .BYTE   0
0088    00              .BYTE   0               ;*
0089    00              .BYTE   0
```

```
                        .PAGE                   ;'PAGE ZERO STUFF FOR FLOATING POINT


                        ;FOR FLOATING POINT INPUT AND OUTPUT ROUTINES

028A  00        SIGNP:  .BYTE   0               ;TEMPORARY SIGN
028B  00        DPFLG:  .BYTE   0               ;SAW DECIMAL POINT
028C  00        GOTFLG: .BYTE   0               ;IF NON-ZERO, WE ACTUALLY INPUT A NUMBER
028D  00        K:      .BYTE   0               ;MISC COUNTER
028E  00        L:      .BYTE   0               ;TEMPORARY
028F  05        M:      .BYTE   5               ;NUMBER OF FALCES BEFORE DECIMAL POINT
0290  05        N:      .BYTE   5               ;NUMBER OF PLACES AFTER DECIMAL POINT

0291  83        FTEN:   .BYTE   $83             ;CONSTANT 10.00
0292  50                .BYTE   $50
0293  00                .BYTE   0
0294  00                .BYTE   0
0295  00                .BYTE   0               ;*

0296  7F        FHALF:  .BYTE   $7F             ;CONSTANT 0.50
0297  40                .BYTE   $40
0298  00                .BYTE   0
0299  00                .BYTE   0
029A  00                .BYTE   0               ;*

029B  80        FONE:   .BYTE   $80             ;CONSTANT 1.0
029C  40                .BYTE   $40
029D  00                .BYTE   0
029E  00                .BYTE   0
029F  00                .BYTE   0               ;*

02A0  00        T:      .BYTE   0               ;TEMPORARY
02A1  00                .BYTE   0
02A2  00                .BYTE   0
02A3  00                .BYTE   0
02A4  00                .BYTE   0               ;*
                ;
                ;
02A5            .DEF    PGZERO=.                ;FLAG END OF OUR PAGE ZERO USAGE
                                                ;FOR ANYONE WHO NEEDS TO KNOW
```

```
                        .PAGE
                        ;TIM IO PACKAGE

                007F            .DEF    RUBCHR=$7F
                72C6            .DEF    SYSTOUT=$72C6   ;DEFINES TIM OUTPUT ROUTINE
                72EE            .DEF    SYSTIN=$72EE    ;DEFINES TIM INPUT ROUTINE
                6E02            .DEF    MPB=$6E02       ;DEFINE INPUT PORT LOCATION
                7320            .DEF    DLY1=$7320      ;TIM BIT DELAY ROUTINE
                7310            .DEF    DLY2=$7310      ;TWO BIT TIME DELAY

00A5- 7F        FILLCH: .BYTE   $7F             ;DEFINES FILL CHARACTER AFTER CR
00A6- FE FF     IVCTOR: .WORD   $FFFE           ;DEFINES INTERRUPT VECTOR LOCATION
00A8- 00        XKEYHL: .BYTE                   ;RANDOM NUMBER SEED

00A9- 29 7F     OUT:    AND#    $7F             ;STRIP PARITY BIT
00AB- 4C C672           JMP     SYSTOUT         ;**PJUMP* TO A CHARACTER

00AE- A2 08     IN:     LDX#    $08             ;SET X TO COUNT EIGHT BITS
00B0  E6 76     IN1:    INC     HASH            ;BUMP RANDOM SEED
00B2  AD 226E           LDA     MPB             ;READ PORT
00B5  4A                LSRA                    ;SHIFT BIT INTO CARRY FLAG
00B6  90 F8             BCC     IN1             ;BRANCH UNTIL START BIT
00B8  A5 68             LDA     ECHFLG          ;TEST IF WE ARE TO ECHO
00BA  D0 03             BNE     NOECH           ;BRANCH IF NOT
00BC  4C F672           JMP     SYSTIN+8        ;*PJMP* TO FINISH INPUT WITH ECHO

00BF  20 2073   NOECH:  JSR     DLY1            ;DELAY TO THE MIDDLE OF A BIT
00C2  20 1073   BIN2:   JSR     DLY2            ;DELAY TO THE MIDDLE OF THE NEXT CELL
00C5  AD 026E           LDA     MPB             ;READ PORT
00C8  4A                LSRA                    ;SHIFT BIT INTO CARY FLAG
00C9  08                PHP                     ;SAVE STATUS (AND BIT IN CARRY)
00CA  98                TYA                     ;GET BYTE FORMING IN Y
00CB  4A                LSRA                    ;MAKE ROOM FOR THE BIT
00CC  28                PLP                     ;GET THE STATUS BACK (WITH BIT IN CARRY)
00CD  90 02             BCC     BIN4            ;BRANCH IF BIT IS ZERO
00CF  09 80             ORA#    $80             ;PUT A ONE INTO THE BYTE
00D1  A8        BIN4:   TAY                     ;PUT IT BACK INTO Y
00D2  CA                DEX                     ;COUNT THE NUMBER OF BITS
00D3  D0 ED             BNE     BIN2            ;BRANCH UNTIL EIGHT LOADED
00D5  49 FF             EOR#    $FF             ;COMPLIMENT DATA (INVERTED INTO PORT)
00D7  20 1073           JSR     DLY2            ;DELAY FOR FIRST STOP BIT
00DA  4C 1073           JMP     DLY2            ;SECOND STOP BIT
```

```
                        .PAGE                   ;'FOCDEF - FOCAL SYMBOL DEFINITIONS'
                ;
                ;
                ;       ALL POSSIBLE SYMBOLIC EQUATES SHOULD BE KEPT IN THIS
                ;       MODULE.
                ;
                ;
                ;
                ;       FOCAL ERROR CODES
                ;
                ;
                ;       ERROR CODES ARE NEGATIVE
                ;
                ;
        00DB            .DEF    BASTRF=%333     ;-37    BAD OR MISSING ARGUMENT IN A STRING FUNCTION
        00DC            .DEF    SVRQ=%334       ;-36    STRING VARIABLE REQUIRED HERE
        00DD            .DEF    SVNA=%335       ;-35    STRING VARIABLE NOT ALLOWED HERE
        00DE            .DEF    ERRO=%336       ;-34    I-O ERROR ON OUTPUT DEVICE
        00DF            .DEF    ARGM=%337       ;-33    ARGUMENT MISSING IN FUNCTION
        00E0            .DEF    XNU32=%340      ;-32    CURRENTLY NOT USED
        00E1            .DEF    WNEXG=%341      ;-31    "WRITE" OF NON-EXISTANT GROUP
        00E2            .DEF    UNRFUN=%342     ;-30    UNRECOGNIZABLE FUNCTION NAME
        00E3            .DEF    PFERR=%343      ;-29    PARENTHESES ERROR IN FUNCTION
        00E4            .DEF    MNEXL=%344      ;-28    "MODIFY" OF NON-EXISTANT LINE
        00E5            .DEF    DONEXG=%345     ;-27    "DO" OF NON-EXISTANT GROUP
        00E6            .DEF    DONEXL=%346     ;-26    "DO" OF NON-EXISTANT LINE
        00E7            .DEF    IFSYN=%347      ;-25    SYNTAX ERROR IN "IF" OR "ON"  COMMAND
        00E8            .DEF    ENEXL=%350      ;-24    "ERASE" OF NON-EXISTANT LINE
        00E9            .DEF    ERRI=%351       ;-23    I-O ERROR ON INPUT DEVICE
        00EA            .DEF    WNEXL=%352      ;-22    "WRITE" OF NON-EXISTANT LINE
        00EB            .DEF    GONEXL=%353     ;-21    "GOTO" NON-EXISTANT LINE
        00EC            .DEF    BADLI=%354      ;-20    BAD LINE NUMBER ON INPUT
        00ED            .DEF    UNKINT=%355     ;-19    UNKNOWN INTERRUPT REQUEST
        00EE            .DEF    UNRBRK=%356     ;-18    UNRECOGNIZABLE TRAP CODE
        00EF            .DEF    RESBRK=%357     ;-17    RESET BUTTON PRESSED
        00F0            .DEF    DEVRNG=%360     ;-16    DEVICE NUMBER OUT OF RANGE
        00F1            .DEF    UFL=%361        ;-15    USELESS "FOR" LOOP
        00F2            .DEF    FBDTRM=%362     ;-14    BAD TERMINATOR IN "FOR"
        00F3            .DEF    NOEQLS=%363     ;-13    NO "=" IN "FOR"
        00F4            .DEF    BADVAR=%364     ;-12    BAD VARIABLE NAME
        00F5            .DEF    FUNILL=%365     ;-11    FUNCTION ILLEGAL HERE
        00F6            .DEF    XNU10=%366      ;-10    NOT USED AT THIS TIME
        00F7            .DEF    XNU9=%367       ;-9     NOT USED AT THIS TIME
        00F8            .DEF    FOVFL=%370      ;-8     FLOATING POINT OVERFLOW
        00F9            .DEF    OPNMIS=%371     ;-7     OPERAND MISSING - EVAL
        00FA            .DEF    PMATCH=%372     ;-6     PARENTHESES MISMATCH - EVAL
        00FB            .DEF    OPRMIS=%373     ;-5     OPERATOR MISSING - EVAL
        00FC            .DEF    ILLNO=%374      ;-4     ILLEGAL LINE NUMBER
        00FD            .DEF    BADCOM=%375     ;-3     UNRECOGNIZABLE COMMAND
        00FE            .DEF    EILLG0=%376     ;-2     ILLEGAL GROUP ZERO USAGE
        00FF            .DEF    LTL=%377        ;-1     LINE TOO LONG
```

```
                        .PAGE                   ;'FOCDEF - FOCAL SYMBOL DEFINITIONS
                ;
                ;
                ;       FUNCTION HASH CODE DEFINITIONS
                ;
                ;
        0036            .DEF    HFABS=%66       ;ABSOLUTE VALUE FUNCTION
        00F4            .DEF    HFOUT=%364      ;OUTPUT FUNCTION
        00B2            .DEF    HFRAN=%260      ;RANDOM NUMBER FUNCTION
        00A5            .DEF    HFINT=%253      ;INTEGERIZE FUNCTION
        00A4            .DEF    HFINR=%244      ;INTEGERIZE AFTER ROUNDING FUNCTION
        0084            .DEF    HFIDV=%204      ;SET INPUT DEVICE FUNCTION
        00B4            .DEF    HFODV=%264      ;SET OUTPUT DEVICE FUNCTION
        005C            .DEF    HFCHR=%134      ;SINGLE ALPHA CHAR INPUT FUNCTION
        0090            .DEF    HFCUR=%220      ;SPECIAL CURSOR ADDRESSING FUNCTION FOR CONSOLES
        0044            .DEF    HFECH=%104      ;ECHO CONTROL FUNCTION
        00AA            .DEF    HFPIC=%252      ;PRIORITY INTERRUPT CONTROL FUNCTION
        0096            .DEF    HFMEM=%226      ;MEMORY EXAMINE-DEPOSIT FUNCTION
        0092            .DEF    HFINI=%222      ;INITIALIZE INPUT DEVICE
        009E            .DEF    HFINO=%236      ;INITIALIZE OUTPUT DEVICE
        005A            .DEF    HFCLI=%132      ;CLOSE INPUT DEVICE FUNCTION
        0066            .DEF    HFCLO=%146      ;CLOSE OUTPUT DEVICE FUCNTION
        0070            .DEF    HFCON=%160      ;SET CONSOLE DEVICE FUNCTION
        00C4            .DEF    HFSBR=%304      ;'SUBROUTINE' CALL FUNCTION
        00AC            .DEF    HFISL=%254      ;INITIALIZE STRING LENGTH FUNCTION
        00FA            .DEF    HFSTI=%372      ;STRING INPUT FUNCTION
        0006            .DEF    HFSTO=%6        ;STRING OUTPUT FUNCTION
        00DE            .DEF    HFSLK=%336      ;STRING "LOCK" FUNCTION
                ;
```

```
                        .PAGE                   ;'FOCDEF - FOCAL SYMBOL DEFINITIONS'
                ;
                ;
                ;
                ;       MORE DEFINITIONS
                ;


        0005            .DEF    NUMBF=5         ;DEFINES NUMBER OF BYTES IN A FLOATING
                                                ;POINT NUMBER

        0008            .DEF    VARSIZ=3+NUMBF  ;SIZE OF A VARIABLE.
                                                ;1 BYTE FOR NAME.
                                                ;2 BYTES FOR SUBSCRIPT.
                                                ;'NUMBF' BYTES FOR VALUE

        007F            .DEF    LINEL=127       ;MAXIMUM LENGTH OF A FOCAL LINE PLUS ONE
                                                ;MUST NOT BE GREATER THAN 127!!! (SEE 'PACKC')
                ;
        00FF            .DEF    EOV=$FF         ;END OF VARIABLES MARKER
        00FE            .DEF    EOP=$FE         ;END OF PROGRAM MARKER
        00FD            .DEF    UMARK=$FD       ;ANOTHER UNIQUE MARKER
        00FC            .DEF    STRMRK=$FC      ;MARKS A "STRING" VARIABLE IN VARIABLE LIST
                ;
        0048            .DEF    STRLEN=72       ;DEFINE DEFAULT STRING LENGTH
                ;
        005C            .DEF    RUBECH=%134     ;DEFINES THE CHAR ('\') ECHOED FOR 'RUBOUT'
        005F            .DEF    LINCHR=%137     ;DEFINES THE CHAR ('←') USED FOR 'LINE-DELETE'
        001B            .DEF    ALTCHR=%33      ;DEFINES THE CHAR USED FOR 'ALTMODE'
                                                ;MAY REQUIRE CHANGING FOR SOME KEYBOARDS
                ;
        00FF            .DEF    DIRLIN=$FF      ;VALUE OF PC IF WE ARE EXECUTING A DIRECT LINE
        00FE            .DEF    STRLIN=$FE      ;VALUE OF PC IF WE ARE EXECUTING CODE IN A STRING
        00FD            .DEF    RETCMD=$FD      ;VALUE OF PC IF A 'RETURN' COMMAND WAS JUST EXECUTED
                ;
        0100            .DEF    STACK=$0100     ;BASE ADDR OF HARDWARE STACK AREA
                ;
                ;
        1000            .DEF    FOCADR=$1000    ;DEFINE STARTING ADDR OF MAIN INTERPRETER
                ;
        2F00            .DEF    PDLBEG=FOCADR+$1F00     ;DEFALLT IS SET FOR 8K SYSTEM
                                                ;*** BASE ADDR FOR START OF SOFTWARE PDL ***
                                                ;*** STACK ACTUALLY STARTS AT PDLBEG+$FF
                                                ;*** AND ADVANCES DOWNWARD IN MEMORY
                                                ;*** CHANGE THIS IF YOU HAVE MORE (OR LESS)
                                                ;*** MEMORY.
```

```
                        .PAGE                   ;'FOCINT - FOCAL CODE INTERPRETER'
                ;
                ;
                ;               ***** FOCINT - CODE INTERPRETER FOR "FOCAL" *****
                ;
                ;
                ;
                ;       THIS IS WHERE THE HEART OF FOCAL LIES.  INPUT PROCESSING,
                ;     COMMAND DECODING, CONTROL FLOW, AND EXECUTION ARE PERFORMED
                ;     IN THIS MODULE.
                ;
                ;
```

```
                        .PAGE                   ;'"RESET" LOGIC'
                    ;
                    ;
                    ;
                    ;       ***** STARTING ADDRESS OF "FOCAL" *****
                    ;
                        1000        .LOC    FOCADR          ;START AT PROPER POINT

                    ;       'FOCAL' COMES HERE ON A 'RESET' (PANIC RESTART)

1000    78      FOCAL:  SEI                     ;MAKE SURE INTERRUPTS ARE DISABLED
1001    D8              CLD                     ;AND DECIMAL MODE IS OFF
1002    A2 FF           LDX#    $FF             ;INIT HARDWARE STACK POINTER
1004    9A              TXS
1005    20 3D10         JSR     SETUP           ;SETUP SOME INITIAL PARAMETERS
1008    20 2710         JSR     INIDEV          ;INITIALIZE THE CONSOLE DEVICE
100B    00              BRK                     ;ENTER FOCAL VIA TRAP
100C    EF              .BYTE   RESBRK          ;INDICATE 'RESET' BUTTON PRESSED


100D    A2 FF   SETUP:  LDX#    $FF             ;INIT SOFTWARE PUSH DOWN LIST
100F    20 1C10         JSR     PDPINI
1012    E8              INX                     ;FORM A ZERO IN X REGISTER
1013    86 68           STX     ECHFLG          ;MAKE SURE CHARACTER ECHOING IS ON
                                                ;* FFALL * INTO 'CLRDEV' TO RESET TO CONSOLE I-O

1015    A5 6A   CLRDEV: LDA     CONDEV          ;SET CONSOLE AS BOTH INPUT AND OUTPUT DEVICES
1017    85 66           STA     IDEV
1019    85 67           STA     ODEV
101B    60              RTS                     ;AND RETURN

                        ;ROUTINE TO INITIALIZE THE SOFTWARE PUSHDOWN LIST POINTER

101C    86 4E   PDPINI: STX     PDP             ;ASSUMED TO CONTAIN '$FF' IN X REG
101E    A5 53           LDA     PDLIST          ;RESET ADDR POINTER
1020    85 4C           STA     PDPADR
1022    A5 54           LDA     PDLIST+1
1024    85 4D           STA     PDPADR+1
1026    60              RTS                     ;AND RETURN

                        ;ROUTINE TO INITIALIZE THE INPUT AND OUTPUT DEVICE

1027    A5 66   INIDEV: LDA     IDEV            ;INITIALIZE INPUT DEVICE
1029    20 7F1E         JSR     INI
102C    A5 67           LDA     ODEV            ;INITIALIZE OUTPUT DEVICE
102E    4C 941E         JMP     INO             ;* PJMP * AND RETURN
```

```
                        .PAGE                   ;'COMMAND TYPE-IN PROCESSING LOOP'
                ;       'FOCAL'  COMES HERE ON AN INTERNAL RESTART.
1031  A2 FF   START:  LDX#    $FF             ;INITIALIZE STACK POINTER
1033  9A              TXS
1034  86 21           STX     DMPSW           ;FLAG THE TRACE OFF
1036  86 25           STX     PC              ;INDICATE WE ARE PROCESSING A DIRECT COMMAND
1038  E8              INX                     ;GET A ZERO
1039  86 20           STX     DEBGSW          ;ALLOW TRACE TO BE ENABLED
103B  86 28           STX     TXTADR          ;INIT TEXT POINTERS TO COMMAND BUFFER
103D  86 2A           STX     TEXTP           ;RESET OFFSET POINTER TO TEXT LINE
103F  86 6D           STX     ACTMSK          ;INDICATE ALL SOFTWARE INTERRUPT CHANNELS ARE INACTIVE
1041  86 6E           STX     EVMASK          ;CLEAR ANY EXTERNAL EVENT FLAGS
1043  86 2E           STX     PRILVL          ;AND INDICATE THAT WE ARE AT LOWEST PRIO LEVEL
1045  E8              INX                     ;GET A 1
1046  86 29           STX     TXTADR+1        ;COMMAND BUFFER IS IN STACK AREA
1048  A9 2A           LDA#    '*              ;LOAD OUR PROMPTING CHARACTER
104A  20 FA18         JSR     PRINTC          ;AND ANNOUNCE OUR PRESENCE
104D  20 CD18 NEXTIC: JSR     READC           ;GO INPUT A CHAR FROM INPUT DEVICE
1050  C9 0A           CMP#    %12             ;LINE FEED?
1052  F0 F9           BEQ     NEXTIC          ;BRANCH IF YES, IGNORE LINE FEEDS ON COMMAND INPUT
1054  20 1F19         JSR     PACKC1          ;STORE CHAR IN COMMAND BUFFER
1057  C9 0D           CMP#    %15             ;CARRIAGE RETURN?
1059  D0 F2           BNE     NEXTIC          ;IF NOT, GO GET ANOTHER
                                              ;IF YES, FALL INTO IRETN
                ;
```

```
                                .PAGE                   ;'COMMAND/INPUT PROCESSOR'
                        ;
                        ;
                        ;
1058  A2 00     IRETN:  LDX#    0                       ;RESET TEXT POINTER
105A  86 2A             STX     TEXTP
105C  86 24             STX     INSW                    ;FLAG TEXT CHAR TO COME FROM MEMORY
105E  CA                DEX                             ;SET TO $FF
105F  20 1C10           JSR     PDPINI                  ;INIT SOFTWARE PDL POINTER
1062  20 A319           JSR     GSPNOR                  ;IGNOR LEADING BLANKS
1065  20 B219           JSR     TESTN1                  ;HAVE A NUMBER?
1068  90 06             BCC     GOTNUM                  ;YES, ITS A NUMBER
106A  C9 2E             CMP#    '.                      ;NO, IS IT A '.'?
106C  D0 13             BNE     INPX1                   ;NOT NUMBER OR '.', MUST BE A DIRECT COMMAND
106E  00                BRK                             ;TRAP (HAVE A '.')
106F  FE                .BYTE   EILLG0                  ;?ILLEGAL GROUP ZERO USAGE
1070  E6 20     GOTNUM: INC     DEBGSW                  ;A DIGIT, DISABLE TRACE FOR PACKING
1072  20 C315           JSR     GETLNC                  ;GET THE LINE NUMBER FROM COMBUF
1075  70 08             BVS     IBADL                   ;00.00 IS A BAD LINE NUMBER
1077  F0 06             BEQ     IBADL                   ;GG.00 IS A BAD LINE NUMBER
1079  20 9117 REPLIN: JSR     INSERT                  ;GO INSERT THIS LINE IN THE TEXT AREA
107C  4C 3110 RSTART: JMP     START                   ;AND START OVER FOR NEXT DIRECT COMMAND
                        ;
107F  00        IBADL:  BRK                             ;TRAP
1080  EC                .BYTE   BADLI                  ;?BAD LINE NUMBER ON INPUT
                        ;
                        ;
                        ;
1081  20 2918  INPX1:  JSR     PUSHJ                   ;PROCESS COMMAND
1084  04 13             .WORD   PROC
1086  A5 26             LDA     PC                      ;GET PROGRAM LINE NUMBER
1088  30 A4             BMI     START                   ;START OVER IF DIRECT COMMAND
108A  20 E416           JSR     EATCR1                  ;EAT TO END OF LINE
108D  20 3F17           JSR     NXTLIN                  ;GO SETUP THE POINTERS TO NEXT LINE
1090  B0 EF             BCS     INPX1                   ;BRANCH IF MORE TO DO
1092  90 9A             BCC     START                   ;BRANCH IF END OF PROGRAM
```

```
                        .PAGE                   ;' ROUTINE TO 'DO' CODE STORED IN A STRING '


                        ;THIS ROUTINE ALLOWS THE EXECUTION OF CODE STORED
                        ;IN A STRING VARIABLE.  IF ONE PLACES CHARACTERS INTO A STRING
                        ;VARIABLE, AND ENDS THEM WITH A CARRIAGE RETURN, THEN A 'DO'
                        ;COMMAND CAN BE USED TO PERFORM THE LINE STORED IN
                        ;THE STRING VARIABLE.

1097  20 B910 DOSTR:    JSR     PUSHDO          ;SAVE CURRENT GOODIES ON STACK
109A  A5 37             LDA     VARADR          ;NOW POINT TO SPECIFIED CHAR IN STRING
109C  85 28             STA     TXTADR          ;WITH TEXT POINTERS
109E  A5 38             LDA     VARADR+1
10A0  85 29             STA     TXTADR+1
10A2  A5 3A             LDA     VSUB+1          ;GET CHAR POSITION IN STRING
10A4  85 2A             STA     TEXTP
10A6  A9 FE             LDA#    STRLIN          ;FLAG THIS
10A8  85 26             STA     PC              ;SO 'FNDLIN' WILL ALWAYS START LOOKING
                                                ;FROM BEGINNING OF PROGRAM TEXT.
10AA  A5 30             LDA     VCHAR           ;STORE THE STRING VARIABLE'S NAME FOR
10AC  85 27             STA     PC+1            ;ERROR MESSAGE PRINT ROUTINE (BERROR).
10AE  20 2918           JSR     PUSHJ           ;NOW "DO" THE STRING. (IT SHOULD HAVE A CR IN IT!)
10B1  21 13             .WORD   PROCES
10B3  20 C010           JSR     POPDO           ;RESTORE CURRENT GOODIES
10B6  20 5818           JSR     POPJ            ;AND RETURN FROM 'DO'.


                        ;ROUTINE TO SAVE NEEDED INFO ON STACK.  USED BY 'DO'

10B9  A2 26   PUSHDO:   LDX#    PC              ;GET ADDR OF START OF SAVE AREA
10BB  A0 09             LDY#    9               ;AND NUMBER OF BYTES TO SAVE
10BD  4C 7818           JMP     PUSHB0          ;* PJMP * PUSH THEM ON STACK AND RETURN

                        ;ROUTINE TO RESTORE NEEDED INFO AFTER THE 'DO'

10C0  A2 2E   POPDO:    LDX#    PC+8            ;GET BASE ADDR TO PUT INFO BACK INTO
10C2  A0 09             LDY#    9               ;AND NUMBER OF BYTES TO RESTORE
10C4  4C 6818           JMP     POPB0           ;* PJMP * POP THEM OFF AND RETURN
```

```
                        .PAGE                   ;'"IF" COMMAND PROCESSOR'
                        ;
                        ;
                        ;       "IF" CONDITIONAL "GOTO" COMMAND PROCESSOR
                        ;
                        ;
                        ;       THROW IN THE "QUIT" COMMAND HERE ALSO
                        ;
                        ;
1031                    .DEF    QUIT=START      ;QUIT COMMAND RE-STARTS FOCAL
                        ;
                        ;
1BC7  20 3E18 ON:       JSR     PUSHA           ;REMEMBER WHO WE ARE (IF OR ON)
1BCA  20 A319 IFON:     JSR     GSPNOR          ;MOVE TO NEXT NON-BLANK
1BCD  C9 28             CMP#    '(              ;LEFT PAREN?
1BCF  F0 02             BEQ     IFCNT1          ;BRANCH IF YES
1BD1  00                BRK                     ;NO, THEN TRAP
1BD2  E7                .BYTE   IFSYN           ;?SYNTAX ERROR IN "IF" OR "ON" COMMAND
1BD3  20 2918 IFCNT1:   JSR     PUSHJ           ;NOW EVALUATE WHAT'S IN PARENS
1BD6  F5 19             .WORD   EVALM1
1BD8  A5 28             LDA     CHAR            ;GET TERMINATOR
1BDA  C9 29             CMP#    ')              ;CLOSING PAREN?
1BDC  F0 02             BEQ     IFCNT2          ;YES, CONTINUE
1BDE  00                BRK                     ;NO, TRAP
1BDF  FA                .BYTE   PMATCH          ;?PARENTHESES MISMATCH
1BE0  20 8419 IFCNT2:   JSR     GETC            ;MOVE PAST RIGHT PAREN
1BE3  A5 81             LDA     FLOSGN          ;GET THE SIGN OF THE EXPRESSION
1BE5  30 26             BMI     IFXCT           ;IF NEGATIVE, GO DO IT NOW
1BE7  10 03             BPL     IF3             ;OTHERWISE LOOK FOR COMMA
1BE9  20 8419 IFCOM:    JSR     GETC            ;GET A CHAR
1BEC  A5 28   IF3:      LDA     CHAR
1BEE  C9 2C             CMP#    ',              ;COMMA?
1BF0  F0 0E             BEQ     IF1             ;BRANCH IF YES
1BF2  C9 3B             CMP#    ';              ;SEMI-COLON?
1BF4  F0 20             BEQ     IFNOP           ;YES, CONTINUE WITH NEXT COMMAND ON THIS LINE
1BF6  C9 0D             CMP#    %15             ;CARRIAGE RETURN?
1BF8  D0 EF             BNE     IFCOM           ;NO, KEEP LOOKING
1BFA  20 4C18           JSR     POPA            ;ADJUST STACK
1BFD  20 5818           JSR     POPJ            ;YES, THEN-EXIT "PROCESS"
1C00  20 8419 IF1:      JSR     GETC            ;MOVE PAST THE COMMA
1C03  20 2624           JSR     CHKZER          ;IS FAC1=0 ?
1C06  F0 05             BEQ     IFXCT           ;BRANCH IF FAC1=0
1C08  20 BD1C           JSR     ZRFAC1          ;SET IT TO ZERO, THIS TIME FOR SURE
1C0B  30 DF             BMI     IF3             ;UNCONDITIONAL BRANCH
1C0D  20 C915 IFXCT:    JSR     GETLNS          ;GO GET THE LINE NUMBER
1C10  08                PHP                     ;SAVE PROCESSOR FLAGS
1C11  20 0B17           JSR     EATECM          ;NOW EAT TILL THE END OF A COMMAND
1C14  28                PLP                     ;GET FLAGS BACK
1C15  70 0C             BVS     IFNOP           ;BRANCH IF NO LINE NUMBER GIVEN
1C17  08                PHP                     ;SAVE THEM AGAIN
1C18  20 4C18           JSR     POPA            ;GET 'WHO WE ARE' SWITCH (IF OR ON)
1C1B  C9 4F             CMP#    'O              ;WAS THIS AN "ON" COMMAND?
1C1D  F0 0A             BEQ     IFDO1           ;YES, GO "DO" THE LINE
1C1F  28                PLP                     ;ADJUST STACK
```

```
690  1120  4C F712          JMP     GOTO1           ;NO, "IF" COMMAND, THEN "GOTO" LINE
691  1123  20 4C18 IFNOP:   JSR     POPA            ;ADJUST STACK
692  1126  4C 0413          JMP     PROC            ;NO LINE NUMBER GIVEN, CONTINUE WITH
693                                                  ;NEXT COMMAND ON THIS LINE
694  1129  68       IFDO1:  PLA                     ;GET STATUS SAVED AFTER 'GETLN' CALL
695  112A  AA               TAX                     ;SAVE IN X REGISTER
696  112B  20 2918          JSR     PUSHJ           ;NOW PERFORM THE 'DO' OF THE LINE OR GROUP
697  112E  B6 11            .WORD   DO1
698  1130  4C 0413          JMP     PROC            ;AND THEN CONTINUE PROCESSING ON THIS LINE
699                 }
700       10C7              .DEF    IF=ON           ;BOTH COMMANDS HAVE COMMON ENTRY POINT
701                 }
```

```
                        .PAGE                   ;'"MODIFY" COMMAND PROCESSOR'
                        ;
                        ;
                        ;       "MODIFY" A LINE OF THE USER'S PROGRAM
                        ;
1133  20 C915 MODIFY:   JSR     GETLNS          ;GET THE LINE NUMBER SPECIFIED
1136  70 02             BVS     BADMOD          ;BRANCH IF 00.00
1138  D0 02             BNE     MODNOK          ;BRANCH IF GG,LL
113A  00          BADMOD: BRK                     ;TRAP
113B  E4                  .BYTE   MNEXL           ;?MODIFY OF NON-EXISTANT LINE
113C  20 8D16 MODNOK: JSR     FINDLN          ;TRY TO FIND THE LINE
113F  90 F9               BCC     BADMOD          ;BRANCH IF COULD NOT FIND
1141  20 BB1D             JSR     CRLF            ;OUTPUT A BLANK LINE
1144  20 3E16             JSR     PRNTLN          ;PRINT THE LINE NUMBER
1147  A0 02               LDY#    2               ;SET OFFSET TO FIRST CHAR ON THE LINE
1149  84 35               STY     TEXTP2
114B  A9 00               LDA#    0
114D  85 28               STA     TXTADR          ;INIT POINTERS TO COMBUF
114F  85 2A               STA     TEXTP           ;OFFSET OF ZERO
1151  A9 01               LDA#    1
1153  85 29               STA     TXTADR+1
1155  20 8D18 MNXTC:   JSR     RNOECH          ;GO WAIT FOR THE GUY TO TYPE A CHAR
1158  C9 0A               CMP#    %12             ;LINE FEED?
115A  F0 25               BEQ     MLOOK1          ;YES, THEN FEED TO END OF LINE
115C  C9 1B 06 AF         CMP#    ALTCHR          ;ALTMODE?
115E  F0 1E               BEQ     MLOOK           ;YES, THEN PICK UP SEARCH CHARACTER
1160  C9 7F               CMP#    RUBCHR          ;RUBOUT
1162  F0 07               BEQ     RNOECH          ;YES, THEN DON'T ECHO
1164  C9 5F               CMP#    LINCHR          ;WAS CHAR THE 'LINE-DELETE' CHARACTER?
1166  F0 03               BEQ     MNOECH          ;BRANCH IF IT IS, DO NOT ECHO
1168  20 FA18             JSR     PRINTC          ;ECHO THE CHARACTER
116B  20 1F19 MNOECH: JSR     PACKC1          ;PACK CHAR INTO COMBUF
116E  C9 0D               CMP#    %15             ;WAS CHAR A CARRIAGE RETURN?
1170  D0 E3               BNE     MNXTC           ;NO, THEN PICK UP NEXT ONE
1172  20 901D MENDL:   JSR     OUTLF           ;FOLLOW WITH A LINE FEED
1175  A9 01               LDA#    1               ;SET COMBUF OFFSET
1177  85 2A               STA     TEXTP           ;FOR 'INSERT'
1179  E6 28               INC     DEBGSW          ;DISABLE TRACE FOR INSERT
117B  4C 7C10             JMP     REPLIN          ;AND GO REPLACE OLD LINE WITH EDITED LINE
                        ;
117E  20 8D18 MLOOK:  JSR     RNOECH          ;SILENTLY GET THE SEARCH CHAR
1181  85 25   MLOOK1: STA     MSCHAR          ;STORE SEARCH CHAR
1183  A4 35   MLOOK2: LDY     TEXTP2          ;GET POINTER TO TEXT IN MEMORY
1185  B1 33               LDA@Y   TXTAD2          ;GET CHAR
1187  C8                  INY                     ;POINT TO NEXT
1188  84 35               STY     TEXTP2
118A  20 1F19             JSR     PACKC1          ;PACK THE CHAR
118D  20 FA18             JSR     PRINTC          ;ECHO IT
1190  C5 25               CMP     MSCHAR          ;WAS IT THE SEARCH CHAR?
1192  F0 C1               BEQ     MNXTC           ;BRANCH IF YES
1194  C9 0D               CMP#    %15             ;CARRIAGE RETURN?
1196  F0 DA               BEQ     MENDL           ;YES, THEN THAT'S IT
1198  D0 E9               BNE     MLOOK2          ;NO, THEN KEEP LOOKING
```

```
                        .PAGE                   ;'"DO" COMMAND PROCESSOR'
                        ;
                        ;
                        ;       "DO" RECURSIVE OPERATE, EXECUTE, OR CALL
                        ;
119A  20 2918   DO:     JSR     PUSHJ           ;CALL THE "DO" SUBROUTINE
119D  A2 11             .WORD   DOX
119F  4C 0413           JMP     PROC            ;AND CONTINUE PROCESSING
                        ;
11A2  A9 00     DOX:    LDA#    0               ;ZERO OUT THE STRING VARIABLE SWITCH
11A4  85 3B             STA     STRSWT
11A6  20 C915           JSR     GETLNS          ;GO GET THE LINE NUMBER TO "DO"
11A9  08                PHP                     ;SAVE FLAGS FROM 'GETLNS'
11AA  A5 3B             LDA     STRSWT          ;WAS EXPRESSION A STRING VARIABLE?
11AC  F0 04             BEQ     DOX1            ;BRANCH IF NOT, PRESS ON
11AE  28                PLP                     ;YES, THEN ADJUST STACK
11AF  4C 9710           JMP     DOSTR           ;* JUMP * CC CODE STORED IN STRING, THEN RETURN
11B2  28        DOX1:   PLP                     ;RESTORE FLAGS FROM 'GETLNS'
11B3  4C B911           JMP     DO2             ;AND ENTER ROUTINE
11B6  8A        DO1:    TXA                     ;ENTER HERE WITH STATUS FROM 'GETLN' IN X
11B7  48                PHA                     ;SAVE ON STACK
11B8  28                PLP                     ;MAKE CURRENT PROCESSOR STATUS
11B9  08        DO2:    PHP                     ;SAVE PROCESSOR STATUS ACCROSS THE SAVE
11BA  20 B910           JSR     PUSHDO          ;SAVE IMPORTANT STUFF ON STACK FOR LATER
11BD  28                PLP                     ;GET THE FLAGS BACK
11BE  70 17             BVS     DOGRP           ;BRANCH IF "DO ALL"
11C0  F0 15             BEQ     DOGRP           ;OR "DO" OF A GROUP
11C2  20 B016   DOONE:  JSR     FINDLN          ;TRY TO FIND THE LINE TO "DO"
11C5  B0 02             BCS     DOCNT1          ;BRANCH IF WE FOUND IT
11C7  00                BRK                     ;TRAP
11C8  E6                .BYTE   DONEXL          ;?"DO" OF NON-EXISTANT LINE
11C9  20 3217   DOCNT1: JSR     NEWLIN          ;SETUP THE NEW LINE
11CC  20 2918           JSR     PUSHJ           ;AND EXECUTE IT
11CF  01 13             .WORD   PROCES
11D1  20 C010   DOCONT: JSR     POPDO           ;NOW RESTORE IMPORTANT STUFF AS IT WAS
11D4  20 5818           JSR     POPJ            ;AND RETURN
```

```
                        .PAGE                   ;'"DO" COMMAND PROCESSOR'
                        ;
                        ;
                        ;
1107  20 8D16 DOGRP:    JSR     FINDLN          ;TRY TO LOCATE SMALLEST LINE OF THE GROUP
110A  A5 51             LDA     TGRP            ;IS IT THE SAME GROUP WE WERE LOOKING FOR?
110C  C5 2C             CMP     GRPNO
110E  F0 02             BEQ     DOGRP1          ;BRANCH IF YES
1110  00                BRK                     ;TRAP
1111  E5                .BYTE   DONEXG          ;?"DO" OF NON-EXISTANT GROUP
1112  A5 51   DOGRP1:   LDA     TGRP            ;GET THE GROUP NUMBER WE ARE "DO"ING
1114  20 3E18           JSR     PUSHA           ;SAVE ON STACK
1117  20 3F17 DOGRPC:   JSR     NXTLIN          ;SET UP POINTERS FOR NEXT LINE
111A  90 1C             BCC     ENDGRP          ;BRANCH IF END OF PROGRAM
111C  20 4C18           JSR     POPA            ;GET GROUP NUMBER WE ARE "DO"ING
111F  20 3E18           JSR     PUSHA           ;SAVE IT AGAIN
1122  09 00             ORA#    0               ;ARE WE DOING GROUP ZERO (ALL)?
1124  F0 04             BEQ     DONEXT          ;YES, THEN ANY LINE IS OK
1126  C5 26             CMP     PC              ;IS THIS LINE OF THE SAME GROUP?
1128  D0 0E             BNE     ENDGRP          ;BRANCH IF NOT
112A  20 2918 DONEXT:   JSR     PUSHJ           ;YES, THEN PROCESS THIS LINE
112D  01 13             .WORD   PROCES
112F  20 E416           JSR     EATCR1          ;EAT TILL A CARRIAGE RETURN
1132  A5 26             LDA     PC              ;GET THE CURRENT LINE NUMBER
1134  C9 FD             CMP#    RETCMD          ;'RETURN' COMMAND SEEN?
1136  D0 DF             BNE     DOGRPC          ;BRANCH IF NOT, CONTINUE
1138  20 4C18 ENDGRP:   JSR     POPA            ;ADJUST STACK
113B  4C 0111           JMP     DOCONT          ;AND RETURN FROM "DO"

                        ;       "RETURN" AND "RESTORE" COMMANDS

113E  20 A619 RETURN:   JSR     SPNOR           ;GET NEXT NON-BLANK
1141  C9 49             CMP#    'I              ;IS THIS A "RESTORE INPUT" (R I)?
1143  F0 0B             BEQ     RESINP          ;BRANCH IF YES
1145  C9 4F             CMP#    'O              ;IS THIS A "RESTORE OUTPUT" (R O)?
1147  F0 0E             BEQ     RESOUT          ;BRANCH IF YES
1149  A9 FD             LDA#    RETCMD          ;NO, THEN IT MUST BE A "RETURN"
114B  85 26             STA     PC              ;SET PC TO SPECIAL VALUE
114D  20 5B16           JSR     POPJ            ;AND EXIT "PROCESS"

1150  A5 68   RESINP:   LDA     IDVSAV          ;RESTORE INPUT DEVICE NUMBER TO WHAT IT WAS
1152  85 66             STA     IDEV            ;BEFORE LAST STRING INPUT
1154  4C 0113           JMP     PROCESS         ;AND CONTINUE EXECUTING ON THIS LINE
1157  A5 69   RESOUT:   LDA     ODVSAV          ;RESTORE OUTPUT DEVICE NUMBER TO WHAT IT WAS
1159  85 67             STA     ODEV            ;BEFORE LAST STRING OUTPUT
115B  4C 0113           JMP     PROCES          ;AND CONTINUE EXECUTING ON THIS LINE
```

```
                        .PAGE                   ;'ERASE COMMAND PROCESSOR'
                        ;
                        ;
                        ;       "ERASE" COMMAND PROCESSOR
                        ;
                        ;
122E  20 A619 ERASE:    JSR     SPNOR           ;GO GET NEXT NON-BLANK
1231  C9 41             CMP#    'A              ;ERASE "ALL"?
1233  F0 25             BEQ     EALL            ;BRANCH IF YES
1235  20 CC15           JSR     GETLN           ;NO, GO GET THE LINE NUMBER TO ERASE
1238  08                PHP                     ;SAVE FLAGS
1239  20 1717           JSR     PUSHTP          ;SAVE TEXT POINTERS
123C  28                PLP                     ;GET FLAGS BACK
123D  70 0F             BVS     EVAR            ;BRANCH IF JUST "ERASE" WITH NO LINE NUMBER
123F  F0 30             BEQ     EGRP            ;BRANCH IF WE ERASE A GROUP
1241  20 8D16           JSR     FINDLN          ;NOW TRY AND LOCATE THE SPECIFIED LINE
1244  B0 02             BCS     ERCONT          ;BRANCH IF WE FOUND IT
1246  00                BRK                     ;TRAP
1247  E8                .BYTE   ENEXL           ;?ERASE OF NON-EXISTANT LINE
1248  20 5817 ERCONT: JSR     DELETE          ;GO ZAP THE LINE
124B  4C 3110 ERDON:  JMP     START           ;AND GO TO DIRECT COMMAND MODE
124E  A0 00   EVAR:   LDY#    0               ;SET OFFSET TO ZERO
1250  A9 FF            LDA#    EOV             ;ERASE ALL THE VARIABLES
1252  91 3E            STA@Y   VARBEG          ;BY FLAGING LIST AS EMPTY
1254  20 1E17          JSR     POPTP           ;RESTORE TEXT POINTERS
1257  4C 0413          JMP     PROC            ;AND CONTINUE PROCESSING THIS LINE
125A  A0 00   EALL:   LDY#    0               ;SET OFFSET TO ZERO
125C  A9 FE            LDA#    EOP             ;ERASE PROGRAM TEXT
125E  91 31            STA@Y   PBADR           ;EXCEPT GROUP ZERO
1260  A5 40            LDA     VARST           ;UPDATE VARBEG
1262  85 3E            STA     VARBEG
1264  A5 41            LDA     VARST+1
1266  85 3F            STA     VARBEG+1
1268  A9 FF            LDA#    EOV
126A  91 3E            STA@Y   VARBEG          ;FLAG VARIABLE LIST AS EMPTY
126C  D0 DD            BNE     ERDON           ;AND START OVER
                        ;
126E  20 5817 ELINE:  JSR     DELETE          ;DELETE THE LINE
1271  20 2517 EGRP:   JSR     TXTINI          ;MUST RESET TEXT POINTER TO START OF PROGRAM
1274  20 8D16           JSR     FINDLN          ;TRY TO FIND THE GROUP
1277  A5 51            LDA     TGRP            ;GET GROUP NUMBER OF LINE FOUND
1279  C5 2C            CMP     GRPNO           ;IS IT OF THE GROUP WE ARE DELETING?
127B  F0 F1            BEQ     ELINE           ;BRANCH IF YES, GO DELETE IT
127D  D0 CC            BNE     ERDON           ;AND RETURN IF ALL LINES IN GROUP DELETED
                        ;
                        ;
                        ;
```

```
                        .PAGE                   ;'WRITE COMMAND PROCESSOR'
                        ;
                        ;
                        ;       "WRITE" OUT PARTS OF THE PROGRAM TEXT
                        ;
                        ;
127F  20 C915 WRITE:    JSR     GETLNS          ;GO GET THE LINE OR GROUP TO WRITE
1282  08                PHP                     ;SAVE CONDITION CODES
1283  20 1717           JSR     PUSHTP          ;SAVE POSITION ON THIS LINE
1286  E6 20             INC     DEBGSW          ;DISABLE TRACE
1288  28                PLP                     ;GET CONDITION CODES AFTER 'GETLN'
1289  70 28             BVS     WALL            ;BRANCH IF "WRITE ALL"
128B  D0 19             BNE     WLINE           ;BRANCH IF WE WRITE A SINGLE LINE
128D  20 8D16 WGRP:     JSR     FINDLN          ;TRY TO FIND THE GROUP
1290  A5 51             LDA     TGRP            ;GET THE GROUP LOCATED IN MEMORY
1292  C5 20             CMP     GRPNO           ;IS IT THE ONE WE ARE LOOKING FOR?
1294  F0 02             BEQ     WGRP1           ;BRANCH IF YES
1296  00                BRK                     ;TRAP
1297  E1                .BYTE   WNEXG           ;?"WRITE" OF NON-EXISTANT GROUP
1298  20 C212 WGRP1:    JSR     WGRPO           ;OUTPUT THE GROUP
129B  20 ED12 WEXIT:    JSR     WCRLF           ;BLANK LINE
129E  C6 20             DEC     DEBGSW          ;ENABLE TRACE AGAIN
12A0  20 1E17           JSR     POPTP           ;RESTORE TEXT POINTERS
12A3  4C 0413           JMP     PROC.           ;AND CONTINUE PROCESSING ON THIS LINE
12A6  20 8D16 WLINE:    JSR     FINDLN          ;TRY TO FIND THE LINE
12A9  B0 02             BCS     WLINE1          ;BRANCH IF WE FOUND IT
12AB  00                BRK                     ;BRK
12AC  EA                .BYTE   WNEXL           ;?"WRITE" OF NON-EXISTANT LINE
12AD  20 ED12 WLINE1:   JSR     WCRLF           ;OUTPUT BLANK LINE
12B0  20 CF12           JSR     WONE            ;OUTPUT THE SINGLE LINE
12B3  4C 9B12           JMP     WEXIT           ;THEN EXIT "WRITE"
12B6  20 8D16 WALL:     JSR     FINDLN          ;GO FIND GROUP ZERO (ALWAYS WINS)
12B9  20 C212 WALL1:    JSR     WGRPO           ;OUTPUT THIS GROUP
12BC  F0 DD             BEQ     WEXIT           ;BRANCH IF END OF PROGRAM
12BE  85 51             STA     TGRP            ;MAKE NEXT GROUP THE GROUP TO OUTPUT
12C0  D0 F7             BNE     WALL1           ;AND GO OUTPUT IT
```

```
                                .PAGE                   ;'MORE WRITE COMMAND PROCESSOR'
                        ;
                        ;
                        ;
                        ;
12C2    20 ED12 WGRP0:  JSR     WCRLF           ;OUTPUT BLANK LINE
12C5    20 CF12 WGRP01: JSR     WONE            ;OUTPUT THIS LINE
12C8    F0 04           BEQ     WEXGRP          ;BRANCH IF END OF PROGRAM
12CA    C5 51           CMP     TGRP            ;IS GROUP OF NEXT LINE THE SAME?
12CC    F0 F7           BEQ     WGRP01          ;BRANCH IF YES
12CE    60      WEXGRP: RTS                     ;OTHERWISE RETURN
                        ;
12CF    20 3E16 WONE:   JSR     PRNTLN          ;OUTPUT THE LINE NUMBER
12D2    20 8419 WONEC:  JSR     GETC            ;GET NEXT CHAR
12D5    20 FA18         JSR     PRINTC          ;OUTPUT IT
12D8    C9 0D           CMP#    %15             ;WAS IT THEN END OF THE LINE?
12DA    D0 F6           BNE     WONEC           ;LOOP IF NOT
12DC    A5 67           LDA     ODEV            ;YES, ARE WE OUTPUTTING TO A STRING?
12DE    30 03           BMI     WEOL            ;BRANCH IF WE ARE, DON'T FOLLOW WITH LINE FEED
12E0    20 931D         JSR     OUTLF           ;NO, THEN FOLLOW WITH A LINE FEED
12E3    20 E416 WEOL:   JSR     EATCR1          ;YES, POINT TO NEXT LINE
12E6    A4 2A           LDY     TEXTP           ;LOAD Y WITH OFFSET
12E8    B1 23           LDA@Y   TXTADR          ;PICK UP GROUP NUMBER OF NEXT LINE
12EA    C9 FE           CMP#    EOP             ;COMPARE TO END OF PROGRAM FLAG
12EC    60      WRET:   RTS                     ;RETURN WITH Z=1 IF END OF PROGRAM
                                                ;RETURN WITH Z=0 IF NOT END OF PROGRAM

12ED    A5 67   WCRLF:  LDA     ODEV            ;ARE WE OUTPUTTING TO A STRING?
12EF    30 FB           BMI     WRET            ;BRANCH IF WE ARE, DON'T ADVANCE
12F1    4C 881D         JMP     CRLF            ;* PJMP * OTHERWISE OUTPUT A CRLF AND RETURN
                        ;
                        ;
                        ;
```

```
                                .PAGE                   ;'MAIN CONTROL AND TRANSFER'
                        ;
                        ;
                        ;       "GOTO" COMMAND
                        ;
12F4    20 C915 GOTO:   JSR     GETLNS          ;GO GET THE LINE NUMBER
12F7    20 8016 GOTO1:  JSR     FINDLN          ;NOW GO TRY TO FIND IT
12FA    B0 02           BCS     GCONT           ;BRANCH IF WE FOUND IT
12FC    00              BRK                     ;TRAP
12FD    EB              .BYTE   GONEXL          ;?GOTO NON-EXISTANT LINE
12FE    20 3217 GCONT:  JSR     NEWLIN          ;GO SETUP THE NEW LINE
                                                ;* PFALL * INTO PROCES
                        ;
1301    20 8419 PROCES: JSR     GETC            ;GET NEXT CHAR
1304    A5 28   PROC:   LDA     CHAR            ;JUST IN CASE CALLED FROM ELSEWHERE
1306    C9 0D           CMP#    %15             ;END OF LINE?
1308    F0 58           BEQ     PC1             ;IF SO, THEN RETURN
130A    20 5413         JSR     GLTEST          ;IS CHAR A TERMINATOR?
130D    F0 F2           BEQ     PROCES          ;IF SO, IGNORE IT
130F    A6 2E   PROCX:  LDX     PRILVL          ;GET OUR CURRENT SOFTWARE INTERRUPT PRIO LEVEL
1311    A5 60           LDA     ACTMSK          ;GET BYTE WHICH INDICATES WHICH CHANNELS ARE ACTIVE
1313    F0 13           BEQ     PROC1           ;BRANCH IF NONE ACTIVE, PRESS ON WITH COMMAND
1315    3D 8625         ANDX    INTTAB          ;MASK OUT ANY THAT ARE NOT ALLOWED TO HAPPEN
1318    F0 0E           BEQ     PROC1           ;NONE ALLOWED TO HAPPEN, PRESS ON
131A    25 6E           AND     EVMASK          ;NOW SEE IF ANY EVENT IS PENDING ON ALLOWED CHANNELS
131C    F0 0A           BEQ     PROC1           ;BRANCH IF NONE PENDING, PRESS ON
131E    A2 08           LDX#    8               ;WE HAVE AN EVENT TO SERVICE (AT LEAST ONE)
1320    DD 8F25 PRILOP: CMPX    BITTAB          ;NOW FIND THE HIGHEST PRIORITY ONE TO SERVICE
1323    B0 43           BCS     EVNTDO          ;BRANCH WHEN WE HAVE IT
1325    CA              DEX                     ;NOT THIS LEVEL, TRY NEXT LOWER
1326    D0 F8           BNE     PRILOP          ;AND LOOP TILL WE FIND IT
                                                ;*** SHOULD NEVER FALL OUT THIS END! ***
1328    A5 28   PROC1:  LDA     CHAR            ;GET CHARACTER WHICH STARTED COMMAND ——— *    Jmp  EXIT
132A    48              PHA                     ;TEMPORARILY SAVE COMMAND CHAR
132B    20 5413 PTERM1: JSR     GLTEST          ;TERMINATOR?
132E    F0 05           BEQ     PTERM2          ;BRANCH IF YES
1330    20 8419         JSR     GETC            ;OTHERWISE SKIP OVER REST OF COMMAND NAME
1333    10 F6           BPL     PTERM1          ;UNCONDITIONAL BRANCH
1335    68      PTERM2: PLA                     ;GET FIRST CHAR OF COMMAND NAME BACK AGAIN
1336    A2 00           LDX#    0               ;INIT TABLE OFFSET TO ZERO
1338    BC 3225 NXTCOM: LDYX    COMTAB          ;GET THE COMMAND CHAR
133B    F0 26           BEQ     ILLCOM          ;ZERO ENDS THE TABLE
133D    DD 3225         CMPX    COMTAB          ;IS THIS CHAR THE COMMAND WE WANT?
1340    F0 03           BEQ     GOTCOM          ;BRANCH IF YES
1342    E8              INX                     ;NO, POINT TO NEXT ONE
1343    D0 F3           BNE     NXTCOM          ;UNCONDITIONAL LOOP FOR NEXT ONE
1345    BC 5825 GOTCOM: LDYX    COMADL          ;GET LOW ORDER ADDR OF ROUTINE
1348    8C 5213         STY     CJADR           ;SAVE IN DUMMY JUMP
134B    BC 4625         LDYX    COMADH          ;GET HIGH ORDER ADDR OF ROUTINE
134E    8C 5313         STY     CJADR+1         ;SAVE IN DUMMY JUMP
                        ;
        1352            .DEF    CJADR=.+1
                        ;
1351    4C 0000         JMP     $0000           ;DISPATCH TO COMMAND ROUTINE
```

```
                                .PAGE                   ;'MAIN CONTROL AND TRANSFER'
                        ;
                        ;
                        ;       GLTEST EXITS WITH Z SET IF CHAR IN THE ACCUMULATOR IS
                        ;       A TERMINATOR (SPACE, COMMA, OR SEMI-COLON).
                        ;
1354  C9 20     GLTEST: CMP#    ' '             ;SPACE?
1356  F0 0A             BEQ     RTS2            ;RETURN IF YES
1358  C9 2C             CMP#    ','             ;COMMA?
135A  F0 06             BEQ     RTS2            ;RETURN IF YES
135C  C9 3B     TSTEOC: CMP#    ';              ;SEMI-COLON?
135E  F0 02             BEQ     RTS2            ;RETURN IF YES
1360  C9 00             CMP#    %15             ;CARRIAGE RETURN?
1362  60        RTS2:   RTS                     ;RETURN WITH Z=1 IF ANY OF THESE
                        ;
1363  00        ILLCOM: BRK                     ;TRAP
1364  FD                .BYTE   BADCOM          ;?UNRECOGNIZABLE COMMAND
                        ;
1365  20 5818   PC1:    JSR     POPJ            ;EXIT 'PROCESS'
                        ;
      1365              .DEF    COMMNT=PC1      ;"COMMENT" COMMAND
                        ;
                        ;ROUTINE TO PERFORM A 'DO' OF A SPECIFIED LINE OR GROUP
                        ;WHEN AN EXTERNAL EVENT HAPPENS ON A SOFTWARE INTERRUPT CHANNEL.

      130E              .DEF    EVRET=PROCX-1   ;DEFINE PLACE TO RETURN TO AFTER EVENT 'DO'

1368  8A        EVNTDO: TXA                     ;SAVE X REGISTER AS IT HAS LEVEL OF INTERRUPT
1369  48                PHA
136A  A9 0E             LDA#    EVRET           ;SETUP RETURN ADDRESS ON STACK
136C  20 3E18           JSR     PUSHA           ;A LA 'PUSHJ'
136F  A9 13             LDA#    >EVRET
1371  20 3E18           JSR     PUSHA
1374  20 B910           JSR     PUSHDO          ;SAVE IMPORTANT STUFF AT THIS LEVEL
1377  68                PLA                     ;GET NEW PRIO LEVEL OF GROUP OR LINE TO 'DO'
1378  AA                TAX
1379  86 2E             STX     PRILVL          ;SET IT AS OUR NEW PRIO LEVEL
137B  BD 8F25           LDAX    BITTAB          ;GET BIT MASK FOR THIS NEW LEVEL
137E  49 FF             EOR#    $FF             ;COMPLEMENT TO MAKE 'AND' MASK
1380  08                PHP                     ;SAVE STATE OF 'I' BIT IN CASE IRQ'S ARE ON
1381  78                SEI                     ;DISABLE IRQ GRANTING
1382  25 6E             AND     EVMASK          ;WHILE WE UPDATE EVENT MASK
1384  85 6E             STA     EVMASK
1386  28                PLP                     ;WE CAN ALLOW IRQ'S AGAIN (IF ON)
1387  BD 7425           LDAX    INTGRP          ;GET THE GROUP NUMBER TO 'DO'
138A  85 2C             STA     GRPNO
138C  BD 7025           LDAX    INTLIN          ;AND THE LINE NUMBER
138F  85 2D             STA     LINENO
1391  20 1216           JSR     GOTLNO          ;NOW SET FLAGS TO DETERMINE WHAT TO 'DO'
1394  70 05             BVS     EVDALL          ;BRANCH IF WE ARE 'DO'ING ALL
1396  F0 03             BEQ     EVDALL          ;OR 'DO' OF A GROUP
1398  4C C211           JMP     DOONE           ;* PJMP * 'DO' ONE LINE, THEN RETURN TO 'EVRET'
139B  4C D711 EVDALL: JMP     DOGRP           ;* PJMP * 'DO' A GROUP, THEN RETURN TO 'EVRET'
```

```
                        .PAGE                   ;'TYPE - ASK COMMAND PROCESSOR'
                ;
                ;
139E  20 681C TDUMP:  JSR     VARINI            ;INIT POINTER TO START OF VARIABLE LIST
13A1  20 681D TDNEXT: JSR     CRLF              ;START ON NEW LINE
13A4  A0 00           LDY#    0                 ;POINT TO VARIABLE NAME
13A6  84 38           STY     STRSWT            ;MAKE SURE STRING SWITCH IS OFF
13A8  B1 37           LDA@Y   VARADR            ;PICK UP THE VARIABLE NAME
13AA  C9 FF           CMP#    EOV               ;END OF VARIABLE LIST?
13AC  F0 52           BEQ     JTASK4            ;BRANCH IF END OF VARIABLE LIST
13AE  C9 FC           CMP#    STRMRK            ;STRING VARIABLE?
13B0  D0 05           BNE     TDCONT            ;BRANCH IF NOT, PRESS ON
13B2  85 38           STA     STRSWT            ;YES, THEN FLAG IT
13B4  C8              INY                       ;AND MOVE PAST MARKER, POINT TO NAME
13B5  B1 37           LDA@Y   VARADR            ;AND PICK UP THE NAME
13B7  20 E913 TDCONT: JSR     PRTVNM            ;PRINT THE VARIABLE'S NAME
13BA  A5 38           LDA     STRSWT            ;IS THIS VARIABLE A STRING VARIABLE?
13BC  D0 45           BNE     TPSTR             ;BRANCH IF YES, DO SPECIAL OUTPUT
13BE  A9 28           LDA#    '(                ;AND THEN START OF SUBSCRIPT
13C0  20 FA18         JSR     PRINTC
13C3  A0 01           LDY#    1                 ;POINT TO SUBSCRIPT
13C5  B1 37           LDA@Y   VARADR            ;GET HIGH ORDER SUBSCRIPT
13C7  85 81           STA     M1                ;SAVE IN FAC1
13C9  C8              INY                       ;POINT TO LOW ORDER SUBSCRIPT
13CA  B1 37           LDA@Y   VARADR            ;GET IT
13CC  85 82           STA     M1+1
13CE  20 4F22         JSR     FLT16             ;FORM A FLOATING POINT NUMBER
13D1  20 6816         JSR     OUTLN0            ;AND OUTPUT IT
13D4  A9 29           LDA#    ')                ;CLOSING PAREN
13D6  20 FA18         JSR     PRINTC
13D9  A9 3D           LDA#    '=                ;PUT IN AN EQUALS FOR FORM
13DB  20 FA18         JSR     PRINTC
13DE  20 A81C         JSR     FETVAR            ;GET THE VARIABLE'S VALUE INTO FLAC
13E1  20 2123         JSR     FPRNT             ;PRINT IT
13E4  20 711C         JSR     NXTVAR            ;POINT TO NEXT VARIABLE IN THE LIST
13E7  D0 B8           BNE     TDNEXT            ;UNCONDITIONALLY LOOP FOR MORE

                        ;ROUTINE TO PRINT OUT A VARIABLE'S NAME

13E9  48        PRTVNM: PHA                       ;SAVE COMPOSITE FORM
13EA  4A                LSRA                      ;EXTRACT ALPHA PART
13EB  4A                LSRA
13EC  4A                LSRA
13ED  09 40             ORA#    $40               ;FORM ASCII
13EF  C9 46             CMP#    'F                ;ACTUALLY '@', SINCE 'F' IS ILLEGAL
13F1  D0 02             BNE     TPNAM             ;NOT SPECIAL, SO PRINT IT
13F3  A9 26             LDA#    '&                ;PRINT A '&' AS SPECIAL VARIABLE NAME
13F5  20 FA18 TPNAM:    JSR     PRINTC            ;AND PRINT IT
13F8  68                PLA                       ;GET BACK HASH
13F9  29 07             AND#    7                 ;EXTRACT NUMBER
13FB  09 30             ORA#    $30               ;FORM  ASCII
13FD  4C FA18           JMP     PRINTC            ;* PJMP * PRINT THE VARIABLE NUMBER, THEN RETURN

1400  4C DD14 JTASK4: JMP     TASK4             ;BRANCH AIC
```

```
1118                            .PAGE                   ;'TYPE - ASK COMMAND PROCESSOR'
1119
1120                            ;HERE TO TYPE OUT A STRING VARIABLE
1121
1122
1123
1124  1403  A9 24    TPSTR:  LDA#    '$              ;INDICATE ITS A STRING VARIABLE
1125  1405  20 FA18          JSR     PRINTC
1126  1408  A9 3D            LDA#    '=              ;DON'T PRINT A SUBSCRIPT ON A STRING VARIABLE
1127  140A  20 FA18          JSR     PRINTC
1128  140D  A9 22            LDA#    '"              ;DELIMIT STRING WITH QUOTES
1129  140F  20 FA18          JSR     PRINTC
1130  1412  A0 02            LDY#    2               ;POINT TO STRING LENGTH
1131  1414  B1 37            LDA@Y   VARADR          ;PICK UP THE STRING LENGTH
1132  1416  85 3C            STA     VSIZE           ;SAVE IT
1133  1418  C8              INY                     ;POINT TO FIRST BYTE OF STRING
1134  1419  98              TYA                     ;NOW UPDATE 'VARADR' TO BASE ADDR OF STRING
1135  141A  20 731C         JSR     UPDVAR
1136  141D  A0 00            LDY#    0               ;POINT TO FIRST BYTE OF STRING
1137  141F  98       TPNXTC: TYA                     ;SAVE OFFSET
1138  1420  48              PHA
1139  1421  B1 37            LDA@Y   VARADR          ;GET BYTE FROM STRING
1140  1423  20 FA18          JSR     PRINTC          ;PRINT THE BYTE
1141  1426  68              PLA                     ;RESTORE POINTER
1142  1427  A8              TAY
1143  1428  C8              INY                     ;POINT TO NEXT BYTE
1144  1429  C4 3C            CPY     VSIZE           ;PRINTED ALL OF STRING YET?
1145  142B  D0 F2            BNE     TPNXTC          ;BRANCH IF MORE TO PRINT
1146  142D  98              TYA                     ;YES, THEN SKIP OVER STRING BY
1147  142E  20 731C         JSR     UPDVAR          ;UPDATING 'VARADR'
1148  1431  A9 22            LDA#    '"              ;CLOSE OFF STRING WITH CLOSING QUOTE
1149  1433  20 FA18          JSR     PRINTC
1150  1436  4C A113         JMP     TDNEXT          ;AND DUMP NEXT VARIABLE
1151
1152  1439  4C 9E13 JTDUMP: JMP     TDUMP           ;BRANCH AID
1153
```

```
                        .PAGE                   ;'TYPE - ASK COMMAND PROCESSOR'
                        ;
                        ;
                        ;
143C  20 2918 TASK1:    JSR     PUSHJ           ;GO GET THE VARIABLE
143F  43 13             .WORD   GETVAR
1441  20 5D1C           JSR     BOMSTV          ;BOMB OUT IF A STRING VARIABLE IS USED IN "ASK"
1444  A5 28             LDA     CHAR            ;SAVE DELIMITER
1446  48                PHA                     ;ON HARDWARE STACK
1447  E6 24             INC     INSW            ;FLAG INPUT FROM KEYBOARD
1449  A2 37   ASKAGN:   LDX#    VARADR          ;SAVE THE VARIABLE'S ADDR
144B  20 7618           JSR     PUSHB2
144E  20 2918           JSR     PUSHJ           ;NOW GO GET USER SUPPLIED DATA
1451  F5 19             .WORD   EVALM1
1453  A9 41             LDA#    'A              ;RESTORE 'ATSW' (SINCE WE MUST BE RECURSIVE!)
1455  85 22             STA     ATSW
1457  A2 38             LDX#    VARADR+1        ;RESTORE VARIABLE'S ADDRESS
1459  20 6618           JSR     POPB2
145C  A5 28             LDA     CHAR            ;GET DELIMITER FROM EVAL
145E  C9 5F             CMP#    LINCHR          ;WAS IT 'LINE-DELETE' CHARACTER
1460  D0 08             BNE     STODAT          ;BRANCH IF NOT, STORE VALUE AWAY
1462  A5 66             LDA     IDEV            ;YES, IS THE INPUT DEVICE
1464  C5 6A             CMP     CONDEV          ;THE CONSOLE?
1466  D0 E1             BNE     ASKAGN          ;BRANCH IF NOT, ASK AGAIN
1468  20 8B1D           JSR     CRLF            ;YES, ADVANCE A LINE
146B  10 DC             BPL     ASKAGN          ;AND ASK AGAIN
146D  20 931C STODAT:   JSR     PUTVAR          ;PLACE DATA IN VARIABLE
1470  C6 24             DEC     INSW            ;FLAG INPUT FROM CORE AGAIN
1472  68                PLA                     ;GET DELIMITER BACK AGAIN
1473  85 28             STA     CHAR
1475  10 13             BPL     TASK            ;UNCONDITIONALLY CONTINUE PROCESSING
                        ;
1477  20 8419 TFORM:    JSR     GETC            ;MOVE PAST "%"
147A  20 C915           JSR     GETLNS          ;GET GG,SS
147D  A5 2C             LDA     GRPNO           ;GET GG
147F  85 8F             M                       ;SAVE AS NUMBER BEFORE DECIMAL POINT
1481  A5 2D             LDA     LINENO          ;GET SS
1483  85 90             N                       ;SAVE AS NUMBER AFTER DECIMAL POINT
1485  4C 8A14           JMP     TASK            ;AND CONTINUE PROCESSING
                        ;
                        ;
                        ;
                        ;
```

```
                                .PAGE                   ;'TYPE - ASK COMMAND PROCESSOR
                        ;
                        ;
                        ;
                        ;
1488  85 22    TYPE:   STA     ATSW                    ;FLAG WHICH ONE IT IS
148A  A9 00    TASK:   LDA#    0                       ;ENABLE THE TRACE
148C  85 23            STA     DEBGSW
148E  20 A619          JSR     SPNOR                   ;LOOK FOR NEXT NON-BLANK
1491  C9 24              CMP#    '$
1493  F0 A4              BEQ     JTDUMP                ;DUMP OUT THE VARIABLE LIST
1495  C9 25              CMP#    '%                    ;FORMAT CONTROL?
1497  F0 0E              BEQ     TFORM                 ;BRANCH IF YES
1499  C9 21              CMP#    '!                    ;SEE IF SPECIAL
149B  F0 36              BEQ     TCRLF                 ;GIVE OUT A CARRIAGE RETURN-LINE FEED
149D  C9 23              CMP#    '#
149F  F0 37              BEQ     TCR                   ;CARRIAGE RETURN ONLY
14A1  C9 22              CMP#    '"
14A3  F0 30              BEQ     TQUOT                 ;TYPE OUT A QUOTED STRING
14A5  C9 2C              CMP#    ',
14A7  F0 34              BEQ     TASK4                 ;IGNORE IN CASE USER WANTS IT TO LOOK PRETTY
14A9  C9 3B              CMP#    ';                    ;END OF COMMAND?
14AB  F0 4E              BEQ     TPROC                 ;YES, THEN BRANCH
14AD  C9 00              CMP#    %15                   ;END OF LINE?
14AF  F0 43              BEQ     TPC1                  ;YES, THEN GO HANDLE IT
14B1  A5 22              LDA     ATSW                  ;NOT SPECIAL CHARACTER, GET COMMAND SWITCH
14B3  C9 41              CMP#    'A                    ;WHICH COMMAND ARE WE DOING?
14B5  F0 85              BEQ     TASK1                 ;BRANCH IF "ASK", AS IT DIFFERS
14B7  20 2918            JSR     PUSHJ                 ;CALL "EVAL" TO EVALUATE THE EXPRESSION
14BA  F8 19              .WORD   EVAL
14BC  A9 54              LDA#    'T                    ;RESTORE 'ATSW' (SINCE WE MUST BE RECURSIVE!)
14BE  85 22              STA     ATSW
14C0  20 2123            JSR     FPRNT                 ;GO OUTPUT IT
14C3  A5 28              LDA     CHAR                  ;GET TERMINATOR FROM "EVAL"
14C5  C9 29              CMP#    ')                    ;SO "TYPE 3)" DOESN'T DIE!
14C7  F0 14              BEQ     TASK4                 ;FLUSH IF WE CON'T LIKE IT
14C9  C9 3D              CMP#    '=                    ;ALSO FLUSH OTHER NASTIES
14CB  F0 10              BEQ     TASK4
14CD  C9 2E              CMP#    '.
14CF  F0 0C              BEQ     TASK4
14D1  D0 B7              BNE     TASK                  ;OTHERWISE, CONTINUE PROCESSING
                        ;
        1488            .DEF    ASK=TYPE
                        ;
                        ;
                        ;
```

```
1243                            .PAGE                   ;'TYPE - ASK COMMAND PROCESSOR'
1244                    ;
1245                    ;
1246  1403  20 8B1D TCRLF:  JSR     CRLF            ;OUTPUT A CARRIAGE RETURN FOLLOWED BY LINE FEED
1247  1406  10 05          BPL     TASK4           ;UNCONDITIONAL BRANCH
1248  1408  A9 0D    TCR:    LDA#    %15             ;OUTPUT A CR
1249  140A  20 F418         JSR     PRINTC
1250  140D  20 8419 TASK4:  JSR     GETC            ;SKIP OVER THIS CHARACTER
1251  1410  10 A8          BPL     TASK            ;UNCONDITIONALLY CONTINUE PROCESSING
1252                    ;
1253  14E2  E6 20   TQUOT:  INC     DEBGSW          ;DISABLE TRACE SO LITERAL ONLY PRINTED ONCE
1254  14E4  20 8419 TQUOT1: JSR     GETC            ;GET NEXT CHAR
1255  14E7  C9 22           CMP#    '"              ;CLOSING QUOTE?
1256  14E9  F0 F2           BEQ     TASK4           ;BRANCH IF YES
1257  14EB  C9 0D           CMP#    %15             ;END OF LINE?
1258  14ED  F0 05           BEQ     TPC1            ;BRANCH IF YES,  (IT TERMINATES STRING)
1259  14EF  20 FA18         JSR     PRINTC          ;OTHERWISE, PRINT THE CHARACTER
1260  14F2  10 F0           BPL     TQUOT1          ;UNCONDITIONALLY LOOP TILL DONE
1261  14F4  A9 00   TPC1:   LDA#    0               ;ENABLE TRACE JUST IN CASE
1262  14F6  85 20           STA     DEBGSW
1263  14F8  4C 6513 FPC1:   JMP     PC1             ;EXIT 'PROCESS'.
1264  14FB  4C 0113 TPROC:  JMP     PROCES          ;CONTINUE PROCESSING ON THIS LINE
1265                    ;
1266                    ;
1267                    ;
1268                    ;
1269                    ;
```

```
                        .PAGE                   ;'"FOR" AND "SET" COMMAND PROCESSING
                        ;
                        ;
                        ;       "FOR" LOOP ITERATION COMMAND
                        ;
                        ;
                        ;
1277 14FE 20 2918 FOR:  JSR     PUSHJ           ;GO GET THE VARIABLE
1278 1501 43 13         .WORD   GETVAR
1279 1503 A5 2B         LDA     CHAR            ;GET TERMINATOR
1280 1505 C9 3D         CMP#    '=              ;"=" SIGN?
1281 1507 F0 02         BEQ     FOR2            ;BRANCH IF YES
1282 1509 00            BRK                     ;NO, TRAP
1283 150A F3            .BYTE   NOEQLS          ;?NO "=" IN 'FOR' OR 'SET'
1284 150B A2 37   FOR2: LDX#    VARADR          ;SAVE THE ADDR OF THE VARIABLE
1285 150D A0 05         LDY#    5               ;AND IT'S PROPERTIES
1286 150F 20 7818       JSR     PUSHB0          ;ON STACK
1287 1512 20 2918       JSR     PUSHJ           ;CALL "EVAL" TO EVALUATE RIGHT HAND
1288 1515 F5 19         .WORD   EVALM1          ;SIDE OF "="
1289 1517 A2 3B         LDX#    VARADR+4        ;GET THE ADDR OF VARIABLE BACK AGAIN
1290 1519 A0 05         LDY#    5
1291 151B 20 6818       JSR     POPB0
1292 151E 20 801C       JSR     BOMSTV          ;BOMB OUT IF LOOP COUNTER IS A STRING VARIABLE
1293 1521 20 951C       JSR     PUTVAR          ;NOT A STRING, SO STORE INITIAL VALUE
1294 1524 A5 2B         LDA     CHAR            ;GET THE EXPRESSION TERMINATOR
1295 1526 C9 2C         CMP#    ',              ;COMMA?
1296 1528 F0 13         BEQ     FINCR           ;BRANCH IF IT'S A "FOR" COMMAND
1297 152A 00      BTFOR: BRK                     ;TRAP
1298 152B F2            .BYTE   FBDTRM          ;?BAD TERMINATOR IN "FOR"
1299
1300
1301                    ;       "SET" COMMAND
1302
1303
1304 152C 20 8419 SET1: JSR     GETC            ;SKIP OVER COMMA
1305 152F 20 2918 SET:  JSR     PUSHJ           ;CALL 'EVAL' TO EVALUATE EXPRESSION
1306 1532 F8 19         .WORD   EVAL
1307 1534 A5 2B         LDA     CHAR            ;GET TERMINATOR
1308 1536 C9 2C         CMP#    ',              ;COMMA?
1309 1538 F0 F2         BEQ     SET1            ;BRANCH IF YES, LOOP FOR ANOTHER EXPRESSION
1310 153A 4C 0413       JMP     PROC            ;NO.  ALL DONE, CONTINUE ON THIS LINE
```

Best cmmd exit point $1304

```
                                .PAGE                   ;'"FOR" COMMAND PROCESSING'
                                ;
                                ;
1530  A2 37   FINCR:  LDX#     VARADR                  ;SAVE THE ADDR OF THE LOOP VARIABLE ON STACK
153F  20 7618           JSR     PUSHB2
1542  20 2918           JSR     PUSHJ                   ;GO GET THE INCREMENT
1545  F5 19             .WORD   EVALM1
1547  A5 28             LDA     CHAR                    ;GET TERMINATOR
1549  C9 2C             CMP#    ',                      ;DID WE GET AN INCREMENT?
154B  F0 16             BEQ     FLIMIT                  ;YES, GO GET THE UPPER LIMIT OF LOOP
154D  C9 3B             CMP#    ';                      ;WAS NO INCREMENT SPECIFIED?
154F  F0 06             BEQ     FINCR1                  ;BRANCH IF NO INCREMENT GIVEN
1551  C9 0D             CMP#    %15                     ;CARRIAGE RETURN?
1553  D0 D5             BNE     BTFOR                   ;NO, THEN BAD TERMINATOR
1555  00                BRK                             ;YES, TRAP
1556  F1                .BYTE   0FL                     ;?USELESS "FOR" LOOP
1557  A2 F8   FINCR1:  LDX#     -NUMBF                  ;GET NEG OF NUMBER OF BYTES
1559  B5 A0   FI1C:    LDAX    FONE+NUMBF              ;GET NEXT BYTE
155B  20 3E18           JSR     PUSHA                   ;PUSH IT ON STACK
155E  E8                INX                             ;POINT TO NEXT ONE
155F  30 F9             BMI     FI1C                    ;AND LOOP TILL ALL PUSHED
1561  10 03             BPL     FSHORT                  ;UNCONDITIONAL BRANCH
1563  20 8618 FLIMIT:  JSR     PHFAC1                  ;SAVE INCREMENT ON STACK
1566  20 2918           JSR     PUSHJ                   ;NOW EVALUATE THE UPPER LOOP LIMIT
1569  F5 19             .WORD   EVALM1
156B  20 8618 FSHORT:  JSR     PHFAC1                  ;SAVE UPPER LIMIT ON STACK
                                                        ;AND ENTER LOOP
```

```
1339                            .PAGE                   ;'LOOP PROCESSOR FOR "FOR" COMMAND"'
1340
1341  156E  20 1717 FCONT:  JSR     PUSHTP              ;NOW SAVE THE TEXT POINTERS ON STACK
1342  1571  A5 26           LDA     PC                  ;SAVE PC ACCROSS CALL
1343  1573  20 3E18         JSR     PUSHA
1344  1576  20 2918         JSR     PUSHJ               ;NOW EXECUTE THE REST OF THE LINE
1345  1579  01 13           .WORD   PROCES
1346  157B  20 4C18         JSR     POPA                ;GET PC BACK
1347  157E  85 73           STA     ITEMP1              ;SAVE IT IN TEMPORARY
1348  1580  20 1E17         JSR     POPTP               ;RESTORE POINTERS FOR POSSIBLE RE-EXECUTE
1349  1583  20 B218         JSR     PLTMP               ;RESTORE UPPER LOOP LIMIT INTO TEMPORARY
1350  1586  20 B51C         JSR     POPIV               ;RESTORE INCREMENT AND VARIABLE ADDR
1351  1589  A5 26           LDA     PC                  ;GET PC
1352  158B  C9 FD           CMP#    RETCMD              ;WAS A 'RETURN' COMMAND JUST EXECUTED?
1353  158D  F0 22           BEQ     FORXIT              ;BRANCH IF YES, THEN EXIT THE LOOP NOW!
1354  158F  A5 7C           LDA     M2                  ;GET THE SIGN OF THE INCREMENT (+ OR -)
1355  1591  08             PHP                         ;SAVE STATUS ON STACK FOR LATER
1356  1592  20 AD1C         JSR     PUSHIV              ;SAVE AGAIN FOR POSSIBLE REPEAT OF LOOP
1357  1595  20 A01C         JSR     FETVAR              ;GO GET THE VARIABLE'S CURRENT VALUE
1358  1598  20 7B22         JSR     FADD                ;ADD THE INCREMENT TO FLAC
1359  159B  20 951C         JSR     PUTVAR              ;STORE AS NEW LOOP COUNTER VALUE
1360  159E  20 9C18         JSR     PHTMP               ;SAVE TEMPORARY ON STACK
1361  15A1  20 A718         JSR     PLFAC2              ;PLACE INTO FAC2
1362  15A4  20 7522         JSR     FSUB                ;SUBTRACT COUNTER FROM UPPER LIMIT
1363  15A7  28              PLP                         ;GET SIGN OF THE INCREMENT
1364  15A8  30 0E           BMI     CNTDWN              ;BRANCH IF NEGATIVE, WE ARE COUNTING DOWN
1365  15AA  A5 81           LDA     FLOSGN              ;GET THE SIGN OF THE NUMBER
1366  15AC  10 13           BPL     MORFOR              ;BRANCH IF REPEAT NECCESSARY
1367  15AE  20 B51C FOREND: JSR     POPIV               ;CLEAN UP STACK
1368  15B1  A5 73   FORXIT: LDA     ITEMP1              ;RESTORE PC
1369  15B3  85 26           STA     PC                  ;IN CASE 'RETURN' ENCOUNTERED
1370  15B5  20 5B18         JSR     POPJ                ;EXIT "FOR" COMMAND
1371  15B8  A5 81   CNTDWN: LDA     FLOSGN              ;ARE WE LESS THAN THE LOOP LIMIT?
1372  15BA  F0 02           BEQ     MORFOR              ;NO, THEN KEEP GOING
1373  15BC  10 F0           BPL     FOREND              ;YES, THEN THAT'S ALL
1374  15BE  20 9C18 MORFOR: JSR     PHTMP               ;PLACE UPPER LIMIT BACK ON THE STACK
1375  15C1  10 AB           BPL     FCONT               ;UNCONDITIONALLY REPEAT LOOP
1376
```

```
1377                            .PAGE                   ;'LINE NUMBER MANIPULATION ROUTINES'
1378                    ;
1379                    ;
1380                    ;       "GETLN" GET A LINE NUMBER FROM PROGRAM TEXT.
1381                    ;       RETURNS WITH V=1 IF "ALL" (00.00), OTHERWISE
1382                    ;       IT RETURNS WITH Z=1 IF GROUP NUMBER ONLY (GG,00)
1383                    ;       AND Z=0 IF INDIVIDUAL LINE NUMBER (GG.LL).
1384
1385
1386  1503  20 2F24 GETLNC: JSR     FINP            ;ONLY ALLOW NUMERIC INPUT
1387  1506  4C EC15        JMP     GETLN1          ;AND ENTER REST OF CODE
1388
1389  1509  20 A619 GETLNS: JSR     SPNOR           ;GET NEXT NON-BLANK
1390  150C  A9 00   GETLN:  LDA#    0               ;ASSUME LINE NUMBER IS ZERO
1391  150E  85 2C           STA     GRPNO
1392  1510  85 2D           STA     LINENO
1393  1512  A5 28           LDA     CHAR            ;GET FIRST CHAR OF EXPRESSION?
1394  1514  C9 2C           CMP#    ',              ;IS EXPRESSION NULL?
1395  1516  F0 3A           BEQ     GOTLNO          ;BRANCH IF YES, THEN WE HAVE THE NUMBER
1396  1518  C9 3D           CMP#    %15             ;ANOTHER FORM OF NULL?
1397  151A  F0 36           BEQ     GOTLNO          ;BRANCH IF YES, THEN WE HAVE THE NUMBER
1398  151C  20 B319         JSR     TESTN           ;DOES EXPRESSION BEGIN WITH A NUMBER?
1399  151F  B0 06           BCS     GETLNX          ;BRANCH IF NOT, THEN MUST BE COMPLEX
1400  1521  20 951D         JSR     GETILN          ;CALL INTEGER LINE NUMBER INPUT FOR SPEED
1401  1524  4C 1216         JMP     GOTLNO          ;WE NOW HAVE THE LINE NUMBER
1402  1527  20 2918 GETLNX: JSR     PUSHJ           ;CALL 'EVAL' TO EVALUATE EXPRESSION
1403  152A  F8 19           .WORD   EVAL
1404  152C  20 8618 GETLN1: JSR     PHFAC1          ;SAVE EXPRESSION VALUE ON STACK
1405  152F  20 2316         JSR     GETL            ;INTEGERIZE AND RANGE CHECK
1406  1532  85 2C           STA     GRPNO           ;SAVE AS GROUP NUMBER
1407  1534  20 3822         JSR     FLOAT           ;NOW FLOAT THE GROUP NUMBER
1408  1537  20 A718         JSR     PLFAC2          ;POP FULL GG.SS INTO FAC2
1409  153A  20 7622         JSR     FSUB            ;SUBTRACT OFF THE GROUP NUMBER
1410  153D  20 3416         JSR     GMUL10          ;MULTIPLY BY 100
1411  1540  20 3416         JSR     GMUL10
1412  1543  A2 96           LDX#    FHALF           ;MOVE CONSTANT .50
1413  1545  A0 78           LDY#    X2
1414  1547  20 0724         JSR     MOVXY
1415  154A  20 7322         JSR     FADD            ;NOW ADD IN THE .50 FOR ROUNDING
1416  154D  20 2316         JSR     GETL            ;INTEGERIZE AND RANGE CHECK
1417  1550  85 2D           STA     LINENO          ;SAVE AS LINE NUMBER (STEP NUMBER)
1418  1552  B8      GOTLNO: CLV                     ;ASSUME NOT 00,00
1419  1553  A5 2C           LDA     GRPNO           ;GET GROUP NUMBER
1420  1555  05 2D           ORA     LINENO          ;'OR' IN THE LINE NUMBER
1421  1557  F0 07           BEQ     GOTALL          ;BRANCH IF BOTH ARE ZERO
1422  1559  A5 2C           LDA     GRPNO           ;GET GROUP NUMBER AGAIN
1423  155B  F0 15           BEQ     BADLNO          ;BAD LINE NUMBER IF GROUP ONLY IS ZERO
1424  155D  A5 2D           LDA     LINENO          ;GROUP NUMBER OK, GET LINE (STEP) NUMBER
1425  155F  60      RTS3:   RTS                     ;RETURN WITH Z=1 IF GROUP ONLY
1426                                                ;RETURN WITH Z=0 IF GG.SS
1427  1560  24 6F   GOTALL: BIT     BITV1           ;EXIT WITH V=1, N=1
1428  1562  60              RTS
```

```
1429                            .PAGE                   ;'LINE NUMBER MANIPULATION ROUTINES'
1430
1431
1432
1433  1623  20 1A23 GETL:  JSR     FIX                 ;FIX THE NUMBER IN FAC1
1434  1626  A5 81          LDA     M1                  ;GET HIGH ORDERS
1435  1628  05 82          ORA     M1+1                ;SMASH THEM TOGETHER
1436  162A  D0 06          BNE     BADLNO              ;LINE NUMBER CAN ONLY BE POSITIVE
1437  162C  A5 83          LDA     M1+2
1438  162E  C9 64          CMP#    100                 ;AND < 100 ?
1439  1630  30 ED          BMI     RTS3
1440  1632  00       BADLNO: BRK                         ;TRAP
1441  1633  FC              .BYTE   ILLNO               ;?ILLEGAL LINE NUMBER
1442
1443  1634  A2 91    GMUL10: LDX#    FTEN                ;MOVE 10.0
1444  1636  A0 7B          LDY#    X2                  ;INTO FAC2
1445  1638  20 0724        JSR     MOVXY
1446  163B  4C A222        JMP     FMUL                ;* PJMP *  FAC1*FAC2=FAC1
```

```
1447                            .PAGE                   ;'PRNTLN - PRINT A LINE NUMBER'
1448                    ;
1449                    ;
1450                    ;       "PRNTLN" PRINT A LINE NUMBER TO OUTPUT DEVICE
1451                    ;
1452                    ;
1453  163E  A4 2A   PRNTLN: LDY     TEXTP           ;GET TEXT POINTER
1454  1640  B1 28           LDA@Y   TXTADR          ;GET GROUP NUMBER
1455  1642  D0 05           BNE     PRNTL1          ;BRANCH IF NOT ZERO
1456  1644  C8             INY                     ;DO NOT PRINT GROUP ZERO LINE NUMBERS
1457  1645  C8             INY
1458  1646  84 2A           STY     TEXTP           ;POINT TO FIRST CHAR IN LINE
1459  1648  60             RTS                     ;AND RETURN
1460
1461  1649  48         PRNTL1: PHA                     ;SAVE THE GROUP NUMBER FOR LATER
1462  164A  C8             INY                     ;POINT TO STEP NUMBER
1463  164B  B1 28           LDA@Y   TXTADR          ;GET STEP NUMBER
1464  164D  C8             INY                     ;MOVE PAST IT
1465  164E  84 2A           STY     TEXTP           ;SAVE POINTER
1466  1650  20 8416         JSR     PFLT            ;FLOAT THE STEP NUMBER
1467  1653  20 1924         JSR     DIV10           ;DIVIDE BY 100
1468  1656  20 1924         JSR     DIV10
1469  1659  20 8618         JSR     PHFAC1          ;SAVE 00.SS FOR LATER
1470  165C  68             PLA                     ;GET THE GROUP NUMBER BACK
1471  165D  20 8416         JSR     PFLT            ;FLOAT IT
1472  1660  20 A718         JSR     PLFAC2          ;RESTORE 00.SS INTO FAC2
1473  1663  20 7B22         JSR     FADD            ;ADD TOGETHER TO FORM GG.SS
1474                                                 ;* PFALL * INTO OUTPUT ROUTINE
1475  1666  A9 02   OUTLN2: LDA#    2               ;ASSUME TWO DIGITS BEFORE THE DECIMAL POINT
1476  1668  AA             TAX                     ;AND TWO DIGITS AFTER
1477  1669  D0 04           BNE     OUTLN           ;UNCONDITIONAL BRANCH
1478  166B  A9 02   OUTLN0: LDA#    2               ;ASSUME TWO DIGITS BEFORE DECIMAL
1479  166D  A2 00   OUTLN1: LDX#    0               ;ASSUME NO DECIMAL POINT
1480  166F  A8      OUTLN:  TAY                     ;SAVE NUMBER BEFORE DECIMAL IN Y REG
1481  1670  A5 8F           LDA     M               ;SAVE OLD FORMAT ON HARDWARE STACK
1482  1672  48             PHA                     ;(COULD BE CALLED FROM ERROR TRAP)
1483  1673  A5 90           LDA     N
1484  1675  48             PHA
1485  1676  84 8F           STY     M               ;STORE NEW FORMAT
1486  1678  86 90           STX     N
1487  167A  20 2123         JSR     FPRNT           ;PRINT NUMBER IN TEMPORARY FORMAT
1488  167D  68             PLA                     ;RESTORE OLD FORMAT
1489  167E  85 90           STA     N
1490  1680  68             PLA
1491  1681  85 8F           STA     M
1492  1683  60             RTS                     ;AND RETURN
1493
1494  1684  85 82   PFLT:  STA     M1+1            ;SAVE IN LOW ORDER
1495  1686  A9 00           LDA#    0               ;MAKE HIGH ORDER ZERO
1496  1688  85 81           STA     M1
1497  168A  4C 4F22         JMP     FLT16           ;* PJMP * AND FLOAT IT
1498
1499
```

```
1500                            .PAGE                   ;'FINDLN - FIND A LINE IN THE STORED PROGRAM'
1501                    ;
1502                    ;
1503                    ;       "FINDLN" RETURNS WITH C=1 IF THE LINE WAS FOUND.
1504                    ;                TXTAD2 POINTS TO THE GROUP NUMBER,
1505                    ;                RETURNS WITH C=0 IF THE LINE WAS NOT LOCATED.
1506                    ;                TXTAD2 POINTS TO GROUP NUMBER OF NEXT HIGHEST LINE NUMBER
1507                    ;               (I. E., WHERE YOU WOULD INSERT THIS LINE)
1508                    ;
1509                    ;
1510 168D  A5 2C   FINDLN: LDA     GRPNO           ;PLACE LINE NUMBER OF LINE WE ARE
1511 168F  85 5A           STA     ITMP2H          ;LOOKING FOR INTO TEMPORARY
1512 1691  A5 2D           LDA     LINENO          ;STEP PART ALSO
1513 1693  85 59           STA     ITMP2L
1514 1695  A9 00           LDA#    0               ;SET FLAG INDICATING FIRST SEARCH
1515 1697  85 38           STA     FSWIT
1516 1699  85 2A           STA     TEXTP           ;ALSO RESET TEXT POINTER TO BEGIINING OF
1517 169B  A5 26           LDA     PC              ;CURRENT LINE.  IS CURRENT LINE DIRECT COMMAND?
1518 169D  10 03           BPL     CHKLIN          ;NO, THEN START SEARCHING FROM PRESENT POSITION
1519 169F  20 2517 FNDINI: JSR     TXTINI          ;YES, THEN RESET TEXT POINTERS TO START OF
1520 16A2  E6 38           INC     FSWIT           ;OF PROGRAM, INDICATE LAST SEARCH
1521 16A4  A4 2A   CHKLIN: LDY     TEXTP           ;GET TEXT POINTER
1522 16A6  B1 28           LDA@Y   TXTADR          ;GET THE GROUP NUMBER
1523 16A8  C9 FE           CMP#    EOP             ;END OF TEXT?
1524 16AA  F0 23           BEQ     NOFIND          ;BRANCH IF YES
1525 16AC  85 56           STA     ITMP1H          ;SAVE FOR COMPARISON
1526 16AE  85 51           STA     TGRP            ;ALSO SAVE IN CASE THIS ONE IS IT
1527 16B0  C8              INY                     ;POINT TO STEP NUMBER
1528 16B1  B1 28           LDA@Y   TXTADR          ;GET IT
1529 16B3  85 55           STA     ITMP1L          ;SAVE IT FOR COMPARISON
1530 16B5  85 52           STA     TLINE           ;ALSO SAVE IN CASE THIS IS IT
1531 16B7  C8              INY                     ;POINT TO FIRST CHAR IN LINE
1532 16B8  84 2A           STY     TEXTP           ;UPDATE TEXT POINTER
1533 16BA  38              SEC                     ;SETUP FOR SUBTRACT
1534 16BB  A5 55           LDA     ITMP1L          ;GET LOW ORDER
1535 16BD  E5 59           SBC     ITMP2L
1536 16BF  85 55           STA     ITMP1L          ;SAVE FOR LATER
1537 16C1  A5 56           LDA     ITMP1H          ;NOW HIGH ORDERS
1538 16C3  E5 5A           SBC     ITMP2H
1539 16C5  30 0E           BMI     FNEXT           ;BRANCH IF THE ONE IN THE TEXT AREA IS SMALLER
1540 16C7  05 55           ORA     ITMP1L          ;NOT BIGGER, IS IT EQUAL?
1541 16C9  F0 0F           BEQ     FOUNDL          ;BRANCH IF WE LOCATED THE LINE
1542 16CB  A5 38           LDA     FSWIT           ;LAST SEARCH ATTEMPT?
1543 16CD  F0 D0           BEQ     FNDINI          ;BRANCH IF NOT,  TRY AGAIN FROM START OF
1544                                               ;PROGRAM.
1545 16CF  18      NOFIND: CLC                     ;FLAG THE FACT WE DIDN'T FIND IT
1546 16D0  A0 00   FNEXIT: LDY#    0               ;RESET POINTER TO GROUP NUMBER
1547 16D2  84 2A           STY     TEXTP
1548 16D4  60              RTS
1549 16D5  20 DD16 FNEXT:  JSR     EATCR           ;FLUSH TO START OF NEXT LINE
1550 16D8  10 CA           BPL     CHKLIN          ;UNCONDITIONALLY LOOP FOR MORE
1551 16DA  38      FOUNDL: SEC                     ;FLAG THE FACT WE FOUND IT
1552 16DB  B0 F3           BCS     FNEXIT          ;AND RETURN
```

```
1553                            .PAGE                   ;'UTILITY ROUTINES FOR TEXT MANIPULATION'
1554                    ;
1555                    ;
1556                    ;       UTILTIY ROUTINES FOR TEXT MANIPULATION
1557                    ;
1558                    ;
1559                    ;       FLUSH TILL A CARRIAGE RETURN
1560                    ;
1561                    ;
1562                    ;
1563  160D  E6 20   EATCR:  INC     DEBGSW          ;DISABLE TRACE
1564  160F  20 8419 EATCRC: JSR     GETC            ;GET NEXT CHAR
1565  16E2  10 02           BPL     EATCNT          ;UNCONDITIONAL BRANCH
1566  16E4  E6 20   EATCR1: INC     DEBGSW          ;DISABLE TRACE
1567  16E6  A5 28   EATCNT: LDA     CHAR            ;GET THE CHAR
1568  16E8  C9 0D           CMP#    %15             ;CARRIAGE RETURN?
1569  16EA  D0 F3           BNE     EATCRC          ;BRANCH IF NOT
1570  16EC  A5 28           LDA     TXTADR          ;YES, CALCULATE START OF NEXT LINE
1571  16EE  18              CLC
1572  16EF  65 2A           ADC     TEXTP           ;ADD IN THE TEXT POINTER
1573  16F1  85 28           STA     TXTADR          ;SAVE IN POINTER
1574  16F3  85 33           STA     TXTAD2          ;AND ALTERNATE POINTER
1575  16F5  A5 29           LDA     TXTADR+1        ;NOW HIGH ORDER
1576  16F7  69 00           ADC#    0
1577  16F9  85 29           STA     TXTADR+1
1578  16FB  85 34           STA     TXTAD2+1        ;AND ALTERNATE POINTER
1579  16FD  A9 00           LDA#    0               ;AND RESET THE TEXT POINTER
1580  16FF  85 2A           STA     TEXTP
1581  1701  85 35           STA     TEXTP2
1582  1703  C6 20           DEC     DEBGSW          ;ALLOW TRACE AGAIN
1583  1705  60              RTS                     ;AND RETURN
1584                    ;
1585                    ;       FLUSH UNTIL END OF COMMAND (SEMI-COLON CR CARRIAGE RETURN)
1586                    ;
1587                    ;
1588  1706  20 8419 EATEC1: JSR     GETC            ;GET NEXT CHAR
1589  1709  10 02           BPL     EATECC          ;UNCONDITIONAL BRANCH
1590  170B  E6 20   EATECM: INC     DEBGSW          ;TURN OFF TRACE
1591  170D  A5 28   EATECC: LDA     CHAR            ;GET THE CHAR
1592  170F  20 5C13         JSR     TSTEOC          ;GO SEE IF ";" OR CARRAIGE RETURN
1593  1712  D0 F2           BNE     EATEC1          ;BRANCH IF NOT
1594  1714  C6 20           DEC     DEBGSW          ;ENABLE TRACE AGAIN
1595  1716  60              RTS                     ;AND RETURN
1596
```

```
                        .PAGE                   ;'UTILITY ROUTINES FOR TEXT MANIPULATION'
                ;
                ;       PUSH THE TEXT POINTERS ON THE STACK
                ;
                ;
1717  A2 28     PUSHTP: LDX#    TXTADR
1719  A0 04             LDY#    4               ;THREE PLUS 'CHAR'
171B  4C 7818           JMP     PUSHB0          ;* PJMP *
                ;
                ;       POP THE TEXT POINTERS OFF THE STACK
                ;
171E  A2 23     POPTP:  LDX#    TXTADR+3
1720  A0 04             LDY#    4
1722  4C 6818           JMP     POPB0           ;* PJMP *
                ;
                ;       INIT TEXT POINTER TO BEGGINNING OF TEXT
                ;
1725  A5 2F     TXTINI: LDA     TXTBEG          ;POINT TO START OF STORED TEXT
1727  85 28             STA     TXTADR
1729  A5 30             LDA     TXTBEG+1
172B  85 29             STA     TXTADR+1
172D  A9 00             LDA#    0               ;INIT OFFSET TO ZERO
172F  85 2A             STA     TEXTP
1731  60                RTS
                ;
                ;       "NEWLIN" SETUP TEXT POINTERS AND PC FOR NEW LINE NUMBER
                ;
1732  A5 2C     NEWLIN: LDA     GRPNO           ;GET THE LINE NUMBER
1734  85 26             STA     PC              ;STORE IN PROGRAM COUNTER
1736  A5 2D             LDA     LINENO
1738  85 27             STA     PC+1
173A  A0 02             LDY#    2               ;POINT TO FIRST CHAR ON LINE
173C  84 2A             STY     TEXTP
173E  60                RTS                     ;AND RETURN
                ;
                ;       "NXTLIN" SETUP TEXT POINTERS AND PC FOR NEXT LINE NUMBER
                ;
173F  A4 2A     NXTLIN: LDY     TEXTP           ;GET TEXT POINTER
1741  B1 28             LDA@Y   TXTADR          ;PICK UP GROUP NUMBER
1743  C9 FE             CMP#    EOP             ;END OF PROGRAM?
1745  F0 0F             BEQ     NONEXT          ;BRANCH IF NO NEXT LINE
1747  85 26             STA     PC              ;SAVE AS NEW LINE NUMBER
1749  C8                INY
174A  B1 28             LDA@Y   TXTADR          ;GET STEP NUMBER
174C  85 27             STA     PC+1            ;STORE IT
174E  C8                INY                     ;POINT TO FIRST CHAR ON THE LINE
174F  84 2A             STY     TEXTP
1751  20 8419           JSR     GETC            ;GET THE FIRST CHAR OF NEW LINE
1754  38                SEC                     ;FLAG THE FACT WE HAVE A NEXT LINE
1755  60                RTS                     ;AND RETURN
1756  18        NONEXT: CLC                     ;INDICATE WE HAVE NO NEXT LINE
1757  60                RTS                     ;AND RETURN
```

```
1649                            .PAGE                   ;'DELETE A LINE OF STORED PROGRAM'
1650                    ;
1651                    ;
1652                    ;       "DELETE" A LINE OF STORED PROGRAM
1653                    ;
1654                    ;
1655  1758  20 1717 DELETE: JSR    PUSHTP          ;SAVE TEXT POINTERS
1656  175B  A0 02           LDY#   2               ;SKIP OVER LINE NUMBER
1657  175D  84 2A           STY    TEXTP
1658  175F  20 DD16         JSR    EATCR           ;SKIP TO THE CARRIAGE RETURN
1659  1762  20 1E17         JSR    POPTP           ;RESTORE POINTER TO START OF LINE TO ZAP
1660  1765  20 1717         JSR    PUSHTP          ;BUT KEEP THEM AROUND
1661  1768  A0 00           LDY#   0               ;SET OFFSET TO ZERO
1662  176A  B1 33   DMVLOP: LDA@Y  TXTAD2          ;GET A CHAR
1663  176C  91 28           STA@Y  TXTADR          ;MOVE IT DOWN
1664  176E  C9 FE           CMP#   EOP             ;END OF TEXT REACHED YET?
1665  1770  F0 09           BEQ    DELDON          ;BRANCH IF YES
1666  1772  C8              INY                    ;NO POINT TO NEXT CHAR TO MOVE
1667  1773  D0 F5           BNE    DMVLOP          ;BRANCH IF NO OVERFLOW ON OFFSET
1668  1775  E6 29           INC    TXTADR+1        ;OVERFLOW, BUMP HIGH ORDERS
1669  1777  E6 34           INC    TXTAD2+1
1670  1779  D0 EF           BNE    DMVLOP          ;UNCONDITIONALLY MOVE NEXT BYTE
1671  177B  C8       DELDON: INY
1672  177C  84 2A           STY    TEXTP           ;SAVE OFFSET
1673  177E  A5 28           LDA    TXTADR          ;GET BASE ADDR
1674  1780  18              CLC
1675  1781  65 2A           ADC    TEXTP           ;ADD IN THE OFFSET
1676  1783  85 3E           STA    VARBEG          ;SAVE AS START OF VARIABLE LIST
1677  1785  A5 29           LDA    TXTADR+1        ;GET HIGH ORDER
1678  1787  69 00           ADC#   0               ;ADD IN THE CARRY
1679  1789  85 3F           STA    VARBEG+1        ;SAVE IT
1680  178B  20 1A18         JSR    INSDON          ;FLAG VARIABLE LIST AS EMPTY
1681  178E  4C 1E17         JMP    POPTP           ;* PJMP * RESTORE POINTERS TO POINT TO
1682                                                ;WHERE WE WOULD INSERT LINE.
1683                    ;
1684                    ;
1685                    ;
```

```
1686                            .PAGE                   ;'INSERT - A LINE IN STORED PROGRAM'
1687                    ;
1688                    ;
1689                    ;       "INSERT" A LINE IN THE STORED PROGRAM TEXT AREA
1690                    ;
1691                    ;
1692  1791  20 1717 INSERT: JSR     PUSHTP          ;SAVE TEXT POINTERS ACCROSS CALL
1693  1794  20 8D16        JSR     FINDLN          ;TRY TO LOCATE THE LINE
1694  1797  90 06          BCC     INSCNT          ;BRANCH IF LINE DOES NOT EXIST
1695  1799  20 5817        JSR     DELETE          ;LINE EXISTS, DELETE IT
1696  179C  20 8D16        JSR     FINDLN          ;RE-FIND TO SET UP POINTERS AGAIN
1697  179F  20 1E17 INSCNT: JSR     POPTP           ;GET COMBUF POINTERS BACK
1698  17A2  C6 2A          DEC     TEXTP           ;POINT TO THE LINE NUMBER DELIMITER
1699  17A4  20 1717        JSR     PUSHTP          ;BUT KEEP THEM AROUND
1700  17A7  A2 33          LDX#    TXTAD2          ;SAVE POINTER TO PLACE TO INSERT
1701  17A9  A0 04          LDY#    4               ;ON STACK
1702  17AB  20 7818        JSR     PUSHB0
1703  17AE  A2 02          LDX#    2               ;SET COUNTER FOR 3 BYTES MINIMUM
1704  17B0  E8        IFCR: INX                     ;COUNT THIS BYTE
1705  17B1  20 8419        JSR     GETC            ;GET IT FROM COMMAND BUFFER
1706  17B4  C9 0D          CMP#    %15             ;CARRIAGE RETURN?
1707  17B6  D0 F8          BNE     IFCR            ;NO, KEEP COUNTING
1708  17B8  86 5F          STX     TEMP1           ;SAVE COUNT TEMPORARILY
1709  17BA  A0 00          LDY#    0               ;OFFSET TO ZERO
1710  17BC  B1 33          LDA@Y   TXTAD2          ;GET THE LAST CHAR TO SLIDE DOWN
1711  17BE  48             PHA                     ;SAVE FOR LATER
1712  17BF  A9 FD          LDA#    UMARK           ;FLAG THE LOC WITH ALL ONES
1713  17C1  91 33          STA@Y   TXTAD2
1714  17C3  A5 3E          LDA     VARBEG          ;GET ADDR OF START OF VARIABLE LIST
1715  17C5  85 28          STA     TXTADR          ;SAVE FOR LATER
1716  17C7  18             CLC
1717  17C8  65 5F          ADC     TEMP1           ;ADD IN AMOUNT TO MOVE DOWNWARD
1718  17CA  85 33          STA     TXTAD2          ;SAVE FOR LATER
1719  17CC  85 3E          STA     VARBEG          ;SAVE AS NEW START OF VARIABLE LIST
1720  17CE  A5 3F          LDA     VARBEG+1        ;NOW HIGH ORDER
1721  17D0  85 29          STA     TXTADR+1
1722  17D2  69 00          ADC#    0
1723  17D4  85 34          STA     TXTAD2+1
1724  17D6  85 3F          STA     VARBEG+1
1725                                                ;AND FALL INTO MOVE LOOP
```

```
                        .PAGE                   ;'INSERT - A LINE IN STORED PROGRAM'
                                                ;PICK UP A BYTE
1708  B1 28     IMVLOP: LDA@Y   TXTADR
170A  C9 FD             CMP#    UMARK           ;END OF MOVE?
170C  F0 0D             BEQ     IMVDON          ;BRANCH IF YES
170E  91 33             STA@Y   TXTAD2          ;NO, THEN SLIDE IT DOWN
1710  88                DEY                     ;DECREMENT OFFSET?
1711  C0 FF             CPY#    $FF             ;OVERFLOW?
1713  D0 F3             BNE     IMVLOP          ;BRANCH IF NOT
1715  C6 29             DEC     TXTADR+1        ;OVERFLOW, BUMP HIGH ORDER ADDRESSES
1717  C6 34             DEC     TXTAD2+1
1719  D0 ED             BNE     IMVLOP          ;UNCONDITIALLY LOOP FOR MORE
171B  68          IMVDON: PLA                     ;GET THE LAST BYTE BACK AGAIN
171C  91 33             STA@Y   TXTAD2          ;STORE IT AWAY
171E  20 1E17           JSR     POPTP           ;RESTORE POINTERS TO PLACE TO INSERT INTO
1721  A2 36             LDX#    TXTAD2+3        ;RESTORE POINTERS TO COMBUF
1723  A0 04             LDY#    4
1725  20 6818           JSR     POPBØ
1728  A4 2A             LDY     TEXTP           ;GET OFFSET
172A  A5 2C             LDA     GRPNO           ;GET THE GROUP NUMBER
172C  91 28             STA@Y   TXTADR          ;STORE IT IN PROGRAM AREA
172E  C8                INY
172F  A5 2D             LDA     LINENO          ;GET THE STEP NUMBER
1731  91 28             STA@Y   TXTADR          ;STORE IT IN PROGRAM AREA
1733  C8                INY                     ;POINT TO WHERE FIRST CHAR GOES
1734  84 2A             STY     TEXTP           ;SAVE IT FOR LATER
1736  A4 35     INSLOP: LDY     TEXTP2          ;GET POINTER TO CHAR
1738  B1 33             LDA@Y   TXTAD2          ;PICK IT UP
173A  C8                INY                     ;BUMP IT
173B  84 35             STY     TEXTP2          ;STORE IT BACK
173D  A4 2A             LDY     TEXTP           ;POINT TO WHERE IT GOES
173F  91 28             STA@Y   TXTADR          ;PUT IT THERE
1741  C9 0D             CMP#    %15             ;CARRIAGE RETURN YET?
1743  F0 05             BEQ     INSDON          ;BRANCH IF YES
1745  C8                INY                     ;NO POINT TO NEXT
1746  84 2A             STY     TEXTP           ;SAVE POINTER
1748  D0 EC             BNE     INSLOP          ;UNCONDITIONALLY LOOP FOR MORE
174A  A0 00     INSDON: LDY#    Ø               ;OFFSET TO ZERO
174C  A9 FF             LDA#    EOV             ;FLAG VARIABLE LIST AS EMPTY
174E  91 3E             STA@Y   VARBEG
1750  A5 3E             LDA     VARBEG          ;AND UPDATE 'VAREND'
1752  85 42             STA     VAREND
1754  A5 3F             LDA     VARBEG+1
1756  85 43             STA     VAREND+1
1758  60                RTS                     ;AND RETURN
                        ;
                        ;
                        ;
                        ;
```

```
1777                            .PAGE                   ;'SOFTWARE STACK MANIPULATION ROUTINES'
1778                    ;
1779                    ;
1780                    ;
1781                    ;       "PUSHJ" PUSH-JUMP TO A ROUTINE
1782                    ;
1783                    ;       CALLING SEQUENCE IS:
1784                    ;
1785                    ;       JSR     PUSHJ                   ;CALL THIS SUBROUTINE
1786                    ;       .WORD   ROUTINE                 ;TWO BYTE ADDR OF ROUTINE TO GO TO
1787                    ;                                       ;*** NOTE: THIS WORD CANNOT OVERLAP
1788                    ;                                       ;*** A PAGE BOUNDARY.
1789                    ;                       <-----          ;RETURN IS HERE VIA "POPJ" ROUTINE
1790                    ;
1791                    ;
1792 1829  68.        PUSHJ:  PLA                     ;GET LOW ORDER RETURN ADDR FROM STACK
1793 182A  A8                 TAY                     ;PLACE IN Y REGISTER
1794 182B  C8                 INY                     ;INCREMENT TO GET LOW ORDER TO JUMP INDIRECT
1795 182C  8C 3C18            STY     PJADR1          ;STORE IN JMP INDIRECT INSTRUCTION
1796 182F  C8                 INY                     ;BUMP FOR THE RTS IN "POPJ"
1797 1830  98                 TYA                     ;PLACE IN ACCUMULATOR
1798 1831  20 3E18            JSR     PUSHA           ;SAVE ON STACK FOR LATER
1799 1834  68                 PLA                     ;GET HIGH ORDER RETURN ADDR
1800 1835  8D 3D18            STA     PJADR1+1        ;STORE IN JMP INDIRECT INSTRUCTION
1801 1838  20 3E18            JSR     PUSHA           ;SAVE FOR LATER RETURN VIA "POPJ"
1802                    ;
1803       183C               .DEF    PJADR1=.+1      ;ADDRESS OF LOW ORDER JUMP INDIRECT
1804                    ;
1805 183B  6C 0000            JMP@    $0000           ;ADDR IS OVERWRITTEN FROM ABOVE CODE
1806                                                    ;THIS JUMP WILL GO TO "ROUTINE".
1807                    ;
1808                    ;
1809                    ;
1810                    ;       "PUSHA" PUSH THE ACCUMULATOR ON THE SOFTWARE STACK
1811                    ;
1812                    ;       CALLING SEQUENCE IS:
1813                    ;       JSR     PUSHA
1814                    ;
1815                    ;
1816 183E  A4 4E      PUSHA:  LDY     PDP             ;GET THE SOFTWARE STACK POINTER
1817 1840  91 4C              STA@Y   PDPADR          ;STORE THE ACCUMULATOR VIA POINTER
1818 1842  88                 DEY                     ;DECREMENT SOFT STACK POINTER
1819 1843  C0 FF              CPY#    $FF             ;IS NEW VALUE $FF ?
1820 1845  D0 02              BNE     PUSHRT          ;NO, THEN BASE ADDR IS OK
1821 1847  C6 4D              DEC     PDPADR+1        ;YES, DECREMENT BASE ADDRESS BY ONE PAGE
1822 1849  84 4E      PUSHRT: STY     PDP             ;STORE UPDATED POINTER
1823 184B  60                 RTS                     ;AND RETURN
1824                    ;
1825                    ;
```

```
                        .PAGE                   ;'SOFTWARE STACK MANIPULATION ROUTINES'
                ;
                ;
                ;       "POPA" POP ITEM OFF SOFTWARE STACK INTO THE ACCUMULATOR
                ;
                ;       CALLING SEQUENCE IS:
                ;       JSR     POPA
                ;
                ;
184C  A4 4E     POPA:   LDY     PDP             ;LOAD THE SOFTWARE STACK POINTER
184E  C8                INY                     ;INCREMENT SO IT POINTS TO NEW ITEM
184F  D0 02             BNE     PHOK            ;BRANCH IF HIGH ORDER BASE ADDR IS OK
1851  E6 4D             INC     PDPADR+1        ;NOT OK, INCREMENT IT BY ONE PAGE
1853  B1 4C     PHOK:   LDA@Y   PDPADR          ;RETRIEVE ITEM FROM SOFT STACK
1855  84 4E             STY     PDP             ;RESTORE UPDATED POINTER
1857  60                RTS                     ;AND RETURN
                ;
                ;
                ;       "POPJ" RETURN TO ADDRESS SAVED BY A CALL TO "PUSHJ"
                ;
                ;       CALLING SEQUENCE IS:
                ;       JSR     POPJ
                ;
                ;
1858  BA       POPJ:  TSX                     ;LOAD X WITH CURRENT HARDWARE STACK POINTER
1859  20 4C18        JSR     POPA            ;GET HIGH ORDER ADDR TO RETURN TO
185C  9D 0201        STAX    STACK+2         ;OVERWRITE RETURN ADDR
185F  20 4C18        JSR     POPA            ;GET LOW ORDER ADDR TO RETURN TO
1862  9D 0101        STAX    STACK+1         ;OVERWRITE RETURN ADDR
1865  60             RTS                     ;NOW RETURN TO THE PROPER PLACE, WHICH
                                             ;IS PAST THE "JSR PUSHJ"
                                             ;               ",WORD ROUTINE"
                                             ;SEQUENCE--->
                ;
                ;
```

```
1862                            ,PAGE                   ;'SOFTWARE STACK MANIPULATION ROUTINES'
1863                    ;
1864                    ;
1865                    ;
1866                    ;       "POPB0" POP BYTES OFF OF STACK INTO PAGE ZERO
1867                    ;
1868  1866  A0 02   POPB2:  LDY#    2               ;ENTRY POINT WHEN WE NEED TWO BYTES ONLY
1869  1868  84 5F   POPB0:  STY     TEMP1           ;SAVE Y REGISTER
1870  186A  20 4C18         JSR     POPA            ;GET A BYTE FROM STACK
1871  186D  95 00           STAX    $0000           ;PUT IT IN PAGE ZERO
1872  186F  A4 5F           LDY     TEMP1           ;GET Y REG BACK
1873  1871  CA              DEX                     ;COUNT X DOWN
1874  1872  88              DEY                     ;DONE YET?
1875  1873  D0 F3           BNE     POPB0           ;LOOP FOR MORE
1876  1875  60              RTS                     ;YES, RETURN
1877                    ;
1878                    ;
1879                    ;       "PUSHB0" PUSH BYTES FROM PAGE ZERO ONTO STACK
1880                    ;
1881                    ;
1882  1876  A0 02   PUSHB2: LDY#    2               ;ENTRY POINT WHEN WE NEED ONLY TWO
1883  1878  84 5F   PUSHB0: STY     TEMP1           ;SAVE Y REGISTER
1884  187A  B5 00           LDAX    $0000           ;GET THE VALUE FROM PAGE ZERO
1885  187C  20 3E18         JSR     PUSHA           ;SAVE ON STACK
1886  187F  A4 5F           LDY     TEMP1           ;GET Y REG BACK
1887  1881  E8              INX                     ;BUMP X TO POINT TO NEXT BYTE
1888  1882  88              DEY                     ;DONE YET?
1889  1883  D0 F3           BNE     PUSHB0          ;LOOP FOR MORE
1890  1885  60              RTS                     ;RETURN
1891
1892
```

```
                        .PAGE                   ;"PUSH" AND "POP" FLOATING POINT NUMBERS

                        ;"PUSH" FAC1 ONTO STACK

1886  A2 F8   PHFAC1: LDX#    -NUMBF            ;GET NEG OF NUMBER OF BYTES TO PUSH
1888  B5 85   PHF1B:  LDAX    X1+NUMBF          ;GET A BYTE OF NUMBER
188A  20 3E18         JSR     PUSHA             ;PUSH IT ON SOFTWARE STACK
188D  E8              INX                       ;POINT TO NEXT ONE
188E  30 F8           BMI     PHF1B             ;LOOP TILL ALL PUSHED
1890  60              RTS                       ;AND RETURN

                        ;"PUSH" FAC2 ONTO STACK

1891  A2 F8   PHFAC2: LDX#    -NUMBF            ;GET NEG OF NUMBER OF BYTES TO PUSH
1893  B5 80   PHF2B:  LDAX    X2+NUMBF          ;GET A BYTE OF NUMBER
1895  20 3E18         JSR     PUSHA             ;PUSH IT ONTO THE STACK
1898  E8              INX                       ;POINT TO NEXT BYTE
1899  30 F8           BMI     PHF2B             ;AND LOOP TILL ALL PUSHED
189B  60              RTS                       ;AND RETURN

                        ;"PUSH" FLOATING POINT TEMPORARY ONTO STACK

189C  A2 F8   PHTMP:  LDX#    -NUMBF            ;GET NEG OF NUMBER OF BYTES TO PUSH
189E  B5 A5   PHTB:   LDAX    T+NUMBF           ;GET A BYTE OF NUMBER
18A0  20 3E18         JSR     PUSHA             ;PUSH IT ONTO STACK
18A3  E8              INX                       ;POINT TO NEXT BYTE
18A4  30 F8           BMI     PHTB              ;AND LOOP TILL ALL ARE PUSHED
18A6  60              RTS                       ;AND RETURN

                        ;"POP" NUMBER ON STACK INTO FAC2

18A7  A2 04   PLFAC2: LDX#    NUMBF-1           ;POINT TO LAST BYTE
18A9  20 4C18 PLF2B:  JSR     POPA              ;POP ITEM FROM STACK INTO ACCUMULATOR
18AC  95 78           STAX    X2                ;STORE INTO FAC2
18AE  CA              DEX                       ;POINT TO NEXT BYTE
18AF  10 F8           BPL     PLF2B             ;LOOP TILL ALL POPPED
18B1  60              RTS                       ;AND RETURN

                        ;"POP" NUMBER FROM STACK INTO FLOATING POINT TEMPORARY

18B2  A2 04   PLTMP:  LDX#    NUMBF-1           ;POINT TO LAST BYTE
18B4  20 4C18 PLTB:   JSR     POPA              ;GET ITEM FROM STACK INTO ACCUMULATOR
18B7  95 A0           STAX    T                 ;STORE IT IN TEMPORARY
18B9  CA              DEX                       ;POINT TO NEXT BYTE
18BA  10 F8           BPL     PLTB              ;AND LOOP TILL ALL POPPED
18BC  60              RTS                       ;AND RETURN
```

```
1939                            .PAGE                   ;'CHARACTER MANIPULATING ROUTINES'
1940                    ;
1941                    ;
1942                    ;       READ ONE CHARACTER WITH NO ECHO
1943                    ;
1944                    ;
1945  18BD  A5 6B   RNOECH: LDA     ECHFLG              ;GET ECHO CONTROL FLAG
1946  18BF  48              PHA                         ;SAVE ON STACK
1947  18C0  A9 01           LDA#    1                   ;NO DISABLE ECHOING
1948  18C2  85 6B           STA     ECHFLG
1949  18C4  20 CD18         JSR     READC               ;GET A CHAR FROM INPUT DEVICE
1950  18C7  AA              TAX                         ;SAVE CHAR IN X REGISTER
1951  18C8  68              PLA                         ;GET OLD ECHO FLAG VALUE BACK
1952  18C9  85 6B           STA     ECHFLG
1953  18CB  8A              TXA                         ;GET THE CHAR INPUT
1954  18CC  60              RTS                         ;AND RETURN
1955                    ;
1956                    ;       "READC" - READ ONE CHARACTER FROM INPUT DEVICE
1957                    ;
1958  18CD  A6 66   READC:  LDX     IDEV                ;GET CURRENT DEVICE NUMBER
1959  18CF  10 03           BPL     READC1              ;BRANCH IF DEVICE NUMBER IS POSITIVE
1960  18D1  4C BF20         JMP     RSTRNG              ;* PJMP * NEGATIVE, READ FROM STRING AND RETURN
1961  18D4  BD 9825 READC1: LDAX    IDSPH               ;GET HIGH ORDER DISPATCH ADDR
1962  18D7  85 60           STA     TEMP1+1             ;STORE IT AWAY
1963  18D9  BD 9D25         LDAX    IDSPL               ;GET LOW ORDER
1964  18DC  85 5F           STA     TEMP1               ;STORE IT AWAY
1965  18DE  20 5E00         JSR     JSRIND              ;AND CALL THE INPUT ROUTINE
1966  18E1  90 05           BCC     READCC              ;BRANCH IF NO ERRORS
1967  18E3  20 1510 IERRI:  JSR     CLRDEV              ;RESET DEVICES ON AN I-O ERROR
1968  18E6  00              BRK                         ;TRAP
1969  18E7  E9              .BYTE   ERRI                ;?I-O ERROR ON INPUT DEVICE
1970  18E8  85 2B   READCC: STA     CHAR                ;SAVE CHAR
1971  18EA  C9 0D           CMP#    CR                  ;IS IT A CR
1972  18EC  F0 03           BEQ     READCE              ;BRANCH IF YES
1973  18EE  A5 2B   READCR: LDA     CHAR                ;GET CHAR BACK
1974  18F0  60              RTS
1975
1976  18F1  A9 0A   READCE: LDA#    LF                  ;FOLLOW CARRIAGE RETURNS WITH LINE FEE
1977  18F3  20 FA18         JSR     PRINTC              ;PRINT IT
1978  18F6  10 F6           BPL     READCR              ;UNCONDITIONAL BRANCH
```

```
1979                            .PAGE                   ;'MORE CHARACTER MANIPULATING ROUTINES'
1980
1981                    ;
1982                    ;       "PRINTC" - PRINT THE CHAR IN ACCUMULATOR OR 'CHAR'
1983                    ;
1984  18F8  A9 20       PSPACE: LDA#    ' '             ;OUTPUT A SPACE
1985  18FA  48          PRINTC: PHA                     ;SAVE THE CHAR IN THE AC
1986  18FB  A6 67               LDX     ODEV            ;GET CURRENT OUTPUT DEVICE NUMBER
1987  18FD  10 06               BPL     PUSEA1          ;BRANCH IF DEVICE NUMBER IS POSITIVE
      18FF  A4 4A               LDY   STOPNT    ;GET POINTER TO NEXT CHAR
      1901  4C B0 20            JMP   WSTRNG    ;* PJMP * WRITE TO STRING
      1904  EA                  NOP             ;PATCH FILL
1990  1905  BD A225    PUSEA1: LDAX    ODSPH           ;GET HIGH ORDER ADDR OF ROUTINE TO DO OUTPUT
1991  1908  85 60              STA     TEMP1+1         ;SAVE IT
1992  190A  BD A725            LDAX    ODSPL           ;GET LOW ORDER ADDR OF ROUTINE
1993  190D  85 5F              STA     TEMP1           ;SAVE IT
1994  190F  68                 PLA                     ;GET CHAR BACK
1995  1910  48                 PHA                     ;BUT SAVE ACCROSS CALL
1996  1911  20 5E00             JSR     JSRIND          ;CALL THE ROUTINE TO DO THE OUTPUT
1997  1914  90 05              BCC     PRRET           ;BRANCH IF NO ERRORS
1998  1916  20 1510  OERRO:    JSR     CLRDEV          ;RESET I-O DEVICES IF ERROR
1999  1919  00                 BRK                     ;TRAP
2000  191A  DE                 .BYTE   ERRO            ;?I-O ERROR ON OUTPUT DEVICE
2001  191B  68          PRRET:  PLA                     ;RESTORE THE CHARACTER
2002  191C  60                 RTS                     ;AND RETURN
2003                    ;
2004                    ;       "PACKC" - ROUTINE TO PACK A CHAR INTO MEMORY
2005                    ;
2006  191D  A5 29       PACKC:  LDA     CHAR            ;GET CHAR
2007  191F  A4 2A       PACKC1: LDY     TEXTP           ;GET THE TEXT POINTER
2008  1921  C9 7F               CMP#    RUBCHR          ;RUBOUT?
2009  1923  F0 13              BEQ     RUB1            ;YES, THEN HANDLE IT
2010  1925  C9 5F              CMP#    LINCHR          ;IS THIS THE 'LINE-DELETE' CHARACTER?
2011  1927  F0 29              BEQ     RUBLIN          ;BRANCH IF IT IS, RUBOUT THE ENTIRE LINE
2012  1929  91 28              STA@Y   TXTADR          ;STORE CHAR INTO MEMORY
2013  192B  C8          PCKRUB: INY                     ;BUMP TEXT POINTER
2014  192C  C0 7F              CPY#    LINEL           ;EXCEEDED MAX LINE LENGTH?
2015  192E  10 03              BPL     PBIG            ;BRANCH IF WE HAVE EXCEEDED IT
2016  1930  84 2A       PCKRET: STY     TEXTP           ;SAVE TEXT POINTER
2017  1932  60          RTS1:   RTS                     ;AND RETURN
2018  1933  00          PBIG:   BRK                     ;TRAP
2019  1934  FF                  .BYTE   LTL             ;?LINE TOO LONG
```

```
2020                            .PAGE                   ;'MORE CHARACTER MANIPULATING ROUTINES'
2021
2022
2023                            ;ROUTINE TO RUB OUT ONE CHARACTER
2024
2025  1935  C0 00    RUB1:     CPY#    0               ;ANYTHING TO RUBOUT?
2026  1937  F0 F7              BEQ     PCKRET          ;BRANCH IF NOT, RETURN
2027  1939  A4 68              LDY     ECHFLG          ;HAS USER ENABLED CHARACTER ECHOING?
2028  193B  D0 0E              BNE     RUB1CC          ;BRANCH IF IT IS DISABLED.  DON'T ECHO ANYTHING
2029  193D  A4 6C              LDY     DELSPL          ;DO WE DO SPECIAL CRT RUBOUT PROCESSING?
2030  193F  F0 05              BEQ     RUB1C           ;BRANCH IF WE DO NOT
2031  1941  20 6F19            JSR     EATTVC          ;YES, THEN EAT THE CHAR OFF THE CRT SCREEN
2032  1944  10 05              BPL     RUB1CC          ;AND UNCONDITIONALLY BRANCH
2033  1946  A9 5C    RUB1C:    LDA#    RUBECH          ;ECHO SPECIAL CHARACTER FOR RUBOUT
2034  1948  20 FA18            JSR     PRINTC
2035  194B  A4 2A    RUB1CC:   LDY     TEXTP           ;LOAD Y REGISTER AGAIN WITH TEXT POINTER
2036  194D  88                DEY                     ;DECREMENT TEXT POINTER
2037  194E  10 E0              BPL     PCKRET          ;RETURN IF STILL POSITIVE
2038  1950  30 D9              BMI     PCKRUB          ;IF PAST BEGINNING, SET TO ZERO
2039
2040
2041                            ;ROUTINE TO RUB OUT THE ENTIRE LINE
2042
2043
2044  1952  C0 00    RUBLIN:   CPY#    0               ;ANYTHING TO RUBOUT?
2045  1954  F0 DA              BEQ     PCKRET          ;BRANCH IF NOT, RETURN
2046  1956  A4 68              LDY     ECHFLG          ;HAS USER ENABLED CHARACTER ECHOING?
2047  1958  D0 0C              BNE     RUBLC           ;BRANCH IF IT IS DISABLED.  DON'T ECHO ANYTHING.
2048  195A  A4 6C              LDY     DELSPL          ;DO WE DO SPECIAL CRT RUBOUT PROCESSING?
2049  195C  F0 08              BEQ     RUBLC           ;BRANCH IF WE DO NOT
2050  195E  20 6F19  RUBLCL:   JSR     EATTVC          ;EAT A CHAR OFF OF CRT SCREEN
2051  1961  C6 2A              DEC     TEXTP           ;ZAP IT FROM BUFFER
2052  1963  D0 F9              BNE     RUBLCL          ;AND LOOP TILL ALL ARE ZAPPED
2053  1965  60                RTS                     ;AND RETURN
2054  1966 A0 00      RUBLC:   LDY#    0
2055  1968 F0 C6               BEQ     PCKRET
2056  196A 20 E8 18   TRACBG:  JSR     READCC          ;CHECK FOR "CR" FOLLOW WITH "LF"
2057  196D 10 8B               BPL     PRINTC          ;ASSUME PARITY BIT 0
2058
2059
2060                            ;ROUTINE TO EAT A CHAR FROM A CRT SCREEN BY SENDING
2061                            ;A "BACKSPACE", "SPACE", "BACKSPACE" SEQUENCE.
2062
2063
2064  196F  20 7519  EATTVC:   JSR     BACKSP          ;OUTPUT A BACKSPACE
2065  1972  20 F818            JSR     PSPACE          ;FOLLOWED BY A SPACE
2066  1975  A9 08    BACKSP:   LDA#    %10             ;GET BACKSPACE CHAR
2067  1977  4C FA18            JMP     PRINTC          ;* PJMP * OUTPUT IT AND RETURN
```

```
                        .PAGE                   ;'MORE CHARACTER MANIPULATION ROUTINES'
                        ;
                        ;
                        ;       "GETC" GET A CHAR FROM MEMORY, ECHO IF TRACE IS ON
                        ;
197A  A4 20     GETCX:  LDY     DEBGSW          ;IS TRACE DISABLED?
197C  D0 17             BNE     GETC1           ;YES, THEN DON'T LOOK AT FLAG
197E  A5 21             LDA     DMPSW           ;FLIP THE STATE OF THE DUMP SWITCH
1980  49 FF            EOR#    $FF
1982  85 21             STA     DMPSW           ;AND STORE IT BACK
1984  A5 24     GETC:   LDA     INSW            ;WHERE DO WE GET THE CHAR FROM?
1986  F0 03             BEQ     GETCC           ;FROM MEMORY
1988  4C CD18           JMP     READC           ;* PJMP * GO GET FROM THE INPUT DEVICE
198B  A4 2A     GETCC:  LDY     TEXTP           ;GET TEXT POINTER
198D  E6 2A             INC     TEXTP           ;BUMP IT NOW TO POINT TO NEXT CHAR
198F  B1 28             LDA@Y   TXTADR          ;GET THIS CHAR
1991  C9 3F             CMP#    '?              ;QUESTION MARK?
1993  F0 E5             BEQ     GETCX           ;YES, GO HANDLE
1995 85 2B      GETC1:  STA     CHAR            ;STORE AWAY FOR OTHER USES
1997 48                 PHA                     ;SAVE IT ON STACK
1998 A5 20              LDA     DEBGSW          ;CHECK TO SEE IF WE PRINT IT
199A 05 21              ORA     DMPSW           ;FOR DEBUGGING
199C D0 19              BNE     TESTN+07        ;NO
199E 68                 PLA                     ;GET BACK CHARACTER
199F 20 6A 19           JSR     TRACBG          ;FIX FOR TRACE BUG
19A2 60                 RTS                     ;CHARACTER IS RETURNED
                        ;       "SPNOR" ROUTINE TO IGNORE LEADING SPACES
                        ;
                        ;
19A3  20 8419   GSPNOR: JSR     GETC            ;CALL GETC FIRST
19A6  A5 2B     SPNOR:  LDA     CHAR            ;GET THE CHAR
19A8  C9 20     SPNOR1: CMP#    ' '             ;IS IT A SPACE?
19AA  F0 F7             BEQ     GSPNOR          ;YES, THEN IGNORE
19AC  60                RTS                     ;NO, RETURN
                        ;
                        ;
                        ;       "TESTN"  TESTS TO SEE IF CHARACTER IS A NUMBER
                        ;
19AD  20 A319   TESTNS: JSR     GSPNOR          ;GET NEXT NON-BLANK
19B0  A5 2B     TESTN:  LDA     CHAR            ;GET CHAR
19B2  48        TESTN1: PHA                     ;SAVE CHARACTER ON STACK
19B3  49 30             EOR#    $30             ;CONVERT TO BCD (IF A NUMBER)
19B5  C9 0A             CMP#    $0A             ;SET C BIT IF GREATER THAN 9
19B7  68                PLA                     ;RESTORE CHARACTER TO ACCUMULATOR
19B8  60                RTS                     ;AND RETURN (C BIT CLEAR IF NUMBER)
                        ;
                        ;
                        ;
```

```
                        .PAGE                   ;'EVAL - EXPRESSION EVALUATOR'
                        ;
                        ;
                        ;       "EVAL" EVALUATE AN EXPRESSION.
                        ;
                        ;       RECURSIVE EXPRESSION EVALUATOR.
                        ;
                        ;
                        ;
                        ;
                        ;
19B9  A9 00   EFUN:     LDA#    0               ;GET A ZERO
19BB  0A      EFUNL:    ASLA                    ;ROTATE LEFT TO HASH
19BC  85 65             STA     ETEMP1          ;SAVE IT
19BE  20 8419           JSR     GETC            ;GET NEXT CHAR OF FUNCTION NAME
19C1  20 321B           JSR     TTERMS          ;TERMINATOR?
19C4  F0 07             BEQ     EFNAME          ;BRANCH IF END OF NAME
19C6  29 1F             AND#    %37             ;KEEP ONLY 5 BITS
19C8  18                CLC
19C9  65 65             ADC     ETEMP1          ;ADD IN THE HASH
19CB  D0 EE             BNE     EFUNL           ;UNCONDITIONALLY LOOP FOR MORE
19CD  C9 28   EFNAME:   CMP#    '(              ;LEFT PAREN?
19CF  F0 02             BEQ     EFUNC           ;BRANCH IF YES
19D1  00                BRK                     ;TRAP
19D2  E3                .BYTE   PFERR           ;?PARENTHESES ERROR IN FUNCTION
19D3  A5 65   EFUNC:    LDA     ETEMP1          ;GET THE HASH CODE FOR FUNCTION NAME
19D5  20 3E18           JSR     PUSHA           ;SAVE FOR LATER
19D8  20 2916           JSR     PUSHJ           ;MOVE PAST PAREN, EVALUATE FIRST ARG
19DB  F5 19             .WORD   EVALM1
19DD  20 4C18           JSR     POPA            ;GET THE NAME BACK AGAIN
19E0  AA                TAX                     ;TRANSFER TO X REGISTER
19E1  20 2918           JSR     PUSHJ           ;AND GO DO THE FUNCTION
19E4  DB 10             .WORD   FUNC
19E6  4C BF1A           JMP     ERPAR           ;GO SEE IF TERMINATOR IS A RIGHT PAREN

                        ;HERE FOR A QUOTED CONSTANT

19E9  20 8419 ECHAR:    JSR     GETC            ;GET CHARACTER FOLLOWING QUOTE
19EC  20 4322           JSR     FLT8            ;AND MAKE IT A FLOATING POINT NUMBER
19EF  20 8419           JSR     GETC            ;MOVE PAST CHARACTER
19F2  4C 751A           JMP     OPNEXT          ;AND CHECK FOR OPERATOR
```

```
2160                            .PAGE                   ;'MAIN ENTRY POINT(S) TO "EVAL"
2161                    ;
2162                    ;
2163  19F5  20 8419  EVALM1: JSR     GETC            ;ENTER HERE TO ADVANCE PAST CURRENT CHAR
2164  19F8  A9 00    EVAL:   LDA#    0               ;ASSUME LOWEST LEVEL ARITHMETIC OPERATION
2165  19FA  85 75             STA     LASTOP
2166  19FC  85 38             STA     STRSWT          ;MAKE SURE STRING VARIABLE SWITCH IS OFF
2167  19FE  20 BD1C           JSR     ZRFAC1          ;ASSUME VALUE OF EXPRESSION IS ZERO
2168  1A01  A5 75    ARGNXT: LDA     LASTOP          ;SAVE LAST OPERATION ON STACK
2169  1A03  20 3E18           JSR     PUSHA
2170  1A06  20 3D1B           JSR     TTERMS          ;GO SEE IF THIS CHAR IS A TERMINATOR
2171  1A09  D0 03             BNE     ECHKC           ;BRANCH IF NOT
2172  1A0B  4C B61A           JMP     ETERM1          ;YES, THEN HANDLE
2173  1A0E  C9 46    ECHKC:  CMP#    'F              ;IS IT A FUNCTION?
2174  1A10  F0 A7             BEQ     EFUN            ;BRANCH IF YES
2175  1A12  C9 27             CMP#    ''              ;IS IT A CHARACTER CONSTANT? ('X)
2176  1A14  F0 03             BEQ     ECHAR           ;BRANCH IF YES
2177  1A16  C9 2E             CMP#    '.              ;IS IT A FRACTION?
2178  1A18  F0 1C             BEQ     ENUM            ;BRANCH IF YES, CALL FLOATING INPUT ROUTINE
2179  1A1A  20 B219           JSR     TESTN1          ;NO, BUT IS IT A NUMBER?
2180  1A1D  B0 28             BCS     EGTVAR          ;BRANCH IF NOT A NUMBER
2181                                                  ;*** START OF KLUDGE HACK TO SPEED THINGS UP ***
2182  1A1F  A5 24             LDA     INSW            ;ARE WE INPUTTING FROM INPUT DEVICE?
2183  1A21  D0 13             BNE     ENUM            ;BRANCH IF YES, CALL FLOATING INPUT ROUTINE
2184  1A23  A2 02             LDX#    2               ;NO, THEN WE CAN LOOK AHEAD TO SEE IF
2185  1A25  A4 2A             LDY     TEXTP           ;CONSTANT IS IN RANGE 0-99
2186  1A27  B1 28    KLOOP:  LDA@Y   TXTADR          ;GET NEXT CHAR
2187  1A29  C8                INY                     ;BUMP POINTER
2188  1A2A  C9 2E             CMP#    '.              ;DOES NUMBER HAVE A FRACTIONAL PART
2189  1A2C  F0 08             BEQ     ENUM            ;BRANCH IF IT DOES, CALL FLOATING POINT INPUT
2190  1A2E  20 B219           JSR     TESTN1          ;IS THIS CHAR A DIGIT 0-9 ALSO?
2191  1A31  B0 09             BCS     FSTNUM          ;BRANCH IF NOT, THEN CALL FAST INPUT ROUTINE
2192  1A33  CA                DEX                     ;CAN ONLY HAVE UP TO TWO DIGITS
2193  1A34  D0 F1             BNE     KLOOP           ;LOOK AT NEXT ONE
2194                                                  ;IF WE FALL OUT OF THE LOOP, WE HAVE TO
2195  1A36  20 2F24  ENUM:   JSR     FIN             ;CALL FLOATING POINT INPUT ROUTINE (SLOW!)
2196  1A39  4C 751A           JMP     OPNEXT          ;AND LOOP FOR OPERATOR
2197  1A3C  A5 28    FSTNUM: LDA     CHAR            ;GET FIRST DIGIT OF NUMBER
2198  1A3E  20 B41D           JSR     GETIN           ;CALL FAST INPUT ROUTINE FOR NUMBERS 0-99
2199  1A41  20 4322           JSR     FLT8            ;CALL FAST ONE BYTE FLOAT ROUTINE
2200  1A44  4C 751A           JMP     OPNEXT          ;AND LOOK FOR OPERATOR
2201                                                  ;*** END OF KLUDGE HACK ***
```

```
2202                            .PAGE                   ;'EVAL - EXPRESSION EVALUATOR'
2203
2204
2205  1A47  20 2918 EGTVAR: JSR     PUSHJ               ;IT MUST BE A VARIABLE, GET VALUE
2206  1A4A  43 1B           .WORD   GETVAR
2207  1A4C  A5 2B           LDA     CHAR                ;GET CHARACTER WHICH TERMINATED THE VARIABLE
2208  1A4E  C9 3D           CMP#    '=                  ;DOES HE WANT SUBSTITUTION?
2209  1A50  D0 23           BNE     OPNEXT              ;BRANCH IF NOT, JUST A TERM
2210  1A52  A2 37           LDX#    VARADR              ;YES, THEN SAVE THE INFO ABOUT THIS VARIABLE
2211  1A54  A0 05           LDY#    5
2212  1A56  20 7818         JSR     PUSHB0              ;ON STACK
2213  1A59  20 2918         JSR     PUSHJ               ;CALL OURSELVES TO EVALUATE THE EXPRESSION
2214  1A5C  F5 19           .WORD   EVALM1
2215  1A5E  A2 3B           LDX#    VARADR+4            ;RESTORE POINTERS TO VARIABLE
2216  1A60  A0 05           LDY#    5
2217  1A62  20 6818         JSR     POPB0
2218  1A65  A5 3B           LDA     STRSWT              ;WAS THE VARIABLE A STRING VARIABLE?
2219  1A67  D0 05           BNE     SETSTR              ;BRANCH IF IT WAS
2220  1A69  20 931C         JSR     PUTVAR              ;NO, THEN STORE EXPRESSION VALUE AS VARIABLE'S
2221  1A6C  F0 07           BEQ     OPNEXT              ;VALUE, AND THIS VALUE IS ALSO VALUE OF THIS TERM
2222  1A6E  20 851F SETSTR: JSR     INTGER              ;KEEP ONLY 8 BITS FOR VALUE
2223  1A71  A4 3A           LDY     VSUB+1              ;POINT TO POSITION IN STRING
2224  1A73  91 37           STA@Y   VARADR              ;STORE IT INTO STRING, FALL INTO 'OPNEXT'
2225  1A75  20 301B OPNEXT: JSR     TTERMS              ;GO SEE IF NEXT NON-SPACE IS SPECIAL
2226  1A78  D0 04           BNE     MISOPR              ;BRANCH IF NOT
2227  1A7A  E0 06           CPX#    6                   ;LEFT PAREN?
2228  1A7C  D0 08           BNE     OPNXT1              ;NO, THAT'S GOOD AS WE CAN'T HAVE ONE HERE
2229  1A7E  00        MISOPR: BRK                       ;TRAP TO ERROR HANDLER
2230  1A7F  FB                .BYTE   OPRMIS              ;?OPERATOR MISSING - EVAL
2231                  ;
2232  1A80  4C 211A JARGN:  JMP     ARGNXT              ;BRANCH AIC
2233                  ;
2234  1A83  20 5818 EVALRT: JSR     POPJ                ;RETURN FROM CALL TO "EVAL"
```

```
2235                            .PAGE                   ;'EVAL - EXPRESSION EVALUATOR'
2236                    ;
2237                    ;
2238  1A86  E2 27   OPNXT1: CPX#   7                     ;IS THIS A DELIMITER?
2239  1A88  30 C2           BMI     OPNXT2              ;BRANCH IF NOT
2240  1A8A  A2 00           LDX#    0                   ;YES, THEN THE OPERATION LEVEL IS LOWEST
2241  1A8C  20 4C18 OPNXT2: JSR     POPA                ;GET LAST OPERATION LEVEL
2242  1A8F  85 5F           STA     TEMP1               ;SAVE IF FOR COMPARE
E4 -2243  1A91  E4 5F           CPX     TEMP1               ;IS THIS OPERATION LEVEL < OR = TO LAST ONE?
2244  1A93  30 02           BMI     DOBOP               ;BRANCH IF YES
2245  1A95  D0 12           BNE     ESTACK              ;BRANCH IF NO
2246  1A97  09 00   DOBOP:  ORA#    0                   ;TO RESET FLAGS AFTER CPX
2247  1A99  85 75           STA     LASTOP              ;YES, THEN GET THE LAST OPERATOR
2248  1A9B  F0 E6           BEQ     EVALRT              ;IF LOWEST LEVEL, THEN WE ARE ALL DONE
2249  1A9D  8A              TXA                         ;SAVE 'THISOP'
2250  1A9E  48              PHA                         ;ON HARDWARE STACK
2251  1A9F  20 A718         JSR     PLFAC2              ;POP PARTIAL RESULT BACK INTO FAC2
2252  1AA2  20 DA1A         JSR     EVBOP               ;AND GO DO THE OPERATION LEAVING THE
2253                                                    ;RESULT IN FLAC
2254  1AA5  68              PLA                         ;GET 'THISOP' BACK
2255  1AA6  AA              TAX
2256  1AA7  10 E3           BPL     OPNXT2              ;UNCONDITIONAL BRANCH WITH NEW PARTIAL
2257                                                    ;RESULT.
```

```
                        .PAGE                   ;'EVAL - EXPRESSION EVALUATOR'
                    ;
                    ;
1AA9  20 3E18 ESTACK: JSR     PUSHA             ;SAVE BACK ON STACK FOR LATER COMPUTATION
1AAC  86 75           STX     LASTOP            ;NOW UPDATE 'LASTOP' TO 'THISOP'
1AAE  20 8618         JSR     PHFAC1            ;SAVE PARITAL RESULT ON STACK
1AB1  20 8419         JSR     GETC              ;SKIP OVER THE OPERATOR
1AB4  10 0A           BPL     JARGN             ;UNCONDITIONAL BRANCH TO PICK UP NEXT ARG
1AB6  E0 06   ETERM1: CPX#    6                 ;LEFT PAREN?
1AB8  D0 12           BNE     ETERM2            ;BRANCH IF NOT
1ABA  20 2918         JSR     PUSHJ             ;ENTERING NEW LEVEL OF NESTING
1ABD  F5 19           .WORD   EVALM1            ;SO CALL OURSELVES TO EVALUATE IT!
1ABF  A5 28   ERPAR:  LDA     CHAR              ;GET THE DELIMITER WHICH ENDED THIS LEVEL
1AC1  48              PHA                       ;SAVE IT MOMENTARILY
1AC2  20 8419         JSR     GETC              ;MOVE PAST IT
1AC5  68              PLA                       ;GET DELIMITER BACK
1AC6  C9 29           CMP#    ')                ;RIGHT PAREN?
1AC8  F0 AB           BEQ     OPNEXT            ;YES.  GO PICK UP NEXT OPERATOR
1ACA  00      EPMISS: BRK                       ;TRAP TO ERROR HANDLER
1ACB  FA              .BYTE   PMATCH            ;?PARENTHESES MISMATCH - EVAL
1ACC  E0 07   ETERM2: CPX#    7                 ;DELIMITER ON RIGHT HAND SIDE?
1ACE  10 04           BPL     ETERM3            ;BRANCH IF YES
1AD0  E0 03           CPX#    3                 ;OR UNARY OPERATOR
1AD2  10 04           BPL     MISOPN            ;NO, THEN IT CAN'T BE HERE
1AD4  A5 75   ETERM3: LDA     LASTOP            ;PICK UP OPERATION LEVEL
1AD6  F0 AE           BEQ     OPNXT1            ;ONLY ALLOW IF AT LOWEST LEVEL
1AD8  00      MISOPN: BRK                       ;TRAP TO ERROR HANDLER
1AD9  F9              .BYTE   OPNMIS            ;?OPERAND MISSING - EVAL
                    ;
1ADA  A6 75   EVBOP:  LDX     LASTOP            ;GET THE ARITHMETIC OPERATION TO PERFORM
1ADC  BD 6925         LDAX    EVOSPH            ;GET THE HIGH ORDER ADDR OF ROUTINE
1ADF  85 60           STA     TEMP1+1           ;STORE IT
1AE1  BD 6E25         LDAX    EVOSPL            ;GET LOW ORDER ADDR OF ROUTINE
1AE4  85 5F           STA     TEMP1             ;STORE IT
1AE6  6C 5F00         JMP@    TEMP1             ;* PJMP * TO ROUTINE TO DO THE OPERATION
```

```
                                .PAGE                   ;'EVALUATE A POWER'


                                ;*** NOTE: THIS ROUTINE IS CURRENTLY RESTRICTED TO RAISING
                                ;***       THE NUMBER TO AN INTEGER POWER WITHIN THE RANGE OF
                                ;***       + OR - 32,767.


1AE9  20 991F  EVPOWR: JSR     INTFIX          ;GET EXPONENT
1AEC  85 55            STA     ITMP1L          ;STORE IT AWAY
1AEE  A5 82            LDA     M1+1            ;AS NUMBER OF TIMES TO DO OPERATION
1AF0  85 56            STA     ITMP1H
1AF2  A2 93            LDX#    FONE            ;GET THE CONSTANT 1.0 INTO FAC1
1AF4  A0 88            LDY#    FAC1
1AF6  20 9724          JSR     MOVXY
1AF9  A5 56            LDA     ITMP1H          ;RAISING TO A NEGATIVE POWER?
1AFB  30 1A            BMI     NPOWR           ;BRANCH IF WE ARE
1AFD  A9 FF    POWRLP: LDA#    $FF             ;POSITIVE POWER, ARE WE DONE YET?
1AFF  C6 55            DEC     ITMP1L
1B01  C5 56            CMP     ITMP1L
1B03  D0 02            BNE     POWR1           ;BRANCH IF HIGH ORDER OK
1B05  C6 56            DEC     ITMP1H          ;DECREMENT HIGH ORDER
1B07  C5 56    POWR1:  CMP     ITMP1H          ;DONE YET?
1B09  F0 32            BEQ     TTRET           ;YES, THEN RETURN
1B0B  20 9118          JSR     PHFAC2          ;NO, SAVE FAC2
1B0E  20 A229          JSR     FMUL            ;NUMBER TIMES ITSELF (EXCEPT FIRST TIME)
1B11  20 A718          JSR     PLFAC2          ;RESTORE NUMBER TO FAC2
1B14  4C FD1A          JMP     POWRLP          ;AND KEEP MULTIPLYING

                                ;HERE IF RAISING TO A NEGATIVE POWER

1B17  A5 56    NPOWR:  LDA     ITMP1H          ;DONE YET?
1B19  F0 22            BEQ     TTRET           ;BRANCH IF ALL DONE
1B1B  E6 55            INC     ITMP1L          ;NO, THEN COUNT UP SINCE COUNT IS NEGATIVE
1B1D  D0 02            BNE     NPOWR1
1B1F  E6 56            INC     ITMP1H          ;INCREMENT HIGH ORDER ALSO
1B21  20 2322  NPOWR1: JSR     SWAP            ;PUT PARTIAL INTO FAC2, 'X' INTO FAC1
1B24  20 9618          JSR     PHFAC1          ;SAVE 'X'
1B27  20 CC22          JSR     FDIV            ;1/(X*X*X*X . . .)
1B2A  20 A718          JSR     PLFAC2          ;RESTORE 'X'
1B2D  4C 171B          JMP     NPOWR           ;AND LOOP TILL DONE
```

```
2336                            .PAGE                   ;'ROUTINES USED BY "EVAL"'
2337                    ;
2338                    ;
2339                    ;
2340                    ;       TEST TO SEE IF CHARACTER IS A SPECIAL TERMINATOR
2341                    ;
2342                    ;
2343  1B33  20 A619 TTERMS: JSR    SPNOR           ;IGNORE SPACES, GET NEXT NON-BLANK CHAR
2344  1B36  A2 0C           LDX#   TRMAX           ;GET MAX TABLE OFFSET
2345  1B38  DD 0124 TRMCHK: CMPX   TRMTAB          ;MATCH?
2346  1B3B  F0 03           BEQ    TTRET           ;YES, RETURN WITH Z=1
2347  1B3D  CA              DEX                    ;POINT TO NEXT ENTRY
2348  1B3E  10 F8           BPL    TRMCHK          ;AND CHECK IT
2349  1B40  60        TTRET: RTS                    ;RETURN (NOTE: Z=0 IF CANNOT FIND)
2350                    ;
```

```
                        .PAGE                   ;'GETVAR - GET A VARIABLE FROM VARIABLE LIST'
                        ;
                        ;
                        ;       "GETVAR" - GET A VARIABLE FROM THE VARIABLE LIST
                        ;                  OTHERWISE CREATE IT AND ASSIGN IT A VALUE OF ZERO
                        ;
                        ;
                        ;
1B3E  4C CA1A   GPMISS: JMP     EPMISS          ;BRANCH AIC
                        ;
1B41  00        GTERR3: BRK                     ;TRAP
1B42  F5                .BYTE   FUNILL          ;?FUNCTION ILLEGAL HERE
                        ;
                        ;
1B43  A9 00     GETVAR: LDA#    0               ;ASSUME VARIABLE IS NOT A STRING VARIABLE
1B45  85 38             STA     STRSWT
1B47  20 A619           JSR     SPNOR           ;(DEFENSIVE!) GET THE CHARACTER
1B4A  C9 26             CMP#    '&              ;IS IT SPECIAL 'FSBR' SCRATCH VARIABLE?
1B4C  F0 0A             BEQ     VAROK           ;YES, THEN NAME IS OK
1B4E  C9 41             CMP#    'A              ;IS IT ALPHABETIC?
1B50  30 04             BMI     VARBAD          ;BRANCH IF NOT
1B52  C9 5B             CMP#    %133            ;'Z' + 1
1B54  30 02             BMI     VAROK           ;BRANCH IF ALPHABETIC
1B56  00        VARBAD: BRK                     ;NOT ALPHABETIC
1B57  F4                .BYTE   BADVAR          ;?BAD VARIABLE NAME
1B58  C9 46     VAROK:  CMP#    'F              ;FUNCTION?
1B5A  F0 E5             BEQ     GTERR3          ;BRANCH IF YES
1B5C  0A                ASLA
1B5D  0A                ASLA
1B5E  0A                ASLA                    ;SHIFT ALPHA LEFT 3
1B5F  48                PHA                     ;SAVE IT
1B60  20 8419           JSR     GETC            ;GET NEXT CHARACTER
1B63  49 30             EOR#    $30             ;CONVERT TO BCD IF A NUMBER
1B65  C9 08             CMP#    $8              ;IS IT 0-7?
1B67  B0 09             BCS     VARDUN          ;IF NOT, NAME IS ON STACK
1B69  85 30             STA     VCHAR           ;IF YES, SAVE IT
1B6B  68                PLA                     ;GET BACK ALPHA PART
1B6C  05 30             ORA     VCHAR           ;PUT THE PARTS TOGETHER
1B6E  48                PHA                     ;AND STICK ON STACK
1B6F  20 8419           JSR     GETC            ;GET A NEW CHARACTER IN CHAR
```

```
                        .PAGE                   ;'GETVAR - GET A VARIABLE FROM VARIABLE LIST'


1872  A5 28   VARDUN: LDA     CHAR
1874  C9 24           CMP#    '$              ;STRING VARIABLE?
1876  D0 05           BNE     VARDN1          ;BRANCH IF NOT, PRESS ON
1878  85 38           STA     STRSWT          ;YES, FLAG THE FACT
187A  20 8419         JSR     GETC            ;AND MOVE PAST THE '$'
187D  68        VARDN1: PLA                     ;GET VARIABLE NAME OFF STACK
187E  20 3E18           JSR     PUSHA           ;PUT NAME ON SOFT STACK
1881  20 A619           JSR     SPNOR           ;GET NEXT NON-BLANK
1884  C9 28           CMP#    '(              ;LEFT PAREN?
1886  F0 06           BEQ     VARSUB          ;BRANCH IF VARIABLE HAS A SUBSCRIPT
1888  A9 00           LDA#    0               ;OTHERWISE ASSUME ZERO
188A  85 3A           STA     VSUB+1          ;ZERO THE SUBSCRIPT
188C  F0 21           BEQ     VARSOK          ;AND PROCESS IT
188E  A5 38     VARSUB: LDA     STRSWT          ;SAVE STRING FLAG
1890  20 3E18           JSR     PUSHA
1893  20 2718           JSR     PUSHJ
1896  F5 19           .WORD   EVALM1          ;CALL EVAL TO CALCULATE SUBSCRIPT
1898  20 4018           JSR     POPA            ;RESTORE STRING FLAG
189B  85 38           STA     STRSWT
189D  A5 28           LDA     CHAR            ;GET TERMINATOR
189F  C9 29           CMP#    ')              ;PAREN MATCH?
18A1  D0 98           BNE     GPMISS          ;BRANCH IF NOT
18A3  20 8419         JSR     GETC            ;MOVE PAST THE RIGHT PAREN
18A6  20 1A23         JSR     FIX             ;MAKE SUBSCRIPT AN INTEGER
18A9  A5 83           LDA     X1+3            ;GET LOW ORDER BYTE
18AB  85 3A           STA     VSUB+1          ;STORE IT
18AD  A5 82           LDA     X1+2            ;GET HIGH ORDER BYTE.  NOTE: 16 BITS ONLY.
18AF  85 39     VARSOK: STA     VSUB            ;SAVE FOR LATER
18B1  20 4018           JSR     POPA            ;GET THE VARIABLE NAME BACK
18B4  85 30           STA     VCHAR           ;SAVE IT
18B6  20 631C FNDVAR: JSR     VARINI          ;SET ADDR TO START OF VARIABLE LIST
18B9  A0 00   CHKVAR: LDY#    0               ;SET OFFSET TO ZERO
18BB  B1 37           LDA@Y   VARADR          ;GET THE VARIABLE NAME
18BD  C9 FF           CMP#    EOV             ;IS THIS THE END OF THE VARIABLE LIST?
18BF  F0 57           BEQ     NOVAR           ;BRANCH IF END OF LIST
18C1  C9 FC           CMP#    STRMRK          ;IS THIS VARIABLE IN THE LIST A STRING VARIABLE?
18C3  F0 09           BEQ     CHKSTR          ;BRANCH IF YES, WE HANDLE DIFFERENTLY
18C5  C5 30           CMP     VCHAR           ;ARE THE NAMES THE SAME?
18C7  F0 37           BEQ     CHKSUB          ;YES, GO SEE IF SUBSCRIPTS ARE EQUAL
18C9  20 711C NOTVAR: JSR     NXTVAR          ;POINT TO NEXT VARIABLE IN LIST
18CC  D0 EB           BNE     CHKVAR          ;UNCONDITIONAL BRANCH TO CHECK NEXT VARIABLE
```

```
2435                            .PAGE                   ;'GETVAR - GET A VARIABLE FROM VARIABLE LIST'
2436
2437
2438  1BCE  A5 33   CHKSTR: LDA     STRSWT              ;ARE WE LOOKING FOR A STRING VARIABLE?
2439  1BD0  F0 1E           BEQ     SKPSTR              ;BRANCH IF NOT, JUST SKIP OVER THIS STRING VARIABLE
2440  1BD2  A5 30           LDA     VCHAR               ;YES, THEN GET IT'S NAME
2441  1BD4  C8              INY                         ;POINT TO NAME OF STRING VARIABLE IN VARIABLE LIST
2442  1BD5  D1 37           CMP@Y   VARADR              ;IS THIS THE ONE WE ARE LOOKING FOR?
2443  1BD7  D0 18           BNE     SKPST1              ;BRANCH IF NOT, JUST SKIP OVER IT
2444  1BD9  C8              INY                         ;YES, THIS IS THE ONE.  GET THE SIZE OF THE
2445  1BDA  B1 37           LDA@Y   VARADR              ;STRING
2446  1BDC  85 30           STA     VSIZE               ;STORE FOR THOSE WHO NEED IT
2447  1BDE  C8              INY                         ;AND UPDATE
2448  1BDF  98              TYA
2449  1BE0  20 731C         JSR     UPDVAR              ;'VARADR' TO POINT TO BASE ADDR OF STRING
2450  1BE3  A4 3A   GETSTC: LDY     VSUB+1              ;GET SUBSCIPT (POSITION) OF BYTE WE WANT
2451  1BE5  B1 37           LDA@Y   VARADR              ;GET THE BYTE WE WANT
2452  1BE7  85 B2           STA     M1+1                ;STORE AS LOW ORDER 8 BITS
2453  1BE9  A9 00           LDA#    0
2454  1BEB  85 B1           STA     M1                  ;ZERO HIGH ORDER
2455  1BED  4C D51E         JMP     FL16PJ              ;* PJMP * FLOAT AND RETURN THE VALUE
2456  1BF0  C8        SKPSTR: INY                       ;MOVE OVER STRING VARIABLE'S NAME
2457  1BF1  C8        SKPST1: INY                       ;POINT TO STRING VARIABLE'S LENGTH
2458  1BF2  B1 37           LDA@Y   VARADR              ;GET STRING LENGTH
2459  1BF4  48              PHA                         ;SAVE IT
2460  1BF5  C8              INY                         ;POINT TO FIRST BYTE IN STRING
2461  1BF6  98              TYA                         ;UPDATE 'VARADR' TO BASE OF STRING
2462  1BF7  20 731C         JSR     UPDVAR
2463  1BFA  68              PLA                         ;GET SIZE OF STRING
2464  1BFB  20 731C         JSR     UPDVAR              ;UPDATE 'VARADR' BY PROPER AMOUNT
2465  1BFE  D0 B9           BNE     CHKVAR              ;AND LOOK FOR NEXT VARIABLE IN THE LIST
2466  1C00  A5 33   CHKSUB: LDA     STRSWT              ;ARE WE LOOKING FOR A STRING VARIABLE
2467  1C02  D0 C6           BNE     NOTVAR              ;BRANCH IF WE ARE, CAN'T BE THIS NUMERIC VARIABLE
2468  1C04  A5 39           LDA     VSUB                ;GET HIGH ORDER SUBSCRIPT WE ARE LOOKING FOR
2469  1C06  C8              INY                         ;POINT TO SUBSCRIPT IN LIST
2470  1C07  D1 37           CMP@Y   VARADR              ;ARE THEY THE SAME?
2471  1C09  D0 BE           BNE     NOTVAR              ;BRANCH IF THIS ONE IS NOT IT
2472  1C0B  A5 3A           LDA     VSUB+1              ;ARE LOW ORDERS ALSO THE SAME?
2473  1C0D  C8              INY
2474  1C0E  D1 37           CMP@Y   VARADR
2475  1C10  D0 B7           BNE     NOTVAR              ;BRANCH IF THEY ARE NOT THE SAME
2476                        ;
2477                        ;
```

```
                        .PAGE                   ;'GETVAR - GET A VARIABLE FROM VARIABLE LIST'
                        ;
                        ;
1C12  20 A31C LOCVAR: JSR     FETVAR            ;GET THE VARIABLE'S VALUE INTO FLAC
1C15  20 5818         JSR     POPJ              ;AND RETURN TO CALLER
1C18  A5 38   NOVAR:  LDA     STRSWT            ;IS THIS A STRING VARIABLE?
1C1A  D0 21           BNE     NOSTR             ;BRANCH IF IT IS A STRING VARIABLE
1C1C  A5 3D           LDA     VCHAR             ;GET THE VARIABLE'S NAME
1C1E  91 37           STA@Y   VARADR            ;STORE IT IN LIST
1C20  C8              INY                       ;POINT TO NEXT IN LIST
1C21  A5 39           LDA     VSUB              ;GET HIGH ORDER SUBSCRIPT
1C23  91 37           STA@Y   VARADR            ;SAVE IT IN LIST
1C25  C8              INY                       ;POINT TO NEXT
1C26  A5 3A           LDA     VSUB+1            ;GET LOW ORDER SUBSCRIPT
1C28  91 37           STA@Y   VARADR            ;SAVE IT IN LIST
1C2A  A9 00           LDA#    0                 ;GET A ZERO
1C2C  A2 06           LDX#    NUMBF+1           ;GET COUNT OF NUMBER OF BYTES IN NUMBER + 1
1C2E  C8      ZERVAR: INY                       ;POINT TO NEXT BYTE IN VARIABLE
1C2F  91 37           STA@Y   VARADR            ;ZERO OUT VARIABLE'S VALUE
1C31  CA              DEX                       ;COUNT THIS BYTE
1C32  D0 FA           BNE     ZERVAR            ;LOOP TILL DONE.  NOTE: EXTRA ZERO AT END
1C34  A9 FF           LDA#    EOV               ;FLAG END OF VARIABLE LIST
1C36  91 37           STA@Y   VARADR            ;FLAG END OF VARIABLE LIST
1C38  20 7F1C         JSR     UPDEND            ;UPDATE THE END OF THE VARIABLE LIST
                        ;
1C3B  D0 D6           BNE     LOCVAR            ;UNCONDITIONAL BRANCH, AS WE HAVE FOUND
                                                ;THE VARIABLE


                        ;HERE WHEN STRING VARIABLE WAS NOT FOUND

1C3D  A9 FC   NOSTR:  LDA#    STRMRK            ;ADD STRING MARKER TO END OF VARIABLE LIST
1C3F  91 37           STA@Y   VARADR
1C41  C8              INY
1C42  A5 3D           LDA     VCHAR             ;ADD IT'S NAME
1C44  91 37           STA@Y   VARADR
1C46  C8              INY
1C47  A5 64           LDA     STRSIZ            ;GET DEFAULT STRING SIZE
1C49  91 37           STA@Y   VARADR            ;STORE AS SIZE OF STRING
1C4B  85 3C           STA     VSIZE             ;ALSO STORE FOR OTHERS WHO NEED TO KNOW
1C4D  C8              INY                       ;POINT TO FIRST BYTE OF STRING
1C4E  98              TYA                       ;UPDATE 'VARADR'
1C4F  20 731C         JSR     UPDVAR
1C52  A0 00           LDY#    0                 ;POINT TO FIRST BYTE OF STRING
1C54  A9 20           LDA#    ' '               ;GET A BLANK
1C56  91 37   STRINI: STA@Y   VARADR            ;SET STRING TO ALL BLANKS
1C58  C8              INY
1C59  C4 64           CPY     STRSIZ            ;DONE YET?
1C5B  D0 F9           BNE     STRINI            ;NO, LOOP TILL STRING IS ALL BLANKS
1C5D  A9 FF           LDA#    EOV               ;GET END OF VARIABLE LIST MARKER
1C5F  91 37           STA@Y   VARADR            ;FLAG END OF LIST
1C61  98              TYA                       ;UPDATE 'VARADR'
1C62  20 811C         JSR     UPDENV
1C65  4C E31B         JMP     GETSTC            ;GET BYTE FROM STRING, AND RETURN
```

```
                        .PAGE                   ;'GETVAR - GET A VARIABLE FROM VARIABLE LIST'

1C68  A5 3E   VARINI: LDA     VARBEG            ;GET ADDR OF START OF VARIABLE LIST
1C6A  85 37           STA     VARADR            ;AND SET UP POINTER
1C6C  A5 3F           LDA     VARBEG+1
1C6E  85 38           STA     VARADR+1
1C70  60              RTS                       ;AND RETURN
                      ;
                      ;
1C71  A9 08   NXTVAR: LDA#    VARSIZ            ;ADD IN SIZE OF NUMERIC VARIABLE
1C73  18      UPDVAR: CLC                       ;SETUP FOR ADDITION
1C74  65 37           ADC     VARADR            ;TO 'VARADR'
1C76  85 37           STA     VARADR
1C78  A5 38           LDA     VARADR+1
1C7A  69 00           ADC#    0
1C7C  85 38           STA     VARADR+1
1C7E  60              RTS

                        ;ROUTINE TO UPDATE THE END OF THE VARIABLE LIST

1C7F  A9 08   UPDEND: LDA#    VARSIZ            ;ADD IN SIZE OF NUMERIC VARIABLE
1C81  18      UPDENV: CLC
1C82  65 37           ADC     VARADR            ;ADD NUMBER IN ACCUMULATOR TO 'VARADR'
1C84  85 42           STA     VAREND            ;AND STORE RESULT IN 'VAREND'
1C86  A5 38           LDA     VARADR+1
1C88  69 00           ADC#    0                 ;ADD IN THE CARRY
1C8A  85 43           STA     VAREND+1
1C8C  60      BOMSVR: RTS                       ;AND RETURN

                        ;ROUTINE TO BOMB OUT IF THE VARIABLE IS A STRING VARIABLE

1C8D  A5 3B   BOMSTV: LDA     STRSWT            ;GET STRING FLAG
1C8F  F0 FB           BEQ     BOMSVR            ;RETURN IF NOT A STRING
1C91  00              BRK                       ;TRAP
1C92  DD              .BYTE   SVNA              ;?STRING VARIABLE NOT ALLOWED HERE
```

```
                                .PAGE                   ;'VARIABLE MANIPULATION UTILITIES'

                                ;       "PUTVAR" PUT NUMBER IN FAC1 INTO THE VARIABLE

1C93  A0 03   PUTVAR: LDY#      3                       ;POINT TO START OF VALUE
1C95  B9 7000 PUTV1:  LDAY      X1-3                    ;GET A BYTE FROM FAC1
1C98  91 37           STA@Y     VARADR                  ;STORE IT INTO VARIABLE
1C9A  C8              INY                               ;POINT TO NEXT BYTE
1C9B  C0 08           CPY#      VARSIZ                  ;REACHED END OF VARIABLE YET?
1C9D  D0 F6           BNE       PUTV1                   ;NO, THEN MOVE SOME MORE
1C9F  60              RTS                               ;*** MUST RETURN WITH Z BIT = 1 ! ***

                                ;       "FETVAR" FETCH VARIABLE VALUE INTO FAC1

1CA1  A0 03   FETVAR: LDY#      3                       ;POINT TO START OF VALUE
1CA2  B1 37   FETV1:  LDA@Y     VARADR                  ;GET A BYTE FROM VARIABLE
1CA4  99 7000         STAY      X1-3                    ;PUT IT INTO FAC1
1CA7  C8              INY                               ;POINT TO NEXT BYTE
1CA8  C0 08           CPY#      VARSIZ                  ;REACHED END OF VARIABLE YET?
1CAA  D0 F6           BNE       FETV1                   ;NO, THEN MOVE ANOTHER BYTE
1CAC  60              RTS                               ;YES, RETURN

                                ;       "PUSHIV" PUSH INCREMENT AND VARIABLE ADDR ON STACK
                                ;                USED BY "FOR" COMMAND.

1CAD  A2 37   PUSHIV: LDX#      VARADR                  ;POINT TO VARIABLE ADDR
1CAF  20 7618         JSR       PUSHB2                  ;PUSH IT ONTO STACK
1CB2  4C 9118         JMP       PHFAC2                  ;* PJMP * PUSH FAC2 ONTO STACK AND RETURN

                                ;       "POPIV" POP INCREMENT AND VARIABLE ADDR OFF STACK

1CB5  20 A716 POPIV:  JSR       PLFAC2                  ;RESTORE INTO FAC2
1CB8  A2 38           LDX#      VARADR+1                ;POINT TO VARIABLE ADDR
1CBA  4C 6618         JMP       POPB2                   ;* PJMP * RESTORE INTO VARIABLE ADDR AND RETURN

                                ;ZERO THE FLOATING POINT ACCUMULATOR FAC1

1CBD  A2 04   ZRFAC1: LDX#      NUMBF-1                 ;POINT TO LAST BYTE
1CBF  A9 00           LDA#      0                       ;LOAD A ZERO
1CC1  95 50   ZRFAC:  STAX      X1                      ;ZERO THE BYTE
1CC3  CA              DEX                               ;POINT TO NEXT ONE
1CC4  D0 FB           BNE       ZRFAC                   ;LOOP TILL ALL OF MANTISSA ZEROED
1CC6  A9 80           LDA#      $80                     ;NOW SET EXPONENT
1CC8  85 50           STA       X1
1CCA  60              RTS                               ;AND RETURN.  MUST RETURN WITH N BIT = 1 !
```

```
                                .PAGE                   ;'INTERRUPT HANDLERS'
                        ;
                        ;
1CC8 -  48      NOTBRK: PHA                             ;SAVE THE PROCESSOR STATUS
1CC9    A9 ED           LDA#    UNKINT                  ;UNKNOWN INTERRUPT
1CCB    48              PHA                             ;SAVE CODE ON STACK
1CCC    D0 22           BNE     BERROR                  ;AND PRINT THE ERROR CODE
1CCE    EA      NMISRV: NOP                             ;CURRENTLY PUNT NMI'S AS UNKNOWNS
1CCF    85 71   INTSRV: STA     ACSAV                   ;SAVE ACCUMULATOR
1CD1    68              PLA                             ;GET THE PROCESSOR STATUS
1CD2    24 73           BIT     MSKBRK                  ;IS B BIT ON?
1CD4    F0 F2           BEQ     NOTBRK                  ;BRANCH TO INTERRUPT SERVICE CHAIN
1CD6 -  85 72           STA     STATUS                  ;SAVE OLD PROCESSOR STATUS
1CD8    68              PLA                             ;GET THE LOW ORDER RETURN ADDR
1CD9    18              CLC                             ;GET READY FOR ADC
1CDA    69 FF           ADC#    $FF                     ;ADD IN A -1
1CDC    85 73           STA     ITEMP1                  ;STORE IT IN PAGE ZERO
1CDE    68              PLA                             ;GET HIGH ORDER RETURN ADDR
1CDF    69 FF           ADC#    $FF                     ;ADD IN A -1
1CE1    85 74           STA     ITEMP1+1                ;STORE IT IN PAGE ZERO
1CE3    98              TYA                             ;GET Y REGISTER
1CE4    48              PHA                             ;SAVE ON STACK
1CE5    A0 00           LDY#    0                       ;OFFSET OF ZERO
1CE7    B1 73           LDA@Y   ITEMP1                  ;GET BRK CODE
1CE9    48              PHA                             ;SAVE ON STACK
1CEA    30 04           BMI     BERROR                  ;BRANCH IF A SOFTWARE DETECTED ERROR
1CEC    68              PLA                             ;POSITIVE ERROR CODE, GET IT BACK
1CED    A9 EE           LDA#    UNRBRK                  ;UNRECOGNIZABLE BREAK
1CEF    48              PHA                             ;SAVE ON STACK
```

```
2648                            .PAGE                   ;'ERROR CODE OUTPUT RCUTINE'
2649
2650
2651
2652  1CF3  A9 FF   BERROR: LDA#    $FF               ;GET -1
2653  1CF5  85 81           STA     M1                ;FOR HIGH ORDER
2654  1CF7  68              PLA                       ;GET THE NEG ERROR CODE
2655  1CF8  85 82           STA     M1+1              ;STORE INTO LOW ORDER
2656  1CFA  20 4F22         JSR     FLT16             ;FLOAT IT
2657  1CFD  20 3D10         JSR     SETUP             ;RESET AND INITIALIZE IMPORTANT STUFF
2658  1D00  20 8B1D         JSR     CRLF              ;ADVANCE A LINE
2659  1D03  A9 3F           LDA#    '?                ;INDICATE AN ERROR
2660  1D05  20 FA18         JSR     PRINTC
2661  1D08  20 6B16         JSR     OUTLN0            ;OUTPUT IT
2662  1D0B  A5 26           LDA     PC                ;GET HIGH ORDER FOCAL STATEMENT COUNTER
2663  1D0D  10 24           BPL     BERR1             ;BRANCH IF ERROR OCCURED IN A STORED STATEMENT
2664  1D0F  C9 FE           CMP#    STRLIN            ;DID ERROR OCCUR WHILE EXECUTING A STRING?
2665  1D11  D0 32           BNE     BERRC             ;NO, THEN ERROR OCCURED IN DIRECT COMMAND
2666  1D13  A9 20           LDA#    '                 ;SPACE FOR LOCKS
2667  1D15  20 FA18         JSR     PRINTC
2668  1D18  A9 40           LDA#    '@                ;NOW AN '@'
2669  1D1A  20 FA18         JSR     PRINTC
2670  1D1D  A9 20           LDA#    '                 ;ANOTHER SPACE FOR LOCKS
2671  1D1F  20 FA18         JSR     PRINTC
2672  1D22  20 1717         JSR     PUSHTP            ;SAVE THE TEXT POINTERS
2673  1D25  A5 26           LDA     PC                ;EXECUTING A STRING WHEN ERROR OCCURED?
2674  1D27  10 0C           BPL     BERR2             ;BRANCH IF NOT, PRINT STATEMENT NUMBER
2675  1D29  A5 27           LDA     PC+1              ;YES, THEN GET THE STRING NAME
2676  1D2B  20 E913         JSR     PRTVNM            ;AND PRINT IT
2677  1D2E  A9 24           LDA#    '$                ;INDICATE IT'S A STRING
2678  1D30  20 FA18         JSR     PRINTC
2679  1D33  D0 0D           BNE     BERR3             ;AND UNCONDITIONALLY PRESS ON
2680  1D35  A9 26   BERR2:  LDA#    PC                ;GET ADDR OF WHERE PROGRAM COUNTER IS STORED
2681  1D37  85 28           STA     TXTADR            ;MAKE TEXT POINTER POINT TO IT
2682  1D39  A9 00           LDA#    0
2683  1D3B  85 29           STA     TXTADR+1          ;HIGH ORDER IS ZERO
2684  1D3D  85 2A           STA     TEXTP
2685  1D3F  20 3E16         JSR     PRNTLN            ;OUTPUT THE LINE NUMBER
2686  1D42  20 1E17 BERR3:  JSR     POPTP             ;RESTORE TEXT POINTERS
2687                                                  ;FALL INTO 'BERRC'
2688
```

```
                        .PAGE                   ;'MORE INTERRUPT HANDLERS'
                        ;
                        ;
1D45  20 8810 BERRC:    JSR     CRLF2           ;ADVANCE TWO LINES
1D48  A0 02             LDY#    0               ;POINT TO FIRST CHAR IN LINE
1D4A  A5 26             LDA     PC              ;DIRECT COMMAND OR STRING?
1D4C  30 02             BMI     OUTCMD          ;BRANCH IF YES
1D4E  A0 02             LDY#    2               ;NO, THEN POINT PAST LINE NUMBER
1D50  C4 2A   OUTCMD:   CPY     TEXTP           ;ARE WE AT FRONT OF LINE?
1D52  F0 31             BEQ     BERRET          ;BRANCH IF YES, DON'T OUTPUT SPECIAL ERROR AID
1D54  98      OUTCML:   TYA                     ;SAVE Y REG ACROSS OUTPUT CALL
1D55  48                PHA
1D56  B1 28             LDA@Y   TXTADR          ;OUTPUT CHAR FROM COMMAND LINE SO USER CAN SEE
1D58  20 FA18           JSR     PRINTC
1D5B  C9 0D             CMP#    %15             ;REACHED END OF LINE YET?
1D5D  F0 05             BEQ     EREOL           ;BRANCH IF YES
1D5F  68                PLA                     ;RESTORE Y REG
1D60  A8                TAY
1D61  C8                INY                     ;POINT TO NEXT CHAR IN THE COMMAND LINE
1D62  D0 F0             BNE     OUTCML          ;AND LOOP TILL ALL OF THE COMMAND LINE HAS BEEN OUTPUT
1D64  68      EREOL:    PLA                     ;ADJUST STACK
1D65  20 8810           JSR     OUTLF           ;FOLLOW WITH A LINE FEED
1D68  C6 2A   CHKERR:   DEC     TEXTP           ;COUNT DOWN NUMBER OF BYTES TILL ERROR
1D6A  A0 00             LDY#    0               ;ASSUME WE COUNT BACK TO ZERO
1D6C  A5 26             LDA     PC              ;DIRECT COMMAND OR STRING?
1D6E  30 02             BMI     CHKERC          ;BRANCH IF YES
1D70  A0 02             LDY#    2               ;NO, THEN WE ONLY COUNT BACK TO LINE NUMBER
1D72  C4 2A   CHKERC:   CPY     TEXTP           ;HAVE WE OUTPUT ENOUGH SPACES TO GET TO ERROR BYTE?
1D74  F0 07             BEQ     EARROW          ;BRANCH IF YES, OUTPUT UPARROW TO FLAG CHARACTER
1D76  A9 20             LDA#    ' '             ;NO, THEN ADVANCE ONE SPACE
1D78  20 FA18           JSR     PRINTC
1D7B  10 EB             BPL     CHKERR          ;AND UNCONDITIONALLY CHECK AGAIN
1D7D  A9 5E   EARROW:   LDA#    '^              ;OUTPUT UPARROW TO INDICATE WHERE ERROR IS
1D7F  20 FA18           JSR     PRINTC
1D82  20 8810           JSR     CRLF2           ;AND ADVANCE FOR LOOKS
1D85  4C 3110 BERRET:   JMP     START           ;AND RESTART
                        ;
                        ;
1D88  20 8810 CRLF2:    JSR     CRLF            ;ADVANCE TWO LINES
1D8B  A9 0D   CRLF:     LDA#    %15             ;A CARRIAGE RETURN
1D8D  20 FA18           JSR     PRINTC
1D90  A9 0A   OUTLF:    LDA#    %12             ;AND A LF
1D92  4C FA18           JMP     PRINTC          ;* PJMP * TO PRINT ROUTINE
```

```
2733                            .PAGE                   ;'INTEGER LINE NUMBER INPUT ROUTINE'
2734
2735
2736                            ;THIS ROUTINE CALLED IF THE FIRST CHAR OF A LINE NUMBER
2737                            ;IS 0-9 FOR ADDED SPEED, AS THE CALL TO 'EVAL' IS POWERFUL
2738                            ;BUT SLOW (SEE 'GETLN').
2739
2740
2741  1D95  20 B41D GETILN: JSR     GETIN           ;GET A TWO-DIGIT INTEGER
2742  1D98  85 2C           STA     GRPNO           ;SAVE AS GROUP NUMBER
2743  1D9A  20 3318         JSR     TTERMS          ;IS TERMINATOR ONE WE RECOGNIZE?
2744  1D9D  F0 31           BEQ     GETIR           ;YES, THEN RETURN
2745  1D9F  C9 2E           CMP#    '.              ;NO, IS IT A PERIOD?
2746  1DA1  D0 29           BNE     GETBAD          ;NO, THEN BAD LINE NUMBER
2747  1DA3  20 AD19         JSR     TESTNS          ;ANOTHER NUMBER?
2748  1DA6  B0 24           BCS     GETBAD          ;NO, THEN ERROR
2749  1DA8  20 B41D         JSR     GETIN           ;YES, THEN GET NEXT NUMBER
2750  1DAB  90 04           BCC     LNOK            ;BRANCH IF TWO DIGITS INPUT
2751  1DAD  AA              TAX                     ;MOVE INTO X
2752  1DAE  BD D11D         LDAX    TENS            ;YES, THEN ASSUME TRAILING ZERO
2753  1DB1  85 20   LNOK:   STA     LINENO          ;SAVE THE LINE (STEP) NUMBER
2754  1DB3  60              RTS                     ;AND RETURN
2755
2756  1DB4  29 0F   GETIN:  AND#    %17             ;MAKE 0-9
2757  1DB6  48              PHA                     ;SAVE ON STACK
2758  1DB7  20 AD19         JSR     TESTNS          ;TEST NEXT NON-BLANK
2759  1DBA  68              PLA                     ;RESTORE SAVED NUMBER
2760  1DBB  B0 13           BCS     GETIR           ;RETURN IF NOT A DIGIT
2761  1DBD  AA              TAX                     ;PLACE SAVED NUMBER INTO X (HIGH ORDER)
2762  1DBE  A5 23           LDA     CHAR            ;GET NEW DIGIT
2763  1DC0  29 0F           AND#    %17             ;FORM 0-9
2764  1DC2  7D D11D         ADCX    TENS            ;ADD IN PROPER HIGH ORDER
2765  1DC5  48              PHA                     ;SAVE NUMBER ON STACK
2766  1DC6  20 AD19         JSR     TESTNS          ;TEST NEXT NON-BLANK
2767  1DC9  68              PLA                     ;GET SAVED NUMBER BACK
2768  1DCA  B0 03           BCS     GETIRC          ;BRANCH IF NOT A NUMBER
2769  1DCC  4C 3216 GETBAD: JMP     BADLNO          ;BAD LINE NUMBER BRANCH AID
2770  1DCF  18      GETIRC: CLC                     ;INDICATE TWO DIGITS INPUT
2771  1DD0  60      GETIR:  RTS                     ;RETURN
2772
2773  1DD1  00      TENS:   .BYTE   0
2774  1DD2  0A              .BYTE   10
2775  1DD3  14              .BYTE   20
2776  1DD4  1E              .BYTE   30
2777  1DD5  28              .BYTE   40
2778  1DD6  32              .BYTE   50
2779  1DD7  3C              .BYTE   60
2780  1DD8  46              .BYTE   70
2781  1DD9  50              .BYTE   80
2782  1DDA  5A              .BYTE   90
2783
2784
2785
```

```
                                .PAGE                   ;'FOCAL FUNCTIONS'

1DEB  A3 00   FUNC:   LDY#    0                 ;SET OFFSET TO ZERO
1DED  8A              TXA                       ;PLACE HASH CODE INTO ACCUMULATOR
1DEE  BE 0E24 FUNC1:  LDXY    FUNTAB            ;GET TABLE VALUE
1DE1  F0 16           BEQ     BADFUN            ;END OF TABLE AND NOT FOUND
1DE3  D9 0E24         CMPY    FUNTAB            ;MATCH YET?
1DE6  F0 06           BEQ     GOTFUN            ;YES, WE FOUND IT
1DE8  C8              INY                       ;NO, POINT TO NEXT ENTRY
1DE9  D0 F3           BNE     FUNC1             ;AND TRY IT
1DEB  B9 1625 GOTFUN: LDAY    FUNADL            ;GET LOW ORDER ADDR OF ROUTINE TO HANDLE
1DEE  85 5F           STA     TEMP1             ;FUNCTION
1DF0  B9 FA24         LDAY    FUNADH            ;GET HIGH ORDER ADDR
1DF3  85 60           STA     TEMP1+1           ;STORE IT
1DF5  6C 5F00         JMP@    TEMP1             ;AND GO TO IT

1DF8  00      BADFUN: BRK                       ;TRAP
1DF9  E2              .BYTE   UNRFUN            ;?UNRECOGNIZABLE FUNCTION NAME

                      ;       "FABS" ABSOLUTE VALUE FUNCTION

1DFA  20 FF23 FABS:   JSR     ABSF1             ;TAKE ABSOLUTE VALUE OF FAC1
1DFD  4C 061E         JMP     FPOPJ             ;* POP * AND RETURN

                      ;       "FINT" INTERGERIZE FUNCTION

1E00  20 991F FINT:   JSR     INTFIX            ;MAKE FAC1 AN INTEGER
1E03  20 3322 FLPOPJ: JSR     FLOAT             ;FLOAT ALL BITS
1E06  20 5818 FPOPJ:  JSR     POPJ              ;AND RETURN

                      ;       "FINR" INTEGERIZE AFTER ROUNDING FUNCTION

1E09  20 951F FINR:   JSR     INTGER            ;FORM ROUNDED INTEGER
1E0C  4C 031E         JMP     FLPOPJ            ;* POP * FLOAT AND RETURN

                      ;ROUTINES TO RANGE CHECK INPUT AND OUTPUT DEVICE NUMBERS

1E0F  C9 03   CHKODV: CMP#    ODEVM             ;COMPARE AC AGAINST MAX ALLOWED
1E11  10 06           BPL     RNGDEV            ;BRANCH IF ERROR
1E13  60      CHKRTS: RTS                       ;RETURN IF OK
1E14  C9 03   CHKIDV: CMP#    IDEVM             ;COMPARE AGAINST MAX
1E16  30 FB           BMI     CHKRTS            ;RETURN IF OK
1E18  C9 FF   RNGDEV: CMP#    $FF               ;MINUS 1?
1E1A  F0 F7           BEQ     CHKRTS            ;BRANCH IF YES, ALWAYS IN RANGE
1E1C  00              BRK                       ;TRAP
1E1D  F8              .BYTE   DEVRNG            ;?DEVICE NUMBER OUT OF RANGE
```

```
                        .PAGE                   ;'MORE FOCAL FUNCTIONS'

                        ;       "FINI" INITIALIZE INPUT DEVICE

1E1E  20 851F FINI:     JSR     INTGER          ;MAKE ARGUMENT INTEGER
1E21  30 06             BMI     INIRET          ;IGNORE IF NEGATIVE
1E23  20 141E           JSR     CHKIDV          ;RANGE CHECK THE DEVICE NUMBER
1E26  20 7F1E           JSR     INI             ;GO CALL APPROPRIATE ROUTINE
1E29  4C 031E INIRET:   JMP     FLPOPJ          ;NO ERRORS - RETURN

                        ;       "FINO" INITIALIZE OUTPUT DEVICE

1E2C  20 851F FINO:     JSR     INTGER          ;MAKE ARGUMENT AN INTEGER
1E2F  30 F8             BMI     INIRET          ;IGNORE IF NEGATIVE
1E31  20 0F1E           JSR     CHKODV          ;CHECK FOR VALIDITY
1E34  20 941E           JSR     INO             ;GO CALL APPROPRIATE ROUTINE
1E37  4C 031E           JMP     FLPOPJ          ;NO ERRORS - RETURN

                        ;       "FCLI" CLOSE INPUT DEVICE

1E3A  20 851F FCLI:     JSR     INTGER          ;MAKE ARGUMENT AN INTEGER
1E3D  30 06             BMI     CLIRET          ;IGNORE IF NEGATIVE
1E3F  20 141E           JSR     CHKIDV          ;RANGE CHECK THE DEVICE NUMBER
1E42  20 A81E           JSR     CLI             ;CALL DEVICE DEPENDENT CODE
1E45  4C 031E CLIRET:   JMP     FLPOPJ          ;NO ERRORS - RETURN
```

```
                        .PAGE                   ;'MORE FOCAL FUNCTIONS'


                        ;       "FCLO" CLOSE OUTPUT DEVICE

1E48  20 851F FCLO:     JSR     INTGER          ;MAKE ARGUMENT AN INTEGER
1E4B  30 F8             BMI     CLIRET          ;IGNORE IF NEGATIVE
1E4D  20 0F1E           JSR     CHKODV          ;RANGE CHECK THE DEVICE NUMBER
1E50  20 841E           JSR     CLO             ;CALL DEVICE DEPENDENT CODE
1E53  4C 031E           JMP     FLPOPJ          ;NO ERRORS - RETURN


                        ;       "FCON" SET CONSOLE DEVICE

1E56  20 851F FCON:     JSR     INTGER          ;MAKE ARGUMENT AN INTEGER
1E59  30 11             BMI     RETCON          ;BRANCH IF NEGATIVE
1E5B  20 141E           JSR     CHKIDV          ;MAKE SURE DEVICE IS IN RANGE FOR BOTH INPUT
1E5E  20 0F1E           JSR     CHKODV          ;AND OUTPUT
1E61  85 6A             STA     CONDEV          ;MAKE IT CURRENT CONSOLE
1E63  20 1510           JSR     CLRDEV          ;MAKE CURRENT I-O DEVICE
1E66  20 2710           JSR     INIDEV          ;INITIALIZE IT FOR INPUT AND OUTPUT
1E69  4C 031E           JMP     FLPOPJ          ;* PJMP * NO ERRORS - RETURN
1E6C  A5 6A     RETCON: LDA     CONDEV          ;GET THE DEVICE NUMBER OF THE CONSOLE
1E6E  20 4322           JSR     FLT8            ;FLOAT IT
1E71  4C 061E           JMP     FPOPJ           ;* PJMP * AND RETURN


                        ;       "FCUR" CONSOLE CURSOR ADDRESSING FUNCTION.
                        ;
                        ;       NOTE:   THIS FUNCTION IS DEVICE DEPENDENT, AND IS HERE
                        ;               PRIMARILY BY POPULAR DEMAND.  THE FUNCTION HAS
                        ;               TWO ARGUMENTS.  THE FIRST IS THE ROW, THE SECOND IS
                        ;               THE COLUMN, OF THE PLACE TO POSITION ON THE CONSOLE
                        ;               DEVICE (USUALLY ASSUMED TO BE A CRT).
                        ;

1E74  20 831F FCUR:     JSR     FI2ARG          ;PICK UP TWO INTEGER ARGS
1E77  20 3219           JSR     CONCUR          ;*** CALL DEVICE DEPENDENT CODE ***
1E7A  B0 29             BCS     JOERRO          ;BRANCH IF ERROR WAS ENCOUNTERED
1E7C  4C 031E           JMP     FLPOPJ          ;* PJMP * AND RETURN
```

```
2912                            .PAGE                   ;'MORE FOCAL FUNCTIONS'
2913
2914
2915                            ;ROUTINES TO DISPATCH TO DEVICE DEPENDENT INITIALIZATION ROUTINE
2916                            ;ENTER EACH WITH THE DEVICE NUMBER IN THE ACCUMULATOR
2917                            ;THEY WILL RETURN ONLY IF NO ERRORS WERE ENCOUNTERED
2918
2919 1E7F AA       INI:        TAX                     ;USE AS OFFSET TO ADDR TABLE
2920 1E80 BD AC25              LDAX    INIAH           ;GET HIGH ORDER ADDR OF ROUTINE TO HANDLE
2921 1E83 85 60                STA     TEMP1+1         ;SAVE IT
2922 1E85 BD B125              LDAX    INIAL           ;GET LOW ORDER ADDR
2923 1E88 85 5F    INIC:       STA     TEMP1           ;SAVE IT
2924 1E8A 18                   CLC                     ;ASSUME SUCCESS
2925 1E8B 20 5E00              JSR     JSRIND          ;CALL THE PROPER ROUTINE FOR THIS DEVICE
2926 1E8E 90 03                BCC     IRTS            ;RETURN IF NO ERRORS
2927 1E90 4C E318              JMP     IERRI           ;ERROR, GO COMPLAIN
2928 1E93 60       IRTS:       RTS
2929
2930 1E94 AA       INO:        TAX                     ;USE AS OFFSET
2931 1E95 BD B625              LDAX    INOAH           ;GET HIGH ORDER ADDR OF ROUTINE TO HANDLE
2932 1E98 85 60                STA     TEMP1+1         ;SAVE IT
2933 1E9A BD BB25              LDAX    INOAL           ;GET LOW ORDER ADDR OF ROUTINE TO HANDLE
2934 1E9D 85 5F    INOC:       STA     TEMP1           ;SAVE IT
2935 1E9F 18                   CLC                     ;ASSUME SUCCESS
2936 1EA0 20 5E00              JSR     JSRIND          ;CALL PROPER ROUTINE FOR THIS DEVICE
2937 1EA3 90 EE                BCC     IRTS            ;RETURN IF NO ERRORS
2938 1EA5 4C 1619  JOERRO:     JMP     OERRO           ;COMPLAIN IF ERROR
2939
2940
2941 1EA8 AA       CLI:        TAX                     ;USE AS OFFSET TO TABLE
2942 1EA9 BD C025              LDAX    CLIAH           ;GET HIGH ORDER ADDR OF DEVICE DEPENDENT CODE
2943 1EAC 85 60                STA     TEMP1+1
2944 1EAE BD C525              LDAX    CLIAL           ;GET LOW ORDER ADDR
2945 1EB1 4C 881E              JMP     INIC            ;* PJMP * CALL DEVICE DEPENDENT CODE AND RETURN
2946
2947
2948 1EB4 AA       CLO:        TAX                     ;USE AS OFFSET TO TABLE
2949 1EB5 BD CA25              LDAX    CLOAH           ;GET HIGH ORDER ADDR OF DEVICE DEPENDENT CODE
2950 1EB8 85 60                STA     TEMP1+1
2951 1EBA BD CF25              LDAX    CLOAL           ;GET LOW ORDER
2952 1EBD 4C 901E              JMP     INOC            ;* PJMP * CALL DEVICE DEPENDENT CODE AND RETURN
```

```
2953                            .PAGE                   ;'MORE FOCAL FUNCTIONS'
2954
2955
2956                            ;       "FMEM" MEMORY EXAMINE-DEPOSIT FUNCTION
2957
2958
2959  1EC0  20 6B1F FMEM:   JSR     FI2ARG          ;PICK UP TWO INTEGER ARGS
2960  1EC3  A4 28           LDY     CHAR            ;GET THE TERMINATOR
2961  1EC5  C0 2C           CPY#    ',              ;ANOTHER ARG?
2962  1EC7  F0 12           BEQ     FMEMD           ;YES, THEN IT'S THE DEPOSIT FUNCTION
2963  1EC9  85 56           STA     ITMP1H          ;SAVE HIGH ORDER ADDR TO EXAMINE
2964  1ECB  86 55           STX     ITMP1L          ;SAVE LOW ORDER ADDR TO EXAMINE
2965  1ECD  A0 00           LDY#    0               ;FORM OFFSET OF ZERO
2966  1ECF  B1 55           LDA@Y   ITMP1L          ;GET DATA STORED IN THE LOCATION
2967  1ED1  85 82   ST16PJ: STA     M1+1            ;SAVE IN INTEGER
2968  1ED3  84 81           STY     M1              ;HIGH ORDER OF ZERO
2969  1ED5  20 4F22 FL16PJ: JSR     FLT16           ;FLOAT A 16 BIT INTEGER
2970  1ED8  4C 061E         JMP     FPOPJ           ;* PJMP * AND RETURN
2971
2972  1EDB  48      FMEMD:  PHA                     ;SAVE HIGH ORDER
2973  1EDC  8A              TXA
2974  1EDD  48              PHA                     ;AND LOW ORDER
2975  1EDE  20 7B1F         JSR     NXIARG          ;PICK UP THE NEXT INTEGER ARG
2976  1EE1  A8              TAY                     ;SAVE IN Y REG FOR A MOMENT
2977  1EE2  68              PLA                     ;GET LOW ORDER ADDR BACK
2978  1EE3  85 55           STA     ITMP1L
2979  1EE5  68              PLA                     ;GET HIGH ORDER ADDR BACK
2980  1EE6  85 56           STA     ITMP1H
2981  1EE8  98              TYA                     ;GET DATA TO DEPOSIT BACK
2982  1EE9  A0 00           LDY#    0               ;SET OFFSET OF ZERO
2983  1EEB  48              PHA                     ;SAVE DATA TO DEPOSIT
2984  1EEC  B1 55           LDA@Y   ITMP1L          ;READ THE LOCATION
2985  1EEE  85 82           STA     M1+1            ;SAVE AS INTEGER
2986  1EF0  84 81           STY     M1              ;HIGH ORDER OF ZERO
2987  1EF2  68              PLA                     ;GET DATA TO DEPOSIT BACK AGAIN
2988  1EF3  91 55           STA@Y   ITMP1L          ;STORE IN THE ADDR
2989  1EF5  A5 28           LDA     CHAR            ;GET TERMINATOR
2990  1EF7  C9 2C           CMP#    ',              ;MORE ARGS?
2991  1EF9  D0 DA           BNE     FL16PJ          ;* PBNE * NO, FLOAT AND RETURN
2992  1EFB  20 2918         JSR     PUSHJ           ;MOVE PAST COMMA,
2993  1EFE  F5 19           .WORD   EVALM1          ;EVALUATE NEXT ARG
2994  1F00  4C C01E         JMP     FMEM            ;AND GO TRY AGAIN
2995
2996
```

```
                        .PAGE                   ;'MORE FOCAL FUNCTIONS'


                        ;       "FOUT" OUTPUT ASCII EQUIVALENT

1F03  20 851F FOUT:     JSR     INTGER          ;FORM INTEGER
1F06  20 FA18           JSR     PRINTC          ;OUTPUT THE CHARACTER
1F09  4C 031E           JMP     FLPOPJ          ;* PJMP * FLOAT AND RETURN

                        ;       "FCHR" RETURN DECIMAL EQUIVALENT OF ASCII CHAR INPUT

1F0C  20 121F FCHR:     JSR     GICHR           ;GET A CHAR FROM INPUT DEVICE
1F0F  4C 051E           JMP     FL16PJ          ;* PJMP * FLOAT AND RETURN

                        ;ROUTINE TO INPUT ONE CHAR FROM INPUT DEVICE INTO FAC1

1F12  A9 00   GICHR:    LDA#    0               ;ZERO HIGH ORDER
1F14  85 81             STA     M1
1F16  A5 28             LDA     CHAR            ;SAVE CURRENT CHAR
1F18  48                PHA
1F19  20 CD18           JSR     READC           ;GET NEXT CHAR FROM INPUT DEVICE
1F1C  85 82             STA     M1+1            ;STORE IN LOW ORDER
1F1E  68                PLA                     ;RESTORE SAVED CHAR
1F1F  85 28             STA     CHAR
1F21  A5 82             LDA     M1+1            ;GET CHAR INPUT INTO ACCUMULATOR
1F23  60                RTS                     ;AND RETURN


                        ;       "FECH" SET CHAR ECHO CONTROL


1F24  20 851F FECH:     JSR     INTGER          ;FORM INTEGER
1F27  85 68             STA     ECHFLG          ;SAVE IN FLAG FOR LATER REFERENCE
1F29  4C 031E           JMP     FLPOPJ          ;* PJMP * FLOAT AND RETURN
```

```
                        .PAGE                   ;'MORE FOCAL FUNCTIONS'


                        ;       "FIDV" SET INPUT DEVICE FUNCTION

1F2C  A2 44   FIDV:     LDX#    STIADR          ;GET ADDR TO STORE STRING INFORMATION
1F2E  20 511F           JSR     GTDEVN          ;GET DEVICE NUMBER (POSSIBLY A STRING)
1F31  20 141E           JSR     CHKIDV          ;RANGE CHECK IT
1F34  A6 66             LDX     IDEV            ;SAVE PREVIOUS VALUE FOR POSSIBLE "RESTORE"
1F36  86 68             STX     IDVSAV
1F38  85 66             STA     IDEV            ;MAKE IT THE CURRENT INPUT DEVICE
1F3A  4C 481F           JMP     FIODRT          ;* PJMP * SET FAC1 TO ZERO, THEN RETURN

                        ;       "FODV" SET OUTPUT DEVICE FUNCTION

1F3D  A2 48   FODV:     LDX#    STOADR          ;GET ADDR TO STORE STRING INFORMATION
1F3F  20 511F           JSR     GTDEVN          ;GET DEVICE NUMBER (POSSIBLY A STRING)
1F42  20 0F1E           JSR     CHKODV          ;RANGE CHECK IT
1F45  A6 67             LDX     ODEV            ;SAVE PREVIOUS VALUE FOR POSSIBLE "RESTORE"
1F47  86 69             STX     ODVSAV
1F49  85 67             STA     ODEV            ;SET AS CURRENT OUTPUT DEVICE
1F4B  20 BD1C FIODRT:   JSR     ZRFAC1          ;RETURN A VALUE OF ZERO FOR THE FUNCTION
1F4E  4C 031F           JMP     FLPOPJ          ;* PJMP * FLOAT AND RETURN


                        ;ROUTINE TO GET A DEVICE NUMBER (POSSIBLY A STRING)


1F51  A5 38   GTDEVN:   LDA     STRSWT          ;WAS ARGUMENT A STRING VARIABLE?
1F53  D0 03             BNE     STRDEV          ;BRANCH IF YES
1F55  4C 851F           JMP     INTGER          ;* PJMP * NO, JUST INTEGERIZE ARG AND RETURN

1F58  A5 37   STRDEV:   LDA     VARADR          ;STORE BASE ADDR OF STRING
1F5A  95 00             STAX    $0000           ;IN POINTER
1F5C  A5 38             LDA     VARADR+1
1F5E  95 01             STAX    $0001
1F60  A5 3A             LDA     VSUB+1          ;GET SUBSCRIPT OF PLACE TO START
1F62  95 02             STAX    $0002
1F64  A5 3C             LDA     VSIZE           ;AND GET MAX SIZE OF STRING
1F66  95 03             STAX    $0003
1F68  A9 FF             LDA#    $FF             ;RETURN DEVICE NUMBER OF -1.
1F6A  60                RTS                     ;AND RETURN
```

```
                        .PAGE                   ;'MORE FOCAL FUNCTIONS'


1F6B  A5 28   FI2ARG: LDA     CHAR              ;GET TERMINATOR
1F6D  C9 2C           CMP#    ','               ;ANOTHER ARG?
1F6F  D0 12           BNE     FARGM             ;BRANCH IF ARG IS MISSING
1F71  20 851F         JSR     INTGER            ;GET A SINGLE BYTE INTEGER
1F74  48              PHA                       ;SAVE ACCROSS "EVAL" CALL
1F75  20 781F         JSR     NXIARG            ;GET ANOTHER ARG
1F78  AA              TAX                       ;SAVE SECOND ARGUMENT
1F79  68              PLA                       ;GET FIRST ARGUMENT
1F7A  60              RTS                       ;AND RETURN

1F7B  20 2918 NXIARG: JSR     PUSHJ             ;MOVE PAST COMMA, EVALUATE NEXT ARGUMENT
1F7E  F5 19           .WORD   EVALM1
1F80  4C 851F         JMP     INTGER            ;* PJMP * FORM SINGLE BYTE INTEGER AND RETURN

1F83  00      FARGM:  BRK                       ;TRAP
1F84  DF              .BYTE   ARGM              ;?ARGUMENT MISSING IN FUCNTION


                        ;ROUTINE TO GENERATE A ROUNDED INTEGER

1F85  A2 96   INTGER: LDX#    FHALF             ;MOVE CONSTANT .50
1F87  A0 78           LDY#    X2                ;INTO FAC2
1F89  20 3724         JSR     MOVXY
1F8C  20 2822         JSR     SWAP              ;PUT .50 INTO FAC1
1F8F  A5 7C           LDA     M2                ;GET SIGN OF FAC2
1F91  10 03           BPL     INTG1             ;OK IF POSITIVE
1F93  20 BA22         JSR     FCOMPL            ;MAKE -.50
1F96  20 7822 INTG1:  JSR     FADD              ;ADD IT IN AS ROUNDING.
1F99  20 1A23 INTFIX: JSR     FIX               ;AND FORM 23 BIT INTEGER
1F9C  A5 83           LDA     M1+2              ;GET LOW ORDER IF CALLER NEEDS IT
1F9E  60              RTS                       ;AND RETURN
```

```
3110                            .PAGE                   ;'MORE FOCAL FUNCTIONS'
3111
3112
3113                            ;       "FPIC" SOFTWARE PRIORITY INTERRUPT CONTROL FUNCTION
3114
3115
3116  1F9F  20 2918 FPICC:  JSR     PUSHJ             ;CALL 'EVAL' TO PICK UP NEXT AREG
3117  1FA2  F5 19           .WORD   EVALM1
3118  1FA4  A5 28   FPIC:   LDA     CHAR              ;GET CHAR WHICH TERMINATED ARGUMENT
3119  1FA6  C9 2C           CMP#    ','               ;IS THERE ANOTHER ARGUMENT TO FOLLOW?
3120  1FA8  D0 09           BNE     FARGM             ;BRANCH IF NOT, GO COMPLAIN.
3121  1FAA  20 851F         JSR     INTGER            ;YES, PICK UP VALUE OF FIRST
3122  1FAD  F0 27           BEQ     PISET             ;BRANCH IF LEVEL TO ENABLE IS LEVEL 0
3123  1FAF  48              PHA                       ;SAVE LEVEL TO ENABLE
3124  1FB0  20 8419         JSR     GETC              ;MOVE PAST COMMA
3125  1FB3  20 C915         JSR     GETLNS            ;AND PICK UP THE LINE NUMBER TO 'DO'
3126  1FB6  68              PLA                       ;GET LEVEL BACK
3127  1FB7  AA              TAX                       ;INTO X REGISTER
3128  1FB8  A5 2C           LDA     GRPNO             ;GET GROUP NUMBER OF LINE TO 'DO'
3129  1FBA  9D 7425         STAX    INTGRP            ;SAVE FOR LATER USE
3130  1FBD  A5 2D           LDA     LINENO            ;GET STEP NUMBER OF LINE TO 'DO'
3131  1FBF  9D 7D25         STAX    INTLIN            ;SAVE FOR LATER USE
3132  1FC2  A5 60           LDA     ACTMSK            ;GET MASK WHICH INDICATES WHICH CHANNELS ARE ACTIVE
3133  1FC4  1D 8F25         ORAX    BITTAB            ;SET THE BIT FOR THE SPECIFIED CHANNEL
3134  1FC7  85 60           STA     ACTMSK            ;MAKING IT NOW ACTIVE
3135  1FC9  A5 28   ENDPIC: LDA     CHAR              ;GET CHAR WHICH TERMINATED SECOND ARG
3136  1FCB  C9 2C           CMP#    ','               ;ANY MORE ARGS?
3137  1FCD  F0 D0           BEQ     FPICC             ;BRANCH IF YES, PICK THEM UP
3138  1FCF  A0 00           LDY#    0                 ;NO, THEN GET A ZERO
3139  1FD1  A5 60           LDA     ACTMSK            ;AND THE CURRENT ACTIVE MASK
3140  1FD3  4C D11E         JMP     ST16PJ            ;* PJMP * STORE, FLOAT, AND RETURN IT AS VALUE
3141                        ;
3142  1FD6  20 7B1F PISET:  JSR     NXIARG            ;GET NEXT ARG AS A NUMBER
3143  1FD9  A5 82           LDA     M1+1              ;IS IT NEGATIVE?
3144  1FDB  30 EC           BMI     ENDPIC            ;YES, THEN THIS CALL IS A NO-OP
3145  1FDD  A5 83           LDA     M1+2              ;NO, GET THE INTEGER VALUE (0-255)
3146  1FDF  85 60           STA     ACTMSK            ;AND STORE THAT AS NEW ACTIVE MASK
3147  1FE1  4C C91F         JMP     ENDPIC            ;AND CHECK FOR MORE ARGUMENTS BEFORE RETURNING
```

```
                        .PAGE                   ;'FOCAL STRING FUNCTIONS'

                        ;       "FISL" INITIALIZE STRING LENGTH

1FE4  20 2918 FISLNX: JSR     PUSHJ             ;PICK UP NEXT ARGUMENT
1FE7  F5 19           .WORD   EVALM1
1FE9  A5 64   FISL:   LDA     STRSIZ            ;SAVE DEFAULT STRING SIZE
1FEB  48              PHA
1FEC  20 851F         JSR     INTGER            ;GET FIRST ARGUMENT WHICH IS SIZE TO SET
1FEF  85 64           STA     STRSIZ
1FF1  20 2020         JSR     FGTSV             ;GET NEXT ARGUMENT WHICH IS A STRING VARIABLE
                                                ;IF NOT PREVIOUSLY DEFINED, IT WILL BE DEFINED
                                                ;WITH SUPPLIED LENGTH.
1FF4  68              PLA                       ;RESTORE DEFAULT STRING LENGTH
1FF5  85 64           STA     STRSIZ
1FF7  A5 28           LDA     CHAR
1FF9  C9 2C           CMP#    ',                ;ANY MORE ARGUMENTS?
1FFB  F0 E7           BEQ     FISLNX            ;BRANCH IF YES, PROCESS THEM
1FFD  4C 861E         JMP     FPOPJ             ;* PUMP * AC, THEN RETURN

                        ;ROUTINE TO GET A STRING VARIABLE FROM PROGRAM TEXT

2000  A5 28   FGTSV:   LDA     CHAR              ;ANY MORE ARGUMENTS IN FUNCTION CALL?
2002  C9 2C           CMP#    ',
2004  D0 15           BNE     FSTRBA            ;BRANCH IF NOT, ERROR
2006  20 8419         JSR     GETC              ;YES, MOVE PAST COMMA
2009  20 2918 FGTSV1: JSR     PUSHJ             ;CALL 'GETVAR' TO GET A VARIABLE
200C  43 18           .WORD   GETVAR
200E  A5 38           LDA     STRSWT            ;WAS IT A STRING VARIABLE?
2010  D0 02           BNE     SVOK              ;BRANCH IF IT WAS
2012  00              BRK                       ;TRAP
2013  DC              .BYTE   SVRQ              ;?STRING VARIABLE REQUIRED HERE
2014  A4 3A   SVOK:   LDY     VSUB+1            ;GET ELEMENT POSITION
2016  A5 37           LDA     VARADR            ;AND LOW ADD
2018  A6 38           LDX     VARADR+1          ;HIGH ORDER BASE ADDR OF STRING
201A  60              RTS                       ;AND RETURN

201B  00        FSTRBA: BRK                     ;TRAP
201C  DB              .BYTE   BASTRF            ;?BAD OR MISSING ARGUMENT IN STRING FUNCTION
```

```
3190                            .PAGE                   ;'FOCAL STRING FUNCTIONS'
3191
3192
3193
3194                            ;       "FSTI" INPUT A STRING FROM INPUT DEVICE
3195
3196
3197  201D  20 7F20 FSTI:   JSR     SETSIO          ;PICK UP ARGS
3198  2020  20 121F FSTINX: JSR     GICHR           ;GET A CHARACTER FROM INPUT DEVICE
3199  2023  C5 3D           CMP     TCHAR           ;IS THIS THE TERMINATOR?
3200  2025  F0 10           BEQ     SENDIO          ;YES, THEN THAT'S ALL FOLKS
3201  2027  A4 3A           LDY     VSUB+1          ;GET SUBSCRIPT TO PLACE CHAR INTO
3202  2029  C9 7F           CMP#    RUBCHR          ;IS THE CHARACTER A RUBOUT?
3203  202B  F0 15           BEQ     RUBSTI          ;BRANCH IF YES, SEE IF WE DO SOMETHING SPECIAL
3204  202D  91 37   FSTOC:  STA@Y   VARADR          ;STORE CHAR THERE
3205  202F  E6 3A           INC     VSUB+1          ;POINT TO NEXT
3206  2031  E6 55           INC     STRCNT          ;COUNT THIS CHARACTER
3207  2033  C6 59           DEC     STRMAX          ;REACHED MAX ALLOWED?
3208  2035  D0 E9           BNE     FSTINX          ;BRANCH IF NOT, INPUT MORE
3209  2037  A9 00   SENDIO: LDA#    0               ;STORE A ZERO IN HIGH ORDER
3210  2039  85 81           STA     M1
3211  203B  A5 55           LDA     STRCNT          ;GET NUMBER ACTUALLY MOVED
3212  203D  85 82           STA     M1+1
3213  203F  4C D51E         JMP     FL16PJ          ;* PJMP # FLOAT AND RETURN
3214
3215                            ;HERE IF A RUBOUT SEEN DURING A STRING INPUT
3216
3217  2042  A6 66   RUBSTI: LDX     IDEV            ;IS THE INPUT DEVICE
3218  2044  E4 6A           CPX     CONDEV          ;THE CONSOLE?
3219  2046  D0 E5           BNE     FSTOC           ;BRANCH IF NOT,  DON'T DO ANYTHING SPECIAL
3220  2048  C4 5D           CPY     STRSAV          ;YES, ARE WE TRYING TO RUBOUT PAST STARTING SUBSCRIPT?
3221  204A  F0 D4           BEQ     FSTINX          ;BRANCH IF SO, DON'T DO ANYTHING, IGNORE RUBOUT
3222  204C  A4 6B           LDY     ECHFLG          ;DOES USER WANT CHARACTER ECHOING?
3223  204E  D0 0E           BNE     RUBSC           ;BRANCH IF ECHOING DISABLED.
3224  2050  A4 6C           LDY     DELSPL          ;DO WE DO FANCY CRT STYLE RUBOUTS?
3225  2052  F0 05           BEQ     RUBS1           ;BRANCH IF NOT.
3226  2054  20 6F19         JSR     EATTVC          ;YES, THEN EAT THE CHAR OFF OF CRT SCREEN
3227  2057  10 05           BPL     RUBSC           ;AND DO COMMON THINGS
3228  2059  A9 5C   RUBS1:  LDA#    RUBECH          ;ECHO PLAIN CHAR TO INDICATE A RUBOUT
3229  205B  20 FA18         JSR     PRINTC
3230  205E  C6 3A   RUBSC:  DEC     VSUB+1          ;PACK UP ONE BYTE IN THE STRING
3231  2060  C6 55           DEC     STRCNT          ;DON'T COUNT THE CHARACTER RUBBED OUT
3232  2062  E6 59           INC     STRMAX
3233  2064  4C 2020         JMP     FSTINX          ;AND GET NEXT CHARCTER
```

```
                        .PAGE                   ;'FOCAL STRING FUNCTIONS'


                        ;       "FSTO" OUTPUT A STRING TO OUTPUT DEVICE

2067  20 7F20 FSTO:     JSR     SETSIO          ;GET ARGS
206A  A4 3A   FSTONX:   LDY     VSUB+1          ;GET SUBSCRIPT OF BYTE IN STRING
206C  B1 37             LDA@Y   VARADR          ;GET THE BYTE
206E  C5 3D             CMP     TCHAR           ;TERMINATOR?
2070  F0 C5             BEQ     SENDIO          ;BRANCH IF YES.
2072  20 FA18           JSR     PRINTC          ;OUTPUT IT
2075  E6 3A             INC     VSUB+1          ;POINT TO NEXT BYTE
2077  E6 55             INC     STRCNT          ;COUNT THIS ONE OUTPUT
2079  C6 59             DEC     STRMAX          ;OUTPUT MAX YET?
207B  D0 ED             BNE     FSTONX          ;BRANCH IF MORE TO OUTPUT
207D  F0 B8             BEQ     SENDIO          ;BRANCH IF WE HAVE HIT LIMIT
```

```
3251                            .PAGE                   ;'FOCAL STRING FUNCTIONS'
3252
3253
3254                            ;ROUTINE TO GET ARGS FOR 'FSTI' AND 'FSTO'
3255
3256 207F  A9 00    SETSIO: LDA#    0               ;GET A ZERO
3257 2081  85 55            STA     STRCNT          ;INIT BYTE COUNT TO ZERO
3258 2083  20 851F          JSR     INTGER          ;GET MAX NUMBER OF CHARACTERS TO MOVE
3259 2086  85 59            STA     STRMAX
3260 2088  20 0020          JSR     FGTSV           ;GET THE STRING VARIABLE
3261 208B  48               PHA                     ;SAVE NEAT STUFF RETURNED
3262 208C  8A               TXA
3263 208D  48               PHA
3264 208E  98               TYA
3265 208F  48               PHA
3266 2090  A5 28            LDA     CHAR            ;IS THE OPTIONAL TERMINATOR ARG SUPPLIED?
3267 2092  C9 2C            CMP#    ',
3268 2094  F0 04            BEQ     SETS1           ;YES, THEN PICK IT UP
3269 2096  A9 FF            LDA#    $FF             ;NO, THEN SET IT TO $FF
3270 2098  D0 08            BNE     SETS2           ;AND ENTER COMMON CODE
3271 209A  20 2918  SETS1:  JSR     PUSHJ           ;MOVE PAST COMMA, PICK UP NEXT ARG
3272 209D  F5 19            .WORD   EVALM1
3273 209F  20 851F          JSR     INTGER          ;FORM INTEGER
3274 20A2  85 3D    SETS2:  STA     TCHAR           ;SAVE TERMINATION CHARACTER
3275 20A4  68               PLA                     ;RESTORE GOOD STUFF
3276 20A5  85 3A            STA     VSUB+1
3277 20A7  85 50            STA     STBSAV          ;REMEMBER SUBSCRIPT TO BEGIN I-O TO/FROM
3278 20A9  68               PLA
3279 20AA  85 38            STA     VARADR+1
3280 20AC  68               PLA
3281 20AD  85 37            STA     VARADR
3282 20AF  60               RTS                     ;AND RETURN
```

```
                        .PAGE                   ;' STRING I-O ROUTINES '

                        ;ROUTINE TO WRITE TO A STRING

20B0 C4 4B      WSTRNG: CPY     STØMAX          ;BEYØND END ØF STRING?
20B2 F0 04              BEQ     WSRET           ;BRANCH IF YES, IGNØRE
20B4 91 48              STA@Y   STØADR          ;NØ, STØRE CHAR IN STRING
20B6 E6 4A              INC     STØPNT          ;PØINT TØ NEXT BYTE
20B8 68         WSRET:  PLA                     ;RESTØRE CHAR FRØM STACK
20B9 60                 RTS                     ;RETURN
2[illegible]BA  A9 0D   IOSRET: LDA#    %15             ;RETURN A CARRIAGE RETURN
2[illegible]BC  85 28           STA     CHAR            ;ALSO IN CHAR
2[illegible]BE  60      IOSRT2: RTS                     ;AND RETURN

                        ;ROUTINE TO INPUT FROM A STRING

[illegible]HF   A4 46   RSTRNG: LDY     STIPNT          ;GET POINTER TO NEXT BYTE
2[illegible]C1  C4 47           CPY     STIMAX          ;BEYOND END OF STRING?
2[illegible]C3  F[illegible] F5         BEQ     IOSRET          ;BRANCH IF YES, RETURN A CARRIAGE RETURN
2[illegible]C5  B1 44           LDA@Y   STIADR          ;NO, GET BYTE FROM STRING
2[illegible]C7  85 28           STA     CHAR            ;SAVE FOR THOSE WHO NEED IT
2[illegible]C9  E6 46           INC     STIPNT          ;AND POINT TO NEXT
2[illegible]CB  60              RTS                     ;AND RETURN
```

```
                        .PAGE                   ;' FSLK - STRING "LOOK" FUNCTION '
                        
                        
                        
2CCC  A5 38   FSLK:     LDA     STRSWT          ;WAS ARG A STRING VARIABLE?
2CCE  D0 02             BNE     FSLK1           ;YES, THEN PROCEED
2CD0  00                BRK                     ;TRAP
2CD1  0B                .BYTE   BASTRF          ;?BAD OR MISSING ARGUMENT IN STRING FUNCTION
2CD2  A5 37   FSLK1:    LDA     VARADR          ;COPY POINTERS INTO STRING1 POINTERS
2CD4  85 55             STA     STRAD1
2CD6  A5 38             LDA     VARADR+1
2CD8  85 56             STA     STRAD1+1
2CDA  A5 3A             LDA     VSUB+1
2CDC  85 57             STA     SBEG1           ;STORE BEGINNING POSITION
2CDE  20 0020           JSR     FGTSV           ;GET NEXT STRING PARAMETER
2CE1  84 58             STY     SEND1           ;STORE ENDING POSITION
2CE3  20 0020           JSR     FGTSV           ;GET STRING2 POINTERS
2CE6  85 59             STA     STRAD2
2CE8  86 5A             STX     STRAD2+1
2CEA  84 5B             STY     SBEG2
2CEC  20 0020           JSR     FGTSV           ;GET ENDING POSITION
2CEF  84 5C             STY     SEND2           ;STORE IT
2CF1  A9 FF             LDA#    $FF             ;ASSUME -1 (STRING NOT FOUND)
2CF3  85 81             STA     M1
2CF5  85 82             STA     M1+1
                        
                        ;SEARCH ROUTINE
                        
2CF7  20 3521 LKFCHR:   JSR     CMPCHR          ;FIRST CHAR MATCH?
2CFA  F0 09             BEQ     FCMAT           ;BRANCH IF YES
2CFC  C4 5C   CHKEOS:   CPY     SEND2           ;NO, REACHED END OF STRING2 ?
2CFE  F0 32             BEQ     SNOTF           ;BRANCH IF YES, STRING1 NOT FOUND IN STRING2
2100  E6 5B             INC     SBEG2           ;NO, POINT TO NEXT CHAR IN STRING2
2102  4C F720           JMP     LKFCHR          ;AND TRY TO FIND FIRST CHAR MATCH
```

```
                                .PAGE                   ;'FOCAL STRING FUNCTIONS'

                                ;HERE WHEN FIRST CHAR OF STRING1 MATCHES A CHAR IN STRING2

2105  20 3E21  FCMAT:  JSR     PUSHSP          ;SAVE CURRENT POSITION IN BOTH STRINGS
2108  A5 57    NXCMAT: LDA     SBEG1           ;REACHED END OF FIRST STRING?
210A  C5 58             CMP     SEND1
210C  F0 13             BEQ     SFOUND          ;BRANCH IF YES, THEN STRING1 WAS FOUND IN STRING2
210E  C4 5C             CPY     SEND2           ;NO, REACHED END OF STRING2?
2110  F0 1D             BEQ     SNOTFP          ;BRANCH IF YES, THEN STRING1 CAN'T BE IN STRING2
2112  E6 57             INC     SBEG1           ;POINT TO NEXT CHAR IN EACH STRING
2114  E6 5B             INC     SBEG2
2116  20 3521           JSR     CMPCHR          ;MATCH?
2119  F0 ED             BEQ     NXCMAT          ;BRANCH IF YES, KEEP CHECKING AS LONG AS THEY MATCH
211B  20 4A21           JSR     POPSP           ;NO, THEN RETURN TO POINT OF FIRST CHAR MATCH
211E  4C FC20           JMP     CHKEOS          ;AND TRY AGAIN FOR FIRST CHAR MATCH

2121  20 4A21 SFOUND: JSR     POPSP           ;RESTORE POINTERS TO POSITION OF FIRST CHAR MATCH
2124  A9 00           LDA#    0               ;STORE 0 IN HIGH ORDER
2126  85 01           STA     M1
2128  A5 5B           LDA     SBEG2           ;RETURN SUBSCRIPT WHERE FIRST CHAR MATCHED
212A  85 02           STA     M1+1
212C  4C D51E           JMP     FL16PJ          ;* PUMP * FLOAT AND RETURN
212F  20 4A21 SNOTFP: JSR     POPSP           ;POP OFF SAVED POINTERS
2132  4C D51E SNOTF:  JMP     FL16PJ          ;FLOAT -1 AND RETURN AS STRING1 WAS NOT FOUND IN STRING2

                                ;ROUTINES USED BY 'FSLK'.

2135  A4 57    CMPCHR: LDY     SBEG1           ;GET CHAR FROM STRING1
2137  B1 55             LDA@Y   STRAD1
2139  A4 5B             LDY     SBEG2           ;GET CHAR FROM STRING2
213B  D1 59             CMP@Y   STRAD2          ;COMPARE THEM
213D  60                RTS                     ;RETURN WITH Z=1 IF THEY ARE THE SAME

213E  98        PUSHSP: TYA                     ;PRESERVE 'Y' REGISTER
213F  48                PHA
2140  A2 55             LDX#    STRAD1          ;SAVE STRING POINTERS ON STACK
2142  A0 08             LDY#    8
2144  20 7818           JSR     PUSHB0
2147  68                PLA                     ;RESTORE 'Y' REGISTER
2148  A8                TAY
2149  60                RTS                     ;AND RETURN

214A  98        POPSP:  TYA                     ;PRESERVE 'Y' REGISTER
214B  48                PHA
214C  A2 5C             LDX#    STRAD1+7        ;RESTORE STRING POINTERS
214E  A0 08             LDY#    8
2150  20 6818           JSR     POPB0
2153  68                PLA                     ;RESTORE 'Y' REGISTER
2154  A8                TAY
2155  60                RTS                     ;AND RETURN
```

```
                        .PAGE                   ;'MORE FOCAL FUNCTIONS'


                        ;       "FSBR" SINGLE VALUED SUBROUTINE CALL

2156  20 EC15 FSBR:     JSR     GETLN1          ;FINISH EVALUATING GROUP OR LINE TO "DO"
2159  08                PHP                     ;SAVE STATUS FLAGS ON STACK
215A  A9 30             LDA#    $30             ;GET CODE NAME FOR VARIABLE '&Ø'
215C  85 3D             STA     VCHAR           ;SAVE AS VARIABLE NAME TO LOOK FOR
215E  A9 00             LDA#    Ø               ;ALSO SET SUBSCRIPT TO ZERO
2160  85 39             STA     VSUB
2162  85 3A             STA     VSUB+1
2164  85 3B             STA     STRSWT          ;MAKE SURE STRING VARIABLE FLAG IS OFF
2166  20 2918           JSR     PUSHJ           ;CALL 'FNDVAR' TO LOCATE '&Ø(Ø)'
2169  B6 19             .WORD   FNDVAR
216B  20 2B22           JSR     SWAP            ;PUT CURRENT VALUE OF '&Ø' INTO FAC2
216E  20 AD1C           JSR     PUSHIV          ;SAVE IT'S VALUE AND ADDR ON STACK
2171  A5 28             LDA     CHAR            ;GET TERMINATOR
2173  C9 2C             CMP#    ',              ;IS THERE ANOTHER ARGUMENT
2175  F0 02             BEQ     FSBR1           ;BRANCH IF YES, PRESS ON
2177  00                BRK                     ;NO, TRAP
2178  0F                .BYTE   ARGM            ;?ARGUMENT MISSING IN FUNCTION
2179  68          FSBR1: PLA                     ;GET FLAGS FROM 'GETLN' INTO ACCUMULATOR
217A  20 3E18           JSR     PUSHA           ;SAVE ON STACK
217D  A2 2C             LDX#    GRPNO           ;SAVE LINE OR GROUP TO "DO"
217F  20 7618           JSR     PUSHB2
2182  20 2918           JSR     PUSHJ           ;MOVE PAST COMMA, EVALUATE NEXT ARGUMENT
2185  F5 19             .WORD   EVALM1
2187  A2 2D             LDX#    LINENO          ;GET LINE OR GROUP TO "DO" BACK
2189  20 6618           JSR     POPB2
218C  20 4C18           JSR     POPA            ;GET 'GETLN' FLAGS BACK
218F  48                PHA                     ;SAVE ON STACK FOR LATER
2190  20 B51C           JSR     POPIV           ;GET VALUE OF '&Ø' AND POINTER TO IT
2193  20 AD1C           JSR     PUSHIV          ;SAVE FOR LATER (VALUE IS IN FAC2)
2196  20 931C           JSR     PUTVAR          ;NOW SET '&Ø' TO ARG VALUE (IN FAC1)
2199  A5 24             LDA     INSW            ;SAVE WHERE INPUT IS COMING FROM
219B  20 3E18           JSR     PUSHA           ;(PROGRAM OR INPUT DEVICE)
219E  A9 00             LDA#    Ø               ;AND FORCE IT TO BE PROGRAM
21A0  85 24             STA     INSW
21A2  68                PLA                     ;GET STATUS FLAGS RETURNED BY 'GETLN'
21A3  AA                TAX                     ;SAVE IN X REGISTER
21A4  20 2918           JSR     PUSHJ           ;NOW PERFORM THE "DO" OF THE LINE OR GROUP
21A7  B6 11             .WORD   DO1
21A9  20 4C18           JSR     POPA            ;RESTORE WHERE INPUT IS COMING FROM
21AC  85 24             STA     INSW
21AE  20 B51C           JSR     POPIV           ;RETURN IS TO HERE, GET POINTERS TO '&Ø'
                                                ;AND OLD VALUE IS IN FAC2.
21B1  20 AD1C           JSR     FETVAR          ;GET CURRENT VALUE IN FAC1
21B4  20 2B22           JSR     SWAP            ;OLD VALUE IN FAC1, CURRENT VALUE IN FAC2
21B7  20 931C           JSR     PUTVAR          ;REPLACE OLD VALUE OF '&Ø' BEFORE CALL
21BA  20 2B22           JSR     SWAP            ;GET CURRENT VALUE OF '&Ø' INTO FAC1
21BD  4C 761E           JMP     FPOPJ           ;RETURN IT AS THE VALUE OF THE 'FSBR'
```

```
3448                            .PAGE               ;'MORE FOCAL FUNCTIONS'
3449
3450
3451                            ;       "FRAN" RANDOM NUMBER GENERATOR,
3452                            ;               RETURNS A FRACTION BETWEEN 0.00 AND 1.00
3453                            ;
3454  21C0  20 851F FRAN:       JSR     INTGER          ;INTEGERIZE ARGUMENT
3455  21C3  F0 0F               BEQ     FRANC           ;BRANCH IF = 0, RETURN NEXT RANDOM NUMBER
3456  21C5  10 04               BPL     FRSET           ;BRANCH IF > 0, SET TO REPEATABILITY
3457  21C7  A5 76               LDA     HASH            ;GET THE RANDOM NUMBER HASH VALUE
3458  21C9  D0 02               BNE     FRNINI          ;AND RANDOMIZE
3459  21CB  A9 55   FRSET:      LDA#    $55             ;SET TO ALTERNATING ZEROES AND ONES
3460  21CD  A2 02   FRNINI:     LDX#    2               ;SETUP LOOP COUNTER
3461  21CF  95 77   FRNILP:     STAX    SEED            ;STORE IN SEED
3462  21D1  CA                  DEX                     ;POINT TO NEXT
3463  21D2  10 FB               BPL     FRNILP          ;AND LOOP TILL DONE
3464  21D4  A9 7F   FRANC:      LDA#    $7F             ;SET EXPONENT OF FAC1
3465  21D6  85 80               STA     X1
3466  21D8  18                  CLC                     ;ADD K TO SEED
3467  21D9  A5 77               LDA     SEED
3468  21DB  69 B1               ADC#    $B1
3469  21DD  85 83               STA     M1+2            ;PUT RESULT IN LOW ORDER
3470  21DF  85 77               STA     SEED            ;ALSO THIS PART IN SEED
3471  21E1  A5 78               LDA     SEED+1
3472  21E3  69 0C               ADC#    $0C
3473  21E5  85 82               STA     M1+1            ;INTO MIDDLE ORDER
3474  21E7  A5 79               LDA     SEED+2
3475  21E9  69 13               ADC#    $13
3476  21EB  29 7F               AND#    $7F             ;KILL SIGN BIT
3477  21ED  85 81               STA     M1              ;INTO HIGH ORDER
3478  21EF  A5 83               LDA     M1+2
3479  21F1  0A                  ASLA                    ;2*17
3480  21F2  18                  CLC
3481  21F3  65 83               ADC     M1+2            ;2*16
3482  21F5  18                  CLC
3483  21F6  65 81               ADC     M1              ;PLUS HIGH ORDER
3484  21F8  85 79               STA     SEED+2          ;NEW SEED
3485  21FA  18                  CLC
3486  21FB  A5 83               LDA     M1+2            ;2*8 ADDED
3487  21FD  65 82               ADC     M1+1
3488  21FF  85 78               STA     SEED+1
3489  2201  A5 82               LDA     M1+1
3490  2203  65 79               ADC     SEED+2
3491  2205  85 79               STA     SEED+2          ;SEED NOW READY FOR NEXT TIME
3492  2207  A9 00               LDA#    0               ;GET A ZERO
3493  2209  20 5722             JSR     NORM0           ;NORMALIZE THE FRACTION
3494  220C  4C 861E             JMP     FPOPJ           ;* PJMP * AND RETURN
```

```
3495                            .PAGE                   ;'FLOATING POINT PACKAGE'
3496
3497
3498                            ;
3499                            ;       *** FLOATING POINT PACKAGE ***
3500                            ;
3501                            ;
3502                            ;       1 BYTE EXPONENT, 4 BYTE MANTISSA (APPROX 9 SIG FIGS).
3503                            ;
3504                            ;
3505                            ;
3506                            ;DEFINITIONS
3507                            ;
3508
3509    0020                    .DEF    SPACE=%40
3510    007F                    .DEF    RUB=%177
3511    000D                    .DEF    CR=%15
3512    000A                    .DEF    LF=%12
```

```
                        ,PAGE                   ;CENTRAL ROUTINES
                        
                        
                        
2210    18      ADD:    CLC
2211    A2 03           LDX#    3               ;* INDEX FOR 4 BYTE ADD
2213    B5 81   ADD1:   LDAX    M1
2215    75 7C           ADCX    M2              ;ADD NEXT BYTE
2217    95 81           STAX    M1
2219    CA              DEX                     ;TO NEXT MORE SIG BYTE
221A    10 F7           BPL     ADD1            ;DO ALL THREE
221C    60              RTS
                        ;
221D    26 7A   MD1:    ASL     SIGN            ;CLEAR LSB OF SIGN
221F    20 2122         JSR     ABSWAP          ;ABS VAL M1, THEN SWAP
2222    24 81   ABSWAP: BIT     M1              ;M1 NEG?
2224    10 05           BPL     ABSWP1          ;NO JUST SWAP
2226    20 BA22         JSR     FCOMPL          ;YES, NEGATE IT
2229    E6 7A           INC     SIGN            ;COMPLEMENT SIGN
222B    38      ABSWP1: SEC                     ;FOR RETURN TO MUL/DIV
                        ;
                        ;SWAP FAC1 WITH FAC2
                        ;
222D    A2 05   SWAP:   LDX#    5               ;* FIVE BYTES TOTAL
222F    94 84   SWAP1:  STYX    EM1
2231    B5 7F           LDAX    X1M1            ;SWAP A BYTE OF FAC1 WITH
2233    B4 7A           LDYX    X2M1            ;FAC2 AND LEAVE A COPY OF
2235    94 7F           STYX    X1M1            ;M1 IN E, E+3 USED
2237    95 7A           STAX    X2M1
2239    CA              DEX                     ;NEXT BYTE
223A    D0 F3           BNE     SWAP1           ;UNTIL DONE
223C    60              RTS
```

```
                                .PAGE                   ;'MORE CENTRAL ROUTINES'


                                ;ROUTINE TO FLOAT 23 BITS OF MANTISSA

2238  A9 96   FLOAT:  LDA#   $96             ;SET EXPONENT TO 22 DECIMAL
223D  85 80             STA    X1
223F  A9 00             LDA#   0              ;ZERO INTO LOW BYTES
2241  F0 14             BEQ    NORM0          ;* BEQ * NORMALIZE IT AND RETURN

                                ;ROUTINE TO DO A FAST FLOAT OF A ONE BYTE QUANTITY


2243  85 81   FLT8:    STA    M1             ;STORE THE BYTE
2245  A9 86             LDA#   $86            ;ASSUME ALREADY SHIFTED 8 PLACES
2247  85 80             STA    X1
2249  A9 00             LDA#   0              ;GET A ZERO
224B  85 82             STA    M1+1           ;ZERO OUT SECOND BYTE OF MANTISSA
224D  F0 06             BEQ    FLOATC         ;* BEQ * CLEAR THIRD BYTE, NORMALIZE AND RETURN
                                ;
                                ;FLOAT A 16 BIT INTEGER IN M1(HIGH) AND M1+1(LOW)
                                ;TO FAC1, FAC2 UNAFFECTED
                                ;
224F  A9 8E   FLT16:   LDA#   $8E
2251  85 80             STA    X1             ;SET EXP TO 14 DEC
2253  A9 00             LDA#   0              ;CLEAR LOW BYTES
2255  85 83   FLOATC:  STA    M1+2
2257  85 84   NORM0:   STA    M1+3           ;*
                                ;* PFALL * NORMALIZE IT AND RETURN

2259  20 2624 NORM:    JSR    CHKZER         ;* IS MANTISSA ZERO?
225C  D0 0F             BNE    NORML          ;BRANCH IF NOT, THEN DO THE NORMALIZE SHIFTING
225E  A9 80             LDA#   $80            ;YES, THEN AVOID MUCH SHIFTING BY SETTING
2260  85 80             STA    X1             ;THE EXPONENT.
2262  60               RTS                   ;AND RETURN
2263  C6 80   NORM1:   DEC    X1
2265  06 84             ASL    M1+3           ;* SHIFT 4 BYTES LEFT
2267  26 83             ROL    M1+2
2269  26 82             ROL    M1+1
226B  26 81             ROL    M1
226D  A5 81   NORML:   LDA    M1             ;NORMASILED CHECK
226F  0A                ASLA                  ;UPPER TWO BITS UNEQUAL?
2270  45 81             EOR    M1
2272  10 EF             BPL    NORM1          ;NO, LOOP TILL THEY ARE
2274  60       RTSN:    RTS
                                ;
                                ;FAC2-FAC1 INTO FAC1
                                ;
2275  20 BA22 FSUB:    JSR    FCOMPL         ;WILL CLEAR CARRY UNLESS ZERO
2278  20 8822 SWPALG:  JSR    ALGNSW         ;RIGHT SHIFT M1 OR SWAP ON CARRY
                                ;
```

```
3596                            .PAGE                   ;'FLOATING POINT "ADD" AND "MULTIPLY"'
3597
3598                            ;
3599                            ;FAC1+FAC2 INTO FAC1
3600                            ;
3601  227B  A5 78   FADD:       LDA     X2
3602  227D  C5 80               CMP     X1              ;EXPONENTS EQUAL?
3603  227F  D0 F7               BNE     SWPALG          ;IF NOT SWAP OR ALIGN
3604  2281  20 0F22             JSR     ADD             ;ADD MANTISSAS
3605  2284  50 03   ADDEND:     BVC     NORM            ;IF COOL, NORMALISE
3606  2286  70 05               BVS     RTLOG           ;OV: SHIFT RIGHT-CARRY IS COOL
3607                            ;SWAP IF CARRY CLEAR, ELSE SHIFT RIGHT ARITHMETICALLY
3608  2288  90 A1   ALGNSW:     BCC     SWAP
3609  228A  A5 81   RTAR:       LDA     M1              ;SIGN INTO CARRY
3610  228C  0A                 ASLA                    ;ARITH SHIFT
3611  228D  E6 80   RTLOG:      INC     X1              ;COMPENSATE FOR SHIFT
3612  228F  F0 39               BEQ     OVFL            ;EXP OUT OF RANGE
3613  2291  A2 F8   RTLOG1:     LDX#    $F8             ;* INDEX FOR 8 BYTE RT SHIFT
3614  2293  A9 80   ROR1:       LDA#    $80
3615  2295  B0 01               BCS     ROR2
3616  2297  0A                 ASLA
3617  2298  56 89   ROR2:       LSRX    E+4             ;* FAKE RORX E*4
3618  229A  15 89               ORAX    E+4
3619  229C  95 89               STAX    E+4
3620  229E  E8                 INX                     ;NEXT BYTE
3621  229F  D0 F2               BNE     ROR1            ;UNTIL DONE
3622  22A1  60                  RTS
3623                            ;
3624                            ;FAC1*FAC2 INTO FAC1
3625                            ;
3626  22A2  20 1C22 FMUL:       JSR     MD1             ;ABS VAL OF M1,M2
3627  22A5  65 80               ADC     X1              ;ADD EXPONENTS
3628  22A7  20 0023             JSR     MD2             ;CHECK & PREP FOR MUL
3629  22AA  18                  CLC
3630  22AB  20 9122 MUL1:       JSR     RTLOG1          ;SHIFT PROD & MPYR RIGHT
3631  22AE  90 03               BCC     MUL2            ;SKIP PARTIAL PROD
3632  22B0  20 0F22             JSR     ADD             ;ADD IN MCAND
3633  22B3  88      MUL2:       DEY                     ;NEXT ITERATION
3634  22B4  10 F5               BPL     MUL1            ;LOOP UNTIL DONE
3635  22B6  46 7A   MDEND:      LSR     SIGN            ;SIGN EVEN OR ODD?
3636  22B8  90 9F   NORMX:      BCC     NORM            ;IF EVEN NORM, ELSE COMP
3637  22BA  38      FCOMPL:     SEC
3638  22BB  A2 04               LDX#    4               ;* 4 BYTE SUBTRACT
3639  22BD  A9 00   COMPL1:     LDA#    0
3640  22BF  F5 80               SBCX    X1
3641  22C1  95 80               STAX    X1
3642  22C3  CA                  DEX                     ;TO MORE SIG BYTE
3643  22C4  D0 F7               BNE     COMPL1          ;UNTIL DONE
3644  22C6  F0 BC               BEQ     ADDEND          ;FIX UP
3645                            ;
3646  22C8  10 46   OVCHK:      BPL     MD3             ;IF POS EXP NO OVF
3647  22CA  00      OVFL:       BRK                     ;TRAP
3648  22CB  F8                  .BYTE   FOVFL           ;FLOATING POINT OVERFLOW
```

```
                                .PAGE                   ;'FLOATING POINT "DIVIDE"'
                                ;
                                ;DIVIDE FAC2 BY FAC1 INTO FAC1
                                ;
22CC  20 1C22 FDIV:  JSR    MD1             ;ABS VAL  M1,M2
22CF  E5 80          SBC    X1              ;SUBTACT EXPONENTS
22D1  20 0023        JSR    MD2             ;SAVE AS RES EXP
22D4  38       DIV1: SEC
22D5  A2 03          LDX#   3               ;* FOR 4 BYTES
22D7  B5 7C    DIV2: LDAX   M2
22D9  F5 85          SBCX   E               ;SUB BYTE OF E FROM M2
22DB  48             PHA
22DC  CA             DEX                    ;NEXT MORE SIG BYTE
22DD  10 F8          BPL    DIV2            ;UNTIL DONE
22DF  A2 FC          LDX#   $FC             ;* FOR 4 BYTE CONC MOVE
22E1  68       DIV3: PLA                    ;DIFF WAS ON STACK
22E2  90 02          BCC    DIV4            ;IF M2<E DONT RESTORE
22E4  95 80          STAX   M2+4            ;*
22E6  E8       DIV4: INX                    ;NEXT LESS SIG BYTE
22E7  D0 F8          BNE    DIV3            ;UNTIL DONE
22E9  26 84          ROL    M1+3            ;*
22EB  26 83          ROL    M1+2
22ED  26 82          ROL    M1+1            ;ROLL QUOTIENT LEFT
22EF  26 81          ROL    M1              ;CARRY INTO LSB
22F1  26 7F          ASL    M2+3            ;*
22F3  26 7E          ROL    M2+2
22F5  26 7D          ROL    M2+1            ;DIVIDEND LEFT
22F7  26 7C          ROL    M2
22F9  B0 CF          BCS    OVFL            ;OVF IS DUE TO UNNORM DIVISOR
22FB  88             DEY                    ;NEXT ITERATION
22FC  D0 D6          BNE    DIV1            ;UNTILL DONE(23 ITERATIONS)
22FE  F0 B6          BEQ    MDEND           ;NORM QUOT &FIX SIGN
2300  86 84    MD2:  STX    M1+3            ;*
2302  86 83          STX    M1+2
2304  86 82          STX    M1+1            ;CLEAR M1
2306  86 81          STX    M1
2308  B0 BE          BCS    OVCHK           ;CHECK FOR OVFL
230A  30 04          BMI    MD3             ;IF NEG NO UNDERFLOW
230C  68             PLA                    ;POP ONE RETURN
230D  68             PLA
230E  90 A8          BCC    NORMX           ;CLEAR X1 AND RETURN
2310  49 80    MD3:  EOR#   $80             ;COMPL SIGN OF EXP
2312  85 80          STA    X1
2314  A0 1F          LDY#   31              ;COUNT FOR 31 (/), 32 (*) ITERATIONS
2316  60             RTS
                                ;
                                ;FAC1 TO 23 BIT SIGNED INTEGER IN M1 (HIGH), M1+1 (MIDDLE), M1+2 (LOW)
                                ;
2317  20 8A22 FIX1:  JSR    RTAR            ;SHIFT MANT,INC EXP
231A  A5 80    FIX:  LDA    X1              ;CHECK EXP
231C  C9 96          CMP#   $96             ;IS EXP 22 (DEC)?
231E  D0 F7          BNE    FIX1            ;NO, SHIFT MORE
2320  60      RTRN: RTS                    ;DONE
```

```
3702                            .PAGE                   ;'FLOATING POINT OUTPUT ROUTINE'
3703
3704  2321  A5 81    FPRNT:  LDA     M1              ;SAVE THE SIGN OF THE NUMBER
3705  2323  85 8A            STA     SIGNP           ;FOR LATER REFERENCE
3706  2325  20 FF23          JSR     ABSF1           ;DEAL ONLY WITH ABS VALUE
3707  2328  20 2624          JSR     CHKZER          ;IS NUMBER = 0 ?
3708  232B  D0 05            BNE     FPR0            ;BRANCH IF NOT, THEN TRY TO DIVIDE DOWN
3709  232D  85 8D            STA     K               ;YES, SOME FLAVOR OF ZERO.  INDICATE THAT WE
3710  232F  48                PHA                     ;WE DID NOT HAVE TO DIVIDE AS ALREADY < 1 ?
3711  2330  F0 49            BEQ     FPR4A           ;AND PUNT DIVIDE DOWN AND ROUNDING CODE
3712  2332  A5 80    FPR0:   LDA     X1              ;GET THE EXPONENT
3713  2334  48                PHA                     ;SAVE FOR LATER REFERENCE
3714  2335  A9 00            LDA#    0               ;ZERO COUNTER WHICH COUNTS HOW MANY TIMES
3715  2337  85 8D            STA     K               ;WE HAD TO DIVIDE TO GET NUMBER < 1.
3716  2339  24 80    FPR1:   BIT     X1              ;IS NUMBER < 1 ?
3717  233B  10 07            BPL     FPR2            ;BRANCH IF YES
3718  233D  20 1924          JSR     DIV10           ;NO, THEN DIVIDE BY 10
3719  2340  E6 8D            INC     K               ;COUNT THE FACT WE DID
3720  2342  10 F5            BPL     FPR1            ;AND CHECK AGAIN
3721  2344  20 8618 FPR2:   JSR     PHFAC1          ;SAVE NUMBER (NOW < 1) ON STACK
3722  2347  A2 96            LDX#    FHALF           ;GET THE CONSTANT .5
3723  2349  A0 80            LDY#    X1              ;INTO FAC1
3724  234B  20 0724          JSR     MOVXY
3725  234E  18                CLC
3726  234F  A5 8D            LDA     K               ;ROUNDING FACTOR IS .5 * 10 ↑ -(K+N)
3727  2351  65 90            ADC     N
3728  2353  85 8E            STA     L
3729  2355  F0 0F            BEQ     FPR4            ;BRANCH IF WE NEED .5 * 10 ↑ 0
3730  2357  A9 09            LDA#    9               ;* IS FACTOR BEYOND OUR PRECISION?
3731  2359  C5 8E            CMP     L
3732  235B  10 02            BPL     FPR3            ;BRANCH IF NOT, THEN ROUNDING FACTOR IS OK
3733  235D  85 8E            STA     L               ;YES, THEN APPLY ROUNDING TO LEAST SIG FIG
3734  235F  20 1924 FPR3:   JSR     DIV10           ;NOW SHIFT .5 INTO PROPER POSITION
3735  2362  C6 8E            DEC     L
3736  2364  D0 F9            BNE     FPR3
3737  2366  20 A718 FPR4:   JSR     PLFAC2          ;GET NUMBER BACK INTO FAC2
3738  2369  20 7822          JSR     FADD            ;ADD IN THE ROUNDING FACTOR
3739  236C  24 80            BIT     X1              ;IS NUMBER STILL < 1 ?
3740  236E  10 08            BPL     FPR4A           ;BRANCH IF IT IS
3741  2370  68                PLA                     ;NO, THEN GET ORIGINAL EXPONENT
3742  2371  30 02            BMI     FPR41           ;BRANCH IF ORIGINAL NUMBER > 1.  DO NOTHING
3743  2373  A5 80            LDA     X1              ;WE GAINED A SIG FIG IN ROUNDING, GET NEW EXP
3744  2375  48        FPR41:  PHA                     ;SAVE EXPONENT FOR LATER
3745  2376  20 1924          JSR     DIV10           ;SCALE NUMBER BACK DOWN
3746  2379  E6 8D            INC     K               ;AND INDICATE WE HAD TO
3747  237B  38        FPR4A:  SEC
3748  237C  A5 8F            LDA     M               ;NOW CALCULATE NUMBER OF LEADING BLANKS NEEDED
3749  237E  E5 8D            SBC     K
3750  2380  85 8E            STA     L               ;INTO L
```

```
3751                            .PAGE                   ;' MORE OF FLOATING POINT OUTPUT ROUTINE '
3752
3753
3754  2382  68                  PLA                     ;GET EXPONENT OF ORIGINAL NUMBER BACK
3755  2383  48                  PHA                     ;SAVE AGAIN FOR LATER
3756  2384  30 02               BMI     FPR4B           ;BRANCH IF ORIGINAL NUMBER NOT < 1 ?
3757  2386  C6 8E               DEC     L               ;IT WAS < 1.  LEAVE ROOM FOR LEADING 0
3758  2388  24 8A     FPR4B:    BIT     SIGNP           ;WAS NUMBER NEGATIVE
3759  238A  10 02               BPL     FPR5            ;BRANCH IF NOT
3760  238C  C6 8E               DEC     L               ;YES, THEN LEAVE ROOM FOR MINUS SIGN
3761  238E  A5 8E     FPR5:     LDA     L               ;ANY BLANKS TO OUTPUT?
3762  2390  30 09               BMI     FPR7            ;BRANCH IF NOT
3763  2392  F0 07               BEQ     FPR7            ;BRANCH IF NOT
3764  2394  20 F818   FPR6:     JSR     PSPACE          ;OUTPUT A BLANK
3765  2397  C6 8E               DEC     L               ;COUNT IT
3766  2399  D0 F9               BNE     FPR6            ;AND LOOP TILL ALL HAVE BEEN OUTPUT
3767  239B  24 8A     FPR7:     BIT     SIGNP           ;WAS NUMBER NEGATIVE?
3768  239D  10 05               BPL     FPR7A           ;BRANCH IF NOT
3769  239F  A9 2D               LDA#    '-              ;YES, OUTPUT LEADING MINUS
3770  23A1  20 FA18             JSR     PRINTC          ;AND FALL INTO NEXT PAGE
3771  23A4  68        FPR7A:    PLA                     ;GET EXPONENT OF ORIGINAL NUMBER BACK AGAIN
3772  23A5  30 03               BMI     FPR8            ;BRANCH IF NOT < 1,
3773  23A7  20 D023             JSR     PZERO           ;YES, THEN GIVE A LEADING ZERO (PEOPLE LIKE IT!)
3774                                                    ;(IT'S ALSO A PAIN TO CHECK FOR!)
```

```
                                ,PAGE                   ;'UTILITIES FOR FLOATING POINT I-O'
                                ;
                                ;NOW FOR THE MEAT OF IT
                                ;

23AA  A9 09   FPR8:   LDA#    9                 ;* GET MAX NUMBER OF SIG FIGS
23AC  85 8E           STA     L                 ;INTO L
23AE  A5 8D   FPR9:   LDA     K                 ;ANY TO OUTPUT BEFORE DECIMAL?
23B0  F0 07           BEQ     FPR11             ;BRANCH IF NO MORE
23B2  20 CC23 FPR10:  JSR     MD0               ;OUTPUT A DIGIT BEFORE DECIMAL
23B5  C6 8D           DEC     K
23B7  D0 F9           BNE     FPR10             ;AND LOOP TILL ALL DONE
23B9  A5 90   FPR11:  LDA     N                 ;GET NUMBER AFTER DECIMAL POINT
23BB  85 8D           STA     K                 ;INTO K
23BD  F0 0C           BEQ     FPRET             ;BRANCH IF NONE TO OUTPUT
23BF  A9 2E           LDA#    '.                ;THERE ARE SOME TO OUTPUT, PRINT THE
23C1  20 FA18         JSR     PRINTC            ;DECIMAL POINT
23C4  20 CC23 FPR12:  JSR     MD0               ;OUTPUT A DIGIT AFTER DECIMAL
23C7  C6 8D           DEC     K                 ;AND LOOP
23C9  D0 F9           BNE     FPR12             ;TILL ALL OUTPUT
23CB  60      FPRET:  RTS                       ;RETURN FROM 'FPRNT' FAC1 IS DESTORYED!
                                ;
                                ;MPY BY 10, PRINT INTEGER AND SUBTRACT IT
                                ;
23CC  C6 8E   MD0:    DEC     L                 ;HAVE WE OUTPUT ALL DIGITS OF SIGNIFICANCE
23CE  10 05           BPL     MD01              ;BRANCH IF NOT, OUTPUT THIS ONE
23D0  A9 30   PZERO:  LDA#    '0                ;YES, THEN OUTPUT A ZERO
23D2  4C FA18         JMP     PRINTC            ;* PJMP * AND RETURN
23D5  A2 91   MD01:   LDX#    FTEN
23D7  A0 78           LDY#    X2
23D9  20 0724         JSR     MOVXY
23DC  20 A222         JSR     FMUL
23DF  A2 80   FDOONE: LDX#    X1                ;SAVE FAC1
23E1  A0 A0           LDY#    T
23E3  20 0724         JSR     MOVXY
23E6  20 1A23         JSR     FIX
23E9  A5 83           LDA     M1+2              ;MAKE ASCII
23EB  29 0F           AND#    $F
23ED  09 30           ORA#    '0
23EF  20 FA18         JSR     PRINTC
23F2  20 3822         JSR     FLOAT             ;NOW SUBTRACT IT
23F5  A2 A0           LDX#    T                 ;RESTORE TO FAC2
23F7  A0 78           LDY#    X2
23F9  20 0724         JSR     MOVXY
23FC  4C 7522         JMP     FSUB              ;PJUMP
                                ;
```

```
                                .PAGE                   ;'MORE FLOATING POINT UTILITIES'
                                ;
                                ;UTILITIES FOR FPRNT
                                ;
23FF  24 81   ABSF1:  BIT     M1
2401  10 03          BPL     ABSFE
2403  20 BA22         JSR     FCOMPL
2406  60        ABSFE: RTS
                      ;
2407  CA        MOVXY: DEX
2408  8E 1224          STX     MOV1+1
240B  88                DEY
240C  8C 1424          STY     MOV2+1
240F  A2 05            LDX#    5               ;*
2411  B5 00    MOV1:   LDAX    0
2413  95 00    MOV2:   STAX    0
2415  CA                DEX
2416  D0 F9            BNE     MOV1
2418  60                RTS
                      ;
2419  20 2822 DIV10:  JSR     SWAP
241C  A2 91            LDX#    FTEN
241E  A0 80            LDY#    X1
2420  20 0724          JSR     MOVXY
2423  4C CC22          JMP     FDIV            ;PJUMP
                      ;
2426  A5 81   CHKZER: LDA     M1              ;GET HIGH ORDER MANTISSA
2428  05 82          ORA     M1+1            ;'OR' ALL BYTES OF MANTISSA TOGETHER
242A  05 83            ORA     M1+2
242C  05 84            ORA     M1+3            ;*
242E  60                RTS                     ;RETURN WITH Z=1 IF MANTISSA = 0.
                      ;
                      ;
```

```
                                .PAGE                   ;'FLOATING POINT INPUT ROUTINE'

242F  A9 00   FINP:   LDA#    0
2431  85 8A            STA     SIGNP           ;SET SIGN +
2433  85 8B            STA     DPFLG           ;RESET DP FLAG
2435  85 8C            STA     GOTFLG          ;NO INPUT YET
2437  85 8D            STA     K               ;NO DIGITS AFTER DECIMAL POINT
2439  85 7B            STA     X2              ;ZERO RESULT
243B  85 7C            STA     M2
243D  85 7D            STA     M2+1
243F  85 7E            STA     M2+2
2441  85 7F            STA     M2+3            ;*
2443  A5 2B            LDA     CHAR            ;GET CHARACTER
2445  C9 2B            CMP#    '+              ;IGNORE *'S
2447  F0 06            BEQ     FINP3
2449  C9 2D            CMP#    '-              ;FLAG IF NEG
244B  D0 05            BNE     FINP2
244D  E6 8A            INC     SIGNP
244F  20 8419 FINP3:  JSR     GETC            ;ANOTHER CHAR
2452  C9 30   FINP2:   CMP#    '0              ;IS IT DIGIT?
2454  90 24            BCC     FINP4           ;NO
2456  C9 3A            CMP#    ':              ;MAYBE...
2458  B0 20            BCS     FINP4           ;NO
245A  A2 91            LDX#    FTEN
245C  A0 80            LDY#    X1
245E  20 0724          JSR     MOVXY           ;FAC2=10.0=FAC1
2461  20 A222          JSR     FMUL
2464  20 2B22          JSR     SWAP            ; INTO FAC2
2467  E6 8C            INC     GOTFLG          ;YES, WE HAVE INPUT
2469  A5 2B            LDA     CHAR
246B  29 0F            AND#    $F              ;MAKE NUMERIC
246D  20 4322          JSR     FLT8            ;AND FLOAT IT
2470  20 7B22          JSR     FADD            ;ADD TO PARTIAL RESULT
2473  20 2B22          JSR     SWAP            ;BACK INTO FAC2
2476  E6 8D            INC     K               ;COUNT DIGITS AFTER DP
2478  D0 D5            BNE     FINP3           ;GET MORE
247A  C9 2E   FINP4:   CMP#    '.              ;DP?
247C  D0 0C            BNE     FINP5           ;NO, END OF #
247E  A5 8B            LDA     DPFLG           ;YES, ALREADY GOT ONE?
2480  D0 08            BNE     FINP5           ;THEN END OF #
2482  E6 8B            INC     DPFLG           ;ELSE FLAG GOT ONE
2484  A9 00            LDA#    0
2486  85 8D            STA     K               ;RESET K
2488  F0 C5            BEQ     FINP3           ;AND GET FRACTION
```

```
                                ,PAGE                   ;'MORE FLOATING POINT INPUT'


                                ;HERE ON END OF NUMBER
24BA  20 2B22 FINP5:  JSR     SWAP              ;RESULT TO FAC1
24BD  A5 8B           LDA     DPFLG             ;ANY DP?
24BF  F0 0B           BEQ     FINP6             ;NO, ITS OK
2491  A5 8D    FINP7:  LDA     K                 ;ELSE ADJUST
2493  F0 07           BEQ     FINP6             ;ADJUST DONE
2495  20 1924          JSR     DIV10             ;RESULT/10.0
2498  C6 8D            DEC     K                 ;K TIMES
249A  D0 F5           BNE     FINP7
249C  A5 8A    FINP6:  LDA     SIGNP             ;NOW ADD SIGN
249E  F0 03           BEQ     FINP8             ;WAS POS
24A0  20 BA22         JSR     FCOMPL            ;WAS NEG
24A3  4C 5922 FINP8:  JMP     NORM              ;PJUMP TO NORMALISE


24A6  A0 00   CONINI: LDY#    0                 ;CLEAR Y REGISTER
24A8  A9 D2           LDA#    <INTSRV           ;GET LOW BYTE OF SERVICE ROUTINE
24AA  91 A6           STA@Y   IVCTOR            ;STORE IN VECTOR
24AC  C8              INY                       ;ADVANCE Y REGISTER
24AD  A9 1C           LDA#    >INTSRV           ;GET HIGH BYTE
24AF  91 A6           STA@Y   IVCTOR            ;PUT IN VECTOR
24B1  18              CLC                       ;INDICATE SUCCESS
24B2  60              RTS

24B3  C9 0D   TVOUT:  CMP#    CR                ;IS IT A CR
24B5  D0 05           BNE     OUT1              ;BRANCH IF NOT
24B7  20 A900         JSR     OUT               ;PRINT THE CR
24BA  A5 A5   FILOUT: LDA     FILLCH            ;GET FILL CHARACTER
24BC  20 A900 OUT1:   JSR     OUT               ;PRINT IT
24BF  18              CLC                       ;INDICATE SUCCESS
24C0  60              RTS

24C1  20 AE00 KEYIN:  JSR     IN                ;GET CHARACTER
24C4  C9 0D           CMP#    CR                ;IS IT A CR?
24C6  D0 05           BNE     FILRON            ;BRANCH IF NOT
24C8  48              PHA                       ;SAVE CHARACTER
24C9  20 BA24         JSR     FILOUT            ;SEND FILL CHARACTER
24CC  68              PLA                       ;RESTORE CHARACTER
24CD  29 7F   FILRON: AND#    $7F               ;CLEAR PARITY
24CF  18              CLC                       ;INDICATE SUCCESS
24D0  60              RTS
```

```
3947                            .PAGE
3948
3949    0200                    .DEF    C0BYTI=0        ;CASSETTE I/O IS LEFT TO USER
3950    0200                    .DEF    C1BYTI=0        ;DEPENDS ON OUT DEVICES AVAILABLE
3951    0200                    .DEF    C0BYTO=0
3952    0200                    .DEF    C1BYTO=0
3953    1932                    .DEF    CONCUR=RTS1     ;NO CURSOR CONTROL ON TIM I/O
```



```
3954                            .PAGE                   ;'FOCEND - TEXT AREAS AND THE LIKE'
3955                    ;
3956                    ;
3957                    ;       ANYTHING THAT FOLLOWS THE INTERPRETER SHOULD BE LOCATED HERE,
3958                    ;   SUCH AS DEFINITIONS OF TEXT AREAS, ETC.
3959                    ;
3960                    ;
3961                    ;
3962                    ;
```

```
                        .PAGE                   ;'FOCEND - TEXT AREAS AND THE LIKE'
                ;
                ;
                ;
                ;
                ;       SPECIAL TERMINATOR CHAR TABLE
                ;
                ;
2401    20      TRMTAB: .ASCII  ' '             ;LEVEL  0       (SPACE)

2402    2B              .ASCII  '+'             ;LEVEL  1

2403    2D              .ASCII  '-'             ;LEVEL  2

2404    2F              .ASCII  '/'             ;LEVEL  3

2405    2A              .ASCII  '*'             ;LEVEL  4

2406    5E              .ASCII  '^'             ;LEVEL  5

2407    28              .ASCII  '('             ;LEVEL  6

2408    29              .ASCII  ')'             ;LEVEL  7       (START OF DELIMITERS)

2409    2C              .ASCII  ','             ;LEVEL  8

240A    3B              .ASCII  ';'             ;LEVEL  9

240B    0D              .BYTE   %15             ;LEVEL  10      (CARRIAGE RETURN)

240C    3D              .ASCII  '='             ;LEVEL  11      (TERMINATOR FOR 'SET')

240D    5F              .BYTE   LINCHR          ;LEVEL 12       ('LINE-DELETE' IS HERE SO
                                                ;               'ASK' CAN ALLOW RE-TYPEIN)
                ;
        000C            .DEF    TRMAX=12        ;MAX TABLE OFFSET FOR ABOVE TABLE
                ;
                ;
```

```
4001                            .PAGE                   ;'FOCEND - FUNCTION DISPATCH TABLES
4002                    ;
4003                    ;
4004                    ;
4005                    ;       THESE FUNCTION DISPATCH TABLES MAY BE PATCHED BY A USER
4006                    ;       TO CALL HIS OWN FUNCTIONS.
4007                    ;
4008                    ;
4009                    ;
4010                    ;       TABLE OF HASH CODES FOR FUNCTION NAMES
4011                    ;
4012                    ;
4013                    ;
4014  240E  36          FUNTAB: .BYTE   HFABS           ;ABSOLUTE VALUE FUNCTION
4015  240F  F4                  .BYTE   HFOUT           ;CHARACTER OUTPUT FUNCTION
4016  24E0  B0                  .BYTE   HFRAN           ;RANDOM NUMBER FUNCTION
4017  24E1  A8                  .BYTE   HFINT           ;INTEGERIZE FUNCTION
4018  24E2  A4                  .BYTE   HFINR           ;INTEGERIZE WITH ROUNDING FUCNTION
4019  24E3  84                  .BYTE   HFIDV           ;INPUT DEVICE FUNCTION
4020  24E4  84                  .BYTE   HFODV           ;OUTPUT DEVICE FUNCTION
4021  24E5  5C                  .BYTE   HFCHR           ;CHARACTER INPUT FUNCTION
4022  24E6  90                  .BYTE   HFCUR           ;CONSOLE CURSOR ADDRESSING FUNCTION
4023  24E7  44                  .BYTE   HFECH           ;ECHO CONTROL FUNCTION
4024  24E8  AA                  .BYTE   HFPIC           ;SOFTWARE PRIORITY INTERRUPT CONTROL FUNCTION
4025  24E9  96                  .BYTE   HFMEM           ;MEMORY EXAMINE-DEPOSIT FUNCTION
4026  24EA  92                  .BYTE   HFINI           ;INITIALIZE INPUT DEVICE FUNCTION
4027  24EB  9E                  .BYTE   HFINO           ;INITIALIZE OUTPUT DEVICE FUNCTION
4028  24EC  5A                  .BYTE   HFCLI           ;CLOSE INPUT DEVICE FUNCTION
4029  24ED  66                  .BYTE   HFCLO           ;CLOSE OUTPUT DEVICE FUNCTION
4030  24EE  70                  .BYTE   HFCON           ;SET CONSOLE DEVICE FUNCTION
4031  24EF  C4                  .BYTE   HFSBR           ;'SUBROUTINE' CALL FUNCTION
4032  24F0  AC                  .BYTE   HFISL           ;INTIALIZE STRING LENGTH
4033  24F1  FA                  .BYTE   HFSTI           ;STRING INPUT FUNCTION
4034  24F2  06                  .BYTE   HFSTO           ;STRING OUTPUT FUNCTION
4035  24F3  DE                  .BYTE   HFSLK           ;STRING "LOOK" FUNCTION
4036  24F4  00                  .BYTE   0               ;SPARE LOCS FOR HACKERS
4037  24F5  00                  .BYTE   0
4038  24F6  00                  .BYTE   0
4039  24F7  00                  .BYTE   0
4040  24F8  00                  .BYTE   0
4041  24F9  00                  .BYTE   0               ;MUST HAVE AT LEAST ONE ZERO TO END TABLE!
```

```
                        .PAGE                   ;'FOCEND - FUNCTION DISPATCH TABLES
                        ;
                        ;
                        ;
24FA 1D     FUNADH:     .HBYTE  FABS
24FB 1F                 .HBYTE  FOUT
24FC 21                 .HBYTE  FRAN
24FD 1E                 .HBYTE  FINT
24FE 1E                 .HBYTE  FINR
24FF 1F                 .HBYTE  FIDV
2500 1F                 .HBYTE  FODV
2501 1F                 .HBYTE  FCHR
2502 1E                 .HBYTE  FCUR
2503 1F                 .HBYTE  FECH
2504 1F                 .HBYTE  FPIC
2505 1E                 .HBYTE  FMEM
2506 1E                 .HBYTE  FINI
2507 1E                 .HBYTE  FINO
2508 1E                 .HBYTE  FCLI
2509 1E                 .HBYTE  FCLO
250A 1E                 .HBYTE  FCON
250B 21                 .HBYTE  FSBR
250C 1F                 .HBYTE  FISL
250D 20                 .HBYTE  FSTI
250E 20                 .HBYTE  FSTO
250F 20                 .HBYTE  FSLK
2510 00                 .BYTE   0               ;SPACE FOR HACKERS
2511 00                 .BYTE   0
2512 00                 .BYTE   0
2513 00                 .BYTE   0
2514 00                 .BYTE   0
2515 00                 .BYTE   0               ;MUST HAVE AT LEAST ONE ZERO TO MARK END OF TABLE!
```

```
                        .PAGE                   ;'FOCEND - FUNCTION DISPATCH TABLES'

                        ;
                        ;       LOW ORDER ADDR OF FUNCTION ROUTINES
                        ;
2516  FA        FUNADL: .BYTE   FABS
2517  03                .BYTE   FOUT
2518  C0                .BYTE   FRAN
2519  00                .BYTE   FINT
251A  09                .BYTE   FINR
251B  2C                .BYTE   FIDV
251C  3D                .BYTE   FODV
251D  0C                .BYTE   FCHR
251E  74                .BYTE   FCUR
251F  24                .BYTE   FECH
2520  A4                .BYTE   FPIC
2521  C2                .BYTE   FMEM
2522  1E                .BYTE   FINI
2523  2C                .BYTE   FINO
2524  3A                .BYTE   FCLI
2525  48                .BYTE   FCLO
2526  56                .BYTE   FCON
2527  56                .BYTE   FSBR
2528  E9                .BYTE   FISL
2529  1D                .BYTE   FSTI
252A  67                .BYTE   FSTO
252B  CC                .BYTE   FSLK
252C  00                .BYTE   0               ;SPACE FOR HACKERS
252D  00                .BYTE   0
252E  00                .BYTE   0
252F  00                .BYTE   0
2530  00                .BYTE   0
2531  00                .BYTE   0               ;MUST HAVE AT LEAST ONE ZERO AT END OF TABLE!
```

```
                        .PAGE                   ;'FOCEND - COMMAND DISPATCH TABLES'
                        ;
                        ;
                        ;       THESE COMMAND DISPATCH TABLES MAY BE PATCHED BY USER TO
                        ;       ADD HIS OWN SPECIAL COMMAND HANDLERS.
                        ;
                        ;
                        ;       COMMAND CHARACTER TABLE
                        ;
                        ;
2532    53      COMTAB: .ASCII  'S'             ;SET COMMAND
2533    49              .ASCII  'I'             ;IF COMMAND
2534    44              .ASCII  'D'             ;DO COMMAND
2535    4F              .ASCII  'O'             ;ON COMMAND
2536    47              .ASCII  'G'             ;GOTO COMMAND
2537    46              .ASCII  'F'             ;FOR COMMAND
2538    52              .ASCII  'R'             ;RETURN COMMAND
2539    54              .ASCII  'T'             ;TYPE COMMAND
253A    41              .ASCII  'A'             ;ASK COMMAND
253B    43              .ASCII  'C'             ;COMMENT COMMAND
253C    45              .ASCII  'E'             ;ERASE COMMAND
253D    57              .ASCII  'W'             ;WRITE COMMAND
253E    4D              .ASCII  'M'             ;MODIFY COMMAND
253F    51              .ASCII  'Q'             ;QUIT COMMAND
2540    00              .BYTE   0 H             ;SPACE FOR HACKERS
2541    00              .BYTE   0 L
2542    00              .BYTE   0 B
2543    00              .BYTE   0
2544    00              .BYTE   0
2545    00              .BYTE   0               ;MUST HAVE ONE ZERO TO END TABLE!
```

```
4152                            .PAGE                   ;'FOCEND - COMMAND DISPATCH TABLES'
4153                    ;
4154                    ;
4155                    ;
4156  2546  15        COMADH: .HBYTE  SET
4157  2547  10                .HBYTE  IF
4158  2548  11                .HBYTE  DO
4159  2549  10                .HBYTE  ON
4160  254A  12                .HBYTE  GOTO
4161  254B  14                .HBYTE  FOR
4162  254C  12                .HBYTE  RETURN
4163  254D  14                .HBYTE  TYPE
4164  254E  14                .HBYTE  ASK
4165  254F  13                .HBYTE  COMMNT
4166  2550  12                .HBYTE  ERASE
4167  2551  12                .HBYTE  WRITE
4168  2552  11                .HBYTE  MODIFY
4169  2553  10                .HBYTE  QUIT
4170  2554  00                .HBYTE  0#              ;SPACE FOR HACKERS
4171  2555  00                .HBYTE  0L
4172  2556  00                .HBYTE  0@
4173  2557  00                .HBYTE  0               ;MUST HAVE ZERO TO END TABLE!
4174                    ;
4175                    ;
4176                    ;       LOW ORDER ADDR OF COMMAND HANDLING ROUTINE
4177                    ;
4178                    ;
4179  2558  2F        COMADL: .BYTE   SET
4180  2559  C7                .BYTE   IF
4181  255A  9A                .BYTE   DO
4182  255B  C7                .BYTE   ON
4183  255C  F4                .BYTE   GOTO
4184  255D  FE                .BYTE   FOR
4185  255E  0E                .BYTE   RETURN
4186  255F  88                .BYTE   TYPE
4187  2560  88                .BYTE   ASK
4188  2561  65                .BYTE   COMMNT
4189  2562  2E                .BYTE   ERASE
4190  2563  7F                .BYTE   WRITE
4191  2564  33                .BYTE   MODIFY
4192  2565  31                .BYTE   QUIT
4193  2566  00                .BYTE   0#              ;SPACE FOR HACKERS
4194  2567  00                .BYTE   0L
4195  2568  00                .BYTE   0@
4196  2569  00                .BYTE   0               ;MUST HAVE A ZERO TO END TABLE!
4197
4198
```

```
4199                            .PAGE                   ;'DISPATCH TABLE FOR "EVBOP" ROUTINE'
4200
4201
4202            2569            .DEF    EVDSPH=.-1
4203
4204    256A    22              .HBYTE  FADD
4205    256B    22              .HBYTE  FSUB
4206    256C    22              .HBYTE  FDIV
4207    256D    22              .HBYTE  FMUL
4208    256E    1A              .HBYTE  EVPOWR
4209
4210            256E            .DEF    EVDSPL=.-1
4211
4212    256F    7B              .BYTE   FADD
4213    2570    75              .BYTE   FSUB
4214    2571    CC              .BYTE   FDIV
4215    2572    A2              .BYTE   FMUL
4216    2573    E9              .BYTE   EVPOWR
```

```
                        .PAGE                   ;'TABLES USED BY SOFTWARE INTERRUPT SYSTEM'.

                        ;TABLE OF GROUP NUMBERS OF LINES TO 'DO' WHEN EVENT HAPPENS
                        ;ONE ENTRY FOR EACH OF THE 8 PRIORITY CHANNELS

2574  00    INTGRP: .BYTE   0
2575  00            .BYTE   0
2576  00            .BYTE   0
2577  00            .BYTE   0
2578  00            .BYTE   0
2579  00            .BYTE   0
257A  00            .BYTE   0
257B  00            .BYTE   0
257C  00            .BYTE   0

                        ;TABLE OF STEP NUMBERS OF LINES TO 'DO' WHEN AN EVENT HAPPENS

257D  00    INTLIN: .BYTE   0
257E  00            .BYTE   0
257F  00            .BYTE   0
2580  00            .BYTE   0
2581  00            .BYTE   0
2582  00            .BYTE   0
2583  00            .BYTE   0
2584  00            .BYTE   0
2585  00            .BYTE   0

                        ;'AND' MASKS USED TO DISABLE ALL BUT HIGHER PRIO CHANNELS.
                        ;INDEXED BY CURRENT CHANNEL NUMBER

2586  FF    INTTAB: .BYTE   %377            ;CHANNEL 0 ENABLES THEM ALL
2587  FE            .BYTE   %376
2588  FC            .BYTE   %374
2589  F8            .BYTE   %370
258A  F0            .BYTE   %360
258B  E0            .BYTE   %340
258C  C0            .BYTE   %300
258D  80            .BYTE   %200
258E  00            .BYTE   0               ;CHANNEL 8 ENABLES NONE

                        ;BIT TABLE CONTAINING A SINGLE BIT FOR EACH CHANNEL POSITION

258F  00    BITTAB: .BYTE   0
2590  01            .BYTE   1
2591  02            .BYTE   2
2592  04            .BYTE   4
2593  08            .BYTE   8
2594  10            .BYTE   16
2595  20            .BYTE   32
2596  40            .BYTE   64
2597  80            .BYTE   128
```

READ / WRITE


```
                        .PAGE                   ;'DISPATCH TABLE FOR I-C DEVICE NUMBERS'
                                                                                  2831
                0003    .DEF    IDEVM=3         ;MAX NUMBER OF INPUT DEVICES
                0003    .DEF    ODEVM=3         ;MAX NUMBER OF OUTPUT DEVICES

2598  24        IDSPH:  .HBYTE  KEYIN           ;DEVICE 0  - KEYBOARD INPUT ROUTINE
2599  00                .HBYTE  C0BYTI          ;DEVICE 1  - CASSETTE #2 INPUT ROUTINE
259A  00                .HBYTE  C1BYTI          ;DEVICE 2  - CASSETTE #1 INPUT ROUTINE
259B  00                .BYTE   0               ;SPACE FOR HACKERS
259C  00                .BYTE   0

259D  C1        IDSPL:  .BYTE   KEYIN
259E  00                .BYTE   C0BYTI
259F  00                .BYTE   C1BYTI
25A0  00                .BYTE   0
25A1  00                .BYTE   0

25A2  24        ODSPH:  .HBYTE  TVOUT           ;DEVICE 0  - TV OUTPUT ROUTINES
25A3  00                .HBYTE  C0BYTO          ;DEVICE 1  - CASSETTE #2 OUTPUT ROUTINE
25A4  00                .HBYTE  C1BYTO          ;DEVICE 2  - CASSETTE #1 OUTPUT ROUTINE
25A5  00                .BYTE   0               ;SPACE FOR HACKERS
25A6  00                .BYTE   0

25A7  B3        ODSPL:  .BYTE   TVOUT
25A8  00                .BYTE   C0BYTO
25A9  00                .BYTE   C1BYTO
25AA  00                .BYTE   0
25AB  00                .BYTE   0
```

INITIALIZE IN/OUT


```
4301                            .PAGE                   ;'DISPATCH TABLE FOR I-C DEVICE NUMBERS'
4302
4303  25AC  19      INIAH:    .HBYTE  RTS1              ;DON'T NEED TO INITIALIZE KEYBOARD
4304  25AD  19                .HBYTE  RTS1              ;USER MUST PROVIDE ROUTINE
4305  25AE  19                .HBYTE  RTS1              ;
4306  25AF  00                .BYTE   0                 ;SPACE FOR HACKERS
4307  25B0  00                .BYTE   0
4308
4309  25B1  32      INIAL:    .BYTE   RTS1
4310  25B2  32                .BYTE   RTS1
4311  25B3  32                .BYTE   RTS1
4312  25B4  00                .BYTE   0
4313  25B5  00                .BYTE   0
4314
4315  25B6  24      INOAH:    .HBYTE  CONINI            ;USE TO STUFF VECTORS WITH BREAK HANDLERS
4316  25B7  19                .HBYTE  RTS1              ;USER PROVIDES ROUTINE
4317  25B8  19                .HBYTE  RTS1              ;
4318  25B9  00                .BYTE   0
4319  25BA  00                .BYTE   0
4320
4321  25BB  A6      INOAL:    .BYTE   CONINI
4322  25BC  32                .BYTE   RTS1
4323  25BD  32                .BYTE   RTS1
4324  25BE  00                .BYTE   0
4325  25BF  00                .BYTE   0
```

0A

31

CLOSE IN/OUT


```
4326                            .PAGE                   ;'DISPATCH TABLE FOR I-C DEVICE HANDLERS'
4327
4328
4329
4330  25C0  19      CLIAH:  .HBYTE  RTS1            ;KEYBOARD DOESN'T NEED A CLOSE ROUTINE
4331  25C1  19              .HBYTE  RTS1            ;USER PROVIDES ROUTINE
4332  25C2  19              .HBYTE  RTS1            ;
4333  25C3  00              .HBYTE  0               ;SPACE FOR HACKERS
4334  25C4  00              .HBYTE  0               ;SPACE FOR HACKERS
4335
4336  25C5  32      CLIAL:  .BYTE   RTS1
4337  25C6  32              .BYTE   RTS1
4338  25C7  32              .BYTE   RTS1
4339  25C8  00              .BYTE   0
4340  25C9  00              .BYTE   0
4341
4342  25CA  19      CLOAH:  .HBYTE  RTS1            ;TV DOESN'T NEED A CLOSE ROUTINE
4343  25CB  19              .HBYTE  RTS1            ;USER PROVIDES ROUTINE
4344  25CC  19              .HBYTE  RTS1
4345  25CD  00              .HBYTE  0               ;SPACE FOR HACKERS
4346  25CE  00              .HBYTE  0
4347
4348  25CF  32      CLOAL:  .BYTE   RTS1
4349  25D0  32              .BYTE   RTS1
4350  25D1  32              .BYTE   RTS1
4351  25D2  00              .BYTE   0
4352  25D3  00              .BYTE   0
```

ØA

4D

```
                        .PAGE                   ;'FOCEND - TEXT AREAS AND THE LIKE'
                ;
                ;
                ;
25D4    00      PRGBEG: .BYTE   $00             ;LINE NUMBER OF 00.00
25D5    00              .BYTE   $00
25D6    20              .ASCII  ' C FOCAL-65 (V3D) 26-AUG-77'
25D7    43
25D8    20
25D9    46
25DA    4F
25DB    43
25DC    41
25DD    4C
25DE    2D
25DF    36
25E0    35
25E1    22
25E2    28
25E3    56
25E4    33
25E5    44
25E6    29
25E7    20
25E8    32
25E9    36
25EA    2D
25EB    41
25EC    55
25ED    47
25EE    2D
25EF    37
25F0    37
25F1    0D              .BYTE   %15
25F2    FE      PBEG:   .BYTE   EOP             ;START OF PROGRAM TEXT AREA
25F3    FF      VEND:   .BYTE   EOV             ;END OF VARIABLE LIST
                ;
                ;
                ;
                ;
0020            .END
```

ABSF1	23FF	CHKVAR	1BB9	DCSTR	1097	EVDSPH	2569
ABSFE	2406	CHKZER	2426	DCX	11A2	EVDSPL	256E
ABSWAP	2221	CJADR	1352	DCX1	11B2	EVMASK	006E
ABSWP1	222A	CLI	1EA8	DPFLG	028B	EVNTDD	1368
ACSAV	0071	CLIAH	25C0	E	0085	EVPOWR	1AE9
ACTMSK	006D	CLIAL	25C5	EALL	125A	EVRET	139E
ADD	220F	CLIRET	1E45	EARROW	1C7D	FABS	1DFA
ADD1	2212	CLO	1EB4	EATCNT	16C6	FAC1	0080
ADDEND	2234	CLOAH	25CA	EATCR	16DD	FAC2	007B
ALGNSW	2288	CLOAL	25CF	EATCR1	16E4	FADD	227B
ALTCHR	001B	CLRDEV	1015	EATCRC	16DF	FARGM	1F83
ARGN	00DF	CMPCHR	2135	EATEC1	1706	FBDTRM	00F2
ARGNXT	1A01	CNTDWN	1588	EATECC	170D	FCHR	1F0C
ASK	1486	CONADH	2546	EATECM	1708	FCLI	1E3A
ASKAGN	1449	CONADL	2558	EATTVC	196F	FCLO	1E48
ATSW	0022	CONNNT	1365	ECHAR	19E9	FCHAT	2105
BACKSP	1975	CONPL1	22BD	ECHFLG	0068	FCOMPL	22BA
BADCOM	00FD	CONTAB	2532	ECHKC	1A0E	FCON	1E56
BADFUN	1DFB	CONCUR	1932	EFNAME	19CD	FCONT	156E
BADLI	00EC	CONDEV	006A	EFLN	1989	FCUR	1E74
BADLNO	1632	CONINI	24A6	EFLNC	1903	FDIV	22CC
BADMOD	113A	CR	000D	EFLNL	19BB	FDOONE	23DF
BADVAR	00F4	CRLF	1D8B	EGRP	1271	FECH'	1F24
BASTRF	00DB	CRLF2	1D88	EGTVAR	1A47	FETV1	1CA2
BERR1	1D13	DEBGSW	0020	EILLG0	00FE	FETVAR	1CA0
BERR2	1D35	DELDON	177B	ELINE	126E	FGTSV	2000
BERR3	1D42	DELETE	1758	EM1	0084	FGTSV1	2009
BERRC	1D45	DELSPL	006C	ENDGRP	1208	FHALF	0096
BERRET	1D85	DEVRNG	00F0	ENDPIC	1FC9	FI1C	1559
BERROR	1CF3	DIRLIN	00FF	ENEXL	00E8	FI2ARG	1F6B
BIN2	00C2	DIV1	22D4	ENLM	1A36	FIDV	1F2C
BIN4	00D1	DIV10	2419	EOP	00FE	FILLCH	00A5
BITTAB	258F	DIV2	22D7	EOV	00FF	FILOUT	248A
BITV1	006F	DIV3	22E1	EPMISS	1ACA	FILRON	24CD
BOMSTV	1C8D	DIV4	22E6	ERASE	122E	FINCR	153D
BOMSVR	1C9C	DLY1	7320	ERCCNT	1248	FINCR1	1557
BSTART	107F	DLY2	731D	ERCON	124B	FINDLN	160D
BTFOR	152A	DMPSW	0021	ERECL	1D64	FINI	1E1E
C0BYTI	0000	DMVLOP	176A	ERPAR	1ABF	FINO	1E2C
C0BYTO	2300	DO	119A	ERRI	00E9	FINP	242F
C1BYTI	0000	DO1	11B6	ERRO	00DE	FINP2	2452
C1BYTO	2000	DO2	11B9	ESTACK	1AA9	FINP3	244F
CHAR	002B	DOBOP	1A97	ETEMP1	0065	FINP4	247A
CHKEOS	20FC	DOCNT1	11C9	ETERM1	1AB6	FINP5	248A
CHKERC	1D72	DOCONT	11D1	ETERM2	1ACC	FINP6	249C
CHKERR	1D68	DOGRP	11D7	ETERM3	1AD4	FINP7	2491
CHKIDV	1E14	DOGRP1	11E2	EVAL	19F8	FINP8	24A3
CHKLIN	16A4	DOGRPC	11E7	EVALM1	19F5	FINR	1E09
CHKODV	1E0F	DONEXG	00E5	EVALRT	1AB3	FINT	1E00
CHKRTS	1E13	DONEXL	00E6	EVAR	124E	FIODRT	1F4B
CHKSTR	1BCE	DONEXT	11FA	EVBOP	1ADA	FISL	1FE9
CHKSUB	1C00	DOONE	11C2	EVCALL	139B	FISLNX	1FE4

FIX	231A	FRANC	21D4	GOTLNC	1612	ILLCOM	1363
FIX1	2317	FRNILP	21CF	GOTNUM	1273	ILLNO	00FC
FL16PJ	1ED5	FRNINI	21CD	GOTO	12F4	IMVDON	17EB
FLOSGN	0081	FRSET	21CB	GOTO1	12F7	IMVLOP	17D8
FLIMIT	1563	FSBR	2156	GPMISS	183E	IN	00AE
FLOAT	223B	FSBR1	2179	GRPNO	022C	IN1	00B0
FLOATC	2255	FSHORT	156B	GSFNOR	19A3	INI	1E7F
FLPOPJ	1E03	FSLK	20CC	GTCEVN	1F51	INIAH	25AC
FLT16	224F	FSLK1	20D2	GTERR3	1841	INIAL	2581
FLT8	2243	FSTI	2010	HASH	0076	INIC	1E88
FMEM	1EC0	FSTINX	2020	HFABS	0036	INIDEV	1227
FMEMD	1ED8	FSTNUM	1A3C	HFCHR	025C	INIRET	1E29
FMUL	22A2	FSTO	2067	HFCLI	005A	INO	1E94
FNDINI	169F	FSTOC	206D	HFCLO	0066	INOAH	25B6
FNDVAR	18B6	FSTONX	206A	HFCCN	0070	INOAL	25B8
FNEXIT	16D0	FSTRBA	2018	HFCLR	0090	INOC	1E9D
FNEXT	16D5	FSUB	2275	HFECH	0044	INPX1	1084
FOCADR	1000	FSWIT	003B	HFICV	0084	INSCNT	179F
FOCAL	1000	FTEN	0091	HFINI	0092	INSDON	181A
FODV	1F3D	FUNADH	24FA	HFINO	009E	INSERT	1791
FONE	2098	FUNADL	2516	HFINR	00A4	INSLOP	1806
FOR	14FE	FUNC	1DD6	HFINT	00A8	INSW	0024
FOR2	1508	FUNC1	1DDE	HFISL	00AC	INTFIX	1F99
FOREND	15AE	FUNILL	00F5	HFMEM	0096	INTG1	1F96
FORXIT	15B1	FUNTAB	24DE	HFOCV	00B4	INTGER	1F85
FOUNDL	160A	GCONT	12FE	HFOLT	00F4	INTGRP	2574
FOUT	1F03	GETBAD	1DCC	HFPIC	00AA	INTLIN	2570
FOVFL	00F8	GETC	1984	HFRAN	0080	INTSRV	1CD2
FPC1	14F8	GETC1	1995	HFSBR	00C4	INTTAB	2586
FPIC	1FA4	GETCC	198B	HFSLK	00DE	IOSRET	20BA
FPICC	1F9F	GETCX	197A	HFSTI	00FA	IOSRT2	20BE
FPOPJ	1E06	GETILN	1D95	HFSTO	0006	IRETN	105B
FPR0	2332	GETIN	1DB4	IBADL	1082	IRTS	1E93
FPR1	2339	GETIR	1DD0	IDEV	0066	ITEMP1	0073
FPR10	23B2	GETIRC	1DCF	IDEVM	0003	ITMP1H	0056
FPR11	23B9	GETL	1623	IDSPH	2598	ITMP1L	0055
FPR12	23C4	GETLN	15CC	IDSPL	2590	ITMP2H	005A
FPR2	2344	GETLN1	15EC	IDVSAV	0068	ITMP2L	0059
FPR3	235F	GETLNC	15C3	IERRI	18E3	IVCTOR	00A6
FPR4	2366	GETLNS	15C9	IF	12C7	JARGN	1A80
FPR41	2375	GETLNX	15E7	IF1	1100	JMPIND	0061
FPR4A	237B	GETRT1	19A0	IF3	10EC	JOERRO	1EA5
FPR4B	2388	GETSTC	15E3	IFCNT1	10D3	JSRIND	005E
FPR5	238E	GETVAR	1B43	IFCNT2	10E2	JTASK4	1400
FPR6	2394	GICHR	1F12	IFCOM	10E9	JTDUMP	1439
FPR7	239B	GLTEST	1354	IFCR	17B0	K	008D
FPR7A	23A4	GMUL10	1634	IFCC1	1129	KEYIN	24C1
FPR8	23AA	GONEXL	00EB	IFNOP	1123	KLOOP	1A27
FPR9	23AE	GOTALL	1620	IFCA	10CA	L	006E
FPRET	23CB	GOTCOM	1345	IFCNSW	0023	LASTOP	0075
FPRNT	2321	GOTFLG	008C	IFSYN	00E7	LF	000A
FRAN	21C0	GOTFUN	1DEB	IFXCT	110D	LINCHR	005F

LINEL	207F	NPOWR	1B17	PHFAC2	1891	READC	18CD
LINEND	0020	NPOWR1	1B21	PHCK	1853	READC1	18D4
LKFCHR	20F7	NUMBF	0005	PHTB	189E	READCC	18E8
LNCK	1DB1	NXCHAT	2106	PHTMP	189C	READCE	18F1
LOCVAR	1C12	NXIARG	1F7B	PISET	1F06	READCR	18EE
LTL	00FF	NXTCOM	1338	PJADR1	183C	REPLIN	107C
M	008F	NXTLIN	173F	PJMPH	0063	RESBRK	00EF
M1	0081	NXTVAR	1C71	PJMPL	0062	RESINP	1220
M2	007C	ODEV	0067	PLF2B	18A9	RESOUT	1227
MD1	221C	ODEVM	0003	PLFAC2	18A7	RETCMD	00FD
MD2	2300	ODSPH	25A2	PLTB	1884	RETCON	1F6C
MD3	2310	ODSPL	25A7	PLTMP	1882	RETURN	120E
MDEND	2286	ODVSAV	0069	PMATCH	02FA	RNGDEV	1E18
MD0	23CC	OERRO	1916	POFA	184C	RNOECH	18BD
MD01	23D5	ON	10C7	POFB0	1868	ROR1	2293
MENDL	1172	OPNEXT	1A75	POFB2	1866	ROR2	2298
MISOPN	1AD8	OPNMIS	00F9	POFDO	12C2	RSTRNG	20BF
MISOPR	1A7E	OPNXT1	1A86	POFIV	1CB5	RTAR	229A
MLOOK	117E	OPNXT2	1A8C	POFJ	1858	RTLOG	2280
MLOOK1	1181	OPRMIS	00FB	POFSP	214A	RTLOG1	2291
MLOOK2	1183	OUT	00A9	POFTP	171E	RTRN	2320
MNEXL	00E4	OUT1	24BC	POWR1	1807	RTS1	1932
MNOECH	116B	OUTCMD	1D50	POWRLP	1AFD	RTS2	1362
MNXTC	1155	OUTCML	1D54	PRGBEG	25D4	RTS3	161F
MODIFY	1133	OUTLF	1D90	PRILOP	1320	RTSN	2274
MODMOK	113C	OUTLN	166F	PRILVL	002E	RUB	007F
MORFOR	15BE	OUTLN0	1668	PRINTC	18FA	RUB1	1935
MOV1	2411	OUTLN1	166D	PRNTL1	1649	RUB1C	1946
MOV2	2413	OUTLN2	1666	PRNTLN	163E	RUB1CC	1948
MOVXY	2407	OVCHK	22C8	PRCC	1304	RUBCHR	007F
MPB	6E02	OVFL	22CA	PRCC1	1328	RUBECH	005C
MSCHAR	0025	PACKC	191D	PRCCES	1301	RUBLC	1966
MSKBRK	0070	PACKC1	191F	PRCCX	130F	RUBLCL	195E
MUL1	22AB	PBADR	0031	PRRET	191B	RUBLIN	1952
MUL2	22B3	PBEG	25F2	PRTVNM	13E9	RUBLR	1968
N	0090	PBIG	1933	PSPACE	18F8	RUBS1	2059
NEWLIN	1732	PC	0026	PTERM1	1328	RUBSC	205E
NEXTIC	104D	PC1	1365	PTERM2	1335	RUBSTI	2742
NMISRV	1CD1	PCKRET	1930	PUSEA1	1905	SBEG1	0057
NOECH	00BF	PCKRUB	1923	PUSHA	183E	SBEG2	005B
NOEOLS	00F3	PDLBEG	2F00	PUSHB0	1878	SEED	0077
NOFIND	16CF	PDLIST	0053	PUSHB2	1876	SEND1	0058
NONEXT	1756	PDP	004E	PUSHDC	1089	SEND2	005C
NORM	2259	PDPADR	004C	PUSHIV	1CAC	SENDIO	2037
NORM0	2257	PDPINI	101C	PUSHJ	1829	SET	152F
NORM1	2263	PDPTMP	004F	PUSHRT	1849	SET1	152C
NORML	2260	PFERR	00E3	PUSHSP	213E	SETS1	209A
NORMX	22B8	PFLT	1684	PUSHTP	1717	SETS2	20A2
NOSTR	1C3D	PGZERO	00A5	PUTV1	1C95	SETSIO	207F
NOTBRK	1CCB	PHF1B	1888	PUTVAR	1C93	SETSTR	1A6E
NOTVAR	1BC9	PHF2B	1893	PZERO	23D0	SETUP	100D
NOVAR	1C18	PHFAC1	1886	QUIT	1031	SFOUND	2121

SIGN	007A	TESTN	19B0	WALL1	12B9
SIGNP	008A	TESTN1	19B2	WCRLF	12ED
SKPST1	1BF1	TESTNS	19AD	WECL	12E3
SKPSTR	1BF0	TEXTP	002A	WEXCRP	12CE
SNOTF	2132	TEXTP2	0035	WEXIT	129B
SNOTFP	212F	TFORM	1477	WGRP	12BD
SPACE	0020	TGRP	0051	WGRP1	1298
SPNOR	19A6	TLINE	0052	WGRPO	12C2
SPNOR1	19A8	TPC1	14F4	WGRPO1	12C5
ST16PJ	1ED1	TPNAM	13F5	WLINE	12A6
STACK	0100	TPNXTC	141F	WLINE1	12AC
START	1031	TPROC	14FB	WNEXG	00E1
STATUS	0072	TPSTR	1403	WNEXL	00EA
STBSAV	005D	TQUOT	14E2	WONE	12CF
STIADR	0044	TQUOT1	14E4	WONEC	12D2
STINAX	0047	TRMAX	000C	WRET	12EC
STINT	0046	TRMCHK	1B35	WRITE	127F
STOADR	0048	TRMTAB	24D1	WSTRNG	2080
STODAT	146D	TSTEOC	135C	X1	0080
STOMAX	004B	TTERMS	1B30	X1M1	007F
STONT	004A	TTRET	1B3D	X2	0078
STRAD1	0055	TVOUT	24B3	X2M1	007A
STRAD2	0059	TXTADD2	0033	XKEYHL	00A8
STRCNT	0055	TXTADR	0026	XNL10	00F6
STRDEV	1F58	TXTBEG	002F	XNL32	00EC
STRINI	1C56	TXTINI	1725	XNL9	00F7
STRLEN	0048	TYPE	1488	ZERVAR	1C2E
STRLIN	00FE	UFL	00F1	ZRFAC	1CC1
STRMAX	0059	UMARK	00FD	ZRFAC1	1CBD
STRMRK	00FC	UNKINT	00ED		
STRSIZ	0064	UNRBRK	00EE		
STRSNT	0038	UNRFUN	00E2		
SVNA	00DD	UPDEND	1C7F		
SVOK	2014	UPDENV	1C81		
SVRO	00DC	UPDVAR	1C73		
SWAP	2223	VARADR	0037		
SWAP1	222D	VARBAD	1B56		
SWPALG	2278	VARBEG	003E		
SYSTIN	72EE	VARDN1	1B7D		
SYSTOU	72C6	VARDUN	1B72		
T	00A0	VAREND	0042		
TASK	148A	VARINI	1C68		
TASK1	143C	VAROK	1B58		
TASK4	14DD	VARSIZ	0008		
TCHAR	003D	VARSOK	1BAF		
TCR	14D8	VARST	0040		
TCRLF	14D3	VARSUB	188E		
TDCONT	13B7	VCHAR	003D		
TDNEXT	13A1	VEND	25F3		
TDUMP	139E	VSIZE	003C		
TEMP1	005F	VSUB	0039		
TENS	1DD1	WALL	1286		

```
FRANC    3455   3464
FRNILP   3461   3463
FRNINI   3458   3462
FRSET    3456   3459
FSBR     3400   4063   4096
FSBR1    3414   3417
FSHORT   1332   1336
FSLK     3311   4067   4100
FSLK1    3312   3315
FSTI     3197   4065   4098
FSTINX   3196   3208   3221   3233
FSTNUM   2191   2197
FSTO     3240   4066   4099
FSTOC    3214   3219
FSTONX   3241   3249
FSTRBA   3175   3188
FSUB     1362   1400   3593   3819   4205   4213
FSWIT      96   1515   1520   1542
FTEN      269   1443   3803   3844   3880
FUNADH   2799   4046
FUNADL   2797   4079
FUNC     2150   2789
FUNC1    2791   2796
FUNILL    381   2362
FUNTAB   2791   2793   4014
GCONT     964    967
GETBAD   2746   2748   2769
GETC      661    665    675    936    970    993   1186   1252   1254   1304   1564   1588   1644   1705
          2080   2099   2132   2156   2158   2163   2264   2272   2382   2390   2398   2416   3124   3176
          3875
GETC1    2076   2088
GETCC    2081   2083
GETCX    2075   2087
GETILN   1408   2741
GETIN    2198   2741   2749   2756
GETIR    2744   2760   2771
GETIRC   2768   2770
GETL     1405   1416   1433
GETLN     848   1390
GETLN1   1387   1404   3400
GETLNC    578   1386
GETLNS    660    707    766    893    962   1187   1389   3125
GETLNX   1399   1402
GETRT1   2091   2093
GETSTC   2452   2530
GETVAR   1159   1278   2206   2365   3178
GICHR    3017   3015   3198
GLTEST    974    991   1017
GMUL10   1410   1411   1443
GONEXL    371    966
GOTALL   1421   1427
GOTCOM   1000   1003
GOTFLG    263   3862   3885
```

```
GOTFUN   2794    2797
GOTLNO   1359    1395    1397    1401    1418
GOTNUM    572     577
GOTO      962    4160    4183
GOTO1     693     963
GPMISS   2359    2415
GRPNO      66     797     880     902    1056    1188    1391    1406    1419    1422    1510    1624    1747    2742
          3128    3419
GSPNOR    570     649    2099    2102    2108
GTDEVN   3039    3049    3061
GTERR3   2361    2377
HASH      211    3457
HFABS     398    4014
HFCHR     435    4021
HFCLI     412    4028
HFCLO     413    4029
HFCON     414    4030
HFCUR     406    4022
HFECH     407    4023
HFIDV     403    4019
HFINI     410    4026
HFINO     411    4027
HFINR     402    4018
HFINT     401    4017
HFISL     416    4032
HFMEM     409    4025
HFODV     404    4020
HFOUT     399    4015
HFPIC     406    4024
HFRAN     400    4016
HFSBR     415    4031
HFSLK     419    4035
HFSTI     417    4033
HFSTO     418    4034
IBADL     579     580     584
IDEV      180     510     525     834    1175    1958    3041    3043    3217
IDEVM    2831    4274
IDSPH    1961    4278
IDSPL    1963    4284
IDVSAV    182     833    3042
IERRI    1967    2927
IF        700    4157    4180
IF1       668     675
IF3       664     666     679
IFCNT1    651     654
IFCNT2    658     661
IFCOM     665     672
IFCR     1734    1737
IFDO1     686     694
IFNOP     670     684     691
IFON      649
IFONSW     53
IFSYN     367     653
```

```
NORMX    3636    3689
NOSTR    2484    2528
NOTBRK   2622    2630
NOTVAR   2433    2467    2471    2475
NOVAR    2428    2483
NPOWR    2311    2326    2335
NPOWR1   2329    2331
NUMBF     429     432    1327    1328    1897    1898    1906    1907    1915    1916    1924    1933    2494    2609
NXCMAT   3347    3355
NXIARG   2975    3084    3089    3142
NXTCOM    997    1002
NXTLIN    594     603    1634
NXTVAR   1098    2433    2541
ODEV      181     511     527     837     940     950    1986    3051    3053
ODEVM    2828    4275
ODSPH    1992    4290
ODSPL    1992    4296
ODVSAV    183     836    3052
OERRO    1998    2038
ON        648     730    4159    4182
OPNEXT   2159    2196    2200    2209    2221    2225    2275
OPNMIS    385    2285
OPNXT1   2228    2238    2283
OPNXT2   2239    2241    2256
OPRMIS    387    2230
OUT       313    3932    3934
OUT1     3931    3934
OUTCMD   2695    2697
OUTCML   2699    2708
OUTLF     736     942    2710    2730
OUTLN    1477    1480
OUTLN0   1391    1478    2661
OUTLN1   1479
OUTLN2   1475
OVCHK    3646    3685
OVFL     3612    3647    3677
PACKC    2006
PACKC1    585     733     748    2007
PBADR      74     867
PBEG       74    4388
PBIG     2015    2015
PC         62     540     591     615     618     627     633     809     814     830    1342    1351    1369    1517
          1625    1627    1638    1641    2662    2673    2675    2680    2694    2713
PC1       973    1029    1031    1263
PCKRET   2016    2026    2037    2045    2057
PCKRUB   2013    2038
PDLBEG    140     461
PDLIST    140     517     519
PDP       131     516    1816    1822    1836    1841
PDPADR    130     518     520    1817    1821    1839    1840
PDPINI    544     516     569
PDPTMP    133
PFERR     363    2142
```