# The ROM PSET Unwound

```
THE ROM PSET UNWOUND - MDJ 2023/03/16

NOTE: LABELS AND COMMENTS ARE LOOSELY BASED ON THE UNRAVELLED II SERIES,
      BUT NOT EXACTLY.

"-" UNDER BYT ==> BYTE(S) ALREADY COUNTED ONCE.

                                              CYC BYT
                                              --- ---
9361 86 01          PSET    LDA     #$01        2    PSET FLAG
9363 20 01                  BRA     L9366       2

9366 97 C2          L9366   STA     SETFLG      2    STORE FLAG
9368 BD B2 6A               JSR     LB26A       3    SYNTAX CHECK FOR `(

B26A C6 28          LB26A   LDB     #'(         2    SYNTAX CHECK FOR `(
B26C 8C                     FCB     SKP2        1    SKIP 2 BYTES
B26D C6 2C                  LDB     #',         2    SYNTAX CHECK FOR COMMA

    ** AN AMUSING BIT OF CODE HERE. ACTUALLY, 8C IS THE
       OPCODE FOR CMPX IMMEDIATE (MC6809 COOKBOOK, PAGE 171.
       THUS:  8C
              C6 2C    =    8C C6 2C = CMPX #$C62C
       BUT, SINCE NO BRANCH INSTRUCTION FOLLOWS, THE THREE
       BYTES 8C C6 2C DO NOTHING EXCEPT TAKE UP SPACE AND
       WASTE CYCLES, WHILE NONETHELESS BEING NECESSARY IN
       ORDER TO AVOID ACTUALLY DOING AN "LDB #'," HERE.
```

```
B26F E1 9F 00 A6            CMPB    [CHARAD]        4    COMPARE B TO CURRENT INPUT
B273 26 02                  BNE     LB277           2    CHAR: SNTX ERR IF NO MATCH

     ** ASSUMING NO SYNTAX ERROR, THE BRANCH WILL NOT BE TAKEN

B275 0E 9F                  JMP     GETNCH          2    GET A CHARACTER FROM BASIC

009F 0C A7          GETNCH  INC     <CHARAD+1       2    INCR INPUT POINTER LOW BYTE
00A1 26 02                  BNE     GETCCH          2    BRANCH IF NOT 0 (NO CARRY)

     ** SINCE OUR PURPOSE HERE IS TO SEE HOW MANY BYTES AND
        CYCLES WE CAN SAVE, WE WILL ASSUME THE BRANCH IS
        TAKEN HERE. THIS WILL ALLOW US TO COMPARE OUR
        PROPOSED IMPROVEMENTS ADGAINST THE LEANEST POSSIBLE
        INTERPRETATION OF THE EXISTING PSET CODE.

00A5 B6             GETCCH  FCB     $B6             1    OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                          2    THESE 2 BYTES CONTAIN ADDR
                                                         OF THE CURRENT CHAR WHICH
                                                         THE BASIC INTERPRETER IS
                                                         PROCESSING
00A8 7E AA 1A               JMP     BROMHK          3    JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK  CMPA    #'9+1           2    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28           2    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28   RTS                     1    CALLED FROM 9368
```

```
936B BD 93 1A              JSR     L931A          3    EVAL HOR & VER AND NORMALIZE

931A BD 92 FC       L931A  JSR     L92FC          3    GO GET HOR & VER COORDINATES

92FC BD B7 34       L92FC  JSR     LB734          3    EVALUATE TWO EXPRESSIONS
                                                        FROM THE BASIC LINE - RETURN
                                                        WITH THE 1ST VALUE IN BINVAL
                                                        AND THE 2ND IN ACCB

B734 8D 07          LB734  BSR     LB73D          2    EVALUATE AN EXPRESSION

B73D BD B1 41       LB73D  JSR     LB141          3    EVALUATE NUMERIC EXPRESSION

B141 8D 13          LB141  BSR     LB156          2    CHECK FOR NUMERIC

B156 8D 6E          LB156  BSR     B1C6           2    BACK UP INPUT POINTER

B1C6 9E A6          LB1C6  LDX     CHARAD         2    GET BASIC'S INPUT POINTER
B1C8 7E AE BB              JMP     LAEBB          3    AND MOVE IT BACK ONE

AEBB 30 1F          LAEBB  LEAX    -1,X           2    MOV TO JUST BEF STRT OF LINE
AEBD 9F A6                 STX     CHARAD         2    RESET BASIC'S INPUT POINTER
AEBF 39                    RTS                    1    CALLED FROM B156

B158 4F                    CLRA                   1    END OF OP PRECEDENCE FLAG
B159 8C                    FCB     SKP2           1    SKIP 2 BYTES
B15A 34 04                 PSHS    B              2    THIS IS SKIPPER
```

** SINCE 8C IS THE OPCODE FOR CMPX IMMEDIATE,
THE "PSHS B" IS SKIPPED.

```
B15C 34 02                  PSHS    A               2    SAVE FLAG (PRECEDENCE FLAG)
B15E C6 01                  LDB     #1              2
B160 BD AC 33               JSR     LAC33           2    FREE RAM ROOM FOR (B) WORDS?

AC33 4F             LAC33   CLRA                    1    ACCD CONTAINS NUMBER OF XTRA
AC34 58                     ASLB                    1    BYTES TO PUT ON STACK
AC35 D3 1F                  ADDD    ARYEND          2    END OF PROGRAM AND VARIABLES
AC37 C3 00 3A               ADDD    #STKBUF         3    ROOM FOR STACK?
AC3A 25 08                  BCS     LAC44           2    BRANCH IF GREATER THAN $FFFF

     ** SINCE > $FFFF WOULD BE AN ERROR, WE WILL
        ASSUME THE BRANCH IS NOT TAKEN.

AC3C 10 DF 17               STS     BOTSTK          3    CUR NEW BOTTOM OF STACK PTR
AC3F 10 93 17               CMPD    BOTSTK          3    WILL WE BE BELOW STACK?
AC42 25 EE                  BCS     LAC32           2    YES - NO ERROR

AC32 39                     RTS                     1    CALLED FROM B160

B163 BD B2 23               JSR     LB223           3    GO EVALUATE AN EXPRESSION

B223 BD 01 8B       LB223   JSR     RVEC15          3    HOOK INTO RAM

018B 7E CE D2       RVEC15  JMP     $CED2           3    DISK BASIC 1.1 VECTOR

CED2 A6 64          DVEC15  LDA     $04,S           2    CHK STACKED PRECEDENCE FLAG
CED4 26 13                  BNE     LCEE9           2    GO IF NOT END OF OP

     ** WE WILL ASSUME IT IS NOT THE END OF AN OPERATION.
        AND THUS THE BRANCH IS TAKEN (LEANEST PSET)
```

```
CEE9 7E 88 46       LCEE9   JMP     XVEC15          3    EXTENDED BASIC EXPR EVAL

8846 35 40          XVEC15  PULS    U               2    PULL RTS ADDRESS; SAVE IN U
8848 0F 06                  CLR     VALTYP          2    SET VARIABLE TYPE TO NUMERIC
884A 9E A6                  LDX     CHARAD          2    CURRENT INPUT POINTER TO X
884C 9D 9F                  JSR     GETNCH          2    GET CHARACTER FROM BASIC

009F 0C A7          GETNCH  INC     <CHARAD+1       -    INCR INPUT POINTER LOW BYTE
00A1 26 02                  BNE     GETCCH          -    BRANCH IF NOT 0 (NO CARRY)

     ** SINCE OUR PURPOSE HERE IS TO SEE HOW MANY BYTES AND
        CYCLES WE CAN SAVE, WE WILL ASSUME THE BRANCH IS
        TAKEN HERE. THIS WILL ALLOW US TO COMPARE OUR
        PROPOSED IMPROVEMENTS ADGAINST THE LEANEST POSSIBLE
        INTERPRETATION OF THE EXISTING PSET CODE.

00A5 B6             GETCCH  FCB     $B6             -    OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                          -    THESE 2 BYTES CONTAIN ADDR
                                                         OF THE CURRENT CHAR WHICH
                                                         THE BASIC INTERPRETER IS
                                                         PROCESSING
00A8 7E AA 1A               JMP     BROMHK          -    JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK  CMPA    #'9+1           -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28           -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28   RTS                     -    CALLED FROM 884C

884E 81 26                  CMPA    #'&'            2    HEX AND OCTAL?
```

```
8850 27 99                BEQ     L87EB          2    GO IF YES

     ** WE WILL ASSUME IT IS DECIMAL AND THUS THE BRANCH
        IS NOT TAKEN.

8852 81 CC                CMPA    #$CC           2    FUNCTION CALL TOKEN?
8854 27 5E                BEQ     L88B4          2    GO IF YES

     ** WE WILL ASSUME IT IS NOT A FUNCTION CALL AND
        THUS THE BRANCH IS NOT TAKEN.

8856 81 FF                CMPA    #$FF           2    SECONDARY TOKEN?
8858 26 08                BNE     L8862          2    GO IF NO

     ** WE WILL ASSUME IT IS NOT A SECONDARY FUNCTION
        AND THUS THE BRANCH IS TAKEN.

8862 9F A6          L8862 STX     CHARAD         2    RESTORE BASIC'S INPUT PTR
8864 6E C4                JMP     ,U             2    CALLED FROM B223

     ** GIVEN THE "PULS U" AT 8846, THIS "JMP ,U" = RTS

B226 0F 06                CLR     VALTYP         2    INIT TYPE FLAG TO NUMERIC
B228 9D 9F                JSR     GETNCH         2    GET AN INPUT CHAR

009F 0C A7          GETNCH INC    <CHARAD+1      -    INCR INPUT POINTER LOW BYTE
00A1 26 02                BNE     GETCCH         -    BRANCH IF NOT 0 (NO CARRY)

     ** SINCE OUR PURPOSE HERE IS TO SEE HOW MANY BYTES AND
        CYCLES WE CAN SAVE, WE WILL ASSUME THE BRANCH IS
        TAKEN HERE. THIS WILL ALLOW US TO COMPARE OUR
```

PROPOSED IMPROVEMENTS ADGAINST THE LEANEST POSSIBLE
INTERPRETATION OF THE EXISTING PSET CODE.

```
00A5 B6             GETCCH  FCB     $B6             -    OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                          -    THESE 2 BYTES CONTAIN ADDR
                                                         OF THE CURRENT CHAR WHICH
                                                         THE BASIC INTERPRETER IS
                                                         PROCESSING
00A8 7E AA 1A               JMP     BROMHK          -    JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK  CMPA    #'9+1           -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28           -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28   RTS                     -    CALLED FROM B228

B22A 24 03                  BCC     LB22F           2    BRANCH IF NOT NUMERIC

     ** WE WILL ASSUME THAT IT IS NUMERIC AND THUS
        THE BRANCH IS NOT TAKEN

B22C 7E BD 12               JMP     LBD12           3    CONVERT ASCII STRING TO
                                                         FLOATING POINT - RETURN
                                                         RESULT IN FPA0

BD12 9E 8A          LBD12   LDX     ZERO            2    (X) = 0
BD14 9F 54                  STX     FP0SGN          2    ZERO OUT FPA0 & SIGN FLAG
BD16 9F 4F                  STX     FP0EXP          2
BD18 9F 51                  STX     FPA0+1          2
BD1A 9F 52                  STX     FPA0+2          2
```

```
BD1C 9F 47                  STX     V47             2    INITIALIZE EXPONENT & SIGN
                                                         FLAG TO ZERO
BD1E 9F 45                  STX     V45             2    INITIALIZE RIGHT DECIMAL CTR
                                                         & DECIMAL PT FLAG TO 0
BD20 25 64                  BCS     LBD86           2    IF CARRY SET (NUMERIC
                                                         CHARACTER), ASSUME ACCA
                                                         CONTAINS FIRST NUMERIC CHAR,
                                                         SIGN IS POSITIVE AND SKIP
                                                         THE RAM HOOK

     ** WE WILL ASSUME A NUMERIC CHARACTER AND THUS THE
        BRANCH IS TAKEN (LEANEST PSET).

BD86 D6 45          LBD86   LDB     V45             2    GET THE RIGHT DECIMAL COUNTER
BD88 D0 46                  SUBD    V46             2    AND SUBTRACT DECIMAL PNT FLAG
BD8A D7 45                  STB     V45             2
BD8C 34 02                  PSHS    A               2    SAVE NEW DIGIT ON STACK
BD8E BD BB 6A               JSR     LBB6A           3    MULTIPLY FPA0 BY 10

BB6A BD BC 5F       LBB6A   JSR     LBC5F           3    TRANSFER FPA0 TO FPA1

BC5F DC 4F          LBC5F   LDD     FP0EXP          2    TRANSFER EXPONENT & MS BYTE
BC61 DD 5C                  STD     FP1EXP          2
BC63 9E 51                  LDX     FPA0+1          2    TRANSFER MIDDLE TWO BYTES
BC65 9F 5E                  STX     FPA1+1          2
BC67 9E 53                  LDX     FPA0+3          2    TRANSFER BOTTOM TWO BYTES
BC69 9F 60                  STX     FPA1+3          2
BC6B 4D                     TSTA                    1    SET FLAGS PER EXPONENT
BC6C 39                     RTS                     1    CALLED FROM BB6A

BB6D 27 0D                  BEQ     LBB7C           2    BRANCH IF EXPONENT = 0
```

```
     ** WE WILL ASSUME THE EXPONENT = 0 (TO BE EXPECTED
        FOR A SCREEN COORDINATE; ALSO FOR LEANEST PSET),
        AND THUS THE BRANCH IS TAKEN

BB7C 39                    RTS                  1    CALLED FROM BD8E

BD91 35 04                 PULS    B            2    GET NEW DIGIT BACK
BD93 C0 30                 SUBB    #'0          2    MASK OFF ASCII
BD95 8D 02                 BSR     LBD99        2    ADD ACCB TO FPA0

BD99 BD BC 2F       LBD99  JSR     LBC2F        3    PACK FPA0 AND SAVE IN FPA3

BC2F 8E 00 40       LBC2F  LDX     #V40         3    POINT X TO MANTISSA OF FPA3
BC32 8C                    FCB     SKP2         1    SKIP TWO BYTES
BC33 9E 3B                 LDX     VARDES       2    POINT X TO VAR DESCRIPTOR

     ** SINCE 8C IS THE OPCODE FOR CMPX IMMEDIATE,
        THE "LDX VARDES" IS SKIPPED.

BC35 96 4F                 LDA     FP0EXP       2    COPY EXPONENT
BC37 A7 84                 STA     ,X           2
BC39 96 54                 LDA     FP0SGN       2    GET MANTISSA SIGN BIT
BC3B 8A 7F                 ORA     #$7F         2    MASK THE BOTTOM 7 BITS
BC3D 94 50                 ANDA    FPA0         2    AND BIT 7 OF MANTISSA SIGN
                                                     INTO BIT 7 OF MS BYTE
BC3F A7 01                 STA     1,X          2    SAVE MS BYTE
BC41 96 51                 LDA     FPA0+1       2    MOVE 2ND MANTISSA BYTE
BC43 A7 02                 STA     2,X          2
BC45 DE 52                 LDU     FPA0+2       2    MOVE BOTTOM 2 MANTISSA BYTES
BC47 EF 03                 STU     3,X          2
```

```
BC49 39                     RTS                 1    CALLED FROM BD99

BD9C BD BC 7C               JSR     LBC7C       2    CONVERT B TO FP NUM IN FPA0

BC7C D7 50          LBC7C   STB     FPA0        2    SAVE ACCB IN FPA0
BC7E 0F 51                  CLR     FPA0+1      2    CLEAR 2ND MANTISSA FPA0 BYTE
BC80 C6 88                  LDB     #$88        2    EXPONENT FOR FPA0 BE INTEGER
BC82 96 50                  LDA     FPA0        2    GET MS BYTE OF MANTISSA
BC84 80 80                  SUBA    #$80        2    SET CARRY IF POSITIVE MTSSA
BC86 D7 4F                  STB     FP0EXP      2    SAVE EXPONENT
BC88 DC 8A                  LDD     ZERO        2    ZERO OUT ACCD AND
BC8A DD 52                  STD     FPA0+2      2    BOTTOM HALF OF FPA0
BC8C 97 63                  STA     FPSBYT      2    CLEAR SUB BYTE
BC8E 97 54                  STA     FP0SGN      2    CLEAR SIGN OF FPA0 MANTISSA
BC90 7E BA 18               JMP     LBA18       3    GO NORMALIZE FPA0

BA18 25 02          LBA18   BCS     LBA1C       2    BRANCH IF POSITIVE MANTISSA

     ** WE WILL ASSUME THAT THE MANTISSA IS POSITIVE
        AND THUS THE BRANCH IS TAKEN.

BA1C 5F             LBA1C   CLRB                1    CLEAR TEMP EXPONENT ACCUM
BA1D 96 50                  LDA     FPA0        2    TEST MSB OF MANTISSA
BA1F 26 2E                  BNE     LBA4F       2    BRANCH IF <> 0

     ** WE WILL ASSUME THAT THE MSB IS <> 0 AND THUS
        THE BRANCH IS TAKEN (LEANEST PSET).

BA4F 2A F3          LBA4F   BPL     LBA44       2    BRANCH IF NOT YET NORMALIZED

     ** WE WILL ASSUME THAT IT IS ALREADY NORMALIZED AND THUS
```

```
        THE BRANCH IS NOT TAKEN (LEANEST PSET).

BA51 96 4F                  LDA     FP0EXP          2    GET CURRENT EXPONENT
BA53 34 04                  PSHS    B               2    SAVE EXPONENT MODIFIER
BA55 A0 E0                  SUBA    ,S+             2    SUBTRACT EXPONENT MODIFIER
                                                         AND CLEAR IT OFF THE STACK
BA57 97 4F                  STA     FP0EXP          2    SAVE AS NEW EXPONENT
BA59 23 DE                  BLS     LBA39           2    SET FPA0 = 0 IF THE
                                                         NORMALIZATION CAUSED MORE
                                                         OR EQUAL NUMBER OF LEFT
                                                         SHIFTS THAN THE SIZE OF THE
                                                         EXPONENT

     ** WE WILL ASSUME THIS DIDN'T HAPPEN AND THUS
        THE BRANCH IS NOT TAKEN (LEANEST PSET).

BA5B 8C                     FCB     SKP2            1    SKIP 2 BYTES
BA5C 25 08                  BCS     LBA66           2    BRANCH IF MANTISSA OVERFLOW

     ** SINCE 8C IS THE OPCODE FOR CMPX IMMEDIATE,
        THE "BCS LBA66" IS SKIPPED.

BA5E 08 63                  ASL     FPSBYT          2    SUB BYTE BIT 7 TO CARRY
BA60 86 00                  LDA     #0              2    CLRA DON'T CHANGE CARRY FLAG
BA62 97 63                  STA     FPSBYT          2    CLEAR THE SUB BYTE
BA64 20 0C                  BRA     LBA72           2    GO ROUND-OFF RESULT

BA72 24 04          LBA72   BCC     LBA78           2    BRANCH IF NO ROUNDOFF NEEDED

     ** WE WILL ASSUME NO ROUNDOFF IS NEEDED AND THUS
        THE BRANCH IS TAKEN (LEANEST PSET).
```

```
BA78 39            LBA78   RTS                 1    CALLED FROM BD9C

BD9F 8E 00 40              LDX     #V40        3    ADD FPA0 TO FPA3
BDA2 7E B9 C2              JMP     LB9C2       3

B9C2 BD BB 2F      LB9C2   JSR     LBB2F       3    UNPACK PACKED FP DATA

BB2F EC 01         LBB2F   LDD     1,X         2    GET TWO MSB BYTES
BB31 97 61                 STA     FP1SGN      2    SAVE MANTISSA SIGN BYTE
BB33 8A 80                 ORA     #$80        2    FORCE BIT 7 OF MANTISSA = 1
BB35 DD 5D                 STD     FPA1        2    SAVE 2 MSB BYTES IN FPA1
BB37 D6 61                 LDB     FP1SGN      2    GET PACKED MANTISSA SGN BYTE
BB39 D8 54                 EORB    FP0SGN      2    XOR FPA0 SIGN BYTE
BB3B D7 62                 STB     RESSGN      2    SAVE ADJUSTED SIGN BYTE
BB3D EC 03                 LDD     3,X         2    GET 2 LSB BYTES OF MANTISSA
BB3F DD 5F                 STD     FPA1+2      2    AND PUT IN FPA1
BB41 A6 84                 LDA     ,X          2    GET EXPONENT FROM (X) AND
BB43 97 5C                 STA     FP1EXP      2    PUT IN EXPONENT OF FPA1
BB45 D6 4F                 LDB     FP0EXP      2    GET EXPONENT OF FPA0
BB47 39                    RTS                 1    CALLED FROM B9C2

B9C5 5D                    TSTB                1    CHECK EXPONENT OF FPA0
B9C6 10 27 02 80           LBEQ    LBC4A       4    GO IF FPA0 = 0

     ** WE WILL ASSUME FPA0 = 0 (LEANEST PSET)
        AND THUS THE BRANCH IS TAKEN.

BC4A 96 61         LBC4A   LDA     FP1SGN      2    MOVE FPA1 TO FPA0
BC4C 97 54         LBC4C   STA     FP0SGN      2
BC4E 9E 5C                 LDX     FP1EXP      2
```

```
BC50 9F 4F                  STX     FP0EXP          2
BC52 0F 63                  CLR     FPSBYT          2
BC54 96 5E                  LDA     FPA1+1          2
BC56 97 51                  STA     FPA0+1          2
BC58 96 54                  LDA     FP0SGN          2
BC5A 9E 5F                  LDX     FPA1+2          2
BC5C 9F 52                  STX     FPA0+2          2
BC5E 39                     RTS                     1    CALLED FROM BD95

BD97 20 98                  BRA     LBD31           2    GET ANOTHER CHAR FROM BASIC

BD31 9D 9F          LBD31   JSR     GETNCH          2    GET NEXT INPUT CHARACTER

009F 0C A7          GETNCH  INC     <CHARAD+1       -    INCR INPUT POINTER LOW BYTE
00A1 26 02                  BNE     GETCCH          -    BRANCH IF NOT 0 (NO CARRY)

     ** SINCE OUR PURPOSE HERE IS TO SEE HOW MANY BYTES AND
        CYCLES WE CAN SAVE, WE WILL ASSUME THE BRANCH IS
        TAKEN HERE. THIS WILL ALLOW US TO COMPARE OUR
        PROPOSED IMPROVEMENTS ADGAINST THE LEANEST POSSIBLE
        INTERPRETATION OF THE EXISTING PSET CODE.

00A5 B6             GETCCH  FCB     $B6             -    OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                          -    THESE 2 BYTES CONTAIN ADDR
                                                         OF THE CURRENT CHAR WHICH
                                                         THE BASIC INTERPRETER IS
                                                         PROCESSING
00A8 7E AA 1A               JMP     BROMHK          -    JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK  CMPA    #'9+1           -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28           -    BRANCH IF > 9
```

```
     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28   RTS                     -    CALLED FROM BD31

BD33 25 51                  BCS     LBD86           2    BRANCH IF NUMERIC CHARACTER

     ** HAVING BEEN THROUGH LBD86 BEFORE, WE WILL ASSUME
        THAT IT IS NOT A NUMERIC CHARACTER AND THUS THE
        BRANCH IS NOT TAKEN (LEANEST PSET).

BD35 81 2E          LBD35   CMPA    #'.             2    DECIMAL POlNT?
BD37 27 28                  BEQ     LBD61           3    GO IF YES

     ** WE WILL ASSUME THAT IT IS NOT A DECIMAL POINT
        AND THUS THE BRANCH IS NOT TAKEN.

BD39 81 45                  CMPA    #'E             2    "E" (SCIENTIFIC NOTATION)?
BD3B 26 28                  BNE     LBD65           2    GO IF NO

     ** WE WILL ASSUME IT IS NOT SCIENTIFIC NOTATION
        AND THUS THE BRANCH IS TAKEN.

BD65 96 47          LBD65   LDA     V47             2    GET EXPONENT, SUBTRACT THE
BD67 90 45                  SUBA    V45             2    NUMBER OF PLACES TO RIGHT
BD69 97 47                  STA     V47             2    OF DECIMAL POINT AND RESAVE
BD6B 27 12                  BEQ     LBD7F           2    EXIT IF ADJUSTED EXPONENT= 0

     ** WE WILL ASSUME THE ADJUSTED EXPONENT IS ZERO
        AND THUS THE BRANCH IS TAKEN.
```

```
BD7F 96 55          LBD7F   LDA     COEFCT          2     GET THE SIGN FLAG
BD81 2A 8E                  BPL     LBD11           2     RETURN IF POSITIVE

     ** WE WILL ASSUME THE RESULT IS NOT POSITIVE AND
        THUS THE BRANCH IS NOT TAKEN (LEANEST PSET).

BD83 7E BE E9               JMP     LBEE9           3     TOGGLE MANTISSA SIGN OF FPA0

BEE9 96 4F          LBEE9   LDA     FP0EXP          2     GET EXPONENT OF FPA0
BEEB 27 02                  BEQ     LBEEF           2     BRANCH IF FPA0 = 0

     ** WE WILL ASSUME FPA0 <>0 AND THUS THE BRANCH
        IS NOT TAKEN.

BEED 03 54                  COM     FP0SGN          2     TOGGLE MANTISSA SIGN OF FPA0
BEEF 39             LBEEF   RTS                     1     CALLED FROM B163

B166 0F 3F                  CLR     TRELFL          2     RESET RELATIONAL OP FLAG
B168 9D A5                  JSR     GETCCH          2     GET CURRENT INPUT CHARACTER

00A5 B6             GETCCH  FCB      $B6            -     OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                          -     THESE 2 BYTES CONTAIN ADDR
                                                          OF THE CURRENT CHAR WHICH
                                                          THE BASIC INTERPRETER IS
                                                          PROCESSING
00A8 7E AA 1A               JMP      BROMHK         -     JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK  CMPA     #'9+1          -     IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS      LAA28          -     BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)
```

```
AA28 39            LAA28   RTS                    -    CALLED FROM B168

B16A 80 B2                 SUBA    #$B2           2    TOKEN FOR >
B16C 25 13                 BCS     LB181          2    BRANCH IF < RELATIONAL OPS

     ** WE WILL ASSUME THAT IT IS LESS AND THUS THE
        BRANCH IS TAKEN (LEANEST PSET)

B181 D6 3F         LB181   LDB     TRELFL         2    GET RELATIONAL OPERATOR FLAG
B183 26 33                 BNE     LB1B8          2    BRANCH IF RELATIONAL COMPARISON

     ** WE WILL ASSUME THAT IT IS NOT A RELATIONAL COMPARISON
        AND THUS THE BRANCH IS NOT TAKEN (LEANEST PSET).

B185 10 24 00 6B           LBCC    LB1F4          4    BRANCH IF > RELATIONAL OP

     ** WE WILL ASSUME THAT IT IS > RELATIONAL COMPARISON
        AND THUS THE BRANCH IS TAKEN (LEANEST PSET).

B1F4 9E 8A         LB1F4    LDX     ZERO          2    POINT X TO DUMMY VALUE (0)
B1F6 A6 E0                  LDA     ,S+           2    GET PRECEDENCE FLAG FROM STK
B1F8 27 26                  BEQ     LB220         2    BRANCH IF END OF EXPRESSION

     ** WE WILL ASSUME THAT IT IS END OF EXPRESSION AND
        THE BRANCH IS TAKEN (LEANEST PSET).

B220 D6 4F         LB220   LDB     FP0EXP         2    GET EXPONENT OF FPA0
B222 39                    RTS                    1    CALLED FROM B141

B143 1C FE         LB143   ANDCC   #$FE           2    CLEAR CARRY FLAG
```

```
B145 7D                     FCB    $7D             1    OP CODE OF TST $1A01
                                                        SKIP TWO BYTES (DO NOT
                                                        CHANGE CARRY FLAG)
B146 1A 01                  ORCC   #1              2    SET CARRY
B148 0D 06                  TST    VALTYP          2    TEST TYPE FLAG
B14A 25 03                  BCS    LB14F           2    BRANCH IF STRING

     ** WE WILL ASSUME IT IS NOT A STRING AND THUS
        THE BRANCH IS NOT TAKEN

B14C 2A 99                  BPL    LB0E7           2    RETURN ON PLUS

     ** WE WILL ASSUME THAT IT IS POSITIVE AND THUS
        THE BRANCH IS TAKEN (POSITIVE IS TO BE
        EXPECTED FOR SCREEN COORDINATES; ALSO FOR
        LEANEST PRESET).

B0E7 39            LB0E7    RTS                    1    CALLED FROM B73D

B740 96 54                  LDA    FP0SGN          2    GET SIGN OF FPA0 MANTISSA
B742 2B C2                  BMI    LB706           2    'ILLEGAL FUNCTION CALL'

     ** WE WILL ASSUME THE BRANCH IS NOT TAKEN

B744 96 4F                  LDA    FP0EXP          2    GET EXPONENT OF FPA0
B746 81 90                  CMPA   #$90            2    COMPARE TO LARGEST POS INT
B748 22 BC                  BHI    LB706           2    'ILLEGAL FUNCTION CALL'

     ** WE WILL ASSUME THE BRANCH IS NOT TAKEN

B74A BD BC C8               JSR    LBCC8           3    SHIFT BINARY POINT TO
```

```
                                                        EXTREME RIGHT OF FPA0

BCC8 D6 4F          LBCC8   LDB    FP0EXP          2    GET EXPONENT OF FPA0
BCCA 27 3D                  BEQ    LBD09           2    ZERO MANTISSA IF FPA0 = 0

     ** WE WILL ASSUME FPA0 <> 0 AND THUS THE BRANCH
        IS NOT TAKEN.

BCCC C0 A0                  SUBB   #$A0            2    SUBTRACT $A0 FROM FPA0 EXP
BCCE 96 54                  LDA    FP0SGN          2    TEST SIGN OF FPA0 MANTISSA
BCD0 2A 05                  BPL    LBCD7           2    BRANCH IF POSITIVE

     ** WE WILL ASSUME THAT IT IS POSITIVE AND THUS
        THE BRANCH IS TAKEN (POSITIVE IS TO BE
        EXPECTED FOR SCREEN COORDINATES; ALSO FOR
        LEANEST PRESET).

BCD7 8E 00 4F       LBCD7   LDX    #FP0EXP         3    POINT X TO FPA0
BCDA C1 F8                  CMPB   #-8             2    EXPONENT DIFFERENCE < -8?
BCDC 2E 06                  BGT    LBCE4           2    GO IF YES

     ** WE WILL ASSUME THAT IT IS NOT AND THUS
        THE BRANCH IS NOT TAKEN.

BCDE BD BA AE               JSR    LBAAE           3    SHIFT FPA0 RIGHT UNTIL
                                                        FPA0 EXPONENT = $A0

BAAE CB 08          LBAAE   ADDB    #8             2    ADD 8 TO DIFFERENCE OF EXPS
BAB0 2F E8                  BLE     LBA9A          2    BRANCH IF EXP DIFF < -8

     ** WE WILL ASSUME THAT IT IS NOT AND THUS
```

```
        THE BRANCH IS NOT TAKEN.

BAB2 96 63                LDA     FPSBYT          2    GET FPA SUB BYTE
BAB4 C0 08                SUBB    #8              2    CAST OUT 8 ADDED IN ABOVE
BAB6 27 0C                BEQ     LBAC4           2    BRANCH IF EXPONENT DIFF = 0

     ** WE WILL ASSUME THAT IT IS EQUAL TO ZERO
        THE BRANCH IS TAKEN.

BAC4 39           LBAC4   RTS                     1    CALLED FROM BCDE

BCE1 0F 5B                CLR     FPCARY          2    CLEAR CARRY IN BYTE
BCE3 39                   RTS                     1    CALLED FROM B74A

B74D 9E 52                LDX     FPA0+2          2    LOWER TWO BYTES OF FPA0
B74F 39                   RTS                     1    CALLED FROM B734

B736 9F 2B                STX    BINVAL           2    STORE IT IN BINVAL
B738 BD B2 6D             JSR    LB26D            3    SYNTAX CHECK FOR A COMMA

B26D C6 2C                LDB     #',             -    SYNTAX CHECK FOR COMMA
B26F E1 9F 00 A6          CMPB    [CHARAD]        -    COMPARE B TO CURRENT INPUT
B273 26 02                BNE     LB277           -    CHAR: SNTX ERR IF NO MATCH

     ** ASSUMING NO SYNTAX ERROR, THE BRANCH WILL NOT BE TAKEN

B275 0E 9F                JMP     GETNCH          -    GET A CHARACTER FROM BASIC

009F 0C A7          GETNCH  INC     <CHARAD+1       -    INCR INPUT POINTER LOW BYTE
00A1 26 02                BNE     GETCCH          -    BRANCH IF NOT 0 (NO CARRY)
```

```
     ** SINCE OUR PURPOSE HERE IS TO SEE HOW MANY BYTES AND
        CYCLES WE CAN SAVE, WE WILL ASSUME THE BRANCH IS
        TAKEN HERE. THIS WILL ALLOW US TO COMPARE OUR
        PROPOSED IMPROVEMENTS ADGAINST THE LEANEST POSSIBLE
        INTERPRETATION OF THE EXISTING PSET CODE.

00A5 B6          GETCCH  FCB     $B6          -    OP CODE OF LDA EXTENDED
00A6 ?? ??       CHARAD                       -    THESE 2 BYTES CONTAIN ADDR
                                                   OF THE CURRENT CHAR WHICH
                                                   THE BASIC INTERPRETER IS
                                                   PROCESSING
00A8 7E AA 1A            JMP     BROMHK       -    JUMP BACK INTO BASIC ROM

AA1A 81 3A       BROMHK  CMPA    #'9+1        -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A               BHS     LAA28        -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39          LAA28   RTS                  -    CALLED FROM B738

B73B 20 CE               BRA    LB70B         2    EVAL EXPRIN RANGE 0<=X<256

B70B BD B1 41    LB70B   JSR    LB141         2    EVAL A NUMERIC EXPRESSION

*****

B141 8D 13       LB141   BSR     LB156        -    CHECK FOR NUMERIC

B156 8D 6E       LB156   BSR     B1C6         -    BACK UP INPUT POINTER

B1C6 9E A6       LB1C6   LDX     CHARAD       -    GET BASIC'S INPUT POINTER
```

```
B1C8 7E AE BB               JMP     LAEBB          -    AND MOVE IT BACK ONE

AEBB 30 1F          LAEBB   LEAX    -1,X           -    MOV TO JUST BEF STRT OF LINE
AEBD 9F A6                  STX     CHARAD         -    RESET BASIC'S INPUT POINTER
AEBF 39                     RTS                    -    CALLED FROM B156

B158 4F                     CLRA                   -    END OF OP PRECEDENCE FLAG
B159 8C                     FCB     SKP2           -    SKIP 2 BYTES
B15A 34 04                  PSHS    B              -    THIS IS SKIPPER

     ** SINCE 8C IS THE OPCODE FOR CMPX IMMEDIATE,
        THE "PSHS B" IS SKIPPED.

B15C 34 02                  PSHS    A              -    SAVE FLAG (PRECEDENCE FLAG)
B15E C6 01                  LDB     #1             -
B160 BD AC 33               JSR     LAC33          -    FREE RAM ROOM FOR (B) WORDS?

AC33 4F             LAC33   CLRA                   -    ACCD CONTAINS NUMBER OF XTRA
AC34 58                     ASLB                   -    BYTES TO PUT ON STACK
AC35 D3 1F                  ADDD    ARYEND         -    END OF PROGRAM AND VARIABLES
AC37 C3 00 3A               ADDD    #STKBUF        -    ROOM FOR STACK?
AC3A 25 08                  BCS     LAC44          -    BRANCH IF GREATER THAN $FFFF

     ** SINCE > $FFFF WOULD BE AN ERROR, WE WILL
        ASSUME THE BRANCH IS NOT TAKEN.

AC3C 10 DF 17               STS     BOTSTK         -    CUR NEW BOTTOM OF STACK PTR
AC3F 10 93 17               CMPD    BOTSTK         -    WILL WE BE BELOW STACK?
AC42 25 EE                  BCS     LAC32          -    YES - NO ERROR

AC32 39                     RTS                    -    CALLED FROM B160
```

```
B163 BD B2 23              JSR     LB223          -    GO EVALUATE AN EXPRESSION

B223 BD 01 8B      LB223   JSR     RVEC15         -    HOOK INTO RAM

018B 7E CE D2      RVEC15  JMP     $CED2          -    DISK BASIC 1.1 VECTOR

CED2 A6 64         DVEC15  LDA     $04,S          -    CHK STACKED PRECEDENCE FLAG
CED4 26 13                 BNE     LCEE9          -    GO IF NOT END OF OP

     ** WE WILL ASSUME IT IS NOT THE END OF AN OPERATION.
        AND THUS THE BRANCH IS TAKEN (LEANEST PSET)

CEE9 7E 88 46      LCEE9   JMP     XVEC15         -    EXTENDED BASIC EXPR EVAL

8846 35 40         XVEC15  PULS    U              -    PULL RTS ADDRESS; SAVE IN U
8848 0F 06                 CLR     VALTYP         -    SET VARIABLE TYPE TO NUMERIC
884A 9E A6                 LDX     CHARAD         -    CURRENT INPUT POINTER TO X
884C 9D 9F                 JSR     GETNCH         -    GET CHARACTER FROM BASIC

009F 0C A7         GETNCH  INC     <CHARAD+1      -    INCR INPUT POINTER LOW BYTE
00A1 26 02                 BNE     GETCCH         -    BRANCH IF NOT 0 (NO CARRY)

     ** SINCE OUR PURPOSE HERE IS TO SEE HOW MANY BYTES AND
        CYCLES WE CAN SAVE, WE WILL ASSUME THE BRANCH IS
        TAKEN HERE. THIS WILL ALLOW US TO COMPARE OUR
        PROPOSED IMPROVEMENTS ADGAINST THE LEANEST POSSIBLE
        INTERPRETATION OF THE EXISTING PSET CODE.

00A5 B6            GETCCH  FCB     $B6            -    OP CODE OF LDA EXTENDED
00A6 ?? ??         CHARAD                         -    THESE 2 BYTES CONTAIN ADDR
```

```
                                                        OF THE CURRENT CHAR WHICH
                                                        THE BASIC INTERPRETER IS
                                                        PROCESSING
00A8 7E AA 1A              JMP     BROMHK         -     JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK CMPA    #'9+1          -     IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                 BHS     LAA28          -     BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28  RTS                    -     CALLED FROM 884C

884E 81 26                 CMPA    #'&'           -     HEX AND OCTAL?
8850 27 99                 BEQ     L87EB          -     GO IF YES

     ** WE WILL ASSUME IT IS DECIMAL AND THUS THE BRANCH
        IS NOT TAKEN.

8852 81 CC                 CMPA    #$CC           -     FUNCTION CALL TOKEN?
8854 27 5E                 BEQ     L88B4          -     GO IF YES

     ** WE WILL ASSUME IT IS NOT A FUNCTION CALL AND
        THUS THE BRANCH IS NOT TAKEN.

8856 81 FF                 CMPA    #$FF           -     SECONDARY TOKEN?
8858 26 08                 BNE     L8862          -     GO IF NO

     ** WE WILL ASSUME IT IS NOT A SECONDARY FUNCTION
        AND THUS THE BRANCH IS TAKEN.

8862 9F A6          L8862  STX     CHARAD         -     RESTORE BASIC'S INPUT PTR
```

```
8864 6E C4                  JMP     ,U              -    CALLED FROM B223

     ** GIVEN THE "PULS U" AT 8846, THIS "JMP ,U" = RTS

B226 0F 06                  CLR     VALTYP          -    INIT TYPE FLAG TO NUMERIC
B228 9D 9F                  JSR     GETNCH          -    GET AN INPUT CHAR

009F 0C A7          GETNCH  INC     <CHARAD+1       -    INCR INPUT POINTER LOW BYTE
00A1 26 02                  BNE     GETCCH          -    BRANCH IF NOT 0 (NO CARRY)

     ** SINCE OUR PURPOSE HERE IS TO SEE HOW MANY BYTES AND
        CYCLES WE CAN SAVE, WE WILL ASSUME THE BRANCH IS
        TAKEN HERE. THIS WILL ALLOW US TO COMPARE OUR
        PROPOSED IMPROVEMENTS ADGAINST THE LEANEST POSSIBLE
        INTERPRETATION OF THE EXISTING PSET CODE.

00A5 B6             GETCCH  FCB     $B6             -    OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                          -    THESE 2 BYTES CONTAIN ADDR
                                                         OF THE CURRENT CHAR WHICH
                                                         THE BASIC INTERPRETER IS
                                                         PROCESSING
00A8 7E AA 1A               JMP     BROMHK          -    JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK  CMPA    #'9+1           -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28           -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28   RTS                     -    CALLED FROM B228

B22A 24 03                  BCC     LB22F           -    BRANCH IF NOT NUMERIC
```

```
     ** WE WILL ASSUME THAT IT IS NUMERIC AND THUS
        THE BRANCH IS NOT TAKEN

B22C 7E BD 12              JMP     LBD12          -    CONVERT ASCII STRING TO
                                                       FLOATING POINT - RETURN
                                                       RESULT IN FPA0

BD12 9E 8A          LBD12  LDX     ZERO           -    (X) = 0
BD14 9F 54                 STX     FP0SGN         -    ZERO OUT FPA0 & SIGN FLAG
BD16 9F 4F                 STX     FP0EXP         -
BD18 9F 51                 STX     FPA0+1         -
BD1A 9F 52                 STX     FPA0+2         -
BD1C 9F 47                 STX     V47            -    INITIALIZE EXPONENT & SIGN
                                                       FLAG TO ZERO
BD1E 9F 45                 STX     V45            -    INITIALIZE RIGHT DECIMAL CTR
                                                       & DECIMAL PT FLAG TO 0
BD20 25 64                 BCS     LBD86          -    IF CARRY SET (NUMERIC
                                                       CHARACTER), ASSUME ACCA
                                                       CONTAINS FIRST NUMERIC CHAR,
                                                       SIGN IS POSITIVE AND SKIP
                                                       THE RAM HOOK

     ** WE WILL ASSUME A NUMERIC CHARACTER AND THUS THE
        BRANCH IS TAKEN (LEANEST PSET).

BD86 D6 45          LBD86  LDB     V45            -    GET THE RIGHT DECIMAL COUNTER
BD88 D0 46                 SUBD    V46            -    AND SUBTRACT DECIMAL PNT FLAG
BD8A D7 45                 STB     V45            -
BD8C 34 02                 PSHS    A              -    SAVE NEW DIGIT ON STACK
BD8E BD BB 6A              JSR     LBB6A          -    MULTIPLY FPA0 BY 10
```

```
BB6A BD BC 5F       LBB6A   JSR     LBC5F           -    TRANSFER FPA0 TO FPA1

BC5F DC 4F          LBC5F   LDD     FP0EXP          -    TRANSFER EXPONENT & MS BYTE
BC61 DD 5C                  STD     FP1EXP          -
BC63 9E 51                  LDX     FPA0+1          -    TRANSFER MIDDLE TWO BYTES
BC65 9F 5E                  STX     FPA1+1          -
BC67 9E 53                  LDX     FPA0+3          -    TRANSFER BOTTOM TWO BYTES
BC69 9F 60                  STX     FPA1+3          -
BC6B 4D                     TSTA                    -    SET FLAGS PER EXPONENT
BC6C 39                     RTS                     -    CALLED FROM BB6A

BB6D 27 0D                  BEQ     LBB7C           -    BRANCH IF EXPONENT = 0

     ** WE WILL ASSUME THE EXPONENT = 0 (TO BE EXPECTED
        FOR A SCREEN COORDINATE; ALSO FOR LEANEST PSET),
        AND THUS THE BRANCH IS TAKEN

BB7C 39                     RTS                     -    CALLED FROM BD8E

BD91 35 04                  PULS    B               -    GET NEW DIGIT BACK
BD93 C0 30                  SUBB    #'0             -    MASK OFF ASCII
BD95 8D 02                  BSR     LBD99           -    ADD ACCB TO FPA0

BD99 BD BC 2F       LBD99   JSR     LBC2F           -    PACK FPA0 AND SAVE IN FPA3

BC2F 8E 00 40       LBC2F   LDX     #V40            -    POINT X TO MANTISSA OF FPA3
BC32 8C                     FCB     SKP2            -    SKIP TWO BYTES
BC33 9E 3B                  LDX     VARDES          -    POINT X TO VAR DESCRIPTOR

     ** SINCE 8C IS THE OPCODE FOR CMPX IMMEDIATE,
```

THE "LDX VARDES" IS SKIPPED.

```
BC35 96 4F                LDA     FP0EXP          -    COPY EXPONENT
BC37 A7 84                STA     ,X              -
BC39 96 54                LDA     FP0SGN          -    GET MANTISSA SIGN BIT
BC3B 8A 7F                ORA     #$7F            -    MASK THE BOTTOM 7 BITS
BC3D 94 50                ANDA    FPA0            -    AND BIT 7 OF MANTISSA SIGN
                                                       INTO BIT 7 OF MS BYTE
BC3F A7 01                STA     1,X             -    SAVE MS BYTE
BC41 96 51                LDA     FPA0+1          -    MOVE 2ND MANTISSA BYTE
BC43 A7 02                STA     2,X             -
BC45 DE 52                LDU     FPA0+2          -    MOVE BOTTOM 2 MANTISSA BYTES
BC47 EF 03                STU     3,X             -
BC49 39                   RTS                     -    CALLED FROM BD99

BD9C BD BC 7C             JSR     LBC7C           -    CONVERT B TO FP NUM IN FPA0

BC7C D7 50          LBC7C STB     FPA0            -    SAVE ACCB IN FPA0
BC7E 0F 51                CLR     FPA0+1          -    CLEAR 2ND MANTISSA FPA0 BYTE
BC80 C6 88                LDB     #$88            -    EXPONENT FOR FPA0 BE INTEGER
BC82 96 50                LDA     FPA0            -    GET MS BYTE OF MANTISSA
BC84 80 80                SUBA    #$80            -    SET CARRY IF POSITIVE MTSSA
BC86 D7 4F                STB     FP0EXP          -    SAVE EXPONENT
BC88 DC 8A                LDD     ZERO            -    ZERO OUT ACCD AND
BC8A DD 52                STD     FPA0+2          -    BOTTOM HALF OF FPA0
BC8C 97 63                STA     FPSBYT          -    CLEAR SUB BYTE
BC8E 97 54                STA     FP0SGN          -    CLEAR SIGN OF FPA0 MANTISSA
BC90 7E BA 18             JMP     LBA18           -    GO NORMALIZE FPA0

BA18 25 02          LBA18 BCS     LBA1C           -    BRANCH IF POSITIVE MANTISSA
```

```
     ** WE WILL ASSUME THAT THE MANTISSA IS POSITIVE
        AND THUS THE BRANCH IS TAKEN.

BA1C 5F             LBA1C   CLRB                    -    CLEAR TEMP EXPONENT ACCUM
BA1D 96 50                  LDA     FPA0            -    TEST MSB OF MANTISSA
BA1F 26 2E                  BNE     LBA4F           -    BRANCH IF <> 0

     ** WE WILL ASSUME THAT THE MSB IS <> 0 AND THUS
        THE BRANCH IS TAKEN (LEANEST PSET).

BA4F 2A F3          LBA4F   BPL     LBA44           -    BRANCH IF NOT YET NORMALIZED

     ** WE WILL ASSUME THAT IT IS ALREADY NORMALIZED AND THUS
        THE BRANCH IS NOT TAKEN (LEANEST PSET).

BA51 96 4F                  LDA     FP0EXP          -    GET CURRENT EXPONENT
BA53 34 04                  PSHS    B               -    SAVE EXPONENT MODIFIER
BA55 A0 E0                  SUBA    ,S+             -    SUBTRACT EXPONENT MODIFIER
                                                         AND CLEAR IT OFF THE STACK
BA57 97 4F                  STA     FP0EXP          -    SAVE AS NEW EXPONENT
BA59 23 DE                  BLS     LBA39           -    SET FPA0 = 0 IF THE
                                                         NORMALIZATION CAUSED MORE
                                                         OR EQUAL NUMBER OF LEFT
                                                         SHIFTS THAN THE SIZE OF THE
                                                         EXPONENT

     ** WE WILL ASSUME THIS DIDN'T HAPPEN AND THUS
        THE BRANCH IS NOT TAKEN (LEANEST PSET).

BA5B 8C                     FCB     SKP2            -    SKIP 2 BYTES
BA5C 25 08                  BCS     LBA66           -    BRANCH IF MANTISSA OVERFLOW
```

```
     ** SINCE 8C IS THE OPCODE FOR CMPX IMMEDIATE,
        THE "BCS LBA66" IS SKIPPED.

BA5E 08 63                ASL     FPSBYT          -    SUB BYTE BIT 7 TO CARRY
BA60 86 00                LDA     #0              -    CLRA DON'T CHANGE CARRY FLAG
BA62 97 63                STA     FPSBYT          -    CLEAR THE SUB BYTE
BA64 20 0C                BRA     LBA72           -    GO ROUND-OFF RESULT

BA72 24 04         LBA72  BCC     LBA78           -    BRANCH IF NO ROUNDOFF NEEDED

     ** WE WILL ASSUME NO ROUNDOFF IS NEEDED AND THUS
        THE BRANCH IS TAKEN (LEANEST PSET).

BA78 39            LBA78  RTS                     -    CALLED FROM BD9C

BD9F 8E 00 40             LDX     #V40            -    ADD FPA0 TO FPA3
BDA2 7E B9 C2             JMP     LB9C2           -

B9C2 BD BB 2F      LB9C2  JSR     LBB2F           -    UNPACK PACKED FP DATA

BB2F EC 01         LBB2F  LDD     1,X             -    GET TWO MSB BYTES
BB31 97 61                STA     FP1SGN          -    SAVE MANTISSA SIGN BYTE
BB33 8A 80                ORA     #$80            -    FORCE BIT 7 OF MANTISSA = 1
BB35 DD 5D                STD     FPA1            -    SAVE 2 MSB BYTES IN FPA1
BB37 D6 61                LDB     FP1SGN          -    GET PACKED MANTISSA SGN BYTE
BB39 D8 54                EORB    FP0SGN          -    XOR FPA0 SIGN BYTE
BB3B D7 62                STB     RESSGN          -    SAVE ADJUSTED SIGN BYTE
BB3D EC 03                LDD     3,X             -    GET 2 LSB BYTES OF MANTISSA
BB3F DD 5F                STD     FPA1+2          -    AND PUT IN FPA1
BB41 A6 84                LDA     ,X              -    GET EXPONENT FROM (X) AND
```

```
BB43 97 5C                 STA     FP1EXP          -    PUT IN EXPONENT OF FPA1
BB45 D6 4F                 LDB     FP0EXP          -    GET EXPONENT OF FPA0
BB47 39                    RTS                     -    CALLED FROM B9C2

B9C5 5D                    TSTB                    -    CHECK EXPONENT OF FPA0
B9C6 10 27 02 80           LBEQ    LBC4A           -    GO IF FPA0 = 0

     ** WE WILL ASSUME FPA0 = 0 (LEANEST PSET)
        AND THUS THE BRANCH IS TAKEN.

BC4A 96 61          LBC4A  LDA     FP1SGN          -    MOVE FPA1 TO FPA0
BC4C 97 54          LBC4C  STA     FP0SGN          -
BC4E 9E 5C                 LDX     FP1EXP          -
BC50 9F 4F                 STX     FP0EXP          -
BC52 0F 63                 CLR     FPSBYT          -
BC54 96 5E                 LDA     FPA1+1          -
BC56 97 51                 STA     FPA0+1          -
BC58 96 54                 LDA     FP0SGN          -
BC5A 9E 5F                 LDX     FPA1+2          -
BC5C 9F 52                 STX     FPA0+2          -
BC5E 39                    RTS                     -    CALLED FROM BD95

BD97 20 98                 BRA     LBD31           -    GET ANOTHER CHAR FROM BASIC

BD31 9D 9F          LBD31  JSR     GETNCH          -    GET NEXT INPUT CHARACTER

009F 0C A7          GETNCH INC     <CHARAD+1       -    INCR INPUT POINTER LOW BYTE
00A1 26 02                 BNE     GETCCH          -    BRANCH IF NOT 0 (NO CARRY)

     ** SINCE OUR PURPOSE HERE IS TO SEE HOW MANY BYTES AND
        CYCLES WE CAN SAVE, WE WILL ASSUME THE BRANCH IS
```

```
        TAKEN HERE. THIS WILL ALLOW US TO COMPARE OUR
        PROPOSED IMPROVEMENTS ADGAINST THE LEANEST POSSIBLE
        INTERPRETATION OF THE EXISTING PSET CODE.

00A5 B6             GETCCH  FCB     $B6             -    OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                          -    THESE 2 BYTES CONTAIN ADDR
                                                         OF THE CURRENT CHAR WHICH
                                                         THE BASIC INTERPRETER IS
                                                         PROCESSING
00A8 7E AA 1A               JMP     BROMHK          -    JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK  CMPA    #'9+1           -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28           -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28   RTS                     -    CALLED FROM BD31

BD33 25 51                  BCS     LBD86           -    BRANCH IF NUMERIC CHARACTER

     ** HAVING BEEN THROUGH LBD86 BEFORE, WE WILL ASSUME
        THAT IT IS NOT A NUMERIC CHARACTER AND THUS THE
        BRANCH IS NOT TAKEN (LEANEST PSET).

BD35 81 2E          LBD35   CMPA    #'.             -    DECIMAL POlNT?
BD37 27 28                  BEQ     LBD61           -    GO IF YES

     ** WE WILL ASSUME THAT IT IS NOT A DECIMAL POINT
        AND THUS THE BRANCH IS NOT TAKEN.

BD39 81 45                  CMPA    #'E             -    "E" (SCIENTIFIC NOTATION)?
```

```
BD3B 26 28                BNE     LBD65          -    GO IF NO

     ** WE WILL ASSUME IT IS NOT SCIENTIFIC NOTATION
        AND THUS THE BRANCH IS TAKEN.

BD65 96 47          LBD65 LDA     V47            -    GET EXPONENT, SUBTRACT THE
BD67 90 45                SUBA    V45            -    NUMBER OF PLACES TO RIGHT
BD69 97 47                STA     V47            -    OF DECIMAL POINT AND RESAVE
BD6B 27 12                BEQ     LBD7F          -    EXIT IF ADJUSTED EXPONENT= 0

     ** WE WILL ASSUME THE ADJUSTED EXPONENT IS ZERO
        AND THUS THE BRANCH IS TAKEN.

BD7F 96 55          LBD7F LDA     COEFCT         -    GET THE SIGN FLAG
BD81 2A 8E                BPL     LBD11          -    RETURN IF POSITIVE

     ** WE WILL ASSUME THE RESULT IS NOT POSITIVE AND
        THUS THE BRANCH IS NOT TAKEN (LEANEST PSET).

BD83 7E BE E9             JMP     LBEE9          -    TOGGLE MANTISSA SIGN OF FPA0

BEE9 96 4F          LBEE9 LDA     FP0EXP         -    GET EXPONENT OF FPA0
BEEB 27 02                BEQ     LBEEF          -    BRANCH IF FPA0 = 0

     ** WE WILL ASSUME FPA0 <>0 AND THUS THE BRANCH
        IS NOT TAKEN.

BEED 03 54                COM     FP0SGN         -    TOGGLE MANTISSA SIGN OF FPA0
BEEF 39             LBEEF RTS                    -    CALLED FROM B163

B166 0F 3F                CLR     TRELFL         -    RESET RELATIONAL OP FLAG
```

```
B168 9D A5                  JSR    GETCCH          -    GET CURRENT INPUT CHARACTER

00A5 B6            GETCCH   FCB     $B6            -    OP CODE OF LDA EXTENDED
00A6 ?? ??         CHARAD                          -    THESE 2 BYTES CONTAIN ADDR
                                                        OF THE CURRENT CHAR WHICH
                                                        THE BASIC INTERPRETER IS
                                                        PROCESSING
00A8 7E AA 1A               JMP     BROMHK         -    JUMP BACK INTO BASIC ROM

AA1A 81 3A         BROMHK   CMPA    #'9+1          -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28          -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39            LAA28    RTS                    -    CALLED FROM B168

B16A 80 B2                  SUBA    #$B2           -    TOKEN FOR >
B16C 25 13                  BCS     LB181          -    BRANCH IF < RELATIONAL OPS

     ** WE WILL ASSUME THAT IT IS LESS AND THUS THE
        BRANCH IS TAKEN (LEANEST PSET)

B181 D6 3F         LB181    LDB     TRELFL         -    GET RELATIONAL OPERATOR FLAG
B183 26 33                  BNE     LB1B8          -    BRANCH IF RELATIONAL COMPARISON

     ** WE WILL ASSUME THAT IT IS NOT A RELATIONAL COMPARISON
        AND THUS THE BRANCH IS NOT TAKEN (LEANEST PSET).

B185 10 24 00 6B            LBCC    LB1F4          -    BRANCH IF > RELATIONAL OP

     ** WE WILL ASSUME THAT IT IS > RELATIONAL COMPARISON
```

```
        AND THUS THE BRANCH IS TAKEN (LEANEST PSET).

B1F4 9E 8A          LB1F4    LDX    ZERO          -    POINT X TO DUMMY VALUE (0)
B1F6 A6 E0                   LDA    ,S+           -    GET PRECEDENCE FLAG FROM STK
B1F8 27 26                   BEQ    LB220         -    BRANCH IF END OF EXPRESSION

     ** WE WILL ASSUME THAT IT IS END OF EXPRESSION AND
        THE BRANCH IS TAKEN (LEANEST PSET).

B220 D6 4F          LB220    LDB    FP0EXP        -    GET EXPONENT OF FPA0
B222 39                      RTS                  -    CALLED FROM B141

B143 1C FE          LB143    ANDCC  #$FE          -    CLEAR CARRY FLAG
B145 7D                      FCB    $7D           -    OP CODE OF TST $1A01
                                                       SKIP TWO BYTES (DO NOT
                                                       CHANGE CARRY FLAG)
B146 1A 01                   ORCC   #1            -    SET CARRY
B148 0D 06                   TST    VALTYP        -    TEST TYPE FLAG
B14A 25 03                   BCS    LB14F         -    BRANCH IF STRING

     ** WE WILL ASSUME IT IS NOT A STRING AND THUS
        THE BRANCH IS NOT TAKEN

B14C 2A 99                   BPL    LB0E7         -    RETURN ON PLUS

     ** WE WILL ASSUME THAT IT IS POSITIVE AND THUS
        THE BRANCH IS TAKEN (POSITIVE IS TO BE
        EXPECTED FOR SCREEN COORDINATES; ALSO FOR
        LEANEST PRESET).

B0E7 39             LB0E7    RTS                  -    CALLED FROM B70B
```

```
*****

B70E BD B3 E9               JSR    LB3E9          3    CONVERT FPA0 TO INTEGER IN ACCD

B3E9 96 54          LB3E9   LDA    FP0SGN         2    GET FPA0 MANTISSA SIGN
B3EB 2B 5D                  BMI    LB44A          2    'FC' ERR IF NEGATIVE NUMBER

     ** WE ASSUME NO ERROR AND THUS THE BRANCH
        IS NOT TAKEN.

B3ED BD B1 43       INTCNV  JSR    LB143          3    'TM' ERROR IF STRING VARIABLE

B143 1C FE          LB143   ANDCC  #$FE           -    CLEAR CARRY FLAG
B145 7D                     FCB    $7D            -    OP CODE OF TST $1A01
                                                        SKIP TWO BYTES (DO NOT
                                                        CHANGE CARRY FLAG)
B146 1A 01                  ORCC   #1             -    SET CARRY
B148 0D 06                  TST    VALTYP         -    TEST TYPE FLAG
B14A 25 03                  BCS    LB14F          -    BRANCH IF STRING

     ** WE WILL ASSUME IT IS NOT A STRING AND THUS
        THE BRANCH IS NOT TAKEN

B14C 2A 99                  BPL    LB0E7          -    RETURN ON PLUS

     ** WE WILL ASSUME THAT IT IS POSITIVE AND THUS
        THE BRANCH IS TAKEN (POSITIVE IS TO BE
        EXPECTED FOR SCREEN COORDINATES; ALSO FOR
        LEANEST PRESET).
```

```
B0E7 39            LB0E7   RTS                     -    CALLED FROM B3ED

B3F0 96 4F                 LDA    FP0EXP           2    GET FPA0 EXPONENT
B3F2 81 90                 CMPA   #$90             2    COMPARE TO 32768
                                                       LARGEST INTEGER EXPONENT AND
B3F4 25 08                 BCS    LB3FE            2    BRANCH IF FPA0 < 32768

     ** WE ASSUME THAT THE EXPONENT IS LESS THAN 32768
        AND THUS THE BRANCH IS TAKEN.

B3FE BD BC C8      LB3FE   JSR    LBCC8            3    CONVERT FPA0 TO A 2-BYTE INT

BCC8 D6 4F         LBCC8   LDB    FP0EXP           -    GET EXPONENT OF FPA0
BCCA 27 3D                 BEQ    LBD09            -    ZERO MANTISSA IF FPA0 = 0

     ** WE WILL ASSUME FPA0 <> 0 AND THUS THE BRANCH
        IS NOT TAKEN.

BCCC C0 A0                 SUBB   #$A0             -    SUBTRACT $A0 FROM FPA0 EXP
BCCE 96 54                 LDA    FP0SGN           -    TEST SIGN OF FPA0 MANTISSA
BCD0 2A 05                 BPL    LBCD7            -    BRANCH IF POSITIVE

     ** WE WILL ASSUME THAT IT IS POSITIVE AND THUS
        THE BRANCH IS TAKEN (POSITIVE IS TO BE
        EXPECTED FOR SCREEN COORDINATES; ALSO FOR
        LEANEST PRESET).

BCD7 8E 00 4F      LBCD7   LDX    #FP0EXP          -    POINT X TO FPA0
BCDA C1 F8                 CMPB   #-8              -    EXPONENT DIFFERENCE < -8?
BCDC 2E 06                 BGT    LBCE4            -    GO IF YES
```

```
     ** WE WILL ASSUME THAT IT IS NOT AND THUS
        THE BRANCH IS NOT TAKEN.

BCDE BD BA AE               JSR    LBAAE          -    SHIFT FPA0 RIGHT UNTIL
                                                       FPA0 EXPONENT = $A0

BAAE CB 08          LBAAE   ADDB    #8            -    ADD 8 TO DIFFERENCE OF EXPS
BAB0 2F E8                  BLE     LBA9A         -    BRANCH IF EXP DIFF < -8

     ** WE WILL ASSUME THAT IT IS NOT AND THUS
        THE BRANCH IS NOT TAKEN.

BAB2 96 63                  LDA     FPSBYT        -    GET FPA SUB BYTE
BAB4 C0 08                  SUBB    #8            -    CAST OUT 8 ADDED IN ABOVE
BAB6 27 0C                  BEQ     LBAC4         -    BRANCH IF EXPONENT DIFF = 0

     ** WE WILL ASSUME THAT IT IS EQUAL TO ZERO
        THE BRANCH IS TAKEN.

BAC4 39             LBAC4   RTS                   -    CALLED FROM B3FE

B401 DC 52                  LDD    FPA0+2         2    GET THE INTEGER
B403 39                     RTS                   1    CALLED FROM B70E

B711 4D                     TSTA                  1    TEST MS BYTE OF INTEGER
B712 26 F2                  BNE    LB706          2    'FC' ERROR IF EXPR > 255

     ** WE ASSUME NO ERROR AND THUS THE BRANCH
        IS NOT TAKEN.

B714 0E A5                  JMP    GETCCH         2    GET CURRENT INPUT CHARACTER
```

```
00A5 B6             GETCCH  FCB     $B6             -    OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                          -    THESE 2 BYTES CONTAIN ADDR
                                                         OF THE CURRENT CHAR WHICH
                                                         THE BASIC INTERPRETER IS
                                                         PROCESSING
00A8 7E AA 1A               JMP     BROMHK          -    JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK  CMPA    #'9+1           -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28           -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28   RTS                     -    CALLED FROM B714

B716 BD B6 86       VAL     JSR    LB686            3    POINT X TO STRING ADDRESS

B686 8D CC          LB686   BSR    LB654            2    GET STR LENGTH AND ADDRESS

B654 BD B1 46       LB654   JSR    LB146            3    'TM' ERR IF VAR TYPE=NUMERIC

B146 1A 01                  ORCC   #1              -    SET CARRY
B148 0D 06                  TST    VALTYP           -    TEST TYPE FLAG
B14A 25 03                  BCS    LB14F            -    BRANCH IF STRING

     ** WE WILL ASSUME IT IS NOT A STRING AND THUS
        THE BRANCH IS NOT TAKEN

B14C 2A 99                  BPL    LB0E7            -    RETURN ON PLUS

     ** WE WILL ASSUME THAT IT IS POSITIVE AND THUS
```

```
        THE BRANCH IS TAKEN (POSITIVE IS TO BE
        EXPECTED FOR SCREEN COORDINATES; ALSO FOR
        LEANEST PRESET).

B0E7 39            LB0E7   RTS                 -    CALLED FROM B654

B657 9E 52                 LDX    FPA0+2       2    GET ADDR OF SELCTD STR DESCR
B659 E6 84                 LDB    ,X           2    GET LENGTH OF STRING
B65B 8D 18                 BSR    LB675        2    STRING DESCR LAST ON STACK?

B675 9C 0D         LB675   CMPX   LASTPT       2    COMPARE TO LAST DESCR ADDR
B677 26 07                 BNE    LB680        2    ON THE STRING STACK, RETURN
                                                     IF DESCRIPTOR

     ** WE WILL ASSUME THAT IT IS AND THUS THE BRANCH
        IS TAKEN (LEANEST PSET).

B680 39            LB680   RTS                 1    CALLED FROM B65B

B65D 26 13                 BNE    LB672        2    GO IF STR DESCR LAST ON STK?

     ** WE WILL ASSUME THAT IT IS THE LAST ON THE STACK
        AND THUS THE BRANCH IS TAKEN (LEANEST PSET).

B672 AE 02         LB672   LDX    2,X          2    PNT X TO ADDR OF STR NOT ON
                                                     STRING STACK
B674 39                    RTS                 1    CALLED FROM B686

B688 0F 06                 CLR    VALTYP       2    SET VARIABLE TYPE TO NUMERIC
B68A 5D                    TSTB                1    SET FLAGS ACCORDING TO LNGTH
B68B 39                    RTS                 1    CALLED FROM B716
```

```
B719 10 27 03 1C            LBEQ   LBA39          4    IF NULL STRING SET FPA0

     ** IT DOES NOT SEEM REASONABLE TO THINK THAT THERE
        WOULD BE A NULL STRING AT THIS POINT. tHEREFORE,
        WE WILL ASSUME THAT THE BRANCH IS NOT TAKEN.

B71D DE A6                  LDU    CHARAD         2    SAVE INPUT POINTER IN REG U
B71F 9F A6                  STX    CHARAD         2    PNT INPT PTR TO ADDR OF STR
B721 3A                     ABX                   1    MOVE PTR TO END OF STR TERM
B722 A6 84                  LDA    ,X             2    GET LAST BYTE OF STRING
B724 34 52                  PSHS   U,X,A          2    SAVE INP PTR, STR TERMINATOR
                                                        ADDRESS AND CHARACTER
B726 6F 84                  CLR    ,X             2    CLEAR STRING TERMINATOR :
                                                        FOR ASCII - FP CONVERSION
B728 9D A5                  JSR    GETCCH         2    GET CURRENT CHARACTER

00A5 B6           GETCCH    FCB     $B6           -    OP CODE OF LDA EXTENDED
00A6 ?? ??        CHARAD                          -    THESE 2 BYTES CONTAIN ADDR
                                                        OF THE CURRENT CHAR WHICH
                                                        THE BASIC INTERPRETER IS
                                                        PROCESSING
00A8 7E AA 1A               JMP     BROMHK        -    JUMP BACK INTO BASIC ROM

AA1A 81 3A        BROMHK    CMPA    #'9+1         -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28         -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39           LAA28     RTS                   -    CALLED FROM B728
```

```
B72A BD BD 12       JSR     LBD12                3    CONVERT AN ASCII STR TO FP

BD12 9E 8A          LBD12   LDX     ZERO         -    (X) = 0
BD14 9F 54                  STX     FP0SGN       -    ZERO OUT FPA0 & SIGN FLAG
BD16 9F 4F                  STX     FP0EXP       -
BD18 9F 51                  STX     FPA0+1       -
BD1A 9F 52                  STX     FPA0+2       -
BD1C 9F 47                  STX     V47          -    INITIALIZE EXPONENT & SIGN
                                                      FLAG TO ZERO
BD1E 9F 45                  STX     V45          -    INITIALIZE RIGHT DECIMAL CTR
                                                      & DECIMAL PT FLAG TO 0
BD20 25 64                  BCS     LBD86        -    IF CARRY SET (NUMERIC
                                                      CHARACTER), ASSUME ACCA
                                                      CONTAINS FIRST NUMERIC CHAR,
                                                      SIGN IS POSITIVE AND SKIP
                                                      THE RAM HOOK

     ** WE WILL ASSUME A NUMERIC CHARACTER AND THUS THE
        BRANCH IS TAKEN (LEANEST PSET).

BD86 D6 45          LBD86   LDB     V45          -    GET THE RIGHT DECIMAL COUNTER
BD88 D0 46                  SUBD    V46          -    AND SUBTRACT DECIMAL PNT FLAG
BD8A D7 45                  STB     V45          -
BD8C 34 02                  PSHS    A            -    SAVE NEW DIGIT ON STACK
BD8E BD BB 6A               JSR     LBB6A        -    MULTIPLY FPA0 BY 10

BB6A BD BC 5F       LBB6A   JSR     LBC5F        -    TRANSFER FPA0 TO FPA1

BC5F DC 4F          LBC5F   LDD     FP0EXP       -    TRANSFER EXPONENT & MS BYTE
BC61 DD 5C                  STD     FP1EXP       -
BC63 9E 51                  LDX     FPA0+1       -    TRANSFER MIDDLE TWO BYTES
```

```
BC65 9F 5E                  STX     FPA1+1          -
BC67 9E 53                  LDX     FPA0+3          -    TRANSFER BOTTOM TWO BYTES
BC69 9F 60                  STX     FPA1+3          -
BC6B 4D                     TSTA                    -    SET FLAGS PER EXPONENT
BC6C 39                     RTS                     -    CALLED FROM B72A

B72D 35 52                  PULS    A,X,U           2    RESTORE CHARACTERS AND PTRS
B72F A7 84                  STA     ,X              2    REPLACE STRING TERMINATOR
B731 DF A6                  STU     CHARAD          2    RESTORE INPUT CHARACTER
B733 39                     RTS                     1    CALLED FROM 92FC

92FF 10 8E 00 BD            LDY     #HORBEG         4    POINT Y TO TEMP STORAGE LOC
9303 C1 C0                  CMPB    #192            2    IS VERT COORD > 191?
9305 25 02                  BLO     L9309           2    GO IF NO

     ** WE WILL ASSUME THAT THE VERTICAL Y-COORDINATE
        IS IN RANGE AND THUS THE BRANCH IS TAKEN.

9309 4F             L9309   CLRA                    1    HIGH ORDER BYTE OF VER COORD
930A ED 22                  STD     $02,Y           2    SAVE VERTICAL COORDINATE
930C DC 2B                  LDD     BINVAL          2    GET RAW HORIZONTAL COORD
930E 10 83 01 00            CMPD    #256            4    IS IT WITHIN RANGE?
9312 25 03                  BLO     L9317           2    GO IF YES

     ** WE WILL ASSUME THAT THE HORIZONTAL X-COORDINATE
        IS IN RANGE AND THUS THE BRANCH IS TAKEN.

9317 ED A4          L9317   STD     ,Y              2    SAVE HORIZONTAL COORDINATE
9319 39                     RTS                     1    CALLED FROM 931A

931D CE 00 BD               LDU     #HORBEG         3    POINT U TO HOR & VER COORDS
```

```
9320 96 B6                  LDA     PMODE           2    GET PHODE
9322 81 02                  CMPA    #$02            2    CHECK MODE
9324 24 06                  BCC     L932C           2    BRANCH IF > 1

     ** FOR OUR "LEANEST PSET" PURPOSES, WE WILL ASSUME
        THAT PMODE > 1 AND THUS THE BRANCH IS TAKEN.

        (ALTHOUGH AT OUR CURRENT LINE COUNT OF 1306, THE
         TERM "LEANEST" MAY SEEM A BIT IRRELEVANT).
         [OR AT LEAST IRREVERENT] :-)

932C 96 B6          L932C   LDA     PMODE           2    GET PMODE
932E 81 04                  CMPA    #$04            2    CHECK PMODE
9330 24 06                  BCC     L9338           2    BRANCH IF PMODE = 4

     ** AGAIN (LEANEST PSET) WE WILL ASSUME THE BRANCH
        IS TAKEN.

9338 39             L9338   RTS                     1    CALLED FROM 936B

936E BD 95 81               JSR     L9581           3    EVALUATE COLOR
                                                         RETURN IN WCOLOR; ALLCOL

9581 BD 95 9A       L9581   JSR     L959A           3    GET THE COLOR OF A BYTE

959A D6 B2          L959A   LDB     FORCOL          2    GET FOREGROUND COLOR
959C 0D C2                  TST     SETFLG          2    CHECK PSET/PRESET FLAG
959E 26 02                  BNE     L95A2           2    BRANCH IF PSET

     ** SINCE WE ARE UNWINDING PSET HERE, THE BRANCH
        WILL DEFINITELY BE TAKEN.
```

```
95A2 D7 B4          L95A2   STB     WCOLOR          2    TEMP STORE COLOR
95A4 86 55                  LDA     #$55            2    CONSIDER A BYTE AS 4 PIXELS
95A6 3D                     MUL                     1    SET COLOR ON ALL 4 PIXElS
95A7 D7 B5                  STB     ALLCOL          2    SAVE BYTE WITH ALL PIXELS
                                                         TURNED ON
95A9 39                     RTS                     1    CALLED FROM 9581

9584 9D A5                  JSR     GETCCH          2    CHECK CURRENT INPUT CHAR

00A5 B6             GETCCH  FCB     $B6             -    OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                          -    THESE 2 BYTES CONTAIN ADDR
                                                         OF THE CURRENT CHAR WHICH
                                                         THE BASIC INTERPRETER IS
                                                         PROCESSING
00A8 7E AA 1A               JMP     BROMHK          -    JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK  CMPA    #'9+1           -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28           -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28   RTS                     -    CALLED FROM 9584

9586 27 10                  BEQ     L9598           2    BRANCH IF NONE

     ** WE WILL ASSUME THAT THERE IS NONE AT THIS POINT
        (LEANEST PSET) AND THUS THE BRANCH IS TAKEN.

9598 0E A5          L9598   JMP     GETCCH          2    CHECK INPUT CHAR & RETURN
```

```
00A5 B6             GETCCH  FCB     $B6             -    OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                          -    THESE 2 BYTES CONTAIN ADDR
                                                         OF THE CURRENT CHAR WHICH
                                                         THE BASIC INTERPRETER IS
                                                         PROCESSING
00A8 7E AA 1A               JMP     BROMHK          -    JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK  CMPA    #'9+1           -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                  BHS     LAA28           -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28   RTS                     -    CALLED FROM 936E

9371 BD B2 67               JSR     LB267           3    SYNTAX CHECK FOR `)`

B267 C6 29          LB267   LDB     #')             2    SYNTAX CHECK FOR `)`
B269 8C                     FCB     SKP2            1    SKIP 2 BYTES
B26A C6 28                  LDB     #'(             2    SYNTAX CHECK FOR `(`

     ** "8C C6 28" IS SKIPPED

B26C 8C                     FCB     SKP2            1    SKIP 2 BYTES
B26D C6 2C                  LDB     #',             2    SYNTAX CHECK FOR COMMA

     ** "8C C6 2C" IS SKIPPED

B26F E1 9F 00 A6            CMPB    [CHARAD]        4    COMPARE ACCB TO CURR INPUT
B273 26 02                  BNE     LB277           2    CHARACTER - SYNTAX ERROR
                                                         IF NO MATCH
```

```
     ** WE WILL ASSUME THERE IS A MATCH AND THUS THE
        BRANCH IS NOT TAKEN.

B275 0E 9F                 JMP     GETNCH          2    GET A CHARACTER

009F 0C A7          GETNCH INC     <CHARAD+1       -    INCR INPUT POINTER LOW BYTE
00A1 26 02                 BNE     GETCCH          -    BRANCH IF NOT 0 (NO CARRY)

     ** SINCE OUR PURPOSE HERE IS TO SEE HOW MANY BYTES AND
        CYCLES WE CAN SAVE, WE WILL ASSUME THE BRANCH IS
        TAKEN HERE. THIS WILL ALLOW US TO COMPARE OUR
        PROPOSED IMPROVEMENTS ADGAINST THE LEANEST POSSIBLE
        INTERPRETATION OF THE EXISTING PSET CODE.

00A5 B6             GETCCH FCB     $B6             -    OP CODE OF LDA EXTENDED
00A6 ?? ??          CHARAD                         -    THESE 2 BYTES CONTAIN ADDR
                                                        OF THE CURRENT CHAR WHICH
                                                        THE BASIC INTERPRETER IS
                                                        PROCESSING
00A8 7E AA 1A              JMP     BROMHK          -    JUMP BACK INTO BASIC ROM

AA1A 81 3A          BROMHK CMPA    #'9+1           -    IS THIS CHAR >=(ASCII 9)+1?
AA1C 24 0A                 BHS     LAA28           -    BRANCH IF > 9

     ** WE WILL ASSUME THE BRANCH IS TAKEN (LEANEST PSET)

AA28 39             LAA28  RTS                    -    CALLED FROM 9371

9374 BD 92 98              JSR    L9298           3    CALCULATE THE ABSOLUTE ADDR
                                                        OF THE BYTE TO PSET/PRESET
                                                        RETURN ADDRESS IN X - THE
```

```
                                                    MASK OF PIXEL TO CHANGE
                                                    RETURNED IN ACCA SET A
                                                    PIXEL ON SCREEN - ABS POSIT
                                                    IN X, MASK IN ACCA, COLOR
                                                    IN ALLCOL

9298 8D F5          L9298   BSR     L928F          2    GO GET JUMP ADDRESS

928F CE 92 9C       L928F   LDU     #L929C         3    JUMP TABLE ADDRESS TO U
9292 96 B6                  LDA     PMODE          2    GET PMODE VALUE
9294 48                     ASLA                   1    MUL ACCA X2: 2 BYTES PER ADR
9295 EE C6                  LDU     A,U            2    GET JUMP ADDRESS
9297 39                     RTS                    1    CALLED FROM 9298

929A 6E C4                  JMP     ,U             2    GO TO IT

     ** FOR LEANEST PSET, WE WILL ASSUME THAT PMODE = 4
        AND THUS #L929C + 8 = #L92A4 AND, SINCE 92A4 IS:

          92A4 92 A6   L92A4   FDB     L92A6      2    PMODE 4

        THE JUMP IS MADE TO L92A6:

92A6 34 44          L92A6   PSHS    U,B            2    SAVE REGISTERS
92A8 D6 B9                  LDB     HORBYT         2    GET NUMBER BYTES PER
                                                         HOR GRAPHIC ROW
92AA 96 C0                  LDA     VERBEG+1       2    GET VERTICAL COORDINATE
92AC 3D                     MUL                    1    CALC VERTICAL BYTE OFFSET
92AD D3 BA                  ADDD    BEGGRP         2    ADD START OF GRAPHIC PAGE
92AF 1F 01                  TFR     D,X            2    SAVE TEMP VALUE IN X REG
92B1 D6 BE                  LDB     HORBEG+1       2    GET HORIZONTAL COORDINATE
```

```
92B3 54                  LSRB                1    DIVIDE BY 8
92B4 54                  LSRB                1
92B5 54                  LSRB                1
92B6 3A                  ABX                 1    ADD HOR BYTE OFFSET
92B7 96 BE               LDA     HORBEG+1    2    GET HORIZONTAL COORDINATE
92B9 84 07               ANDA    #$07        2    KEEP ONLY BITS 0-2, WHICH
                                                  CONTAIN THE NUMBER OF THE
                                                  PIXEL IN THE BYTE
92BB CE 92 DD            LDU     #L92DD      3    POINT U TO MASK LOOKUP TABLE
92BE A6 C6               LDA     A,U         2    GET PIXEL MASK
92C0 35 C4               PULS    B,U,PC      2    CLEAR STACK AND RTS
                                                  CALLED FROM 9374

9377 E6 84          L9377 LDB    ,X          2    GET BYTE FROM THE SCREEN
9379 34 04               PSHS    B           2    SAVE IT ON STACK
937B 1F 89               TFR     A,B         2    PUT PIXEL MASK IN ACCB
937D 43                  COMA                1    INVERT PIXEL MASK
937E A4 84               ANDA    ,X          2    SET THE PIXEL
9380 D4 B5               ANDB    ALLCOL      2    CONVERT PIXEL IN THE PIXEL
                                                  MASK TO THE PROPER COLOR
9382 34 04               PSHS    B           2    SAVE IT ON STACK
9384 AA E0               ORA     ,S+         2    'OR' IT INTO THE REST OF
                                                  THE PIXELS
9386 A7 84               STA     ,X          2    PUT IT ON SCREEN
9388 A0 E0               SUBA    ,S+         2    SUBTRACT OLD BYTE FROM NEW
                                                  BYTE; ACCA=0 IF NEW
                                                  BYTE = OLD BYTE
938A 9A DB               ORA     CHGFLG      2    'OR' DIFFERENCE WITH
                                                  CHANGE FLAG
938C 97 DB               STA     CHGFLG      2    SAVE IT
938E 39                  RTS                 1    END OF ROM PSET UNWOUND
```

```
** THIS UNWINDING OF THE ROM PSET NOW ENDS
   AT A LINE COUNT OF 1487.

=====
```