```
%-------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------
% CI-2
%
% FLOATING POINT COMPUTATION INTERPRETER FOR PERSEUS-8 AND PERSEUS-9 (CPU:R6502)
%
% FILE_NAME: ASSEMBLY_CODE_CI-2_V1_5_0.PDF
%
% SPECIFICATION: SEE APPENDIX
%
% HAND ASSEMBLED
%
% VERSION AUG/05/2021      VER. 1.0.0
%         AUG/27/2021      VER. 1.1.0    IF THE MANTISSA IS ZERO, SET THE EXPONET PART TO ZERO.
%                                        OUTPUT SPACE AFTER DISPLAYING ZERO.
%         APR/09/2022      VER. 1.2.0    BUG FIX FOR SQRT FUNCTION ARGUMENT ZERO.
%         JULY/12/2022     VER. 1.2.1    TEMPORARY PATCH FOR ERROR IN PROM MAKING ($E6A4-$E6AE)
%         AUG/06/2022      VER. 1.3.1    PREVENT OUTPUT SPACE CODE IN FORMAT ;; ($E05D,EE30,FC57)
%         OCT/07/2022      VER. 1.3.2    CORRECTING LABEL NAME ERROR. ($E390)
%         DEC/09/2022      VER. 1.4.0    PREVENT CALCULATION STACK DESTRUCTION EVERY 256 CHAR.
%                                        ($F396-F3AE)
%         MAR/11/2023      VER. 1.5.0    ALLOW USER PROGRAM AREA LIMIT TO BE SET. ($E7A3-$E830)
%                                         (ADDRESS $0082 MUST BE SET TO $XX AT FIRST STARTUP.)
%                                        CORRECT NORMALISE($E2D0-) TO MAKE EXPONENT PART THE SAME
%                                        AS THE MINIMUM VALUE,1E-39,WHEN THE MANTISSA PART IS 0.
%
%-------------------------------------------------------------------------------------
% COPYRIGHT (C) 2021-2023 MITSURU YAMADA. ALL RIGHTS RESERVED.
%
%-------------------------------------------------------------------------------------
% ADDRESS MAPPING:
%             SYSTEM VARIABLES           (ADDRESS  $0000 - $00A3)
%             SYSTEM STACK               (ADDRESS  $0100 - $01FF)
%             LINE BUFFER                (ADDRESS  $0200 - $027F)
%             DISPLAY BUFFER             (ADDRESS  $0280 - $02FF)
%             USER VARIABLES             (ADDRESS  $0300 - $03FF)
%             USER ARRAY VARIABLES       (ADDRESS  $0400 - $0FFF)
%             USER PROGRAM AREA          (ADDRESS  $1000 - $XXFF)   (V.1.5.0  MAR/11/2023)
%             SYSTEM PROGRAM             (ADDRESS  $E000 - $FFFF)
%-------------------------------------------------------------------------------------
% SYSTEM VARIABLES
%             SYMBOL                     DATA
              LR0                        $0000        LINE NUMBER OF THE NEW LINE
              LR1                        $0002        LINE NUMBER OF PROGRAM AREA
              LINE_LENGTH_1              $0004
              LINE_LENGTH_2              $0005
              TEMP_9                     $0006        ZERO SUPPRESS FLAG
              (TEMP_8)                   $0007        EXPONENT CORRECTION VALUE
              TEMP_7 (SIGN_FLAG)         $0008
              TEMP_8 (TEMP_0)            $0009        (SAVING R0_3,EXPONENT PORTION OF R1_3)
              TEMP_1 (COL_COUNT)         $000A        LOOP COUNTER, EXPONENT PORTION OF R0_3
              TEMP_2,MUL_COUNT,QUO_COUNT $000B        FOR SAVING ACC. FOR LOOP COUNTING
              TEMP_3                     $000C        EXPONENT CORRECTION, SHIFT OF MANTISSA
              TEMP_4                     $000D
              TEMP_5                     $000E
              TEMP_6                     $000F
              R0                         $0010        32BIT FLOATING POINT REGISTER
              R1                         $0014        32BIT FLOATING POINT REGISTER
              DR0                        $0018        48BIT MULTIPLIER REGISTER
              DR1                        $0020        48BIT MULTIPLIER REGISTER
              DR2                        $0028        48BIT MULTIPLIER REGISTER
              R2                         $0030        32BIT FLOATING POINT REGISTER
              R3                         $0034        32BIT FLOATING POINT REGISTER
              P0                         $0040        16BIT POINTER
              P1                         $0042        16BIT POINTER CALCULATED BY R0
              P2                         $0044        16BIT POINTER
              P3                         $0046        MEMORY WRITE POINTER FOR VARIABLE
              P4                         $0048        16BIT POINTER FOR SAVING P0
              P5                         $004A        MEMORY WRITE POINTER FOR ARRAY
              RUN_FLAG                   $0050        EXECUTION FLAG
              DISP_FLAG                  $0051        DISLAY FLAG
              VAR_0                      $0052        DETECTED VARIABLE ADDRESS OFFSET
```

```
             VAR_1                      $0053        DESTINATION VARIABLE ADDRESS OFFSET
             TEMP_10                    $0054        FOR SAVING IY
             TEMP_11                    $0055        FOR SAVING P0
             TEMP_12                    $0056        FOR SAVING P0+1
             TEMP_13                    $0057        FOR SAVING ACC
             MONI_1                     $0058        SHIFI MONITOR CONVERSION FLOAT TO BINARY
             MEM_VAL                    $005A        READING VALUE FROM MEMORY
             MONI_2                     $005B        MONITOR FOR READING VALUE FROM MEMORY
             TEMP_14,15,16,17           $005C        32BIT REGISTER FOR ARRAY DUMMY WRITE
             R4                         $0060        32BIT FLOATING POINT REGISTER
             R5                         $0064        32BIT FLOATING POINT REGISTER
             R6                         $0068        32BIT FLOATING POINT REGISTER
             R7                         $006C        32BIT FLOATING POINT REGISTER
             R8 (LOOP_COUNTER_2)        $0070        32BIT FLOATING POINT REGISTER
             R9                         $0074        32BIT FLOATING POINT REGISTER
             R10                        $0078        32BIT FLOATING POINT REGISTER
             R11                        $007C        32BIT FLOATING POINT REGISTER
             P6                         $0080        16BIT POINTER FOR TABLE OF FUNCTION
             MEM_END_U                  $0082        PROGRAM AREA LIMIT UPPER 8BIT (V.1.5.0)
             LOOP_COUNTER               $0090        8BIT LOOP COUNTER FOR FUNCTION
             FUNC_SGN_FLAG              $0091        SIGN FLAG FOR SINE FUNCTION
             FUNC_SIN_FLG2              $0092
             R12                        $00A0        32BIT FLOATING POINT REGISTER
%
             LINE_HEAD                  $0200        HEAD ADDRESS OF LINE BUFFER
             OUT_BUF                    $0280        HEAD ADDRESS OF DISPLAY BUFFER
             VAR_HEAD                   $0300        HEAD ADDRESS OF USER VARIABLES
             PROG_P0_1                  $1000        HEAD ADDRESS OF USER PROGRAM AREA
%
             ACIA_STATUS                $8000        ACIA(68B50) STATUS REGISTER
             ACIA_DATA                  $8001        ACIA(68B50) DATA REGISTER
%
%-------------------------------------------------------------------------------
% CI-2 START ADDRESS                   $E8D0
%
%-------------------------------------------------------------------------------
%                          ADDRESS(HEX) DATA(HEX)
%-------------------------------------------------------------------------------
% RESET VECTOR
%-------------------------------------------------------------------------------
             .ORIGIN  $FFFC
                                FFFC   D0 E8
%
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
%---------------------------------   UNIT TEST   ---------------------------------
%-------------------------------------------------------------------------------
%
%-------------------------------------------------------------------------------
% UNIT TEST FOR LINE EDITOR WITHOUT SERIAL INTERFACE.
%-------------------------------------------------------------------------------
             .ORIGIN  $E050
%
EDITOR_TEST_1CLC                    E050   18
             LDX  #$FF              E051   A2 FF
             TXS                    E053   9A
             JSR  LINE_EDITOR_1     E054   20 40 E7
L01          JMP  L01               E057   4C 57 E0
%
%-------------------------------------------------------------------------------
% PATCH TO PREVENT OUTPUT SPACE CODE IN FORMAT ;;     (V.1.3.1, AUG/06/2022)
%-------------------------------------------------------------------------------
             .ORIGIN  $E05D
%
PATCH_ELP_5  JMP. ENDLINE_PRC_5A E05D   4C 57 FC
%
%-------------------------------------------------------------------------------
% UNIT TEST FOR FLOATING POINT FOUR ARITHMETIC OPERATION.
%-------------------------------------------------------------------------------
             .ORIGIN  $E060
%
ARITH_TEST   CLC                    E060   18
```

```
             LDX  #$FF           E061   A2 FF
             TXS                 E063   9A
             JSR  INIT_ACIA      E064   20 80 E0
             JSR  INIT_REGISTER  E067   20 8B E0
L01          JSR  IN_1LINE       E06A   20 C0 E0
             JSR  GET_FLOAT_3    E06D   20 F7 E0
             JSR  NORMALISE_1    E070   20 D0 E2
             NOP                 E073   EA
             NOP                 E074   EA
             NOP                 E075   EA
             JSR  DISP_FLOAT_1   E076   20 E0 E1
             JSR  DIST_BUFFER_1  E079   20 D0 E1
             BNE  L01            E07C   D0 EC
             BEQ  L01            E07E   F0 EA
%
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
%-------------------------  I/O & FLOATING POINT CONVERSION   ---------------------
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
INIT_ACIA    LDA  #$03           E080   A9 03
             STA  ACIA_STATUS    E082   8D 00 80
             LDA  #$15           E085   A9 15
             STA  ACIA_STATUS    E087   8D 00 80
             RTS                 E08A   60
%
%-------------------------------------------------------------------------------
INIT_REGISTERRTS                 E08B   60
%
%-------------------------------------------------------------------------------
% INPUT ONE CHARACTER FROM SERIAL INTERFACE.  (OUTPUT PARAMETER: ACC)
%-------------------------------------------------------------------------------
             .ORIGIN  $E090
%
IN_1CHA      LDA  ACIA_STATUS    E090   A0 00 80
             LSR                 E093   4A
             BCC  IN_1CHA        E094   90 FA
             LDA  ACIA_DATA      E096   AD 01 80     INPUT ONE CHARACTER.
             RTS                 E099   60
%
%-------------------------------------------------------------------------------
% OUTPUT ONE CHARACTER TO SERIAL INTERFACE.    (INPUT PARAMETER: ACC)
%-------------------------------------------------------------------------------
OUT_1CHA     PHA                 E09A   48
L02          LDA  ACIA_STATUS    E09B   AD 00 80
             LSR                 E09E   4A
             LSR                 E09F   4A
             BCC  L02            E0A0   90 F9
             PLA                 E0A2   68
             STA  ACIA_DATA      E0A3   8D 01 80     OUTPUT ONE CHARACTER.
             CMP  #$0D           E0A6   C9 0D
             BEQ  L03            E0A8   F0 05
             CMP  #$0A           E0AA   C9 0A
             BEQ  L04            E0AC   F0 05
             RTS                 E0AE   60           IF IT'S NOT 'CR' OR 'LF', EXIT.
%
L03          LDA  #$0A           E0AF   A9 0A        IN THE CASE OF 'CR', 'LF' IS SENT AFTER 'CR'.
             BNE  OUT_1CHA       E0B1   D0 E7
%
L04          LDA  #$0D           E0B3   A9 0D        IN THE CASE OF 'LF', PUT 'CR' TO ACC,
             RTS                 E0B5   60           AND EXIT.
%
%-------------------------------------------------------------------------------
% ENTERING SINGLE LINE
%-------------------------------------------------------------------------------
             .ORIGIN  $E0B6
%
IN_1LINE_1   LDA  #$00           E0B6   A9 00
             LDX  #$00           E0B8   A2 00
L01          STA  LINE_HEAD,X    E0BA   9D 00 02     CLEAR BUFFER ($0200-$02FF)
             INX                 E0BD   E8
             BNE  L01            E0BE   D0 FA
```

```
IN_1LINE      LDX  #$00           E0C0   A2 00
L06           JSR  IN_1CHA        E0C2   20 90 E0
              JSR  OUT_1CHA       E0C5   20 9A E0
              CMP  #$08           E0C8   C9 08        IS IT 'BS' CODE?
              BEQ  L05            E0CA   F0 09
              STA  LINE_HEAD,X    E0CC   9D 00 02
              INX                 E0CF   E8
              CMP  #$0D           E0D0   C9 0D
              BNE  L06            E0D2   D0 EE        REPEAT UNTIL 'CR' CODE IS DETECTED.
              RTS                 E0D4   60
%
L05           CPX  #$00           E0D5   E0 00        IS THE LINE POINTER POSITION
              BEQ  L06            E0D7   F0 E9         THE INITIAL POSITION?
              DEX                 E0D9   CA           LINE POINTER - 1.
              BNE  L06            E0DA   D0 E6
              BEQ  L06            E0DC   F0 E4
%
%-------------------------------------------------------------------------------
% PATCH FOR GET FLOATING POINT VALUE.
%-------------------------------------------------------------------------------
              .ORIGIN  $E0E0
%
L08           INC  TEMP_6         E0E0   E6 0F        SET DECIMAL POINT FLAG.
              JMP  L07            E0E2   4C 28 E1
%
              NOP                 E0E5   EA
              NOP                 E0E6   EA
L09           LDY  TEMP_6         E0E7   A4 0F
              BEQ  L11            E0E9   F0 02        IF DECIMAL POINT FLAG=0, EXIT.
              DEC  TEMP_3         E0EB   C6 0C        IF THERE IS ALREADY A DECIMAL POINT,
L11           JMP  L06            E0ED   4C 11 E1      DECIMAL POINT COUNTER -1.
%
%-------------------------------------------------------------------------------
% GET FLOATING POINT VALUE FROM ASCII CODES ON LINE BUFFER.
%-------------------------------------------------------------------------------
              .ORIGIN  $E0F7
%
GET_FLOAT_3   LDX  #$00           E0F7   A2 00
GET_FLOAT_1   JSR  INIT_REG       E0F9   20 B3 E1
L02           JMP  GET_FLOAT_4    E0FC   4C B0 E6     ANALYZE ONE CHARACTER.
L13           CMP  #$2D           E0FF   C9 2D
              BEQ  L01            E101   F0 2C        IF IT'S '-' CODE, GO TO SIGN PROCESS.
              CMP  #$45           E103   C9 45
              BEQ  GET_EXP_1      E105   F0 30        IF IT'S 'E' CODE, GO TO INPUT EXPONENT PART.
              CMP  #$0D           E107   C9 0D
              BEQ  CORRECT_EXP_1  E109   F0 60        IF IT'S 'CR' CODE, CORRECT EXPONENT,EXIT.
              CMP  #$2E           E10B   C9 2E        IF IT'S '.'CODE,
              BNE  L09            E10D   D0 D8         GO TO DECIMAL POINT PROCESS.
              BEQ  L08            E10F   F0 CF
%
L06           STX  TEMP_5         E111   86 0E        SAVE IX.
              STA  TEMP_2         E113   85 0B        SAVE ACC.
              LDX  #$04           E115   A2 04        SET SHIFT BITS.
              LDY  #$13           E117   A0 13
              LDA  #$FF           E119   A9 FF        SELECT R0_0.
              JSR  SHIFT_L_NBIT   E11B   20 A0 E1     SHIFT LEFT.
              LDX  TEMP_5         E11E   A6 0E
              LDA  TEMP_2         E120   A5 0B
              AND  #$0F           E122   29 0F        EXTRACT LOWER 4 BITS OF ASCII CODE.
              ORA  R0_0           E124   05 10
              STA  R0_0           E126   85 10        PUT BCD ONE DIGIT IN R0_0 LOWER 4 BITS.
L07           INX                 E128   E8
              CPX  #$FF           E129   E0 FF
              BNE  L02            E12B   D0 CF
              BEQ  GET_EXP_1      E12D   F0 08
%
L01           LDA  #$80           E12F   A9 80        SIGN FLAG D7='1'.
              STA  TEMP_7         E131   85 08
              NOP                 E133   EA
              NOP                 E134   EA
              BNE  L07            E135   D0 F1
%
```

```
GET_EXP_1     INX                 E137   E8
              LDA  LINE_HEAD,X    E138   BD 00 02      ANALYZE THE NEXT CHARACTER OF ‘E’ CODE.
              CMP  #$2D           E13B   C9 2D         IF IT’S ‘-‘ CODE,
              BEQ  L03            E13D   F0 24          GO TO EXPONENT SIGN PROCESS.
              JMP  GET_EXP_3      E13F   4C 90 E1
%
              NOP                 E142   EA
GET_EXP_2     STA  TEMP_2         E143   85 0B         SAVE ACC.
              LDA  R0_3           E145   A5 13
              STA  TEMP_0         E147   85 09         SAVE R0_3.
              ASL                 E149   0A            SHIFT LEFT 4 BITS R0_3.
              ASL                 E14A   0A
              ASL                 E14B   0A
              ASL                 E14C   0A
              AND  #$30           E14D   29 30         EXTRACT D5,D4
              STA  TEMP_1         E14F   85 0A
              LDA  TEMP_0         E151   A5 09
              AND  #$C0           E153   29 C0         EXTRACT D7.D6 OF R0_3.
              ORA  TEMP_1         E155   05 0A         SYNTHESIZE D7.D6,D5,D4.
              STA  R0_3           E157   85 13
              LDA  TEMP_2         E159   A5 0B         NEW ONE ASCII CHARACTER.
              AND  #$0F           E15B   29 0F         EXTRACT LOWER 4 BITS OF ASCII CODE.
              ORA  R0_3           E15D   05 13
              STA  R0_3           E15F   85 13         SYNTHESIZE EXPONENT VALUE AND NEW VALUE.
              BCC  GET_EXP_1      E161   90 D4
%
L03           LDA  TEMP_7         E163   A5 08
              ORA  #$40           E165   09 40
              STA  TEMP_7         E167   85 08         SIGN FLAG D6=’1’.
              BNE  GET_EXP_1      E169   D0 CC
%
%-------------------------------------------------------------------------------
% CORRECT EXPONENT PART
%-------------------------------------------------------------------------------
CORRECT_EXP_1 LDA  TEMP_7         E16B   A5 08
              ASL                 E16D   0A
              BPL  L10            E16E   10 0B         IS EXPONENT POSITIVE?
              SEC                 E170   38            IF IT’S NEGATIVE, CONVERT TO COMPLEMENT.
              SED                 E171   F8
              LDA  #$00           E172   A9 00
              SBC  R0_3           E174   E5 13         0 – EXPONENT(BCD)
              SEC                 E176   38
              SBC  #$20           E177   E9 20         AND MORE, - 20(BCD),
              STA  R0_3           E179   85 13          TO MAKE IT COMPLEMENT OF 80(BCD)
L10           LDA  R0_3           E17B   A5 13
              CLC                 E17D   18
              SED                 E17E   F8
              ADC  TEMP_3         E17F   65 0C         EXPONENT CORRECTION ADDITION.
              AND  #$7F           E181   29 7F         PEVENT THE RESULT OVERLAPPING
              STA  R0_3           E183   85 13          SIGN OF MANTISSA.
              LDA  TEMP_7         E185   A5 08         SIGN FLAG.
              AND  #$80           E187   29 80         EXTRACT SIGN FLAG OF MANTISSA.
              ORA  R0_3           E189   05 13         SYNTHESIS EXPONENT AND MANTISSA SIGN.
              STA  R0_3           E18B   85 13
              RTS                 E18D   60
%
%-------------------------------------------------------------------------------
% PATCH FOR GET FLOATING POINT VALUE.
%-------------------------------------------------------------------------------
              .ORIGIN  $E190
%
GET_EXP_3     CMP  #$20           E190   C9 20         SPACE CODE?
              BEQ  L01            E192   F0 07
              CMP  #$0D           E194   C9 0D         ‘CR’ CODE?
              BEQ  CORRECT_EXP_1  E196   F0 D3
              JMP  GET_EXP_2      E198   4C 43 E1      GO TO EXPONENT PROCESS.
%
L01           JMP  G_FLOAT_4_L12  E19B   4C BA E6      GO TO CORRECTION EXPONENT
%                                                       AND MOVE R0 TO R1, ANALYZE NEXT CHARACTER.
%-------------------------------------------------------------------------------
% SHIFT LEFT 3 BYTES SECTION.
%-------------------------------------------------------------------------------
```

```
              .ORIGIN  $E1A0
%
SHIFT_L_NBIT  STA  P0+1           E1A0   85 41
              STY  P0             E1A2   84 40
L69           CLC                 E1A4   18
              LDY  #$FD           E1A5   A0 FD
L68           LDA  (P0),Y         E1A7   B1 40
              ROL                 E1A9   2A
              STA  (P0),Y         E1AA   91 40
              INY                 E1AC   C8
              BNE  L68            E1AD   D0 F8
              DEY                 E1AF   CA
              BNE  L69            E1B0   D0 F2
              RTS                 E1B2   60
%
%-------------------------------------------------------------------------------
% INITIALIZE REGISTERS.
%-------------------------------------------------------------------------------
INIT_REG      LDA  #$06           E1B3   A9 06
              STA  TEMP_3         E1B5   85 0C        INITIALIZE DECIMAL POINT COUNTER
              LDA  #$00           E1B7   A9 00        CLEAR R0_0-3
              STA  R0_0           E1B9   85 10
              STA  R0_1           E1BB   85 11
              STA  R0_2           E1BD   85 12
              STA  R0_3           E1BF   85 13
              STA  TEMP_6         E1C1   85 0F        CLEAR DECIMAL POINT FLAG.
              STA  TEMP_7         E1C3   85 08        CLEAR NEGATIVE FLAG OF MANTISSA
              RTS                 E1C5   60           AND EXPONENT PART.

%
%-------------------------------------------------------------------------------
% DISPLAY BUFFER
%      SERIAL OUTPUT THE ASCII CODE ON THE DISPLAY BUFFER UNTIL THE ‘CR’ CODE IS DETECTED.
%-------------------------------------------------------------------------------
              .ORIGIN  $E1D0
%
DISP_BUFFER_1 LDY  #$00           E1D0   A0 00        INITIALIZE BUFFER POINTER.
L11           LDA  OUT_BUF,Y      E1D2   B9 80 02
              INY                 E1D5   C8
              BEQ  L12            E1D6   F0 07        TERMINATE AT 256 CHARACTERS.
              JSR  OUT_1CHA       E1D8   20 9A E0
              CMP  #$0D           E1DB   C9 0D        IF ‘CR’ CODE IS DETECTED, TERMINATE.
              BNE  L11            E1DD   D0 F3
L12           RTS                 E1DF   60
%
%-------------------------------------------------------------------------------
% DISPLAY FLOATING POINIT VALUE.
%      CONVERT FLOATING POINT REGISTER VALUE TO ASCII CODES, AND STORE IT TO DISPLAY BUFFER.
%-------------------------------------------------------------------------------
              .ORIGIN  $E1E0
%
DISP_FLOAT_1  LDA  #$07           E1E0   A9 07
              STA  TEMP_8         E1E2   85 07
              LDY  #$00           E1E4   A0 00
              STY  TEMP_9         E1E6   84 06
              LDX  #$03           E1E8   A2 03
              LDA  R0_3           E1EA   A5 13
              BPL  L03            E1EC   10 06
              LDA  #$2D           E1EE   A9 2D        IF SIGN IS NEGATIVE, OUTPUT ‘-‘ CODE.
              STA  OUT_BUF,Y      E1F0   99 80 02
              INY                 E1F3   C8
DET_UPPER     LDA  R0_0-1,X       E1F4   B5 0F        EXTRACT R0 UPPER 4 BITS.
              LSR                 E1F6   4A
              LSR                 E1F7   4A
              LSR                 E1F8   4A
              LSR                 E1F9   4A
              STA  TEMP_2         E1FA   85 0B
              BEQ  L01            E1FC   F0 02        IS IT ZERO?
              INC  TEMP_9         E1FE   E6 06        IT IS NOT ZERO, FLAG +1
L01           LDA  TEMP_9         E200   A5 06        IF FLAG=0, DO NOT OUTPUT.
              BEQ  DET_LOWER      E202   F0 16
              LDA  TEMP_2         E204   A5 0B
```

```
            ORA  #$30           E206   09 30        CONVERT TO ASCII CODE.
            STA  OUT_BUF,Y      E208   99 80 02     OUTPUT TO DISPLAY BUFFER.
            INY                 E20B   C8           DISPLAY BUFFER POINTER + 1.
            DEC  TEMP_8         E20C   C6 07        EXPONENT CORRECTION VALUE - 1
            LDA  TEMP_9         E20E   A5 06
            CMP  #$01           E210   C9 01
            BNE  DET_LOWER      E212   D0 06        IF ZERO SUPPRESS FLAG=1,
            JMP  FLOAT_PATCH_1  E214   4C B5 E2      OUTPUT DECIMAL POINT.
            %
            NOP                 E217   EA
            NOP                 E218   EA
            NOP                 E219   EA
DET_LOWER   LDA  R0_0-1,X       E21A   B5 0F        EXTRACT R0 LOWER 4 BITS.
            AND  #$0F           E21C   29 0F
            STA  TEMP_2         E21E   85 0B
            BEQ  LO4            E220   F0 02        IS IT ZERO?
            INC  TEMP_9         E222   E6 06        IT IS NOT ZERO, FLAG +1
L04         LDA  TEMP_9         E224   A5 06        IF FLAG=0, DO NOT OUTPUT.
            BEQ  L06            E226   F0 18
            LDA  TEMP_2         E228   A5 0B
            NOP                 E22A   EA
            NOP                 E22B   EA
            ORA  #$30           E22C   09 30        CONVERT TO ASCII CODE.
            STA  OUT_BUF,X      E22E   99 80 02     OUTPUT TO DISPLAY BUFFER.
            INY                 E231   C8           DISPLAY BUFFER POINTER + 1.
            DEC  TEMP_8         E232   C6 07        EXPONENT CORRECTION VALUE - 1
            LDA  TEMP_9         E234   A5 06
            CMP  #$01           E236   C9 01
            BNE  L06            E238   D0 06
            JMP  FLOAT_PATCH_2  E23A   4C C0 E2     IF ZERO SUPPRESS FLAG=1,
%                                                    OUTPUT DECIMAL POINT.
            NOP                 E23D   EA
            NOP                 E23E   EA
            NOP                 E23F   EA
L06         DEX                 E240   CA           R0 POINTER  - 1.
            BNE  DET_UPPER      E241   D0 B1
OUT_EXP_1   CPY  #$00           E243   C0 00        DISPLAY EXPONENT PART.
            BNE  L07            E245   D0 0C
            JMP  DISP_ZERO      E247   4C 67 E4     IF MANTISSA=0, DISPLAY ‘0’, SPACE ’CR’.
%                                                    (AUG.27,2021)
            .ORIGIN  $E253
%
L07         JMP OUT_EXP_2       E253   4C 90 E2     IF MANTISSA IS NOT ZERO, GO TO
%                                                    DISPLAY EXPONENT PART.
%-------------------------------------------------------------------------------
            .ORIGIN  $E25A
%
OUT_EXP_4   BCC  L08            E25A   90 0D        IF IT’S GREATER THAN $3F, IT’S NEGATIVE.
            LDA  #$2D           E25C   A9 2D
            STA  OUT_BUF,Y      E25E   99 80 02     OUTPUT ‘-‘ CODE.
            INY                 E261   C8
            SEC                 E262   38
            LDA  #$00           E263   A9 00        CONVERT COMPLEMENT TO ABSOLUTE.
            SBC  TEMP_2         E265   E5 0B
            STA  TEMP_2         E267   85 0B
L08         LDA  TEMP_2         E269   A5 0B
            LSR                 E26B   4A           SHIFT RIGHT 4 BIT.
            LSR                 E26C   4A
            LSR                 E26D   4A
            LSR                 E26E   4A
            BEQ  L09            E26F   F0 06        IF UPPER IS ZERO, DO NOT OUTPUT.
            ORA  #$30           E271   09 30
            STA  OUT_BUF,Y      E273   99 80 02
            INY                 E276   C8
L09         LDA  TEMP_2         E277   A5 0B
            AND  #$0F           E279   29 0F
            NOP                 E27B   EA
            NOP                 E27C   EA           IF LOWER IS ZERO, DO NOT OUTPUT.
            ORA  #$30           E27D   09 30
            STA  OUT_BUF,Y      E27F   99 80 02
            INY                 E282   C8
            LDA  #$0D           E283   A9 0D
```

```
              STA  OUT_BUF,Y       E285   99 80 02      OUTPUT 'CR' CODE.
              RTS                  E288   60
%
%-------------------------------------------------------------------------------
              .ORIGIN  $E290
%
OUT_EXP_2     LDA  #$45            E290   A9 45
              STA  OUT_BUF,Y       E292   99 80 02      OUTPUT 'E' CODE.
              INY                  E295   C8            DISPLAY BUFFER POINTER + 1.
              LDA  R0_3            E296   A5 13         EXTRACT EXPONENT PART.
              AND  #$7F            E298   29 7F
              STA  TEMP_2          E29A   85 0B         SAVE ACC.
              JMP  TEST_2          E29C   4C BF EC
%
%-------------------------------------------------------------------------------
OUT_EXP_5     SEC                  E29F   38
              SBC  TEMP_8          E2A0   E5 07         EXPONENT CORRECTION
              STA  TEMP_2          E2A2   85 0B          BY DECIMAL POINT POSITION.
              ASL                  E2A4   0A
              BPL  L07             E2A5   10 07         IF D6='1', ADD 20(BCD)
              LDA  TEMP_2          E2A7   A5 0B          TO MAKE IT COMPLEMENT OF 100(BCD).
              CLC                  E2A9   18
              ADC  #$20            E2AA   69 20
              STA  TEMP_2          E2AC   85 0B
L07           LDA  TEMP_2          E2AE   A5 0B
              CMP  #$3F            E2B0   C9 3F         IS IT GREATER THAN %3F?
              JMP  OUT_EXP_4       E2B2   4C 5A E2
%
%-------------------------------------------------------------------------------
% PATCH FOR DISPLAY FLOATING VALUE.
%      WHEN A DECIMAL POINT IS OUTPUT, ADD 1 TO THE ZERO SUPPRESS FLAG,
%      AND DO NOT DISPLAY THE DECIMAL POINT NEXT TIME.
%-------------------------------------------------------------------------------
FLOAT_PATCH_1 LDA  #$2E            E2B5   A9 2E
              STA  OUT_BUF,Y       E2B7   99 80 02
              INY                  E2BA   C8
              INC  TEMP_9          E2BB   E6 06
              JMP  DET_LOWER       E2BD   4C 1A E2
%
%-------------------------------------------------------------------------------
FLOAT_PATCH_2 LDA  #$2E            E2C0   A9 2E
              STA  OUT_BUF,Y       E2C2   99 80 02
              INY                  E2C5   C8
              INC  TEMP_9          E2C6   E6 06
              JMP  L06             E2C8   4C 40 E2
%
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
%-----------------------   FOUR ARITHMETIC OPERATIONS   ------------------------
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
%
%-------------------------------------------------------------------------------
% NORMALISE  (SHIFT MANTISSA TO R0_2 UPPER, AND CORRECT EXPONENT PART. (V.1.5.0  MAR/11/2023)
%-------------------------------------------------------------------------------
              .ORIGIN  $E2D0
%
NORMALISE_1   LDA  #$06            E2D0   A9 06
              STA  TEMP_1          E2D2   85 0A         INITIALIZE LOOP COUNTER.
              LDA  #$00            E2D4   A9 00
              STA  TEMP_3          E2D6   85 0C         INITIALIZE EXPONENT CORRECTION VALUE.
L02           LDA  R0_2            E2D8   A5 12
              AND  #$F0            E2DA   29 F0         EXTRACT R_0 UPPER.
              BNE  CORRECT_EXP_2   E2DC   D0 14         IF IT IS NOT ZERO, TERMINATE SHIFTING.
              LDX  #$04            E2DE   A2 04         SHIFT BITS.
              LDY  #$13            E2E0   A0 13
              LDA  #$FF            E2E2   A9 FF
              JSR  SHIFT_L_NBIT1   E2E4   20 54 EC      SHIFT LEFT.
              INC  TEMP_3          E2E7   E6 0C         EXPONENT CORRECTION VALUE + 1.
              DEC  TEMP_1          E2E9   C6 0A
              BNE  L02             E2EB   D0 EB         SHIFT ENDS TO THE BOTTOM.
              LDA  #$42            E2ED   A9 42         WHEN THE MANTISSA IS ZERO,
```

```
             STA  R0_3           E2EF   85 13         SET THE EXPONENT PART TO MINIMUM.
             RTS                 E2F1   60                  (MAR/11/2023  V.1.5.0)
%
CORRECT_EXP_2SED                 E2F2   F8
             LDA  R0_3           E2F3   A5 13
             STA  TEMP_0         E2F5   85 09        EXPONENT CORRECTION.
             AND  #$7F           E2F7   29 7F        REMOVE SIGN BIT OF MANTISSA.
             STA  TEMP_2         E2F9   85 0B
             ASL                 E2FB   0A
             BPL  L01            E2FC   10 07
             LDA  TEMP_2         E2FE   A5 0B        IF D6='1', ADD 20(BCD),
             CLC                 E300   18           TO MAKE IT COMPLEMENT OF 100(BCD).
             ADC  #$20           E301   69 20
             STA  TEMP_2         E303   85 0B
L01          LDA  TEMP_2         E305   A5 0B
             SEC                 E307   38
             SBC  TEMP_3         E308   E5 0C        SUBTRACT EXPONENT CORRECTION VALUE.
             STA  TEMP_2         E30A   85 0B
             CMP  #$3F           E30C   C9 3F        IF IT'S GREATER THAN $3F, IT'S NEGATIVE.
             BCC  L03            E30E   90 07
             LDA  TEMP_2         E310   A5 0B        SUBTRACT 20(BCD)
             SEC                 E312   38           TO MAKE IT COMPLEMENT OF 80(BCD).
             SBC  #$20           E313   E9 20
             STA  TEMP_2         E315   85 0B
L03          LDA  TEMP_0         E317   A5 09
             AND  #$80           E319   29 80
             ORA  TEMP_2         E31B   05 0B        RESTORE THE SIGN BIT OF MANTISSA.
             STA  R0_3           E31D   85 13
             RTS                 E31F   60
%

%-------------------------------------------------------------------------------
% ALIGN EXPONENT PART BEFORE FLOATING POINIT ADDITION AND SUBTRACTION.
%-------------------------------------------------------------------------------
             .ORIGIN  $E335
%
PRE_ADD_3    SEC                 E335   38
             SBC  TEMP_2         E336   E5 0B        EXPONENT OF R1 - EXPONENT OF R0.
             STA  TEMP_2         E33B   85 0B
             BEQ  L01            E33A   F0 17        IF THE EXPONENT IS THE SAME, BRANCH.
             BCC  L02            E33C   90 16        IN CASE R1 < R0, BRANCH
             JSR  SHIFT_WIDTH    E33E   20 C2 E3     IN CASE R1 > R0, SET SHIFT WIDTH.
             NOP                 E341   EA
             LDA  R0_3           E342   A5 13
             AND  #$80           E344   29 80
             ORA  TEMP_0         E346   05 09        SYNTHESIS R0_3 SIGN AND R1_3 EXPONENT.
             STA  R0_3           E348   85 13
             LDX  TEMP_3         E34A   A6 0C        SET SHIFT WIDTH.
             LDY  #$10           E34C   A0 10
             LDA  #$00           E34E   A9 00
             JSR  SHIFT_R_NBIT   E350   20 6F E3     SHIFT RIGHT R0
L01          RTS                 E353   60
%
L02          SEC                 E354   38
             LDA  #$00           E355   A9 00
             SBC  TEMP_2         E357   E5 0B        IN CASE R1 < R0, COMPLEMENT DIFFERENCE.
             JSR SHIFT_WIDTH     E359   20 C2 E3     SET SHIFT WIDTH.
             NOP                 E35C   EA
             LDA  R1_3           E35D   A5 17
             AND  #$80           E35F   29 80
             ORA  TEMP_1         E361   05 0A        SYNTHESIS R1_3 SIGN AND R0_3 EXPONENT.
             STA  R1_3           E363   85 17        MOVE TO R1_3.
             LDX  TEMP_3         E365   A6 0C
             LDY  #$14           E367   A0 14
             LDA  #$00           E369   A9 00
             JSR  SHIFT_R_NBIT   E36B   20 6F E3
             RTS                 E36E   60
%
%-------------------------------------------------------------------------------
% SHIFT RIGHT WITH 3 BYTE SECTION.
%      SHIFT BITS: IX, REGISTER HEAD ADDRESS LOWER: IY, REGISTER HEAD ADDRESS UPPER: IX,
%-------------------------------------------------------------------------------
```

```
SHIFT_R_NBIT STY  P0             E36F   84 40
             STA  P0+1           E371   85 41
L67          CLC                 E373   18
             LDY  #$02           E374   A0 02
L66          LDA  (P0),Y         E376   B1 40
             ROR                 E378   6A
             STA  (P0),Y         E379   91 40
             DEY                 E37B   88
             BPL  L66            E37C   10 F8
             DEX                 E37E   CA
             BNE  L67            E37F   D0 F2
             RTS                 E381   60
%
%-------------------------------------------------------------------------------
% ALIGN EXPONENT PART BEFORE FLOATING POINIT ADDITION AND SUBTRACTION.
%-------------------------------------------------------------------------------
             .ORIGIN  $E390
%
PRE_ADD_1    SEC                 E390   F8           (CORRECTING LABEL NAME OCT/07/2022)
             LDA  R0_3           E391   A5 13        EXPONENT PART OF R0
             AND  #$7F           E393   29 7F        REMOVE SIGN OF MANTISSA.
             STA  TEMP_1         E395   85 0A
             ASL                 E397   0A
             CLC                 E398   18
             BMI  L03            E399   30 09        IF NEGATIVE (D6='1'), BRANCH.
             LDA  TEMP_1         E39B   A5 0A        IF POSITIVE (D6='0'),
             ADC  #$60           E39D   69 60         ADD 60(BCD).
             STA  TEMP_2         E39F   85 0B
             CLC                 E3A1   18
             BCC  L04            E3A2   90 06
%
L03          LDA  TEMP_1         E3A4   A5 0A        IN CASE OF NEGATIVE, ADD 80(BCD).
             ADC  #$80           E3A6   69 80
             STA  TEMP_2         E3A8   85 0B
L04          LDA  R1_3           E3AA   A5 17        EXPONENT PART OF R1.
             AND  #$7F           E3AC   29 7F        REMOVE SIGN OF MANTISSA.
             STA  TEMP_0         E3AE   85 09
             ASL                 E3B0   0A
             CLC                 E3B1   18
             BMI  L05            E3B2   30 07        IF NEGATIVE (D6='1'), BRANCH.
             LDA  TEMP_0         E3B2   A5 09        IF POSITIVE (D6='0'),
             ADC  #$60           E3B6   69 60         ADD 60(BCD).
             CLC                 E3B8   18
             BCC  L06            E3B9   90 04
%
L05          LDA  TEMP_0         E3BB   A5 09
             ADC  #$80           E3BD   69 80        IN CASE OF NEGATIVE, ADD 80(BCD).
L06          JMP  PRE_ADD_3      E3BF   4C 35 E3
%
%-------------------------------------------------------------------------------
% PATCH FOR PRE_ADD.  (LIMIT THE SHIFT WIDTH TO 24 BITS.)
%-------------------------------------------------------------------------------
SHIFT_WIDTH  CMP  #$06           E3C2   C9 06
             BCC  L01            E3C4   90 02
             LDA  #$06           E3C6   A9 06
L01          ASL                 E3C8   0A
             ASL                 E3C9   0A
             STA  TEMP_3         E3CA   85 0C
             RTS                 E3CC   60
%
%-------------------------------------------------------------------------------
% FLOATING POINT ADDITION.
%-------------------------------------------------------------------------------
             .ORIGIN  $E3D0
%
ADD_FLOAT_1  JSR  PRE_ADD_4      E3D0   20 F8 E3     ALIGN EXPONENT, CONVERT TO COMPLEMENT.
             LDX  #$FC           E3D3   A2 FC        32 BITS MANTISSA ADDITION,
             CLC                 E3D5   18
L01          LDA  $18,X          E3D6   B5 18        R0 = R1 + R0.
             ADC  $14,X          E3D8   75 14
             STA  $14,X          E3DA   95 14
             INX                 E3DC   E8
```

```
             NOP                 E3DD   EA
             NOP                 E3DE   EA
             BNE  L01            E3DF   D0 F5
L03          JSR  COMP_2_ABS_1   E3E1   20 26 E4     CONVERT COMPLEMENT TO ABSOLUTE WITH SIGN.
             RTS                 E3E4   60
%
%-------------------------------------------------------------------------------
% FLOATING POINT SUBTRACTION.
%-------------------------------------------------------------------------------
SUB_FLOAT_1  JSR  PRE_ADD_4      E3E5   20 F8 E3     ALIGN EXPONENT, CONVERT TO COMPLEMENT.
             LDX  #$FC           E3E8   A2 FC        32 BITS MANTISSA SUBTRACTION,
             SEC                 E3EA   38
L02          LDA  $18,X          E3EB   B5 18        R0 = R1 - R0.
             SBC  $14,X          E3ED   F5 14
             STA  $14,X          E3EF   95 14
             INX                 E3F1   E8
             NOP                 E3F2   EA
             NOP                 E3F3   EA
             BNE  L02            E3F4   D0 F5
             BEQ  L03            E3F6   F0 E9
%
%-------------------------------------------------------------------------------
% PREPROCESSING OF ADDITION AND SUBTRACTION.
%-------------------------------------------------------------------------------
PRE_ADD_4    JSR  PRE_ADD_1      E3F8   20 90 E3     ALIGN EXPONENT.
             LDA  R0_3           E3FB   A5 13
             AND  #$7F           E3FD   29 7F        SAVE EXPONENT OF R0_3
             STA  TEMP_1         E3FF   85 0A
             LDA  #$FF           E401   A9 FF        SET R1 TO POINTER P0.
             STA  P0+1           E403   85 41
             LDA  #$18           E405   A9 18
             STA  P0             E407   85 40
             JSR  ABS_2_COMP_1   E409   20 14 E4     CONVERT R1 TO COMPLEMENT.
             LDA  #$14           E40C   A9 14
             STA  P0             E40E   85 40
             JSR  ABS_2_COMP_1   E410   20 14 E4     CONVERT R0 TO COMPLEMENT.
             RTS                 E413   60
%
%-------------------------------------------------------------------------------
% CONVERT 24 BITS ABSOLUTE MANTISSA TO 32 BITS BCD COMPLEMENT.
%-------------------------------------------------------------------------------
ABS_2_COMP_1 LDY  #$FF           E414   A0 FF
             LDA  (P0),Y         E416   B1 40        EXTRACT SIGN BIT.
             AND  #$80           E418   29 80
             STA  (P0),Y         E41A   91 40
             BPL  L01            E41C   10 07        IF IT'S POSITIVE, DO NOT CONVERT.
             LDA  #$00           E41E   A9 00
             STA  (P0),Y         E420   91 40
             JSR  COMP_2         E422   20 5A E4     COMPLIMENT CONVERSION.
             RTS                 E425   60
%
%-------------------------------------------------------------------------------
% CONVERT 32 BITS BCD COMPLIMENT TO 24 BITS ABSOLUTE MANTISSA.
%-------------------------------------------------------------------------------
COMP_2_ABS_1 LDA  #$00           E426   A9 00
             STA  TEMP_7         E428   85 08
             LDA  R0_3           E42A   A5 13
             BPL  L02            E42C   10 07        IF MANTISSA IS POSITIVE, DO NOT COMPLEMENT.
             JSR  COMP_2         E42E   20 5A E4     WHEN MANTISSA IS NEGATIVE, COMPLEMENT.
             LDA  #$80           E431   A9 80
             STA  TEMP_7         E433   85 08        SET NEGATIVE TO MANTISSA SIGN FLAG.
L02          LSR  R0_3           E435   46 13
             BCC  L03            E437   90 1A        IF D0='1' OF R0_3, DO FOLLOWING PROCESS.
             LDX  #$04           E439   A2 04
             LDY  #$10           E43B   A0 10
             LDA  #$00           E43D   A9 00
             JSR  SHIFT_R_NBIT   E43F   20 6F E3     SHIFT RIGHT 4 BIT R0_3-R0_1.
             LDA  #$10           E442   A9 10
             ORA  R0_2           E444   05 12        SET '1' TO R0_2 UPPER 4 BITS.
             STA  R0_2           E446   85 12
             LDA  TEMP_1         E448   A5 0A
             CLC                 E44A   18
```

```
             ADC  #$01           E44B   69 01        EXPONENT PART CORRECTION + 1
             AND  #$7F           E44D   29 7F
             STA  TEMP_1         E44F   85 0A
             NOP                 E451   EA
             NOP                 E452   EA
L03          LDA  TEMP_1         E453   A5 0A
             ORA  TEMP_7         E455   05 08        SYNTHESIS R0_3 AND SIGN OF MANTISSA.
             STA  R0_3           E457   85 13
             RTS                 E459   60
%
%-------------------------------------------------------------------------------
% COMPLEMENT CONVERSION CORE.
%-------------------------------------------------------------------------------
COMP_2       SEC                 E45A   38
             LDY  #$FC           E45B   A0 FC
L04          LDA  #$00           E45D   A9 00
             SBC  (P0),Y         E45F   F1 40
             STA  (P0),Y         E461   91 40
             INY                 E463   C8
             BNE  L04            E464   D0 F7
             RTS                 E466   60
%
%-------------------------------------------------------------------------------
% DISPLAY ZERO.     (V.1.1.0 AUG/27/2021)
%-------------------------------------------------------------------------------
DISP_ZERO    LDA  #$30           E467   A9 30
             STA  OUT_BUF,Y      E469   99 80 02     OUTPUT '0' CODE.
             INY                 E46C   C8
             LDA  #$20           E46D   A9 20
L01          STA  OUT_BUF,Y      E46F   99 80 02     OUTPUT 9 SPACES.
             INY                 E472   C8
             CPY  #$09           E473   C0 09
             BNE  L01            E475   D0 F8
             LDA  #$0D           E477   A9 0D
             STA  OUT_BUF,Y      E479   99 80 02     OUTPUT 'CR' CODE.
             RTS                 E47C   60
%
%-------------------------------------------------------------------------------
% FLOATING POINT MULTIPLICATION.
%-------------------------------------------------------------------------------
             .ORIGIN  $E4A0
%
MUL_FLOAT_1  JSR  PRE_MUL_2      E4A0   20 F3 E4     PREPROCESSING,EXPONENT ADDITION.
             LDX  #$00           E4A3   A2 00
L01          LDA  $10,X          E4A5   B5 10        MOVE MULTIPLIER R0_0-R0_2 TO DR0_0-DR0_2.
             STA  $18,X          E4A7   95 18
             LDA  $14,X          E4A9   B5 14        MOVE MULTIPLICAND R1_0-R1_2 TO DR1_0-DR1_2
             STA  $20,X          E4AB   95 20
             INX                 E4AD   E8
             CPX  #$03           E4AE   E0 03
             BNE  L01            E4B0   D0 F3
             JSR  MUL_BCD_3      E4B2   20 C0 E4     MULTIPLICATION CORE.
             JSR  AFTER_OPE_3    E4B5   20 82 E6     POST-PROCESSING.
             RTS                 E4B8   60
%
%-------------------------------------------------------------------------------
% MULTIPLICATION CORE FOR MANTISSA.  (RESULT,DR2 = MULTIPLICAND,DR1 * MULTIPLIER,DR0)
%-------------------------------------------------------------------------------
             .ORIGIN  $E4C0
%
MUL_BCD_3    LDA  #$06           E4C0   A9 06        INITIALIZE DIGIT COUNTER.
             STA  COL_COUNT      E4C2   85 0A
L08          LDA  DR0_0          E4C4   A5 18        EXTRACT LEAST SIGNIFICANT 4 BIT
             AND  #$0F           E4C6   29 0F         OF MULTIPLIER 4 BIT.
             STA  MUL_COUNT      E4C8   85 0B        IT BECOMES MULTIPLIER COUNTER VALUE.
             BEQ  L01            E4CA   F0 10        IF MULTIPLIER COUNTER=0 ,DO NOT ACCUMULATE
L03          LDX  #$FA           E4CC   A2 FA        INITIALIZE ACCUMULATION POINTER.
             CLC                 E4CE   18
L02          LDA  $2E,X          E4CF   B5 2E        LOAD FROM RESULT REGISTER DR2.
             ADC  $26,X          E4D1   75 26        ACCUMULATE MULTIPLICAND DR1.
             STA  $2E,X          E4D3   95 2E        MOVE RESULT TO DR2.
             INX                 E4D5   E8           ACCUMULATION POINTER + 1
```

```
             BNE  L02             E4D6   D0 F7         REPEAT UNTIL MOST SIGNIFICANT DIGIT.
             DEC  MUL_COUNT       E4D8   C6 0B         MULTIPLIER COUNTER - 1.
             BNE  L03             E4DA   D0 F0         REPEAT UNTIL MULTIPLIER COUNTER = 0.
L01          LDX  #$04            E4DC   A2 04
             LDY  #$18            E4DE   A0 18
             LDA  #$00            E4E0   A9 00
             JSR  SHIFT_R_NBIT    E4E2   20 6F E3      SHIFIT 4 BIT RIGHT MULTIPLIER REGISTER DR0.
             LDX  #$04            E4E5   A2 04
             LDY  #$26            E4E7   A0 26
             LDA  #$FF            E4E9   A9 FF
             JSR  SHIFT_L_NBIT2   E4E9   20 1F E5      SHIFT 4 BIT LEFT MULTIPLICAND REGISTER DR1
             DEC  COL_COUNT       E4EE   C6 0A
             BNE  L08             E4F0   D0 D2
             RTS                  E4F2   60
%
%-------------------------------------------------------------------------------
% PREPROCESSING OF FLOATING POINT MULTIPLICATION
%-------------------------------------------------------------------------------
PRE_MUL_2    SED                  E4F3   F8           IF THE MANTISSA SIGN OF R1 AND R0 ARE
             LDA  R0_3            E4F4   A5 13          DIFFERENT, SET $80 TO THE SIGN FLAG.
             EOR  R1_3            E4F6   45 17
             AND  #$80            E4F8   29 80
             STA  SIGN_FLAG       E4FA   85 08
             JSR  MUL_EXP_2       E4FC   20 61 E5      ADDITION OF 7 BIT EXPONENT PART.
             LDA  R0_3            E4FF   A5 13
             AND  #$7F            E501   29 7F
             ORA  SIGN_FLAG       E503   05 08         SYNTHESIS SIGN AND EXPONENT PART.
             STA  R0_3            E505   85 13
             LDX  #$00            E507   A2 00         CLEAR MULTIPLICAND DR1 UPPER AND
L02          LDA  #$00            E509   A9 00          CLEAR PRODUCT DR2.
             STA  $23,X           E50B   95 23
             INX                  E50D   E8
             CPX  #$0B            E50E   E0 0B
             BNE  L02             E510   D0 F7
             RTS                  E512   60
%
%-------------------------------------------------------------------------------
% POST-PROCESSING OF MULTIPLICATION.
%-------------------------------------------------------------------------------
AFTER_OPE_1  LDX  #$00            E513   A2 00         MOVE MANTISSA RESULT DR2 TO R0
L01          LDA  $2B,X           E515   B5 2B
             STA  $10,X           E517   95 10
             INX                  E519   E8
             CPX  #$03            E51A   E0 03
             BNE  L01             E51C   D0 F7
             RTS                  E51E   60
%
%-------------------------------------------------------------------------------
% SHIFT LEFT WITH 6 BYTE SECTION.
%-------------------------------------------------------------------------------
SHIFT_L_NBIT2STA  P0+1            E51F   85 41
             STY  P0              E521   84 40
L02          CLC                  E523   18
             LDY  #$FA            E524   A0 FA
L01          LDA  (P0),Y          E526   B1 40
             ROL                  E528   2A
             STA  (P0),Y          E529   91 40
             INY                  E52B   C8
             BNE  L01             E52C   D0 F8
             DEX                  E52E   CA
             BNE  L02             E52F   D0 F2
             RTS                  E531   60
%
%-------------------------------------------------------------------------------
% ADDITION OF 7 BITS EXPONENT PART.
%-------------------------------------------------------------------------------
MUL_EXP_1    LDA  R0_3            E532   A5 13
             ASL                  E534   0A
             BPL  L01             E535   10 09         IF R0_3 EXPONENT IS NEGATIVE, ADD 20(BCD)
             LDA  R0_3            E537   A5 13          TO MAKE IT A COMPLEMENT TO 100(BCD).
             AND  #$7F            E539   29 7F
             CLC                  E53B   18
```

```
            ADC  #$20           E53C   69 20
            STA  R0_3           E53E   85 13
L01         LDA  R1_3           E540   A5 17        IF R1_3 EXPONENT IS NEGATIVE, ADD 20(BCD)
            ASL                 E542   0A           TO MAKE IT A COMPLEMENT TO 100(BCD).
            BPL  L02            E543   10 09
            LDA  R1_3           E545   A5 17
            AND  #$7F           E547   29 7F
            CLC                 E549   18
            ADC  #$20           E54A   69 20
            STA  R1_3           E54C   85 17
L02         LDA  R0_3           E54E   A5 13
            CLC                 E550   18
            ADC  R1_3           E551   65 17        ADDITION (R1 EXPONENT + R0 EXPONENT)
            STA  R0_3           E553   85 13
            CMP  #$3F           E555   C9 3F        IF RESULT IS MORE THAN $3F, IT'S NEGATIVE.
            BCC  L03            E557   90 07         SUBTRACT 20(BCD)
            LDA  R0_3           E559   A5 13         TO MAKE IT A COMPLEMENT TO 80(BCD).
            SEC                 E55B   38
            SBC                 E55C   E9 20
            STA  R0_3           E55E   85 13
L03         RTS                 E560   60
%
%-------------------------------------------------------------------------------
% REMOVE SIGN OF MANTISSA FROM R0,R1 BEFORE ADD 7 BITS EXPONENT PART.
%-------------------------------------------------------------------------------
MUL_EXP_2   LDA  R0_3           E561   A5 13
            AND  #$7F           E563   29 7F
            STA  R0_3           E565   85 13
            LDA  R1_3           E567   A5 17
            AND  #$7F           E569   29 7F
            STA  R1_3           E56B   85 17
            JMP  MUL_EXP_1      E56D   4C 32 E5
%
%-------------------------------------------------------------------------------
% FLOATING POINT DIVISION.
%-------------------------------------------------------------------------------
            .ORIGIN  $E570
%
DIV_FLOAT_1 JSR  PRE_DIV_1      E570   20 89 E5     PREPROCESSING,EXPONENT SUBTRACTION.
            LDX  #$00           E573   A2 00
L01         LDA  $10,X          E575   B5 10        MOVE R0 TO DR0 UPPER.
            STA  $1B,X          E577   95 1B
            LDA  $14,X          E579   B5 14        MOVE R1 TO DR1 UPPER.
            STA  $23,X          E57B   95 23
            INX                 E57D   E8
            CPX  #$03           E57E   E0 03
            BNE  L01            E580   D0 F3
            JSR  DIV_BCD_3      E582   20 DC E5     DIVISION CORE.
            JSR  AFTER_OPE_2    E585   20 4D E6     POST-PROCESSING.
            RTS                 E588   60
%-------------------------------------------------------------------------------
% PREPROCESSING OF FLOATING POINT DIVISION
%-------------------------------------------------------------------------------
PRE_DIV_1   SED                 E589   F8          IF THE MANTISSA FLAG OF R1 AND R2 ARE
            LDA  R0_3           E58A   A5 13         DIFFERENT, SET $80 TO THE SIGN FLAG.
            EOR  R1_3           E58C   45 17
            AND  #$80           E58E   29 80
            STA  SIGN_FLAG      E590   85 08
            JSR  DIV_EXP_1      E592   20 A9 E5     SUBTRACTION OF 7 BIT EXPONENT PART.
            LDA  R0_3           E595   A5 13
            AND  #$7F           E597   29 7F
            ORA  SIGN_FLAG      E599   05 08        SYNTHESIS SIGN AND EXPONENT PART.
            STA  R0_3           E59B   85 13
            LDX  #$00           E59D   A2 00        CLEAR DR0,DR1,DR2
            LDA  #$00           E59F   A9 00
L02         STA  $18,X          E5A1   95 18
            INX                 E5A3   E8
            CPX  #$16           E5A4   E0 16
            BNE  L02            E5A6   D0 F9
            RTS                 E5A8   60
%
%-------------------------------------------------------------------------------
```

```
% SUBTRACTION OF 7 BIT EXPONENT PART.
%-------------------------------------------------------------------------------
DIV_EXP_1     LDA  R1_3           E5A9   A5 17        REMOVE SIGN OF MANTISSA FROM R0_3,R1_3.
              AND  #$7F           E5AB   29 7F
              STA  R1_3           E5AD   85 17
              LDA  R0_3           E5AF   A5 13
              AND  #$7F           E5B1   29 7F
              STA  R0_3           E5B3   85 13
              ASL                 E5B5   0A
              BPL  L01            E5B6   10 07        IF R0_3 EXPONENT IS NEGATIVE, ADD 20(BCD)
              LDA  R0_3           E5B8   A5 13          TO MAKE IT A COMPLEMENT TO 100(BCD).
              CLC                 E5BA   18
              ADC  #$20           E5BB   69 20
              STA  R0_3           E5BD   85 13
L01           LDA  R1_3           E5BF   A5 17
              ASL                 E5C1   0A
              BPL  L02            E5C2   10 07        IF R1_3 EXPONENT IS NEGATIVE, ADD 20(BCD)
              LDA  R1_3           E5C4   A5 17         TO MAKE IT A COMPLEMENT TO 100(BCD).
              CLC                 E5C6   18
              ADC  #$20           E5C7   69 20
              STA  R1_3           E5C9   85 17
L02           LDA  R1_3           E5CB   A5 17
              SEC                 E5CD   38
              SBC  R0_3           E5CE   E5 13        SUBTRACTION (R1 EXPONENT - R0 EXPONENT)
              STA  R0_3           E5D0   85 13
              CMP  #$3F           E5D2   C9 3F        IF RESULT IS MORE THAN $3F, IT'S NEGATIVE.
              BCC  L03            E5D4   90 05         SUBTRACT 20(BCD)
              SEC                 E5D6   38            TO MAKE IT A COMPLEMENT TO 80(BCD).
              SBC  #$20           E5D7   E9 20
              STA  R0_3           E5D9   85 13
L03           RTS                 E5DB   60
%
%-------------------------------------------------------------------------------
% DIVISION CORE FOR MANTISSA. (QUOTIENT,DR2 = DIVIDEND,DR1 / DIVISOR,DR0)
%-------------------------------------------------------------------------------
DIV_BCD_3     LDA  #$00           E5DC   A9 00        INITIALIZE DIGIT COUNTER.
              STA  COL_COUNT      E5DE   85 0A
L07           NOP                 E5E0   EA
              NOP                 E5E1   EA
              NOP                 E5E2   EA
              NOP                 E5E3   EA
              NOP                 E5E4   EA
              NOP                 E5E5   EA
              NOP                 E5E6   EA
              NOP                 E5E7   EA
              NOP                 E5E8   EA
L10           BEQ  DET_ZERO_DR0   E5E9   F0 38
L09           LDA  #$FF           E5EB   A9 FF
              STA  QUO_COUNT      E5ED   85 0B        RESET QUOTIENT COUNTER.
L04           INC  QUO_COUNT      E5EF   E6 0B        QUOTIENT COUNTER + 1.
              LDX  #$FA           E5F1   A2 FA        12 DIGIT BCD SUBTRACTION
              SEC                 E5F3   38
L03           LDA  $26,X          E5F4   B5 26        LOAD DIVIDEND TO DR1.
              SBC  $1E,X          E5F6   F5 1E        SUBTRACT DIVISOR DR0.
              STA  $26,X          E5F8   95 26        MOVE RESULT TO DR1.
              INX                 E5FA   E8           DIGIT POINTER + 1.
              BNE  L03            E5FB   D0 F7
              BCS  L04            E5FD   B0 F0        IF RESULT IS POSITIVE, REPEAT SUBTRACT.
L08           LDX  #$04           E5FF   A2 04        IF SESULT IS NEGATIVE,
              LDY  #$2E           E601   A0 2E         SHIFT LEFT 4 BITS QUOTIENT DR2.
              LDA  #$FF           E603   A9 FF
              JSR  SHIFT_L_NBIT2  E605   20 1F E5
              LDA  DR2_0          E608   A5 28        SET QUOTIENT COUNTER VALUE TO LOWER 4 BIT
              ORA  QUO_COUNT      E60A   05 0B         OF LEAST SIGNIFICANT BYTE OF QUOTIENT DR2.
              STA  DR2_0          E60C   85 28
              LDX  #$FA           E60E   A2 FA        RESTORING PROCESS.
              CLC                 E610   18
L06           LDA  $26,X          E611   B5 26        DR1 = DR1 + DR0.
              ADC  $1E,X          E613   75 1E
              STA  $26,X          E615   95 26
              INX                 E617   E8
              BNE  L06            E618   D0 F7
```

```
             INC  COL_COUNT      E61A   E6 0A         DIGIT COUNTER + 1
             LDA  COL_COUNT      E61C   A5 0A
             CMP  #$07           E61E   C9 07         FINISHED SHIFTING AND SUBTRACTING THE
             JMP  DIV_PATCH_2    E620   4C 73 E6       DIVISOR FOR 6 DIGITS?
%
%-------------------------------------------------------------------------------
% JUDGEMENT OF DR0 IS ZERO.
%-------------------------------------------------------------------------------
DET_ZERO_DR0 LDX  #$06           E623   A2 06         ADDRESS POINTER.
             LDY  #$06           E625   A0 06         COUNTER OF ZERO BYTES.
L02          LDA  $17,X          E627   B5 17
             BNE  L01            E629   D0 01
             DEY                 E62B   88
L01          DEX                 E62C   CA
             BNE  L02            E62D   D0 F8
             TYA                 E62F   98
             BEQ  L03            E630   F0 02
             BNE  L09            E632   D0 B7         GO TO NON ZERO CASE PROCESS.
L03          LDA  #$00           E634   A9 00         ZERO CASE PROCESS.
             STA  QUO_COUNT      E636   85 0B         RESET QUOTIENT COUNTER.
             BEQ  L08            E638   F0 C5         GO TO SHIFT LEFT 4 BITS QUOTIENT.
%
%-------------------------------------------------------------------------------
% 6 BYTE REGISTER RIGHT SHIFT
%      SHIFT BIT: IX, REGISTER HEAD ADDRESS LOWER: IY, REGISTER HEAD ADDRESS UPPER: ACC
%-------------------------------------------------------------------------------
SHIFT_R_NBIT2STY  P0             E63A   84 40
             STA  P0+1           E63C   85 41
L67          CLC                 E63E   18
             LDY  #$05           E63F   A0 05
L66          LDA  (P0),Y         E641   B1 40
             ROR                 E643   6A
             STA  (P0),Y         E644   91 40
             DEY                 E646   88
             BPL  L66            E647   10 F8
             DEX                 E649   CA
             BNE  L67            E64A   D0 F2
             RTS                 E64C   60
%
%-------------------------------------------------------------------------------
% POST-PROCESSING OF DIVISION.
%      SHIFTING QUOTIENT AND EXPONENT PART CORRECTION.
%-------------------------------------------------------------------------------
AFTER_OPE_2  LDA  DR2_3          E64D   A5 2B         IF THE BYTE ABOVE DECIMAL POINT
             BEQ  L01            E64F   F0 16          IN THE QUOTIENT REGISTER IS ZERO,
             LDX  #$04           E651   A2 04          IT IS NOT SHIFTED.
             LDY  #$28           E653   A0 28
             LDA  #$00           E655   A9 00
             JSR  SHIFT_R_NBIT2  E657   20 3A E6      SHIFT 4 BIT RIGHT QUOTIENT REGISTER.
             LDA  R0_3           E65A   A5 13
             AND  #$7F           E65C   29 7F         REMOVE SIGN OF MANTISSA.
             CLC                 E65E   18
             ADC  #$01           E65F   69 01         EXPONENT + 1
             AND  #$7F           E661   29 7F         REMOVE D7
             ORA  SIGN_FLAG      E663   05 08         SYNTHESIS SIGN AND CORRECTED EXPONENT.
             STA  R0_3           E665   85 13
L01          LDX  #$00           E667   A2 00         MOVE DR2_0-DR2_2 TO DR0_0-DR0_2
L02          LDA  $28,X          E669   B5 28
             STA  $10,X          E66B   95 10
             INX                 E66D   E8
             CPX  #$03           E66E   E0 03
             BNE  L02            E670   D0 F7
             RTS                 E672   60
%
%-------------------------------------------------------------------------------
% PATCH FOR SHIFTING RIGHT 4BIT DIVISOR DR0.
%-------------------------------------------------------------------------------
DIV_PATCH_2  BEQ  L01            E673   F0 0C
             LDX  #$04           E675   A2 04         SHIFTING RIGHT 4BIT DIVISOR DR0.
             LDY  #$18           E677   A0 18
             LDA  #$00           E679   A9 00
             JSR  SHIFT_R_NBIT2  E67B   20 3A E6
```

```
             JMP  L10             E67E   4C E9 E5
L01          RTS                  E681   60
%
%-------------------------------------------------------------------------------
% POST-PROCESSING OF MULTIPLICATION.
%      SHIFTING RESULT AND EXPONENT PART CORRECTION.
%-------------------------------------------------------------------------------
AFTER_OPE_3  LDA  DR2_5           E682   A5 2D        EXTRACT MOST SIGNIFICANT 2 DIGIT OF RESULT.
             CMP  #$10            E684   C9 10
             BCS  L01             E686   B0 16        IF IT'S 10 OR MORE, DO NOT SHIFT.
             LDX  #$04            E688   A2 04        SHIFT LEFT 4 BITS RESULT REGISTER
             LDY  #$2E            E68A   A0 2E
             LDA  #$FF            E68C   A9 FF
             JSR  SHIFT_L_NBIT2   E68E   20 1F E5
             LDA  R0_3            E691   A5 13        REMOVE SIGN OF MANTISSA FROM EXPONENT BYTE.
             AND  #$7F            E693   29 7F
             JSR  MUL_PATCH_4     E695   20 A3 E6     EXPONENT - 1
             AND  #$7F            E698   29 7F        REMOVE D7
             ORA  SIGN_FLAG       E69A   05 08        SYNTHESIS SIGN AND CORRECTED EXPONENT.
             STA  R0_3            E69A   85 13
L01          LDX  #$00            E69E   A2 00
             JMP  AFTER_OPE_1     E6A0   4C 13 E5
%
%-------------------------------------------------------------------------------
MUL_PATCH_4  SEC                  E6A3   38
             PHA                  E6A4   48          PATCH FOR ERROR IN PROM MAKING (V1.2.1)
             PLA                  E6A5   68          PATCH FOR ERROR IN PROM MAKING (V1.2.1)
             SUB  #$01            E6A6   E9 01       IN THE CASE OF '00(BCD) - 01(BCD),
             CMP  #$99            E6A8   C9 99        CONVERT RESULT '99(BCD) TO 79(BCD).
             BNE  L02             E6AA   D0 02
             LDA  #$79            E6AC   A9 79
L02          RTS                  E6AE   60
%
%-------------------------------------------------------------------------------
% PATCH FOR FOUR ARITHMETIC OPERATIONS UNIT TEST.
%-------------------------------------------------------------------------------
             .ORIGIN  $E6B0
%
GET_FLOAT_4  LDA  LINE_HEAD,X     E6B0   BD 00 02
             CMP  #$20            E6B3   C9 20        IS IT SPACE CODE?
             BEQ  G_FLOAT_4_L12   E6B5   F0 03
             JMP  L13             E6B7   4C FF E0     GO TO PROCESSING EXCEPT FOR SPACE CODE
G_FLOAT_4_L12INX                  E6BA   E8           AND ARITHMETIC OPERATIONS.
             LDA  LINE_HEAD,X     E6BB   BD 00 02
             CMP  #$2B            E6EB   C9 2B        IS THE '+' CODE NEXT TO THE SPACE CODE?
             BNE  L14             E6C0   D0 12
             INX                  E6C2   E8
             LDA  LINE_HEAD,X     E6C3   BD 00 02
             CMP  #$0D            E6C6   C9 0D        IS THE 'CR' CODE NEXT TO THE SPACE CODE?
             BNE  L15             E6C8   D0 09
             JSR  CORRECT_EXP_1   E6CA   20 6B E1     EXPONENT CORRECTION.
             JSR  NORMALISE_1     E6CD   20 D0 E2     NORMALISE.
             JSR  ADD_FLOAT_1     E6D0   20 D0 E3     ADDITION.
L15          RTS                  E6D3   60
%
L14          CMP  #$2D            E6D4   C9 2D        IS IT '-' CODE?
             BNE  L16             E6D6   D0 12
             INX                  E6D8   E8
             LDA  LINE_HEAD,X     E6D9   BD 00 02
             CMP  #$0D            E6DC   C9 0D        IS THE 'CR' CODE NEXT TO THE '-' CODE?
             BNE  L17             E6DE   D0 36
             JSR  CORRECT_EXP_1   E6E0   20 6B E1     EXPONENT CORRECTION.
             JSR  NORMALISE_1     E6E3   20 D0 E2     NORMALISE.
             JSR  SUB_FLOAT_1     E6E6   20 E5 E3     SUBTRACTION.
             RTS                  E6E9   60
%
L16          CMP  #$2A            E6EA   C9 2A        IS IT '*' CODE?
             BNE  L18             E6EC   D0 12
             INX                  E6EE   E8
             LDA  LINE_HEAD,X     E6EF   BD 00 02
             CMP  #$0D            E6F2   C9 0D        IS THE 'CR' CODE NEXT TO THE '*' CODE?
             BNE  L19             E6F4   D0 09
```

```
              JSR  CORRECT_EXP_1  E6F6   20 6B E1      EXPONENT CORRECTION.
              JSR  NORMALISE_1    E6F9   20 D0 E2      NORMALISE.
              JSR  MUL_FLOAT_1    E6FC   20 A0 E4      MULTIPLICATION.
L19           RTS                 E6FF   60
%
L18           CMP  #$2F           E700   C9 2F         IS IT ‘/‘ CODE?
              BNE  L20            E702   D0 16
              INX                 E704   E8
              LDA  LINE_HEAD,X    E705   BD 00 02
              CMP  #$0D           E708   C9 0D         IS THE ‘CR’ CODE NEXT TO THE ‘/‘ CODE?
              BNE  L21            E70A   D0 09
              JSR  CORRECT_EXP_1  E70C   20 6B E1      EXPONENT CORRECTION.
              JSR  NORMALISE_1    E70F   20 D0 E2      NORMALISE.
              JSR  DIV_FLOAT_1    E712   20 70 E5      DIVISION.
L21           RTS                 E715   60
%
L17           DEX                 E716   CA           RETURN THE POINTER TO SPACE CODE POSITION.
              NOP                 E717   EA
              NOP                 E718   EA
              NOP                 E719   EA
L20           DEX                 E71A   CA
              STX  TEMP_5         E71B   86 0E
              JSR  CORRECT_EXP_1  E71D   20 6B E1      EXPONENT CORRECTION.
              JSR  NORMALISE_1    E720   20 D0 E2      NORMALISE.
              LDA  R0_0           E723   A5 10         MOVE R0 TO R1.
              STA  R1_0           E725   85 14
              LDA  R0_1           E727   A5 11
              STA  R1_1           E729   85 15
              LDA  R0_2           E72B   A5 12
              STA  R1_2           E72D   85 16
              LDA  R0_3           E72F   A5 13
              STA  R1_3           E72F   85 17
              JSR  INIT_REG       E733   20 B3 E1      CLEAR R0.
              LDX  TEMP_5         E736   A6 0E
              INX                 E738   E8
              JMP  GET_FLOAT_4    E739   4C 72 E4      RETURN TO ANALYZE ONE CHARACTER.
%
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
%------------------------------   LINE EDITOR   --------------------------------
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
              .ORIGIN  $E740
%
LINE_EDITOR_1 LDA  LINE_HEAD_L    E740   A9 00
              STA  P0             E742   85 40
              LDA  LINE_HEAD_H    E744   A9 02
              STA  P0+1           E746   85 41
              JSR  GET_LINE_LENG  E748   20 C5 E7      GET THE LENGTH OF A NEW LINE.
              STY  LINE_LENGTH_1  E74B   84 04
              LDX  #LR0_ADDRESS   E74D   A2 00
              JSR  GET_LINE_NUMB  E74F   20 3C E8      GET THE LINE NUMBER OF A NEW LINE.
              LDA  #PROG_PO_1_L   E752   A9 00
              STA  P0             E754   85 40
              LDA  #PROG_PO_1_H   E756   A9 10
              STA  P0+1           E758   85 41         SET PROGRAM AREA HEAD ADDRESS TO P0.
L01           LDX  #LR1_ADDRESS   E75A   A2 02
              JSR  GET_LINE_NUMB  E75C   20 3C E8      GET THE LINE NUMBER OF PROGRAM AREA.
              JSR  CMP16_1        E75F   20 69 E8      COMPARE BOTH LINE NUMBERS.
              BEQ  SWAP_1         E762   F0 0B         IF IT’S A SAME, EXCHANGE THE LINES.
              BCS  SEARCH_NEXT    E764   B0 0F         IF LR1<LR0, SEARCH THE NEXT LINE NUMBER.
INSERT_LINE   LDY  LINE_LENGTH_1  E766   A4 04         WHEN LR1>LR0, MOVE FORWARD THE NEXT LARGER
              JSR  MOV_BLOCK_F1   E768   20 16 E8       LINE NUMBER AFTER THE NEW LINE.
              JMP  LINE_EDITOR_2  E76B   4C F5 F0      INSERT NEW LINE.
              NOP                 E76E   EA
%
SWAP_1        JSR  DELETE_LINE    E76F   20 87 E7      DELETE THE LINE.
              CLC                 E772   18
              BCC  INSERT_LINE    E773   90 F1
%
SEARCH_NEXT   JSR  SEARCH_NEXT_L  E775   20 A3 E7      ADVANCE THE POINTER TO THE NEXT LINE NUMBER
              BCC  L01            E778   90 E0
```

```
             JSR  PUT_NEW_LINE    E77A   20 92 E7
             JSR  SEARCH_NEXT_L   E77D   20 A3 E7
             LDA  #$25            E780   A9 25
             LDY  #$00            E782   A0 00
             STA  (P0),Y          E784   91 40
             RTS                  E786   60
%
%-------------------------------------------------------------------------------
% DELETE THE CURRENT LINE POINTED P0, AND PADDING THE PROGRAM BACK.
%-------------------------------------------------------------------------------
DELETE_LINE  JSR  GET_LINE_LENG   E787   20 C5 E7     GET THE LENGTH OF THE CURRENT LINE.
             STY  LINE_LENGTH_2   E78A   84 05
             NOP                  E78C   EA
             NOP                  E78D   EA
             JSR  MOV_BLOCK_B2    E78E   20 74 E8     PADDING THE PROGRAM BACK
             RTS                  E791   60            BY LENGTH OF CURRENT LINE.
%
%-------------------------------------------------------------------------------
% MOVE THE NEW LINE TO THE POSITION POINTED P0. IF THE LENGTH OF NEW LINE = 0, NOTHING.
%-------------------------------------------------------------------------------
PUT_NEW_LINE LDA  LINE_LENGTH_1   E792   A5 04
             BEQ  L02             E794   F0 0C
             LDY  #$00            E796   A0 00        INITIALIZE POINTER.
L01          LDA  LINE_HEAD,Y     E798   B9 00 02     LOAD ONE CHARACTER FROM LINE BUFFER.
             STA  (P0),Y          E79B   91 40        STORE THE CHARACTER TO PROGRAM AREA.
             INY                  E79D   C8
             CMP  #$0D            E79E   C9 0D        IF IT'S 'CR' CODE, EXIT.
             BNE  L01             E7A0   D0 F6
L02          RTS                  E7A2   60
%
%-------------------------------------------------------------------------------
% ADVANCE THE POINTER P0,P0+1 TO THE HEAD OF THE NEXT LINE NUMBER.
%      IF THE NEXT LINE NUMBER DOES NOT EXIST, SET CARRY FLAG.       (V.1.5.0  MAR/11/2023)
%-------------------------------------------------------------------------------
SEARCH_NEXT_LLDA  #$FF            E7A3   A9 FF        LIMIT LOWER 8BIT
             STA  P1              E7A5   85 42
             LDA  MEM_END_U       E7A7   A5 82        LIMIT UPPER 8BIT (V.1.5.0  MAR/11/2023)
             STA  P1+1            E7A9   85 43        THE END OF PROGRAM AREA.
             LDY  #$00            E7AB   A0 00
L03          LDA  (P0),Y          E7AD   B1 40        LOAD ONE CHARACTER FROM THE PROGRAM.
             PHA                  E7AF   48           SAVE ACC.
             JSR  INC_POINTER_1   E7B0   20 FC E7     ADVANCE THE POINTER.
             PLA                  E7B3   68
             CMP  #$0D            E7B4   C9 0D        IF IT'S 'CR' CODE, EXIT.
             JMP  PATCH_2         E7B6   4C 84 E8
%
SEARCH_N_L01 CMP  #$25            E7B9   C9 25        IF IT'S '%' CODE, SET CARRY AND EXIT.
             BNE  L02             E7BB   D0 02
             SEC                  E7BD   38
             RTS                  E7BE   60
%
L02          JSR  CMP_POINTER_1   E7BF   20 31 E8     IF IT'S END OF PROGRAM AREA, EXIT.
             BNE  L03             E7C2   D0 E9
             RTS                  E7C4   60
%
%-------------------------------------------------------------------------------
% MEASURE THE LENGTH OF CURRENT POINTER POSITION BY P0,P0+1 TO THE NEXT 'CR' CODE.
%      INPUT PARAMETER: P0,P0+1,  OUTPUT PARAMETER: IY (LINE LENGTH)
%-------------------------------------------------------------------------------
GET_LINE_LENGLDY  #$00            E7C5   A0 00
L02          LDA  (P0),Y          E7C7   B1 40
             CMP  #$0D            E7C9   C9 0D
             BEQ  L01             E7CB   F0 03
             INY                  E7CD   C8
             BNE  L02             E7CE   D0 F7
L01          DEY                  E7D0   88
             LDA  (P0),Y          E7D1   B1 40
             CMP  #$3A            E7D3   C9 3A
             BNE  L03             E7D5   D0 03
             LDY  #$00            E7D7   A0 00
             RTS                  E7D9   60
%
```

```
L03           INY                 E7DA   C8
              INY                 E7DB   C8
              RTS                 E7DC   60
%
%-------------------------------------------------------------------------------
% BLOCK TRANSFER TO SMALLER MEMORY ADDRESS.
%      DESTINATION HEAD ADDRES: P0,P0+1,  TRANSFER ADDRESS OFFSET: IY    (V.1.5.0  MAR/11/2023)
%-------------------------------------------------------------------------------
MOV_BLOCK_B1  LDA  #$FF           E7DD   A9 FF        LIMIT LOWER 8BIT
              STA  P1             E7DF   85 42
              LDA  MEM_END_U      E7E1   A5 82        LIMIT UPPER 8BIT (V.1.5.0  MAR/11/2023)
              STA  P1+1           E7E3   85 43
              STY  TEMP_0         E7E5   84 09        SAVE IY.
L03           LDY  TEMP_0         E7E7   A4 09        RESTORE IY.
              LDA  (P0),Y         E7E9   B1 40        LOAD FROM SOURCE POSITION.
              LDY  #$00           E7EB   A0 00
              STA  (P0),Y         E7ED   91 40        STORE TO DESTINATION POSITION.
              CMP  #$25           E7EF   C9 25        IF IT'S '%' CODE, EXIT.
              BEQ  L04            E7F1   F0 08
              JSR  INC_POINTER_1  E7F3   20 FC E7
              JSR  CMP_POINTER_1  E7F6   20 31 E8
              BNE  L03            E7F9   D0 EC
L04           RTS                 E7FB   60
%
%-------------------------------------------------------------------------------
% INCREMENT 16 BIT POINTER P0,P0+1.
%-------------------------------------------------------------------------------
INC_POINTER_1 CLD                 E7FC   D8
              CLC                 E7FD   18
              LDA  P0             E7FE   A5 40
              ADC  #$01           E800   69 01        LOWER 8 BIT +1
              STA  P0             E802   85 40
              BCC  L05            E804   90 02
              INC  P0+1           E806   E6 41        IF CARRY EXIST LOWER 8 BIT, UPPER 8BIT +1.
L05           RTS                 E808   60
%
%-------------------------------------------------------------------------------
% DECREMENT 16 BIT POINTER P1,P1+1.
%-------------------------------------------------------------------------------
DEC_POINTER_1 CLD                 E809   D8
              SEC                 E80A   38
              LDA  P1             E80B   A5 42
              SBC  #$01           E80D   E9 01        LOWER 8 BIT -1
              STA  P1             E80F   85 42
              BCS  L06            E811   B0 02        IF BORROW EXIST LOWER 8 BIT, UPPER 8BIT -1.
              DEC  P1+1           E813   C6 43
L06           RTS                 E815   60
%
%-------------------------------------------------------------------------------
% BLOCK TRANSFER TO LARGER MEMORY ADDRESS.
%      SOURCE HEAD ADDRES: P0,P0+1,  TRANSFER ADDRESS OFFSET: IY       (V.1.5.0  MAR/11/2023)
%-------------------------------------------------------------------------------
MOV_BLOCK_F1  LDA  #$00           E816   A9 00        AREA LIMIT ADDRESS - 255 BYTE
              STA  P1             E818   85 42
              LDA  MEM_END_U      E81A   A5 82        LIMIT UPPER 8BIT (V.1.5.0 MAR/11/2023)
              STA  P1+1           E81C   85 43
              STY  TEMP_0         E81E   84 09
L07           LDY  #$00           E820   A0 00
              LDA  (P1),Y         E822   B1 42
              LDY  TEMP_0         E824   A4 09
              STA  (P1),Y         E826   91 42
              JSR  DEC_POINTER_1  E828   20 09 E8
              JSR  CMP_POINTER_1  E82B   20 31 E8
              BCS  L07            E82E   B0 F0
              RTS                 E830   60
%
%-------------------------------------------------------------------------------
% COMPARE 16 BIT POINTER P1,P1+1 AND P0,P0+1.
%-------------------------------------------------------------------------------
CMP_POINTER_1 LDA  P1+1           E831   A5 43
              CMP  P0+1           E833   C5 41        COMPARE UPPER 8 BIT
              BNE  L01            E835   D0 04
```

```
             LDA  P1              E837   A5 42
             CMP  P0              E839   C5 40        COMPARE LOWER 8 BIT
L01          RTS                  E83B   60
%
%-------------------------------------------------------------------------------
% CONVERT ASCII CODE LINE NUMBER TO 2 BYTE BCD VALUE.
%      INPUT PARAMETER: P0,P0+1,  OUTPUT PARAMETER: IY (LINE NUMBER REGISTER ADDRESS)
%-------------------------------------------------------------------------------
GET_LINE_NUMB LDA  #$00           E83C   A9 00
             STA  $00,X           E83E   95 00
             STA  $01,X           E840   95 01
             LDY  #$00            E842   A0 00
L10          JMP  PATCH_3         E844   4C B8 E8
             NOP                  E847   EA
GET_LINE_L14 BEQ  L10             E848   F0 10
             CMP  #$20            E84A   C9 20        IGNORE SPACE CODE.
             BEQ  L13             E84C   F0 09
             JSR  SHIFT_L_NUM1    E84E   20 5B E8     4 BIT LEFT SHIFT LINE NUMBER REGISTER.
             AND  #$0F            E851   29 0F        EXTRACT LOWER 4 BIT OF ASCII CODE.
             ORA  $00,X           E853   15 00        SYNTHESIZE LINE NUMBER LOWER.
             STA  $00,X           E855   95 00
L13          INY                  E857   C8           ADVANCE LINE BUFFER POINTER.
             BNE  L10             E858   D0 EA
             RTS                  E85A   60
%
%-------------------------------------------------------------------------------
% 4 BIT LEFT SHIFT LINE NUMBER REGISTER.
%-------------------------------------------------------------------------------
SHIFT_L_NUM1 STY  TEMP_0          E85B   84 09        SAVE IY.
             LDY  #$04            E85D   A0 04        SET 4 BIT SHIFT.
L11          ASL  $00,X           E85F   16 00        LEFT SHIFT  2-BYTE SECTION.
             ROL  $01,X           E861   36 01
             DEY                  E863   88
             BNE  L11             E864   D0 F9
             LDY  TEMP_0          E866   A4 09        RESTORE IY.
             RTS                  E868   60
%
%-------------------------------------------------------------------------------
% COMPARE LINE NUMBER REGISTERS
%-------------------------------------------------------------------------------
CMP16_1      LDA  LR0_1           E869   A5 01        COMPARE UPPER 8 BIT.
             CMP  LR1_1           E86B   C5 03        IF LR1>LR0, SET BORROW (CLEAR CARRY)
             BNE  L12             E86D   D0 04
             LDA  LR0_0           E86F   A5 00        IF UPPER IS SAME, COMPARE LOWER.
             CMP  LR1_0           E871   C5 02        IF LR1>LR0, SET BORROW (CLEAR CARRY)
L12          RTS                  E873   60
%
%-------------------------------------------------------------------------------
% PATCH FOR MOVE_BLOCK_B1 TO SAVE P0,P0+1.
%-------------------------------------------------------------------------------
MOVE_BLOCK_B2 LDA  P0             E874   A5 40
             PHA                  E876   48
             LDA  P0+1            E877   A5 41
             PHA                  E879   48
             JSR  MOVE_BLOCK_B1   E87A   20 DD E7
             PLA                  E87D   68
             STA  P0+1            E87E   85 41
             PLA                  E880   68
             STA  P0              E881   85 40
             RTS                  E883   60
%
%-------------------------------------------------------------------------------
% PATCH FOR SEARCH_NEXT_L TO CLEAR CARRY WHEN ‘CR’ CODE IS DETECTED AND EXIT.
%-------------------------------------------------------------------------------
PATCH_2      BNE  L01             E884   D0 02
             CLC                  E886   18
             RTS                  E887   60
L01          JMP  SEARCH_N_L01    E888   4C B9 E7
%
%-------------------------------------------------------------------------------
% EDITOR UNIT TEST WITH SERIAL INTERFACE.
%-------------------------------------------------------------------------------
```

```
EDITOR_TEST_2 CLC                 E88B   18
              LDX  #$FF           E88C   A2 FF
              TXS                 E88E   0A
              JSR  INIT_ACIA      E88F   20 80 E0      INITIALIZE SERIAL INTERFACE
L01           JSR  IN_1LINE       E892   20 C0 E0      INPUT ONE LINE.
              JSR  LINE_EDITOR_1  E895   20 40 E7      LINE EDITOR.
              JSR  DISP_PROGRAM1  E898   20 9E E8      DISPLAY PROGRAM.
              JMP  L01            E89B   4C 92 E8
%
%-------------------------------------------------------------------------------
DISP_PROGRAM1 LDA  #PROG_PO_1_L   E89E   A9 00
              STA  P0             E8A0   85 40
              LDA  #PROG_PO_1_H   E8A2   A9 10
              STA  P0+1           E8A4   85 41
              LDY  #$00           E8A6   A0 00         SET PROGRAM AREA TO POINTER.
L03           LDA  (P0),Y         E8A8   B1 40         LOAD ONE CHARACTER FROM PROGRAM AREA.
              CMP  #$25           E8AA   C9 25         IF IT'S '%' CODE, EXIT.
              BEQ  L02            E8AC   F0 09
              JSR  OUT_1CHA       E8AE   20 9A E0      OUTPUT ONE CHARACTER TO SERIAL INTERFACE.
              JSR  INC_POINTER_1  E8B1   20 FC E7
              CLC                 E8B4   18
              BCC  L03            E8B5   90 F1
L02           RTS                 E8B7   60
%
%-------------------------------------------------------------------------------
% PATCH FOR GET_LINE_NUMB
%      WHEN THE END CODE IS DETECTED, RETURN THE VALUE 9999 TO LR.
%      ENSURE THAT LINE NUMBERS WITH GREATER THAN CURRENT MAXIMUM LINE NUMBER ARE INSERTED CORRECTLY.
%-------------------------------------------------------------------------------
PATCH_3       LDA  (P0),Y         E8B8   B1 40         LOAD ONE CHARACTER.
              CMP  #$25           E8BA   C9 25         IS IT '%' CODE?
              BEQ  L03            E8BC   F0 05
              CMP  #$3A           E8BE   C9 3A         IS IT ':' CODE?
              JMP  GET_LINE_L14   E8C0   4C 48 E8      RETURN FROM THIS PATCH.
%
L03           LDA  #$99           E8C3   A9 99         SET 9999 TO LR
              STA  $00,X          E8C5   95 00
              STA  $01,X          E8C7   95 01
              RTS                 E8C9   60
%
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
%--------------------------  ANALYSIS AND EXECUTION   ---------------------------
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------

% CI-2 MAIN PROGRAM
%-------------------------------------------------------------------------------
              .ORIGIN  $E8D0
%
CI-2_MAIN     LDX  #$FF           E8D0   A2 FF         INITIALIZE STACK POINTER.
              TXS                 F8D2   9A
              JSR  INIT_ACIA      E8D3   20 80 E0      INITIALIZE SERIAL INTERFACE.
              JSR  INIT_RUNFLAG   E8D6   20 0D ED      INITIALIZE RUN FLAG.
              JSR  GREETING       E8D9   20 E0 F2      GREETING MESSAGE.
L01           JSR  IN_1LINE_2     E8DC   20 B9 F0      ENTER A SINGLE LINE.
              JSR  ANA_1LINE      E8DF   20 F0 E8      ANALYSIS & EXECUTE SINGLE LINE.
              JMP  L01            E8E2   4C DC E8      REPEAT FROM PROMPT.
%
%-------------------------------------------------------------------------------
% MAIN LOOP OF ANALYSIS & EXECUTE SINGLE LINE.
%-------------------------------------------------------------------------------
              .ORIGIN  $E8F0
%
ANA_1LINE     JSR  SET_PO_LB      E8F0   20 64 EA      SET POINTER TO LINE BUFFER BEGINNING.
              JSR  DET_LINE_NUM1  E8F3   20 14 ED      LINE NUMBER DETECTION.
              BCC  MAIN_LOOP_6    E8F6   90 03
              JMP  LINE_EDITOR_1  E8F8   4C 40 E7      IF A LINE NUMBER IS DETECTED, GO TO EDITOR.
MAIN_LOOP_6   JSR  INIT_REG_4     E8FB   20 97 F0      CLEAR R0 ETC.
MAIN_LOOP_5   LDY  #$00           E8FE   A0 00
MAIN_LOOP_1   JMP  MAIN_LOOP_3    E900   4C FA EC
MAIN_LOOP_4   BCS  MAIN_LOOP_2    E903   B0 03         IGNORE THE LINE NUMBER
```

```
             JSR  SET_P0_LB       E905   20 64 EA
MAIN_LOOP_2  LDA  (P0),Y          E908   B1 40        IF NO LINE NUMBER, ANALYSE SINLE CHARACTER.
             CMP  #$20            E90A   C9 20
             BNE  L03             E90C   D0 03        IF IT'S A SPACE CODE,
             JMP  IN_NEW_VAR      E90E   4C 13 EA       START ENTERING A NEW VARIABLE.
L03          CMP  #$21            E911   C9 21
             BNE  L04             E913   D0 03
             JMP  RUN_CONT        E915   4C 95 EB     IF IT'S '!' CODE, GO TO EXECUTION.
L04          CMP  #$23            E918   C9 23
             BNE  L05             E91A   D0 03
             JMP  COMMENT         E91C   4C F9 ED     IF IT'S '#' CODE, GO TO COMMENT PROCESS,
L05          CMP  #$24            E91F   C9 24
             BNE  L06             E921   D0 03
             JMP  POINTER_VAR_1   E923   4C 86 EF     IF IT'S '$' CODE, GO TO POINTER VARIABLES.
L06          CMP  #$29            E926   C9 29
             JMP  MAIN_LOOP_6     E928   4C DE F1     GO TO MAIN_LOOP_6
             NOP                  E92B   EA
             NOP                  E92C   EA
MAIN_LOOP_7  CMP  #$2B            E92D   C9 2B        RETURN FROM MAIN_LOOP_6
             BNE  L08             E92F   D0 03
             JMP  ADD_1           E931   4C 26 EB     IF IT'S '+' CODE, GO TO ADDITION.
L08          CMP  #$2D            E934   C9 2D
             BNE  L09             E936   D0 03
             JMP  SUB_2           E938   4C A8 ED     IF IT'S '-' CODE, GO TO SUBTRACTION.
L09          CMP  #$2A            E93B   C9 2A
             BNE  L10             E93D   D0 03
             JMP  MUL_1           E93F   4C 54 EB     IF IT'S '*' CODE, GO TO MULTIPLICATION.
L10          CMP  #$2F            E942   C9 2F
             BNE  L11             E944   D0 03
             JMP  DIV_1           E946   4C 67 EB     IF IT'S '/' CODE, GO TO DIVISION.
L11          CMP  #$2E            E949   C9 2E
             BNE  L12             E94B   D0 03
             JMP  DEC_POINT       E94D   4C 0B EA     IF IT'S '.' CODE, GO TO DECIMAL POINT.
L12          CMP  #$45            E950   C9 45
             BNE  L13             E952   D0 03
             JMP  GET_EXP_4       E954   4C C7 E9     IF IT'S 'E' CODE, GO TO EXPONENT PROCESS.
L13          CMP  #$3B            E957   C9 3B
             BNE  L14             E959   D0 03
             JMP  DISP_CONT       E95B   4C A2 EA     IF IT'S ';' CODE, GO TO OUTPUT PROCESS.
L14          CMP  #$3D            E95E   C9 3D
             BNE  L15             E960   D0 03
             JMP  SUBSTITUTE_3    E962   4C 6D EA     IF IT'S '='CODE, GO TO SUBSTITUTE BOOKING.
L15          CMP  #$3F            E965   C9 3F
             BNE  L16             E967   D0 03
             JMP  CONDITION_1     E969   4C 37 ED     IF IT'S '?' CODE, GO TO CONDITION PROCESS.
L16          CMP  #$40            E96C   C9 40
             BNE  L17             E96E   D0 03
             JMP  DISP_LIST_2     E970   4C F3 F0     IF IT'S '@' CODE, GO TO PROGRAM DUMP.
L17          CMP  #$41            E973   C9 41
             BCC  L18             E975   90 03
             JMP  VARIABLE        E977   4C CB EA     IF $41 OR MORE, GO TO VARIABLE PROCESS.
L19          CMP  #$0D            E97A   C9 0D
             BNE  L20             E97C   D0 03
             JMP  ENDLINE_PROC1   E97E   4C B9 ED     IF IT'S 'CR' CODE, GO TO END LINE PROCESS.
L20          CMP  #$25            E981   C9 25        IF IT'S '%' CODE, TERMINATE EXECUTION.
             BNE  L34             E983   D0 02
             BEQ  L36             E985   F0 0A
L34          CMP  #$30            E987   C9 30
             BCS  L35             E989   B0 02
             BCC  L36             E98B   90 04
L35          CMP  #$3A            E98D   C9 3A
             BCC  GET_FLOAT_5     E98F   90 03        IF $30-$39 CODE, GO TO NUMERIC INPUT.
L36          JMP  STOP_RUNNING    E991   4C B1 EA
%
%-------------------------------------------------------------------------------
GET_FLOAT_5  LDX  TEMP_6          E994   A6 0F
             BEQ  L21             E996   F0 02
             DEC  TEMP_3          E998   C6 0C
L21          STY  TEMP_5          E99A   84 0E
             STA  TEMP_2          E99C   85 0B
             LDX  #$04            E99E   A2 04
             LDY  #$13            E9A0   A0 13
```

```
             LDA  #$FF           E9A2   A9 FF
             JSR  SHIFT_L_NBIT1  E9A4   20 54 EC
             LDY  TEMP_5         E9A7   A4 0E
             LDA  TEMP_2         E9A9   A5 0B
             AND  #$0F           E9AB   29 0F
             ORA  R0_0           E9AD   05 10
             STA  R0_0           E9AF   85 10
GET_FLOAT_6  JSR  INC_POINTER_1  E9B1   20 FC E7
             LDA  (P0),Y         E9B4   B1 40
             JSR  DET_CHR_1      E9B6   20 4E EA
             BCS  L22            E9B9   B0 03
             JMP  MAIN_LOOP_2    E9BB   4C 08 E9
L22          JSR  CORRECT_EXP_1  E9BE   20 6B E1
             JRT  NORMALISE_1    E9C1   20 D0 E2
             JMP  MAIN_LOOP_2    E9C4   4C 08 E9
%
%-------------------------------------------------------------------------------
GET_EXP_4    JSR  INC_POINTER_1  E9C7   20 FC E7
L25          LDA  (P0),Y         E9CA   B1 40        ANALYZE THE NEXT CHARACTER OF 'E'
             CMP  #$2D           E9CC   C9 2D        IS IT A '-' CODE?
             BEQ  L23            E9CE   F0 33
GET_EXP_2    STA  TEMP_2         E9D0   85 0B        SAVE THE ACCUMULATOR VALUE.
             LDA  R0_3           E9D2   A5 13
             STA  TEMP_0         E9D4   85 09        SAVE THE R0_3 VALUE.
             ASL                 E9D6   0A
             ASL                 E9D7   0A
             ASL                 E9D8   0A
             ASL                 E9D9   0A
             AND  #$30           E9DA   29 30        EXTRACT D5,D4.
             STA  TEMP_1         E9DC   85 0A
             LDA  TEMP_0         E9DE   A5 09
             AND  #$C0           E9E0   29 C0        EXTRACT D7,D6 OF R0_3.
             ORA  TEMP_1         E9E2   05 0A
             STA  R0_3           E9E4   85 13
             LDA  TEMP_2         E9E6   A5 0B        THE NEW ASCII CARACTER.
             AND  #$0F           E9E8   29 0F        EXTRACT LOWER 4BIT OF THE ASCII CODE.
             ORA  R0_3           E9EA   05 13
             STA  R0_3           E9EC   85 13
             JSR  INC_POINTER_1  E9EE   20 FC E7     ADVANCE THE POINTER TO THE NEXT.
             LDA  (P0),Y         E9F1   B1 40
             JSR  DET_CHR_2      E9F3   20 56 EA
             BCS  L24            E9F6   B0 02
             BCC  L25            E9F8   90 D0
L24          JSR  CORRECT_EXP_1  E9FA   20 6B E1     IF THE NEXT CHARACTER IS NOT A NUMBER,
             JSR  NORMALISE_1    E9FD   20 D0 E2      DO EXPONENTIAL CORRECTION & NORMALIZATION
             JMP  MAIN_LOOP_2    EA00   4C 08 E9      AND RETURN TO THE MAIN LOOP 2.
%
%-------------------------------------------------------------------------------
% PROCESS OF EXPONENT '-'.
%-------------------------------------------------------------------------------
L23          LDA  TEMP_7         EA03   A5 08
             ORA  #$40           EA05   09 40
             STA  TEMP_7         EA07   85 08        SET '1' TO SIGN FLAG D6.
             BNE  GET_EXP_4      EA09   D0 BC
%
%-------------------------------------------------------------------------------
% PROCESS OF DECIMAL POINT.
%-------------------------------------------------------------------------------
DEC_POINT    INC  TEMP_6         EA0B   E6 0F        SET DECIMAL POINT FLAG
             JSR  INC_POINTER_1  EA0D   20 FC E7     ADVANCE THE POINTER TO THE NEXT.
             JMP  MAIN_LOOP_2    EA10   4C 08 E9     RETURN TO THE MAIN LOOP 2.
%
%-------------------------------------------------------------------------------
% PROCESS OF SPACE CODE.
%-------------------------------------------------------------------------------
IN_NEW_VAR   JSR  PUSH_FP_REG    EA13   20 1F EA     PUSH THE ARITHMETIC STACK.
             JSR  INIT_REG_2     EA16   20 FA EA     INITIALIZE R0,
             JSR  INC_POINTER_1  EA19   20 FC E7      ALLOW THE NEXT NUMBER TO BE ENTERED.
             JMP  MAIN_LOOP_2    EA1C   4C 08 E9     RETURN TO THE MAIN LOOP 2.
%
%-------------------------------------------------------------------------------
% PUSH THE ARITHMETIC REGISTER STACK.
```

```
%-------------------------------------------------------------------------------------
PUSH_FP_REG   STY  TEMP_5          EA1F   84 0E        SAVE THE IY.
              LDX  #$30            EA21   A2 30
              LDY  #$34            EA23   A0 34
              JSR  MOV_FP_REG      EA25   20 39 EA     MOVE R1 TO R3.
              LDX  #$14            EA28   A2 14
              LDY  #$30            EA2A   A0 30        MOVE R1 TO T2.
              JSR  MOV_FP_REG      EA2C   20 39 EA
              LDX  #$10            EA2F   A2 10
              LDY  #$14            EA31   A0 14        MOVE R0 TO R1.
              JSR  MOV_FP_REG      EA33   20 39 EA
              LDY  TEMP_5          EA36   A4 0E
              RTS                  EA38   60
%
%-------------------------------------------------------------------------------------
% MOVE REGISTER. SOURCE HEAD ADDRESS: IX, DESTINATION HEAD ADDRESS: IY
%-------------------------------------------------------------------------------------
MOV_FP_REG    LDA  $00,X           EA39   B5 00
              STA  $0000.Y         EA3B   99 00 00
              LDA  $01,X           EA3E   B5 01
              STA  $0001,Y         EA40   99 01 00
              LDA  $02,X           EA43   B5 02
              STA  $0002,Y         EA45   99 02 00
              LDA  $03,X           EA48   B5 03
              STA  $0003.Y         EA4A   99 03 00
              RTS                  EA4D   60
%
%-------------------------------------------------------------------------------------
% CHARACTER DETECTOR.  (IF IT'S A NUMBER,'.','E', CLEAR CARRY, OTHERS SET CARRY.)
%-------------------------------------------------------------------------------------
DET_CHR_1     CMP  #$2E            EA4E   C9 2E        '.' CODE?
              BEQ  L27             EA50   F0 10
              CMP  #$45            EA52   C9 45        'E' CODE?
              BEQ  L27             EA54   F0 0C
DET_CHR_2     CMP  #$30            EA56   C9 30        MORE THAN $30?
              BCS  L26             EA58   B0 02
              BCC  L28             EA5A   90 04
L26           CMP  #$3A            EA5C   C9 3A        UNDER $39?
              BCC  L27             EA5E   90 02
L28           SEC                  EA60   38
              RTS                  EA61   60
L27           CLC                  EA62   18           IF IT'S MORE THAN $30 & UNDER $39,
              RTS                  EA63   60            EXIT WITH CARRY CLEAR.
%
%-------------------------------------------------------------------------------------
% SET HEAD POSITION OF LINE BUFFER TO P0,P0+1.
%-------------------------------------------------------------------------------------
SET_P0_LB     LDA  #$00            EA64   A9 00
              STA  P0              EA66   85 40
              LDA  #$02            EA68   A9 02
              STA  $P0+1           EA6A   85 41
              RTS                  EA6C   60
%
%-------------------------------------------------------------------------------------
% SUBSTITUTE PATCH  (INITIALIZE R0 TO MAKE IT POSSIBLE T0 ENTER THE FOLLOWING NUMBERS.)
%-------------------------------------------------------------------------------------
SUBSTITUTE_3  JSR  INIT_REG_2      EA6D   20 FA EA
              LDA  VAR_0           EA70   A5 52
              STA  VAR_1           EA72   85 53        SET NORMAL VARIABLE OFFSET TO VAR_1.
              JSR  INC_POINTER_1   EA74   20 FC E7     ADVANCE THE POINTER.
              JMP  MAIN_LOOP_2     EA77   4C 08 E9     RETURN TO MAIN LOOP 2.

%
%-------------------------------------------------------------------------------------
              .ORIGIN  $EAA2
%
DISP_CONT     INC  DISP_FLAG       EAA2   E6 51
              JMP  VAR_RET_1       EAA4   4C E6 EA
%
%-------------------------------------------------------------------------------------
% SET HEAD POSITION OF PROGRAM AREA TO P0,P0+1.
%-------------------------------------------------------------------------------------
```

```
            .ORIGIN  $EAA8
%
SET_P0_PR   LDA  PROG_P0_1_L    EAA8   A9 00
            STA  P0             EAAA   85 40
            LDA  PROG_P0_1_H    EAAC   A9 10
            STA  P0+1           EAAE   85 41
            RTS                 EAB0   60
%
%-------------------------------------------------------------------------------
% TERMINATE THE EXECUTION.
%-------------------------------------------------------------------------------
STOP_RUNNING LDA  #$00          EAB1   A9 00
            STA  RUN_FLAG       EAB3   85 50
            RTS                 EAB5   50
%
%-------------------------------------------------------------------------------
% NORMAL VARIABLE PROCESS.
%      SHIFT THE LOWER 6 BITS OF THE VARIABLE CHARACTER BY 2 BIT
%      TO MAKE IT AN ADDRESS OFFSET, AND LOAD THE VARIABLE VALUE INTO R0.
%-------------------------------------------------------------------------------
            .ORIGIN  $EACB
%
VARIABLE    AND  #$3F           EACB   29 3F        EXTRACT LOWER 6 BITS OF ASCII CODE.
            ASL                 EACD   0A           SHIFT LEFT BY 2 BIT.
            ASL                 EACE   0A
            STA  VAR_0          EACF   85 52        SAVE THE DETECTED ADDRESS OFFSET.
            TAX                 EAD1   AA           MAKE THE IY THE ADDRESS OFFSET.
            LDA  VAR_HEAD,X     EAD2   BD 00 03     SUBSTITUTE VARIABLE VALUE TO R0.
            STA  R0_0           EAD5   85 10
            LDA  VAR_HEAD+1,X   EAD7   BD 01 03
            STA  R0_1           EADA   85 11
            LDA  VAR_HEAD+2,X   EADC   BD 02 03
            STA  R0_2           EADF   85 12
            JMP  VAR_PATCH_1    EAE1   4C A2 F2
            NOP                 EAE4   EA
            NOP                 EAE5   EA
VAR_RET_1   JSR  INC_POINTER_1  EAE6   20 FC E7     ADVANCE POINTER TO THE NEXT.
            JMP  MAIN_LOOP_2    EAE9   4C 08 E9     RETURN TO THE MAIN LOOP 2.
%
%-------------------------------------------------------------------------------
% INITIALIZE REGISTERS.
%-------------------------------------------------------------------------------
            .ORIGIN  $EAEE
%
INIT_REG_3  LDA  #$5C           EAEE   A9 5C        INITIALIZE MEMORY WRITE POINTER.
            STA  P3             EAF0   85 46
            LDA  #$00           EAF2   A9 00
            STA  P3+1           EAF4   85 47
INIT_REG_1  LDA  #$FC           EAF6   A9 FC        INITIALIZE SUBSTITUTE BOOKING ADDRESS.
            STA  VAR_1          EAF8   85 53
INIT_REG_2  LDA  #$06           EAFA   A9 06
            STA  TEMP_3         EAFC   85 0C        INITIALIZE DECIMAL POSITION COUNTER.
            LDA  #$00           EAFE   A9 00
            STA  R0_0           EB00   85 10        CLEAR R0.
            STA  R0_1           EB02   85 11
            STA  R0_2           EB04   85 12
            STA  R0_3           EB06   85 13
            STA  TEMP_6         EB08   85 0F        CLEAR DECIMAL FLAG.
            STA  TEMP_7         EB0A   85 08        CLEAR SIGN FLAG OF MANTISSA & EXPONENT.
            STA  DISP_FLAG      EB0C   85 51
            RTS                 EB0E   60
%
%-------------------------------------------------------------------------------
% ADDITION.         (R0 = R1 + R0)
%-------------------------------------------------------------------------------
            .ORIGIN  $EB26
%
ADD_1       STY  TEMP_10        EB26   84 54        SAVE IY.
            JSR  ADD_FLOAT_2    EB28   20 C5 EC     FLOATING POINT ADDITION.
            JSR  NORMALISE_1    EB2B   20 D0 E2     NORMALISE.
            JSR  PULL_FP_REG    EB2E   20 82 EB     PULL THE ARITHMETIC STACK.
            LDY  TEMP_10        EB31   A4 54
```

```
            JSR  INC_POINTER_1  EB33   20 FC E7     ADVANCE THE POINTER TO THE NEXT.
            JMP  MAIN_LOOP_2    EB36   4C 08 E9     RETURN TO THE MAIN LOOP 2.
%
%-------------------------------------------------------------------------------------
% SUBTRACTION.      (R0 = R1 - R0)
%-------------------------------------------------------------------------------------
SUB_1       INY                 EB39   C8           ADVANCE THE POINTER TO THE NEXT.
            LDA  (P0),Y         EB3A   B1 40
            CMP  #$30           EB3C   C9 30        THE NEXT CHARACTER IS MORE THAN $30
            BCS  MANT_SIGN      EB3E   B0 3A         VARIABLE,VALUE, '-' IS DETERMINED SIGN.
SUB1_1      DEY                 EB40   88
            STY  TEMP_10        EB41   84 54
            JSR  SUB_FLOAT_2    EB43   20 CF EC     FLOATING POINT SUBTRACTION.
            JSR  NORMALISE_1    EB46   20 D0 E2     NORMALISE.
            JSR  PULL_FP_REG    EB49   20 82 EB     PULL THE ARITHMETIC STACK.
            LDY  TEMP_10        EB4C   A4 54
            JSR  INC_POINTER_1  EB4E   20 FC E7     ADVANCE THE POINTER TO THE NEXT.
            JMP  MAIN_LOOP_2    EB51   4C 08 E9     RETURN TO THE MAIN LOOP 2.
%
%-------------------------------------------------------------------------------------
% MULTIPLICATION.   (R0 = R1 * R0)
%-------------------------------------------------------------------------------------
MUL_1       STY  TEMP_10        EB54   84 54        SAVE IY.
            JSR  MUL_FLOAT_2    EB56   20 D9 EC     FLOATING POINT MULTIPLICATION.
            JSR  NORMALISE_1    EB59   20 D0 E2     NORMALISE.
            JSR  PULL_FP_REG    EB5C   20 82 EB     PULL THE ARITHMETIC STACK.
            LDY  TEMP_10        EB5F   A4 54
            JSR  INC_POINTER_1  EB61   20 FC E7     ADVANCE THE POINTER TO THE NEXT.
            JMP  MAIN_LOOP_2    EB64   4C 08 E9     RETURN TO THE MAIN LOOP 2.
%
%-------------------------------------------------------------------------------------
% DIVISION.         (R0 = R1 / R0)
%-------------------------------------------------------------------------------------
DIV_1       STY  TEMP_10        EB67   84 54        SAVE IY.
            JSR  DIV_FLOAT_1    EB69   20 E3 EC     FLOATING POINT DIVISION.
            JSR  NORMALISE_1    EB6C   20 D0 E2     NORMALISE.
            JSR  PULL_FP_REG    EB6F   20 82 EB     PULL THE ARITHMETIC STACK.
            LDY  TEMP_10        EB72   A4 54
            JSR  INC_POINTER_1  EB74   20 FC E7     ADVANCE THE POINTER TO THE NEXT.
            JMP  MAIN_LOOP_2    EB77   4C 08 E9     RETURN TO THE MAIN LOOP 2.
%
%-------------------------------------------------------------------------------------
% SIGN PROCESS OF MANTISSA.
%-------------------------------------------------------------------------------------
MANT_SIGN   DEY                 EB7A   88           MOVE BACK THE POINTER
            LDA  #$80           EB7B   A9 80        SET '1' TO SIGN FLAG.
            STA  TEMP_7         EB7D   85 08
            JMP  GET_FLOAT_6    EB7F   4C B1 E9     RETURN TO THE NUMBER INPUT
%
%-------------------------------------------------------------------------------------
% PULL THE ARITHMETIC REGISTER STACK.
%-------------------------------------------------------------------------------------
PULL_FP_REG STY  TEMP_5         EB82   84 0E        SAVE IY.
            LDX  #$30           EB84   A2 30
            LDY  #$14           EB86   A0 14
            JSR  MOV_FP_REG     EB88   20 39 EA     MOVE R2 TO R1
            LDX  #$34           EB8B   A2 34
            LDY  #$30           EB8D   A0 30
            JSR  MOV_FP_REG     EB8F   20 39 EA     MOVE R3 TO R2
            LDY  TEMP_5         EB92   A4 0E
            RTS                 EB94   60
%
%-------------------------------------------------------------------------------------
% EXECUTION OF EACH LINE.
%-------------------------------------------------------------------------------------
RUN_CONT    LDA  RUN_FLAG       EB95   A5 50
            BNE  L01            EB97   D0 08
            INC  RUN_FLAG       EB99   E6 50        IF IT IS NOT RUN STATE,
            JSR  SET_P0_PR      EB9B   20 A8 EA      SET POINTER TO HEAD OF PROGRAM
            JMP  MAIN_LOOP_5    EB9E   4C FE E8      AND SET RUN FLAG AS '1'.
L01         JSR  SET_LINE_NUM   EBA1   20 B9 EB     IF IT IS RUN STATE,
            JSR  SET_P0_PR      EBA4   20 A8 EA      SET INTEGER PART OF R0 TO LR0
```

```
L03          LDX  LR1_ADDRESS     EBA7   A2 02        GET LINE NUMBER OF PROGRAM
             JSR  GET_LINE_NUMB   EBA9   20 3C E8
             JSR  CMP16_1         EBAC   20 69 E8     COMPARE LINE NUMBERS
             BEQ  L02             EBAF   F0 05
             JSR  SEARCH_NEXT_L   EBB1   20 A3 E7     IF IT IS NOT SAME, MOVE NEXT LINE.
             BCC  L03             EBB4   90 F1
L02          JMP  MAIN_LOOP_5     EBB6   4C FE E8     EXECUTE FROM THE DETECTED LINE
%
%-------------------------------------------------------------------------------------
% CONVERT R0 TO INTEGER 4 DIGIT, AND SUBSTITUTE IT TO LR0
%-------------------------------------------------------------------------------------
SET_LINE_NUM LDX  #$00            EBB9   A2 00
             LDA  R0_1            EBBB   A5 11        MOVE R0_1 TO LR0_0
             STA  LR0_0           EBBD   85 00
             LDA  R0_2            EBBF   A5 12        MOVE R0_2 TO LR0_1
             STA  LR0_1           EBC1   85 01
             LDA  R0_3            EBC3   A5 13        LOAD EXPONENT OF R0
             CMP  #$01            EBC5   C9 01        IS THE EXPONENT IS 1?
             BEQ  L04             EBC7   F0 0F
             CMP  #$02            EBC9   C9 02        IS THE EXPONENT IS 2?
             BEQ  L05             EBCB   F0 0E
             CMP  #$03            EBCD   C9 03        IS THE EXPONENT IS 3?
             BEQ  L06             EBCF   F0 0D
             CMP  #$04            EBD1   C9 04        IS THE EXPONENT IS 4?
             BEQ  L07             EBD3   F0 0C
             JMP  STOP_RUNNING    EBD5   4C B1 EA     IF IT IS OUT OF RANGE, TERMINATE EXEC.
L04          JSR  SHIFT_R_NUM_1   EBD8   20 E2 EB     SHIFT LR0 BY 4 BIT TO THE RIGHT
L05          JSR  SHIFT_R_NUM_1   EBDB   20 E2 EB
L06          JSR  SHIFT_R_NUM_1   EBDE   20 E2 EB
L07          RTS                  EBE1   60
%
%-------------------------------------------------------------------------------------
% SHIFT LINE NUMBER REGISTER LR0 BY 4 BIT TO THE RIGHT.
%-------------------------------------------------------------------------------------
SHIFT_R_NUM_1 STY  TEMP_5         EBE2   84 0E        SAVE IY.
             LDY  #$04            EBE4   A0 04        SET SHIFT TO 4BIT.
L08          LSR  $01,X           EBE6   56 01        SHIFT RIGHT 2 BYTE WIDTH.
             ROR  $00,X           EBE8   76 00
             DEY                  EBEA   88
             BNE  L08             EBEB   D0 F9
             LDY  TEMP_5          EBED   A4 0E
             RTS                  EBEF   60
%
%-------------------------------------------------------------------------------------
% CONDITION PROCESS.   (=, >, <, >=, <=)
%-------------------------------------------------------------------------------------
CONDITION    STY  TEMP_10         EBF0   84 54        SAVE IY.
             JSR  SUB_FLOAT_2     EBF2   20 CF EC     R1 - R0.
             JSR  NORMALISE_1     EBF5   20 D0 E2     NORMALISE.
             JSR  INC_POINTER_1   EBF8   20 FC E7     ADVANCE POINTER.
CONDITION_2  LDA  (P0),Y          EBFB   B1 40        ANALYZE THE NEXT OF '?' CODE.
             CMP  #$3E            EBFD   C9 3E        IS IT '>'?
             BEQ  L09             EBFF   F0 0B
             CMP  #$3C            EC01   C9 3C        IS IT '<'?
             BEQ  L10             EC03   F0 22
             CMP  #$3D            EC05   C9 3D        IS IT '='?
             BEQ  L11             EC07   F0 37
             JMP  MAIN_LOOP_2     EC09   4C 08 E9
L09          JSR  INC_POINTER_1   EC0C   20 FC E7
             LDA  (P0),Y          EC0F   B1 40        ANALYZE MORE NEXT CHARACTER.
             CMP  #$3D            EC11   C9 3D        IS IT '='?
             BEQ  L12             EC13   F0 0C        GO TO '>=' PRCESS.
             LDA  R0_3            EC15   A5 13        '>' PROCESS.
             BPL  L13             EC17   10 02        IS COMPARISON RESULT PLUS?
             BMI  L15             EC19   30 29
L13          LDA  R0_2            EC1B   A5 12        IS MANTISSA IS ZERO?
             BNE  L17             EC1D   D0 30        IS THE SIGN IS PULS AND NOT ZERO?
             BEQ  L15             EC1F   F0 23        IF SIGN IS PULS AND ZERO, GO THAT LINE PROC.
L12          LDA  R0_3            EC21   A5 13
             BPL  L14             EC23   10 27        IF THE SIGN IS PULS, GO THAT LINE PROCESS.
             BMI  L15             EC25   30 1D         ELSE GO TO NEXT LINE.
%
```

```
L10          JSR  INC_POINTER_1  EC27   20 FC E7
             LDA  (P0),Y         EC2A   B1 40        ANALYZE MORE NEXT CHARACTER.
             CMP  #$3D           EC2C   C9 3D        IS IT '='?
             BEQ  L16            EC2E   F0 06        GO TO '<=' PRCESS.
             LDA  R0_3           EC30   A5 13        '<' PROCESS.
             BMI  L17            EC32   30 1B        IF SIGN IS MINUS, GO THAT LINE PROC.
             BPL  L15            EC34   10 0E         ELSE GO TO NEXT LINE.
L16          LDA  R0_3           EC36   A5 13
             BMI  L14            EC38   30 12        IF SIGN IS MINUS, GO THAT LINE PROC.
             LDA  R0_2           EC3A   A5 12
             BEQ  L14            EC3C   F0 0E        IF RESULT IS ZERO, GO THAT LINE PROC.
             BNE  L15            EC3E   D0 04         ELSE GO TO NEXT LINE.
%
L11          LDA  R0_2           EC40   A5 12
             BEQ  L14            EC42   F0 08        IF RESULT IS ZERO, GO THAT LINE PROC.
L15          JSR  SEARCH_NEXT_L  EC44   20 A3 E7      ELSE GO TO NEXT LINE.
             LDY  TEMP_10        EC47   A4 54
             JMP  MAIN_LOOP_5    EC49   4C FE E8
%
L14          JSR  INC_POINTER_1  EC4C   20 FC E7
L17          LDY  TEMP_10        EC4F   A4 54
             JMP  MAIN_LOOP_2    EC51   4C 08 E9     RETURN TO THE MAIN LOOP 2.
%
%-------------------------------------------------------------------------------------
% SHIFT LEFT 3 BYTE WIDTH.   (SAVING P0,P0+1)
%-------------------------------------------------------------------------------------
SHIFT_L_NBIT1STA  TEMP_13        EC54   85 57        SAVE ACCUMULATOR.
             LDA  P0             EC56   A5 40        SAVE P0.
             PHA                 EC58   48
             LDA  P0+1           EC59   A5 41        SAVE P0+1.
             PHA                 EC5B   48
             LDA  TEMP_13        EC5C   A5 57
             JSR  SHIFT_L_NBIT   EC5E   20 A0 E1     SHIFT LEFT 3 BYTE WIDTH.
             PLA                 EC61   68
             STA  P0+1           EC62   85 41        RETURN P0+1.
             PLA                 EC64   68
             STA  P0             EC65   85 40        RETURN P0.
             RTS                 EC67   60
%
%-------------------------------------------------------------------------------------
% CONVERT FLOATING POINT VALUE TO BINARY
%-------------------------------------------------------------------------------------
FLOAT_TO_INT8JSR  SHIFT_INT_1    EC68   20 A0 EE     BOTTOMING R0.
             JSR  FLOAT_2_INT16  EC6B   20 3B EE     CONVERT R0 VALUE TO 16 BIT BINARY.
             LDX  VAR_1          EC6E   A6 53        RESTORE THE BROKEN R0.
             LDA  VAR_HEAD,X     EC70   BD 00 03
             STA  R0_0           EC73   85 10
             LDA  VAR_HEAD+1,X   EC75   BD 01 03
             STA  R0_1           EC78   85 11
             LDA  VAR_HEAD+2,X   EC7A   BD 02 03
             STA  R0_2           EC7D   85 12
             LDA  VAR_HEAD+3,X   EC7F   BD 03 03
             STA  R0_3           EC82   85 13
             RTS                 EC84   60
%
%-------------------------------------------------------------------------------------
% SAVE POINTER P0,P0+1
%-------------------------------------------------------------------------------------
             .ORIGIN  $EC86
%
SAVE_P0      STA  TEMP_13        EC86   85 57
             LDA  P0             EC88   A5 40
             STA  TEMP_11        EC8A   85 55
             LDA  P0+1           EC8C   A5 41
             STA  TEMP_12        EC8E   85 56
             LDA  TEMP_13        EC90   A5 57
             RTS                 EC92   60
%
%-------------------------------------------------------------------------------------
% RESTORE POINTER P0,P0+1
%-------------------------------------------------------------------------------------
LOAD_P0      LDA  TEMP_11        EC93   A5 55
```

```
             STA  P0              EC95   85 40
             LDA  TEMP_12         EC97   A5 56
             STA  P0+1            EC99   85 41
             RTS                  EC9B   60
%
%-------------------------------------------------------------------------------
             .ORIGIN  $ECBF
%
TEST_2       STA  TEMP_14         ECBF   85 58        R0_3 MONITOR
             SED                  ECC1   F8          SET DECIMAL MODE
             JMP  OUT_EXP_5       ECC2   4C 9F E2
%
%-------------------------------------------------------------------------------
% FLOATING POINT ADDITION.
%-------------------------------------------------------------------------------
ADD_FLOAT_2  JSR  SAVE_P0         ECC5   20 86 EC
             JSR  ADD_FLOAT_1     ECC8   20 D0 E3
             JSR  LOAD_P0         ECCB   20 93 EC
             RTS                  ECCE   60
%
%-------------------------------------------------------------------------------
% FLOATING POINT SUBTRACTION.
%-------------------------------------------------------------------------------
SUB_FLOAT_2  JSR  SAVE_P0         ECCF   20 86 EC
             JSR  SUB_FLOAT_1     ECD2   20 E5 E3
             JSR  LOAD_P0         ECD5   20 93 EC
             RTS                  ECD8   60
%
%-------------------------------------------------------------------------------
% FLOATING POINT MULTIPLICATION.
%-------------------------------------------------------------------------------
MUL_FLOAT_2  JSR  SAVE_P0         ECD9   20 86 EC
             JSR  MUL_FLOAT_1     ECDC   20 A0 E4
             JSR  LOAD_P0         ECDF   20 93 EC
             RTS                  ECE2   60
%
%-------------------------------------------------------------------------------
% FLOATING POINT DIVISION.
%-------------------------------------------------------------------------------
DIV_FLOAT_2  JSR  SAVE_P0         ECE3   20 86 EC
             JSR  DIV_FLOAT_1     ECE6   20 70 E5
             JSR  LOAD_P0         ECE9   20 93 EC
             RTS                  ECEC   60
%
%-------------------------------------------------------------------------------
% PATCH OF MAIN LOOP.
%-------------------------------------------------------------------------------
             .ORIGIN  $ECFA
%
MAIN_LOOP_3  LDA  (P0),Y          ECFA   B1 40        ANALYZE ONE CHARACTER
             STA  MONI_1          ECFC   85 58        CHARACTER MONITOR
             CMP  #$25            ECFE   C9 25
             BEQ  L01             ED00   F0 06        IF IT'S '%'CODE, SET RUN FLAG TO ZERO.
             JSR  DET_LINE_NUM_1  ED02   20 14 ED     DETECT LINE NUMBER.
             JMP  MAIN_LOOP_4     ED05   4C 03 E9     RETURN TO MAIN LOOP.
L01          LDA  #$00            ED08   A9 00
             STA  RUN_FLAG        ED0A   85 50
             RTS                  ED0C   60
%
%-------------------------------------------------------------------------------
% INITIALIZE RUN FLAG AT START UP.
%-------------------------------------------------------------------------------
INIT_RUNFLAG SED                  ED0D   F8
             CLC                  ED0E   18
             LDA  #$00            ED0F   A9 00
             STA  RUN_FLAG        ED11   85 50
             RTS                  ED13   60
%
%-------------------------------------------------------------------------------
% DETECT LINE NUMBER.
%      ':' IS DETECTED, SET CARRY. P0 POINTS THE NEXT CHARACTER OF ':'.
%      ':' IS NOT DETECTED, CLEAR CARRY. P0 POINIS THE POSITION BEFORE EXECUTION THE ROUTINE.
```

```
%-------------------------------------------------------------------------------------------
DET_LIN_NUM_1 JSR SAVE_P0          ED14   20 86 EC
              LDX  #$05            ED17   A2 05        DOES THE ':' EXIST BY THE FOURTH LETTER?
              LDY  #$00            ED19   A0 00
L29           LDA  (P0),Y          ED1B   B1 40
              PHA                  ED1D   48
              JSR  INC_POINTER_1   ED1E   20 FC E7
              PLA                  ED21   68
              CMP  #$3A            ED22   C9 3A
              BEQ  L30             ED24   F0 08
              DEX                  ED26   CA
              BNE  L29             ED27   D0 F2
              CLC                  ED29   18
              JSR  LOAD_P0         ED2A   20 93 EC
              RTS                  ED2D   60
%
L30           SEC                  ED2E   38
              RTS                  ED2F   60
%
%-------------------------------------------------------------------------------------------
% PATCH OF CONDITION PROCESS.
%-------------------------------------------------------------------------------------------
              .ORIGIN  $ED37
%
CONDITION_1   STY  TEMP_10         ED37   84 54        SAVE IY.
              JSR  SUB_FLOAT_2     ED39   20 CF EC     R1 - R0
              JSR  NORMALISE_1     ED3C   20 D0 E2     NORMALIZE
              JSR  INC_POINTER_1   ED3F   20 FC E7     ADVANCE THE POINTER
              LDY  TEMP_10         ED42   A4 54        RETURN IY
              JMP  CONDITION_2     ED44   4C FB EB     GO TO CONDITION PROCESS
%
%-------------------------------------------------------------------------------------------
% DISPLAY PROGRAM LIST
%-------------------------------------------------------------------------------------------
DISP_LIST_1   LDA  P0               ED47   A5 40        DOES ANY CHARACTER EXIST BEFORE '@'?
              CMP  #$01            ED49   C9 00
              BNE  L01             ED4B   D0 2F
              JSR  SET_P0_PR       ED4D   20 A8 EA     SET POINTER TO THE HEAD OF PROGRAM AREA.
L10           LDY  #$00            ED50   A0 00
L03           LDA  (P0),Y          ED52   B1 40        GET ONE CHARACTER.
              CMP  #$0D            ED54   C9 0D        'CR' CODE?
              BEQ  L02             ED56   F0 0D
              CMP  #$25            ED58   C9 25        PROGRAM END '%' CODE?
              BEQ  L04             ED5A   F0 1F
L09           JSR  OUT_1CHA        ED5C   20 9A E0     OUT ONE CHARACTER TO SERIAL INTERFACE.
              JSR  INC_POINTER_1   ED5F   20 FC E7     ADVANCE THE POINTER.
              CLC                  ED62   18
              BCC  L03             ED63   90 ED        REPEAT.
L02           JMP  L11             ED65   4C 97 ED     WAIT TO PUSH A KEY
%
              NOP                  ED68   EA
              NOP                  ED69   EA
              NOP                  ED6A   EA
              NOP                  ED6B   EA
%
L05           LDA  (P0),Y          ED6C   B1 40        IF 'CR' IS PUSHED, DISPLAY TO THE END.
              CMP  #$25            ED6E   C9 25
              BEQ  L04             ED70   F0 09
              JSR  OUT_1CHA        ED72   20 9A E0
              JSR  INC_POINTER_1   ED75   20 FC E7
              CLC                  ED78   18
              BCC  L05             ED79   90 F1
L04           RTS                  ED7B   60
%
L01           JSR  SET_LINE_NUM    ED7C   20 B9 EB     CASE OF ENTERING BEFORE '@'.
              JSR  SET_P0_PR       ED7F   20 A8 EA     SET THE HEAD OF PROGRAM AREA TO P0.
L07           LDX  LR1_ADDRESS     ED82   A2 02
              JSR  GET_LINE_NUMB   ED84   20 3C E8     GET LINE NUMBER OF PROGRAM AREA.
              JSR  CMP16_1         ED87   20 69 E8     COMPARE TWO LINE NUMBERS.
              BEQ  L10             ED8A   F0 C4
              JSR  SEARCH_NEXT_L   ED8C   20 A3 E7     IF IT IS NOT THE SAME, GO TO NEXT LINE.
              BCC  L07             ED8F   90 F1
```

```
             BCS  L03            ED91   B0 BF
L08          LDA  #$0D           ED93   A9 0D        OUTPUT 'CR' CODE.
             BNE  L09            ED95   D0 C5
L11          JSR  IN_1CHA        ED97   20 90 E0
             CMP  #$0D           ED9A   C9 0D        IF 'CR' IS PUSHED, DISPLAY TO THE END.
             BEQ  L05            ED9C   F0 CE
             CMP  #$0A           ED9E   C9 0A        IF 'LF' IS PUSHED, EXIT.
             BNE  L08            EDA0   D0 F1
             LDA  #$0D           EDA2   A9 0D
             JSR  OUT_1CHA       EDA4   20 9A E0
             RTS                 EDA7   60
%
%-------------------------------------------------------------------------------
% PROCESS '-' CODE.
%-------------------------------------------------------------------------------
SUB_2        INY                 EDA8   C8          ADVANCE THE POINTER
             LDA  (P0),Y         EDA9   B1 40
             CMP  #$30           EDAB   C9 30        IS THE NEXT CHARACTER OF '-'
             BCS  L01            EDAD   B0 03         MORE THAN $30? (VALUE,':',VARIABLE)
L02          JMP  SUB_1_1        EDAF   4C 40 EB     IF LESS THAN $2F, GO TO SUBTRACTION.
L01          CMP  #$3B           EDB2   C9 3B
             BEQ  L02            EDB4   F0 F9        IF IT'S ';' GO TO SUBTRACTION.
             JMP  MANT_SIGN      EDB6   4C 7A EB     ELSE GO TO SIGN PROCESS OF VALUE & VARIABLE.
%
%-------------------------------------------------------------------------------
% END LINE PROCESS    (';' NO OUTPUT, ';;' OUTPUT WITH NO NEW LINE)
%-------------------------------------------------------------------------------
ENDLINE_PROC1JSR  SUBSTITUTE_5   EDB9   20 62 F0
             LDA  DISP_FLAG      EDBC   A5 51
             CMP  #$01           EDBE   C9 01
             BEQ  ENDLINE_PROC3  EDC0   F0 0F        IF DISP_FLAG=1, NO DISPLAY
             CMP  #$02           EDC2   C9 02
             BEQ  L02            EDC4   F0 15        IF DISP_FLAG=2, DISPLAY R0 & NO NEW LINE.
             JMP  ENDLINE_PROC2  EDC6   4C 17 EE     IF DISP_FLAG=0, DISPLAY R0 & NEW LINE.
ENDLINE_PROC4JSR  DISP_BUFFER_2  EDC9   20 E6 ED     DISPLAY UNTIL BEFORE THE 'CR' CODE.
             LDA  #$0D           EDCC   A9 0D        NEW LINE
L04          JSR  OUT_1CHA       EDCE   20 9A E0
ENDLINE_PROC3JMP  ENDLINE_PROC5  EDD1   4C AB F0
%
             .ORIGIN  $EDDC
%
L02          JSR  DISP_FLOAT     EDDC   20 E0 E1     IF DISP_FLAG=2, DISPLAY R0 & NO NEW LINE.
ENDLINE_PROC5JSR  DISP_BUFFER_2  EDDF   20 E6 ED
             LDA  #$20           EDE2   A9 20        OUTPUT SPACE CODE
             BNE  L04            EDE4   D0 E8
%
%-------------------------------------------------------------------------------
% OUTPUT ASCII CODES IN THE DISPLAY BUFFER TO SERIAL INTERFACE
%-------------------------------------------------------------------------------
DISP_BUFFER_2LDY  #$00           EDE6   A0 00
L07          LDA  OUT_BUF,Y      EDE8   B9 80 02
             INY                 EDEB   C8
             BEQ  L05            EDEC   F0 0A
             CMP  #$0D           EDEE   C9 0D        OUTPUT UNTIL BEFORE THE 'CR' CODE.
             BEQ  L05            EDF0   F0 06
             JSR  OUT_1CHA       EDF2   20 9A E0
             CLC                 EDF5   18
             BCC  L07            EDF6   90 F0
L05          RTS                 EDF8   60
%
%-------------------------------------------------------------------------------
% COMMENT PROCESS  (AFTER '#' CODE IN A LINE, RECOGNIZE CHARACTERS AS COMMENTS.)
%-------------------------------------------------------------------------------
COMMENT      LDX  #$00           EDF9   A2 00
             LDY  #$00           EDFB   A0 00
L02          JSR  INC_POINTER_1  EDFD   20 FC E7     ADVANCE THE POINTER.
             LDA  (P0),Y         EE00   B1 40        GET A ONE CHARACTER.
             CMP  #$3B           EE02   C9 3B        IF IT'S ';', EXIT
             BEQ  L01            EE04   F0 0A
             CMP  #$0D           EE06   C9 0D        IF IT'S 'CR', EXIT
             BEQ  L01            EE08   F0 06
             STA  OUT_BUF,X      EE0A   9D 80 02     WRITE TEXT TO DISPLAY BUFFER.
```

```
             INX                 EE0D   E8           ADVANCE THE DISPLAY BUFFER POINTER.
             BNE  L02            EE0E   D0 ED
L01          LDA  #$04           EE10   A9 04
             STA  DISP_FLAG      EE12   85 51
             JMP  COMMENT_1      EE14   4C 33 EE
%
%-------------------------------------------------------------------------------
ENDLINE_PROC2CMP  #$00           EE17   C9 00        IF DISP_FLAG=0, DISPLAY R0 & NEW LINE.
             BEQ  L03            EE19   F0 0C
             CMP  #$04           EE1B   C9 04        IF DISP_FLAG=4, DISPLAY COMMENT & NEW LINE.
             BEQ  L04            EE1D   F0 0B
             CMP  #$05           EE1F   C9 05        IF DISP_FLAG=5, DO NOT DISPLAY COMMENT.
             BEQ  L05            EE21   F0 0A
             CMP  #$06           EE23   C9 06        IF DISP_FLAG=6, DISP COMMENT & NO NEW LINE.
             BEQ  L06            EE25   F0 09
L03          JSR  DISP_FLOAT     EE27   20 E0 E1     PUT ASCII CODE CONVERTED FROM R0 TO BUFFER.
L04          JMP  ENDLINE_PROC4  EE2A   4C C9 ED     GO TO OUTPUT BUFFER & NEW LINE PROCESS
L05          JMP  ENDLINE_PROC3  EE2D   4C D1 ED     GO TO WITHOUT OUTPUT PROCESS
L06          JMP  PATCH_ELP_5    EE30   4C 5D E0     PATCH TO PREVENT SPACE CODE IN ;; (V.1.3.1)
%
COMMENT_1    LDA  #$0D           EE33   A9 0D        PUT 'CR' CODE ON THE DISPLAY BUFFER END.
             STA  OUT_BUF,X      EE45   9D 80 02
             JMP  MAIN_LOOP_2    EE38   4C 03 E9     RETURN TO MAIN LOOP 2.
%
%-------------------------------------------------------------------------------
% CONVERT BOTTOMING R0 VALUE TO 16 BIT BINARY AND SET P1,P1+1 TO IT.
%-------------------------------------------------------------------------------
FLOAT_2_INT16LDA  #$00           EE3B   A9 00
             STA  P1             EE3D   85 42        CLEAR P1,P1+1.
             STA  P1+1           EEDF   85 43
             STA  P2+1           EE41   85 45
             LDA  #$01           EE43   A9 01
             STA  P2             EE45   85 44        SET $0001 TO P2,P2+1.
             LDA  R0_0           EE47   A5 10
             JSR  LOWER_INTEG_1  EE49   20 79 EE     ACCUMULATE WEIGHT OF 10^0.
             LDA  #$0A           EE4C   A9 0A
             STA  P2             EE4E   85 44        SET $000A TO P2,P2+1.
             LDA  R0_0           EE50   A5 10
             JSR  UPPER_INTEG_1  EE52   20 80 EE     ACCUMULATE WEIGHT OF 10^1.
             LDA  #$64           EE55   A9 64
             STA  P2             EE57   85 44        SET $0064 TO P2,P2+1.
             LDA  R0_1           EE59   A5 11
             JSR  LOWER_INTEG_1  EE5B   20 79 EE     ACCUMULATE WEIGHT OF 10^2.
             LDA  #$03           EE5E   A9 03
             STA  P2+1           EE60   85 45
             LDA  #$E8           EE62   A9 E8
             STA  P2             EE64   85 44        SET $03E8 TO P2,P2+1.
             LDA  R0_1           EE66   A5 11
             JSR  UPPER_INTEG_1  EE68   20 80 EE     ACCUMULATE WEIGHT OF 10^3.
             LDA  #$27           EE6B   A9 27
             STA  P2+1           EE6D   85 45
             LDA  #$10           EE6F   A9 10
             STA  P2             EE71   85 44         SET $2710 TO P2,P2+1.
             LDA  R0_2           EE73   A5 12
             JSR  LOWER_INTEG_1  EE75   20 79 EE     ACCUMULATE WEIGHT OF 10^4.
             RTS                 EE78   60
%
%-------------------------------------------------------------------------------
% ACCUMULATE THE COUNT OF LOWER 4 BIT OF ACC
%-------------------------------------------------------------------------------
LOWER_INTEG_1AND  #$0F           EE79   29 0F        EXTRACT LOWER 4 BIT.
             TAX                 EE7B   AA
             JSR  INTEG16_1      EE7C   20 54 EF     16 BIT ACCUMULATION.
             RTS                 EE7F   60
%
%-------------------------------------------------------------------------------
% ACCUMULATE THE COUNT OF UPPER 4 BIT OF ACC
%-------------------------------------------------------------------------------
UPPER_INTEG_1LSR                 EE80   4A           4 BIT SHIFT RIGHT.
             LSR                 EE81   4A
             LSR                 EE82   4A
             LSR                 EE83   4A
```

```
             TAX                 EE84   AA
             JSR  INTEG16_1      EE85   20 54 EF      16 BIT ACCUMULATION.
             RTS                 EE88   60
%
%-------------------------------------------------------------------------------
% BOTTOMING R0 MANTISSA VALUE
%-------------------------------------------------------------------------------
             .ORIGIN  $EEA0
%
SHIFT_INT_1  JSR  SAVE_P0        EEA0   20 86 EC
             LDA  R0_3           EEA3   A5 13
             AND  #$07           EEA5   29 07         EXTRACT 3 BIT OF EXPONENT.
             STA  TEMP_2         EEA7   85 0B
             CLD                 EEA9   D8
             SEC                 EEAA   38
             LDA  #$06           EEAB   A9 06         CALC RIGHT SHIFT BIT (6 - EXPONENT) * 4.
             SBC  TEMP_2         EEAD   E5 0B
             ASL                 EEAF   0A
             ASL                 EEB0   0A
             STA  MONI_1         EEB1   85 58         SHIFT BIT MONITOR.
             TAX                 EEB3   AA           SET SHIFT BIT TO IX.
             LDY  #$10           EEB4   A0 10
             LDA  #$00           EEB6   A9 00
             JSR  SHIFT_R_NBIT   EEB8   20 6F E3      3 BYTE SHIFT RIGHT
             JSR  LOAD_P0        EEBB   20 93 EC
             RTS                 EEBE   60
%
%-------------------------------------------------------------------------------
% CONVERT 8 BIT BINARY OF ACC TO BCD INTEGER, AND SET R0 TO IT.
%-------------------------------------------------------------------------------
INT8_TO_FLOATSED                 EEBF   F8
             STA  MEM_VAL        EEC0   85 5A
             LDA  #$00           EEC2   A9 00         CLEAR R0,R0+1.
             STA  R0_0           EEC4   85 10
             STA  R0_1           EEC6   85 11
             STA  R0_2           EEC8   85 12
             STA  R1_0           EECA   85 14
             STA  R1_1           EECC   85 15
             STA  R1_2           EECE   85 16
             ASL  MEM_VAL        EED0   06 5A         EXTRACT D7 OF MEM_VAL TO CARRY BIT.
             BCC  L01            EED2   90 0F         IF D7=0, DO NOT ADD WEIGHT.
             LDA  #$28           EED4   A9 28         SET 128(DEC) TO R1.
             STA  R1_0           EED6   85 14
             LDA  #$01           EED8   A9 01
             STA  R1_1           EEDA   85 15
             LDA  #$00           EEDC   A9 00
             STA  R1_2           EEDE   85 16
             JSR  ADD_ABS_BCD_1  EEE0   20 6A EF
L01          ASL  MEM_VAL        EEE3   06 5A         EXTRACT D6 OF MEM_VAL TO CARRY BIT.
             BCC  L02            EEE5   90 0B
             LDA  #$64           EEE7   A9 64         SET 64(DEC) TO R1.
             STA  R1_0           EEE9   85 14
             LDA  #$00           EEEB   A9 00
             STA  R1_1           EEED   85 15
             JSR  ADD_ABS_BCD_1  EEEF   20 6A EF
L02          LDA  #$00           EEF2   A9 00
             STA  R1_1           EEF4   85 15         CLEAR R1_1
             ASL  MEM_VAL        EEF6   06 5A         EXTRACT D5 OF MEM_VAL TO CARRY BIT.
             BCC  L03            EEF8   90 07
             LDA  #$32           EEFA   A9 32         SET 32(DEC) TO R1.
             STA  R1_0           EEFC   85 14
             JSR  ADD_ABS_BCD_1  EEFE   20 6A EF
L03          ASL  MEM_VAL        EF01   06 5A         EXTRACT D4 OF MEM_VAL TO CARRY BIT.
             BCC  L04            EF03   90 07
             LDA  #$16           EF05   A9 16         SET 16(DEC) TO R1.
             STA  R1_0           EF07   85 14
             JSR  ADD_ABS_BCD_1  EF09   20 6A EF
L04          ASL  MEM_VAL        EF0C   06 5A         EXTRACT D3 OF MEM_VAL TO CARRY BIT.
             BCC  L05            EF0E   90 0B
             LDA  #$08           EF10   A9 08         SET 8(DEC) TO R1.
             STA  R1_1           EF12   85 14
             LDA  #$00           EF14   A9 00
```

```
             STA  R1_1           EF16   85 15
             JSR  ADD_ABS_BCD_1  EF18   20 6A EF
L05          ASL  MEM_VAL        EF1B   06 5A        EXTRACT D2 OF MEM_VAL TO CARRY BIT.
             BCC  L06            EF1D   90 07
             LDA  #$04           EF1F   A9 04        SET 4(DEC) TO R1.
             STA  R1_0           EF21   85 14
             JSR  ADD_ABS_BCD_1  EF23   20 6A EF
L06          ASL  MEM_VAL        EF26   06 5A        EXTRACT D1 OF MEM_VAL TO CARRY BIT.
             BCC  L07            EF28   90 07
             LDA  #$02           EF2A   A9 02        SET 2(DEC) TO R1.
             STA  R1_0           EF2C   85 14
             JSR  ADD_ABS_BCD_1  EF2E   20 6A EF
L07          ASL  MEM_VAL        EF31   06 5A        EXTRACT D0 OF MEM_VAL TO CARRY BIT.
             BCC  L08            EF33   90 07
             LDA  #$01           EF35   A9 01        SET 1(DEC) TO R1.
             STA  R1_0           EF37   85 14
             JSR  ADD_ABS_BCD_1  EF39   20 6A EF
L08          RTS                 EF3C   60
%
%-------------------------------------------------------------------------------
% ACCUMULATE P1,PI+1 BY 16 BIT VALUE P2,P2+1.   (ACCUMULATION COUNT: IX)
%-------------------------------------------------------------------------------
             .ORIGIN  $EF54
%
INTEG16_1    CLD                 EF54   D8
             CPX  #$00           EF55   E0 00        IF IX=0, DO NOT ACCUMULATE
             BEQ  L01            EF57   F0 10
L02          CLC                 EF59   18           CARRY CLEAR
             LDA  P1             EF5A   A5 42
             ADC  P2             EF5C   65 44        ACCUMULATE LOWER 8BIT
             STA  P1             EF5E   85 42
             LDA  P1+1           EF60   A5 43
             ADC  P2+1           EF62   65 45        ACCUMULATE UPPER 8BIT
             STA  P1+1           EF64   85 43
             DEX                 EF66   CA           COUNTER -1
             BNE  L02            EF67   D0 F0        REPEAT UNTIL COUNTER=0
L01          RTS                 EF69   60
%
%-------------------------------------------------------------------------------
% ADDITION OF MANTISSA 24 BIT ABSOLUTE BCD.
%-------------------------------------------------------------------------------
             .ORIGIN  $EF6A
%
ADD_ABS_BCD_1SED                 EF6A   F8
             CLC                 EF6B   18
             LDA  R0_0           EF6C   A5 10
             ADC  R1_0           EF6E   65 14
             STA  R0_0           EF70   85 10
             LDA  R0_1           EF72   A5 11
             ADC  R1_1           EF74   65 15
             STA  R0_1           EF76   85 11
             LDA  R0_2           EF78   A5 12
             ADC  R1_2           EF7A   65 16
             STA  R0_2           EF7C   85 12
             RTS                 EF7E   60
%-------------------------------------------------------------------------------
% POINTER VARIABLE.       (8BIT MEMORY VALUE ADDRESSED R0 IS LOADED TO R0)
%-------------------------------------------------------------------------------
             .ORIGIN  $EF86
%
POINTER_VAR_1JSR  SHIFT_INT_1    EF86   20 A0 EE     ALIGN THE DECIMAL POINT OF R0.
             JSR  FLOAT_2_INT16  EF89   20 3B EE     CONVERT R0 VALUE TO 16 BIT BINARY.
             LDY  %$01           EF8C   A0 01
             LDA  (P0),Y         EF8E   B1 40
             CMP  #$3D           EF90   C9 3D        IF THE NEXT CHARACTER IS '=',
             BEQ  L01            EF92   F0 19         DO NOT READ MEMORY
             DEY                 EF94   88
             LDA  (P0),Y         EF95   B1 42        READ MEMORY POINTED BY R0
             STA  MEM_VAL        EF97   85 5A
             STA  MONI_2         EF99   85 5B        MONITOR FOR READ VALUE
             JSR  INT8_TO_FLOAT  EF9B   20 BF EE     CONVERT BYNARY TO FLOATING, AND PUT TO R0.
             LDA  #$05           EF9E   A9 06        CORRECT EXPONENT AS AN INTEGER.
```

```
              STA  R0_3            EFA0   85 13
              JSR  NORMALISE_1     EFA2   20 D0 E2      NORMALISE.
L02           JSR  INC_POINTER_1   EFA5   20 FC E7      ADVANCE THE POINTER.
              LDY  #$00            EFA8   A0 00
              JMP  MAIN_LOOP_2     EFAA   4C 08 E9      RETURN TO MAIN LOOP 2.
%
L01           LDA  #$FC            EFAD   A9 FC         IF THE NEXT CHARACTER IS '=',
              STA  VAR_1           EFAF   85 53          RESET THE NORMAL VARIABLE POINTER.
              LDA  P1              EFB1   A5 42         SET MEMORY WRITE POINTER.
              STA  P3              EFB3   85 46
              LDA  P1+1            EFB5   A5 43
              STA  P3+1            EFB7   85 47
              JSR  INC_POINTER_1   EFB9   20 FC E7
              JSR  INIT_REG_2      EFBC   20 FA EA      INITIALIZE R0,
              CLC                  EFBF   18             ALLOW THE FOLLOWING VALUES TO BE ENTERED.
              BCC  L02             EFC0   90 E3
%
%-------------------------------------------------------------------------------
% INPUT STATE.      (PAUSE THE EXECUTION AND ENTER THE ONE-LINE INPUT STATE)
%-------------------------------------------------------------------------------
INPUT_STATE   LDA  #$00            EFC2   A9 00
              STA  RUN_FLAG        EFC4   85 50         PAUSE THE EXECUTION.
              LDA  P0              EFC6   A5 40         SAVE THE POINTER P0 TO P4.
              STA  P4              EFC8   85 48
              LDA  P0+1            EFCA   A5 41
              STA  P4+1            EFCC   85 49
              JSR  IN_1LINE_1      EFCE   20 B6 E0      INPUT ONE-LINE.
              JSR  ANA_1LINE       EFD1   20 F0 E8      ANALYZE ONE-LINE.
              LDA  P4              EFD4   A5 48         RESTORE THE POINTER P0.
              STA  P0              EFD6   85 40
              LDA  P4+1            EFD8   A5 49
              STA  P0+1            EFDA   85 41
              JSR  SEARCH_NEXT_L   EFDC   20 A3 E7      ADVANCE THE POINTER TO THE NEXT LINE.
              LDA  #$01            EFDF   A9 01         RETURN TO THE EXECUTION.
              STA  RUN_FLAG        EFE1   85 50
              LDY  #$00            EFE3   A0 00
              JMP  MAIN_LOOP_5     EFE5   4C FE E8      RETURN TO MAIN LOOP 5
%
%-------------------------------------------------------------------------------
% ARRAY VARIABLE.
%      LOAD VALUE OF ARRAY VARIABLE ')X, )Y, )Z' POINTED BY R0(0-255) TO R0
%-------------------------------------------------------------------------------
              .ORIGIN  $F00C
%
VECTOR_1      STA  TEMP_0          F00C   85 09         SAVE ASCII CODE.
              JSR  SHIFT_INT_1     F00E   20 A0 EE      ALIGN DECIMAL POSITION OF R0.
              JSR  FLOAT_2_INT16   F011   20 3B EE      CONVERT R0 TO BINARY, AND PUT IT ONTO P1.
              ASL  P1              F014   06 42         2 BIT SHIFT LEFT P1,P1+1
              ROL  P1+1            F016   26 43
              ASL  P1              F018   06 42
              ROL  P1+1            F01A   26 43
              LDA  TEMP_0          F01C   A5 09         RESTORE ASCII CODE
              AND  #$03            F01E   29 03         EXTRACT LOWER 2 BIT
              EOR  #$03            F020   49 03         INVERT LOWER 2 BIT
              ASL                  F022   0A            2 BIT SHIFT LEFT
              ASL                  F023   0A
              ORA  P1+1            F024   05 43         SYNTHESIZE WHTH P1+1
              ORA  #$00            F026   09 00         SET '0000' TO UPPER 4 BIT.
              JMP  VECTOR_PATCH1   F028   4C A2 F0      IF THE NEXT CHARACTER IS '=',
              NOP                  F02B   EA             DO NOT READ ARRAY VARIABLE.
VECTOR_PATCH2 CMP  #$3D            F02C   C9 3D
              BEQ  L01             F02E   F0 1D
              LDY  #$00            F030   A0 00
              LDA  (P1),Y          F032   B1 42         READ ARRAY VARIABLE AND PUT ONTO R0.
              STA  R0_0            F034   85 10
              INY                  F036   C8
              LDA  (P1),Y          F037   B1 42
              STA  R0_1            F039   85 11
              INY                  F03B   C8
              LDA  (P1),Y          F03C   B1 42
              STA  R0_2            F03E   85 12
              INY                  F040   C8
```

```
             LDA  (P1),Y          F041   B1 42
             STA  R0_3            F043   85 13
L04          JSR  INC_POINTER_1   F045   20 FC E7      ADVANCE THE POINTER.
             LDY  #$00            F048   A0 00
             JMP  MAIN_LOOP_2     F04A   4C 08 E9      RETURN TO MAIN LOOP 2.
%
L01          LDA  #$FC            F04D   A9 FC         IF THE NEXT CHARACTER IS '=',
             STA  VAR_1           F04F   85 53          RESET THE NORMAL VARIABLE POINTER.
             LDA  P1              F051   A5 42
             STA  P5              F053   85 4A         SET VALUE TO MEMORY WRITE POINTER P5.
             LDA  P1+1            F055   A5 43
             STA  P5+1            F057   85 4B
             JSR  INC_POINTER_1   F059   20 FC E7
             JSR  INIT_REG_2      F05C   20 FA EA      INITIALIZE R0,
             CLC                  F05F   18             ALLOW THE FOLLOWING VALUES TO BE ENTERED.
             BCC  L04             F060   90 E3
%
%-------------------------------------------------------------------------------
% SUBSTITUTION PROCESS FOR NORMAL VARIABLES.
%-------------------------------------------------------------------------------
SUBSTITUTE_5 LDX  VAR_1           F062   A6 53         SET VARIABLE OFFSET TO IX
             LDA  R0_0            F064   A5 10         MOVE R0 TO NORMAL VARIABLE.
             STA  VAR_HEAD,X      F066   9D 00 03
             LDA  R0_1            F069   A5 11
             STA  VAR_HEAD+1,X    F06B   9D 01 03
             LDA  R0_2            F06E   A5 12
             STA  VAR_HEAD+2,X    F070   9D 02 03
             LDA  R0_3            F073   A5 13
             STA  VAR_HEAD+3,X    F075   9D 03 03
%
             JSR  FLOAT_TO_INT8   F078   20 68 EC      CONVERT R0 TO BINARY, AND PUT IT TO P1.
             LDA  P1              F07B   A5 42         LOAD A BINARY VALUE TO ACC.
             LDY  #$00            F07D   A0 00
             STA  (P3),Y          F07F   91 46         WRITE ACC VALUE TO MEMORY POINTED BY P3.
%
             LDA  R0_0            F081   A5 10         SUBSTITUTE TO ARRAY VARIABLE POINTED BY P5.
             STA  (P5),Y          F083   91 4A
             INY                  F085   C8
             LDA  R0_1            F086   A5 11
             STA  (P5),Y          F088   91 4A
             INY                  F08A   C8
             LDA  R0_2            F08B   A5 12
             STA  (P5),Y          F08D   91 4A
             INY                  F08F   C8
             LDA  R0_3            F090   A5 13
             STA  (P5),Y          F092   91 4A
             LDY  #$00            F094   A0 00
             RTS                  F096   60
%
%-------------------------------------------------------------------------------
% PATCH FOR INITIALIZE REGISTERS.
%-------------------------------------------------------------------------------
INIT_REG_4   LDA  #$5C            F097   A9 5C         INITIALIZE ARRAY WRITE POINTER.
             STA  P5              F099   85 4A
             LDA  #$00            F09B   A9 00
             STA  P5+1            F09D   85 4B
             JMP  INIT_REG_3      F09F   4C EE EA
%
%-------------------------------------------------------------------------------
VECTOR_PATCH1 STA  P1+1           F0A2   85 43
             LDY  #$01            F0A4   A0 01
             LDA  (P0),Y          F0A6   B1 40
             JMP  VECTOR_PATCH2   F0A8   4C 2C F0
%
%-------------------------------------------------------------------------------
% PATCH FOR CALCULATOR LIKE OPERATION.
%-------------------------------------------------------------------------------
ENDLINE_PROC5 LDA  RUN_FLAG       F0AB   A5 50         IF RUN_FLAG='0', PUSH ARITHMETIC STACK.
             BNE  L03             F0AD   D0 04
             JSR  PUSH_FP_REG     F0AF   20 1F EA
             RTS                  F0B2   60
L03          JSR  INC_POINTER_1   F0B3   20 FC E7      IF RUN_FLAG='1', ADVANCE THE POINTER.
```

```
             JMP  MAIN_LOOP_6     F0B6   4C FB E8      RETURN TO MAIN LOOP 6.
%
%-------------------------------------------------------------------------------------
% DISPLAY PROMPT '>' CODE.
%-------------------------------------------------------------------------------------
IN_1LINE_2   LDA  #$3E            F0B9   A9 3E         PROMPT '>' CODE.
             JSR  OUT_1CHA        F0BB   20 9A E0      OUTPUT ONE CHARACTER.
             JMP  IN_1LINE_1      F0BE   4C B6 E0      GO TO INPUT ONE LINE.
%
%-------------------------------------------------------------------------------------
% CHECK FOR PROGRAM.
%      IF THE NEXT CHARACTER OF 'CR' IS NOT '%' OR NUMERAL, REMOVE ILLEGAL GAPS.
%-------------------------------------------------------------------------------------
CHECK_PROG_1 LDA  #$00            F0C1   A9 00
             STA  P0              F0C3   85 40
             LDA  #$10            F0C5   A9 10
             STA  P0+1            F0C7   85 41
L04          JSR  SEARCH_NEXT_L   F0C9   20 A3 E7      ADVANCE THE POINTER TO THE NEXT OF 'CR'.
             LDA  (P0),Y          F0CC   B1 40         INPUT ONE CHARACTER.
             CMP  #$25            F0CE   C9 25         IF IT'S '%' CODE, NORMAL TERMINATION.
             BEQ  L01             F0D0   F0 0C
             CMP  #$30            F0D2   C9 30
             BCS  L02             F0D4   B0 02         IF IT'S MORE THAN $30, BRANCH.
             BCC  L03             F0D6   90 07         IF IT'S LESS THAN $30, GO TO CORRECTION.
L02          CMP  #$3A            F0D8   C9 3A
             BCS  L03             F0DA   B0 03         IF IT'S MORE THAN $30, GO TO CORRECTION.
             BCC  L04             F0DC   90 EB          ELSE ADVANCE THE POINTER TO THE NEXT 'CR'.
L01          RTS                  F0DE   60
%
L03          INY                  F0DF   C8            FOLLOWING IS CORRECTION PROCESS.
             LDA  (P0),Y          F0E0   B1 40
             CMP  #$25            F0E2   C9 25         MEASURE LENGTH UNTIL '%' CODE DETECTION.
             BEQ  L05             F0E4   F0 07
             CMP  #$0D            F0E6   C9 0D         DOES 'CR' CODE DETECT?
             BEQ  L06             F0E8   F0 02
             BNE  L03             F0EA   D0 F3
L06          INY                  F0EC   C8            POINT OF THE NEXT OF 'CR' IS OFFSET TO MOVE.
L05          STY  #$58            F0ED   84 58
             JSR  MOV_BLOCK_B1    F0EF   20 DD E7      BLOCK TRANSFER OF WIDTH INDICATED BY IY.
             RTS                  F0F2   60
%
%-------------------------------------------------------------------------------------
%
DISP_LIST_2  JSR  SAVE_P0         F0F3   20 86 EC
             JSR  CHECK_PROG_1    F0F6   20 C1 F0      EXECUTE THE PROGRAM CHECK
             JSR  LOAD_P0         F0F9   20 93 EC       BEFORE DISPLAY PROGRAM LIST.
             JMP  DISP_LIST_1     F0FC   4C 47 ED
%
%-------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------
%---------------------------    ELEMENTARY FUNCTIONS    -------------------------------
%-------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------
%
%-------------------------------------------------------------------------------------
% SQUARE ROOT BY NEWTON'S METHOD.    Y = (Y + X / Y) *0.5          (V.1.2.0  APR/09/2022)
%-------------------------------------------------------------------------------------
SQRT_1       LDA  R0_0            F0FF   A5 10
             BNE  L04             F101   D0 09
             LDA  R0_1            F103   A5 11
             BNE  L04             F105   D0 05
             LDA  R0_2            F107   A5 12
             BEQ  L06             F109   F0 0E         IF MANTISSA IS ZERO QUIT THE FUNCTION.
             NOP                  F10B   EA
L04          LDA  R0_3            F10C   A5 13
             BPL  L05             F10E   10 03         IF MANTISSA IS NEGATIVE,
             JMP  ERROR_PROC      F110   4C 10 F1       GO TO ERROR PROCESS.
L05          JSR  INIT_VAL_3      F113   20 21 F1      GENERATE INITIAL VALUE OF NEWTON METHOD.
             JSR  NEWTON_METD_1   F116   20 70 F1      NEWTON'S METHOD.
L06          RTS                  F119   60
%
%-------------------------------------------------------------------------------------
```

```
% GENERATE INITIAL VALUE OF NEWTON'S METHOD.
%-------------------------------------------------------------------------------
             .ORIGIN  $F121
%
INIT_VAL_3   LDX  #$10           F121   A2 10        STORE ARGUMENT R0 TO R4
             LDY  #$60           F123   A0 60
             JSR  MOV_FP_REG     F125   20 39 EA
             LDA  R4_3           F128   A5 63        EXPONENT OF THE ARGUMENT
             ASL                 F12A   0A
             BMI  L02            F12B   30 0C        IF EXPONENT IS NEGATIVE, BRANCH.
             LDA  R4_3           F12D   A5 63
             JSR  INIT_VAL_4     F12F   20 50 F1     SET THE HALF EXPONENT OF ARGUMENT.
L06          STA  R5_3           F132   85 67
             LDA  #$10           F134   A9 10
             STA  R5_2           F136   85 66
             RTS                 F138   60
%
L02          LDA  R4_3           F139   A5 63
             STA  TEMP_13        F13B   85 57
             LDA  #$80           F13D   A9 80
             SED                 F13F   F8
             SEC                 F140   38
             SBC  TEMP_13        F141   E5 57        CONVERT TO ABSOLUTE, 80(BCD) - EXPONENT.
             JSR  INIT_VAL_4     F143   20 50 F1
             BEQ  L03            F146   F0 05        IF INITIAL VALUE IS ZERO, DO NOT COMPLEMENT
             LDA  #$80           F148   A9 80
             SEC                 F14A   38
             SBC  R0_2           F14B   E5 12
L03          CLC                 F14D   18
             BCC  L06            F14E   90 E2
%
%-------------------------------------------------------------------------------
% R0 = 0.5 * (EXPONENT OF ARGUMENT).
%-------------------------------------------------------------------------------
INIT_VAL_4   STA  R1_2           F150   85 16
             LDA  #$05           F152   A9 05
             STA  R0_2           F154   85 12
             LDA  #$01           F156   A9 01
             STA  R1_3           F158   85 17
             STA  R0_3           F15A   85 13
             LDA  #$00           F15C   A9 00
             STA  R5_0           F15E   85 64
             STA  R1_0           F160   85 14
             STA  R0_0           F162   85 10
             STA  R5_1           F164   85 65
             STA  R1_1           F166   85 15
             STA  R0_1           F168   85 11
             JSR  MUL_FLOAT_2    F16A   20 D9 EC
             LDA  R0_2           F16D   A5 12
             RTS                 F16F   60
%
%-------------------------------------------------------------------------------
% NEWTON'S METHOD.
%-------------------------------------------------------------------------------
NEWTON_METD_1LDA  #$08           F170   A9 08        SET 8 TO LOOP COUNTER.
             STA  LOOP_COUNTER_2 F172   85 70
L01          LDX  #$60           F174   A2 60
             LDY  #$14           F176   A0 14
             JSR  MOV_FP_REG     F178   20 39 EA     MOVE ARGUMENT R4 TO R1.
             LDX  #$64           F17B   A2 64
             LDY  #$10           F17D   A0 10
             JSR  MOV_FP_REG     F17F   20 39 EA     MOVE PROVISIONAL VALUE Y,R5 TO R0.
             JSR  DIV_FLOAT_1    F182   20 E3 EC     R0 = R1 / R0.
             JSR  NORMALISE_1    F185   20 D0 E2
             LDX  #$64           F188   A2 64
             LDY  #$14           F18A   A0 14
             JSR  MOV_FP_REG     F18C   20 39 EA     MOVE Y,R5 TO R1
             JSR  ADD_FLOAT_2    F18F   20 C5 EC     R0 = Y + X / Y
             JSR  NORMALISE_1    F192   20 D0 E2     NORMALISE
             LDA  #$00           F195   A9 00        SET 0.5 TO R1
             STA  R1_0           F197   85 14
             STA  R1_1           F199   85 15
```

```
              STA  R1_3            F19B   85 17
              LDA  #$50            F19D   A9 50
              STA  R1_2            F19F   85 16
              JSR  MUL_FLOAT_2     F1A1   20 D9 EC      R0 = 0.5 * (Y + X /Y)
              JSR  NORMALISE_1     F1A4   20 D0 E2
              LDX  #$10            F1A7   A2 10
              LDY  #$64            F1A9   A0 64
              JSR  MOV_FP_REG      F1AB   20 39 EA      MOVE R0 TO R5, UPDATE PROVISIONAL VALUE.
              DEC  LOOP_COUNTER_2  F1AE   C6 70
              BNE  L01             F1B0   D0 C2
              RTS                  F1B2   60
%
%-------------------------------------------------------------------------------------
% LOAD TABLE VALUES FOR FUNCTIONS.
%      TABLE HEAD ADDRESS: P6,P6+1, TABLE INDEX: ACC, DESTINATION HEAD ADDRESS: IX
%-------------------------------------------------------------------------------------
              .ORIGIN  $F1C7
%
LOAD_TABLE_1  ASL                  F1C7   0A            2 BIT LEFT SHIFT TABLE INDEX.
              ASL                  F1C8   0A
              TAY                  F1C9   A8            SET INDEX TO IY.
              LDA  (P6),Y          F1CA   B1 80
              STA  $00,X           F1CC   95 00         LOAD FLOATING VALUES FROM TABLE
              INY                  F1CE   C8
              LDA  (P6),Y          F1CF   B1 80
              STA  $01,X           F1D1   95 01
              INY                  F1D3   C8
              LDA  (P6),Y          F1D4   B1 80
              STA  $02,X           F1D6   95 02
              INY                  F1D8   C8
              LDA  (P6),Y          F1D9   B1 80
              STA  $03,X           F1DB   95 03
              RTS                  F1DD   60
%
%-------------------------------------------------------------------------------------
% PATCH FOR MAIN LOOP
%-------------------------------------------------------------------------------------
MAIN_LOOP_6   BNE  L07             F1DE   D0 03
              JMP  SUB_LOOP_2      F1E0   4C FB F1      IF IT'S ')' CODE, GO TO SUB LOOP.
L07           CMP  #$26            F1E0   C9 26
              BNE  L25             F1E5   D0 03
              JMP  INPUT_STATE     F1F7   4C C2 EF      IF IT'S '&' CODE, GO TO INPUT PROCESS.
L25           CMP  #$22            F1EA   C9 22
              BNE  L26             F1EC   D0 03
              JMP  LOAD_PROGRAM    F1EE   4C 01 F3      IF IT'S '"' CODE, GO TO LOAD PROGRAM.
L26           JMP  MAIN_LOOP_7     F1F1   4C 2D E9      RETURN TO MAIN_LOOP.
%
%-------------------------------------------------------------------------------------
% SUB LOOP
%-------------------------------------------------------------------------------------
              .ORIGIN  $F1FB
%
SUB_LOOP_2    JSR  INC_POINTER_1   F1FB   20 FC E7      ADVANCE THE POINTER TO THE NEXT OF ')'.
              LDA  (P0),Y          F1FE   B1 40
              CMP  #$58            F200   C9 58
              BCC  L01             F202   90 07         IF IT'S $57 OR LESS, BRANCH
              CMP  #$5B            F204   C9 5B
              BCS  L01             F206   B0 03         IF IT'S $57 OR MORE, BRANCH
              JMP  VECTOR_1        F208   4C 0C F0      GO TO ARRAY VARIABLE PROCESS.
L01           CMP  #$51            F20B   C9 51
              BNE  L02             F20D   D0 03
              JMP  FUNCTION_SQRT   F20F   4C DE FB      IF IT'S 'Q' CODE, GO TO SQUARE ROOT FUNC.
L02           CMP  #$53            F212   C9 53
              BNE  L03             F214   D0 03
              JMP  FUNCTION_SIN2   F216   4C 4B F4      IF IT'S 'S' CODE, GO TO SINE FUNCTION.
L03           CMP  #$43            F219   C9 43
              BNE  L04             F21B   D0 03
              JMP  FUNCTION_COS    F21D   4C DA F5      IF IT'S 'C' CODE, GO TO COSINE FUNCTION.
L04           CMP  #$54            F220   C9 54
              BNE  L05             F222   D0 03
              JMP  FUNCTION_TAN    F224   4C E6 F5      IF IT'S 'T' CODE, GO TO TANGENT FUNCTION.
L05           CMP  #$4D            F227   C9 4D
```

```
             BNE  L06             F229   D0 03
             JMP  FUNCTION_ASIN   F22B   4C 92 F6     IF IT'S 'M' CODE, GO TO ARCSINE FUNCTION.
L06          CMP  #$4E            F22E   C9 4E
             BNE  L07             F230   D0 03
             JMP  FUNCTION_ACOS   F232   4C 47 F7     IF IT'S 'N' CODE, GO TO ARCCOSINE FUNCTION.
L07          CMP  #$4F            F235   C9 4F
             BNE  L08             F237   D0 03
             JMP  FUNCTION_ATAN   F239   4C F2 F5     IF IT'S 'O' CODE, GO TO ARCTANGENT FUNC.
L08          CMP  #$45            F23C   C9 45
             BNE  L09             F23E   D0 03
             JMP  FUNCTION_EXP1   F240   4C 64 F7     IF IT'S 'E' CODE, GO TO EXPONENTIAL FUNC.
L09          CMP  #$4C            F243   C9 4C
             BNE  L10             F245   D0 03
             JMP  FUNCTION_LN1    F247   4C 75 F8     IF IT'S 'L' CODE, GO TO NATURAL LOG. FUNC.
L10          CMP  #$50            F24A   C9 50
             BNE  L11             F24C   D0 03
             JMP  FUNCTION_PWR1   F24E   4C B4 FA     IF IT'S 'P' CODE, GO TO EXPONENTIATON.
L11          CMP  #$55            F251   C9 55
             BNE  L12             F253   D0 03
             JMP  FUNCTION_HSIN   F255   4C D4 FA     IF IT'S 'U' CODE, GO TO  HYPERBOLIC SINE.
L12          CMP  #$56            F258   C9 56
             BNE  L13             F25A   D0 03
             JMP  FUNCTION_HCOS   F25C   4C 21 FB     IF IT'S 'V' CODE, GO TO HYPERBOLIC COSINE.
L13          CMP  #$57            F25F   C9 57
             BNE  L14             F261   D0 03
             JMP  FUNCTION_HTAN   F263   4C 6E FB     IF IT'S 'W' CODE, GO TO HYPERBOLIC TANGENT.
L14          CMP  #$41            F266   C9 41
             BNE  L15             F268   C0 03
             JMP  FUNCTION_ABS    F26A   4C EA FB     IF IT'S 'A' CODE, GO TO ABSOLUTE FUNCTION.
L15          CMP  #$49            F26D   C9 49
             BNE  L16             F26F   C0 03
             JMP  FUNCTION_INT    F271   4C FD FB     IF IT'S 'I' CODE, GO TO INTEGER COSINE.
L16          CMP  #$52            F274   C9 52
             BNE  L17             F276   D0 03
             JMP  FUNCTION_RND    F278   4C 7B F2     IF IT'S 'R' CODE, GO TO RANDOM NUMBER FUNC.
L17          JSR  INC_POINTER_1   F27B   20 FC E7
             JMP  MAIN_LOOP_2     F27E   4C 08 E9     RETURN TO MAIN LOOP 2.
%
%-------------------------------------------------------------------------------
% PATCH FOR FORMULA OF MINUS SUBSTITUTION'A=-B'
%-------------------------------------------------------------------------------
             .ORIGIN  $F2A2
%
VAR_PATCH_1  LDA  VAR_HEAD+3,X    F2A2   BD 03 03
             STA  R0_3            F2A5   85 13
             LDA  TEMP_7          F2A7   A5 08        LOAD SIGN FLAG.
             BPL  L01             F2A9   10 06
             LDA  R0_3            F2AB   A5 13        IF SIGN FLAG IS NEGATIVE,
             EOR  #$80            F2AD   49 80         INVERT SIGN OF MANTISSA OF R0_3.
             STA  R0_3            F2AF   85 13
L01          JMP  VAR_RET_1       F2B1   4C E6 EA
%
%-------------------------------------------------------------------------------
%-----------------------  GREETING MESSAGE TEXT     (V.1.5.0  MAR/11/2023)  ----------
%-------------------------------------------------------------------------------
             .ORIGIN  $F2B4
%
                                  F2B4   0D 43 49 2D        'CR'   'C'    'I'    '-'
                                  F2B8   32 20 56 31        '2'    ' '    'V'    '1'
                                  F2BC   2E 35 20 4D        '.'    '5'    ' '    'M'
                                  F2C0   2E 59 41 4D        '.'    'Y'    'A'    'M'
                                  F2C4   41 44 41 20        'A'    'D'    'A'    ' '
                                  F2C8   32 30 32 33        '2'    '0'    '2'    '3'
                                  F2CC   0D 22              'CR'   '"'
%
%-------------------------------------------------------------------------------
% DISPLAY GREETING MESSAGE.
%-------------------------------------------------------------------------------
             .ORIGIN  $F2E0
%
GREETING     JSR  SET_P0          F2E0   20 E7 F2
             JSR  OUT_TEXT        F2E3   20 F0 F2
```

```
            RTS                 F2E6   60
%
%-------------------------------------------------------------------------------------
SET_P0      LDA  #$B0           F2E7   A9 B4
            STA  P0             F2E9   85 40
            LDA  #$F2           F2EB   A2 F2
            STA  P0+1           F2ED   85 41
            RTS                 F2EF   60
%
%-------------------------------------------------------------------------------------
OUT_TEXT    LDY  #$00           F2F0   A0 00
L01         LDA  (P0),Y         F2F2   B1 40
            CMP  #$22           F2F4   C9 22
            BEQ  L02            F2F6   F0 08        IF IT'S '"' CODE, TERMINATE DISPLAYING.
            JSR  OUT_1CHA       F2F8   20 9A E0     OUTPUT ONE CHARACTER.
            JSR  INC_POINTER_1  F2FB   20 FC E7     ADVANCE THE POINTER.
            BCC  L01            F2FE   90 F2
L02         RTS                 F300   60
%
%-------------------------------------------------------------------------------------
% LOAD PROGRAM FROM SERIAL INTERFACE.
%     WHEN RECEIVING IS COMPLETE, PRESS 'CR' KEY ON THE TERMINAL TO EXIT.
%     THE END CODE '%' IS PLACED AT THE END OF THE PROGRAM.
%-------------------------------------------------------------------------------------
LOAD_PROGRAM LDA  #$00          F301   A9 00
            STA  P0             F303   85 40
            LDA  #$10           F305   A9 10
            STA  P0+1           F307   85 41        SET PROGRAM AREA HEAD $1000 TO P0.
            LDY  #$00           F309   A0 00
            LDX  #$00           F30B   A2 00        INITIALIZE CHARACTER COUNTER.
L03         JSR  IN_1CHA        F300   20 90 E0     INPUT ONE CHARACTER FROM SERIAL INTERFACE.
            CMP  #$0D           F310   C9 0D
            BNE  L01            F312   D0 06
            CPX  #$01           F314   E0 01        IF IT'S 'CR' CODE AND CHARACTER COUNTER=1,
            BEQ  L02            F316   F0 0B         EXIT.
            LDX  #$00           F318   A2 00        OTHERWISE RESET CHARACTER COUNTER.
L01         INX                 F31A   E8
            STA  (P0),Y         F31B   91 40        PUT RECEIVED CHARACTER TO MEMORY.
            JSR  INC_POINTER_1  F31D   20 FC E7     ADVANCE THE POINTER.
            CLC                 F320   18
            BCC  L03            F321   90 EA        REPEAT.
L02         LDA  #$25           F323   A9 25
            STA  (P0),Y         F325   91 40        PUT THE END CODE '%' AT THE PROGRAM END.
            RTS                 F327   60

%
%-------------------------------------------------------------------------------------
% POSTPROCESSING OF FUNCTIONS
%-------------------------------------------------------------------------------------
            .ORIGIN  $F38E
%
FUNC_2      LDX  #$74           F38E   A2 74        RESTORE R1
            LDY  #$14           F390   A0 14
            JSR  MOV_FP_REG     F392   20 39 EA
            RTS                 F395   60
%
%-------------------------------------------------------------------------------------
% PREPROCESSING OF FUNCTIONS                                  (V.1.4.0  DEC/09/2022)
%-------------------------------------------------------------------------------------
            .ORIGIN  $F396
%
FUNC_1      LDA  RUN_FLAG       F396   A5 50        DOES NOT PULL CALCULATION STACK
            BNE  L01            F398   D0 0D         IN RUN MODE. (V.1.4.0 DEC/09/2022)
            LDA  P0             F39A   A5 40
            CMP  #$01           F39C   C9 01        DOES NOT EXIST VARIABLES NOR NUMBERS?
            BNE  L01            F39E   D0 07
            LDX  #$14           F3A0   A2 14
            LDY  #$10           F3A2   A0 10        MOVE R1 TO R0
            JSR  MOV_FP_REG     F3A4   20 39 EA      TO MAKE IT POSSIBLE TO EXECUTE.
L01         LDX  #$14           F3A7   A2 14
            LDY  #$74           F3A9   A0 74
            JSR  MOV_FP_REG     F3AB   20 39 EA     SAVE R1 TO R9
```

```
             RTS                 F3AE   60
%
%-------------------------------------------------------------------------------------
% SINE FUNCTION MACLAURIN SERIES.
%      Y = X * (K4 - X^2 * (K3 - X^2 * (K2 - X^2 * (K1 - K0 * X^2))))
%-------------------------------------------------------------------------------------
             .ORIGIN  $F3B2
%
MACLAURIN_1  LDX  #$78           F3B2   A2 78
             LDY  #$14           F3B6   A0 14
             JSR  MOV_FP_REG     F3B6   20 39 EA      MOVE ARGUMENT X,R10 TO R1.
             LDX  #$78           F3B9   A2 78
             LDY  #$10           F3BB   A0 10
             JSR  MOV_FP_REG     F3BD   20 39 EA      MOVE R10 TO R0.
             JSR  MUL_FLOAT_2    F3C0   20 D9 EC      R0 = X * X.
             JSR  NORMALISE_1    F3C3   20 D0 E2      NORMALISE.
             LDX  #$10           F3C6   A2 10
             LDY  #$7C           F3C8   A0 7C
             JSR  MOV_FP_REG     F3CA   20 39 EA      MOVE X^2 TO R11.
             NOP                 F3CD   EA
             NOP                 F3CE   EA
             NOP                 F3CF   EA
             NOP                 F3D0   EA
             NOP                 F3D1   EA
             NOP                 F3D2   EA
             NOP                 F3D3   EA
             NOP                 F3D4   EA
             LDA  #$00           F3D5   A9 00         SET ZERO TO TABLE INDEX.
             LDX  #$14           F3D7   A2 14
             JSR  LOAD_TABLE_1   F3D9   20 C7 F1      LOAD COEFFICIENT K0 TO R1.
             JSR  MUL_FLOAT_2    F3DC   20 D9 EC      R0 = K0 * X^2.
             JSR  NORMALISE_1    F3DF   20 D0 E2
             LDA  #$01           F3E2   A9 01
             LDX  #$14           F3E4   A2 14
             JSR  LOAD_TABLE_1   F3E6   20 C7 F1      LOAD K1 TO R1.
             JSR  SUB_FLOAT_2    F3E9   20 CF EC      R0 = K1 - R0.
             JSR  NORMALISE_1    F3EC   20 D0 E2
             LDX  #$7C           F3EF   A2 7C
             LDY  #$14           F3F1   A0 14
             JSR  MOV_FP_REG     F3F3   20 39 EA      MOVE X^2 TO R1.
             JSR  MUL_FLOAT_2    F3F6   20 D9 EC      R0 = X^2 * R0.
             JSR  NORMALISE_1    F3F9   20 D0 E2
             LDA  #$02           F3FC   A9 02
             LDX  #$14           F3FE   A2 14
             JSR  LOAD_TABLE_1   F400   20 C7 F1      LOAD K2 TO R1.
             JSR  SUB_FLOAT_2    F403   20 CF EC      R0 = K2 - R0.
             JSR  NORMALISE_1    F406   20 D0 E2
             LDX  #$7C           F409   A2 7C
             LDY  #$14           F40B   A0 14
             JSR  MOV_FP_REG     F40D   20 39 EA      MOVE X^2 TO R1.
             JSR  MUL_FLOAT_2    F410   20 D9 EC      R0 = X^2 * R0.
             JSR  NORMALISE_1    F413   20 D0 E2
             LDA  #$03           F416   A9 03
             LDX  #$14           F418   A2 14
             JSR  LOAD_TABLE_1   F41A   20 C7 F1      LOAD K3 TO R1.
             JSR  SUB_FLOAT_2    F41D   20 CF EC      R0 = K3 - R0.
             JSR  NORMALISE_1    F420   20 D0 E2
             LDX  #$7C           F423   A2 7C
             LDY  #$14           F425   A0 14
             JSR  MOV_FP_REG     F427   20 39 EA      MOVE X^2 TO R1.
             JSR  MUL_FLOAT_2    F42A   20 D9 EC      R0 = X^2 * R0.
             JSR  NORMALISE_1    F42D   20 D0 E2
             LDA  #$04           F430   A9 04
             LDX  #$14           F432   A2 14
             JSR  LOAD_TABLE_1   F434   20 C7 F1      LOAD K4 TO R1.
             JSR  SUB_FLOAT_2    F437   20 CF EC      R0 = K4 - R0.
             JSR  NORMALISE_1    F43A   20 D0 E2
             LDX  #$78           F43D   A2 78
             LDY  #$14           F43F   A0 14
             JSR  MOV_FP_REG     F441   20 39 EA      MOVE X TO R1.
             JSR  MUL_FLOAT_2    F444   20 D9 EC      R0 = X * R0.
             JSR  NORMALISE_1    F447   20 D0 E2
```

```
             RTS                 F44A   60
%
%-------------------------------------------------------------------------------------
% SINE FUNCTION
%-------------------------------------------------------------------------------------
FUNC_SIN_2   JSR  FUNC_1         F44B   20 96 F3     INDIVIDUAL EXECUTION PROCESS, SAVE R1.
             JSR  SIN_1          F44E   20 5F F4     SINE FUNCTION WITH ARGUMENTS INTERVAL.
             JSR  FUNC_2         F451   20 8E F3     RESTORE R1
             NOP                 F454   EA
             NOP                 F455   EA
             NOP                 F456   EA
FUNC_RETURN_1JSR  INC_POINTER_1  F457   20 FC E7     ADVANCE THE POINTER.
             LDY  #$00           F45A   A0 00
             JMP  MAIN_LOOP_2    F45C   4C 08 E9     RETURN TO MAIN LOOP 2
%
%-------------------------------------------------------------------------------------
% SINE FUNCTION WITH CONVERSION OF ARGUMENTS PER PI/2 INTERVAL.
%      ARGUMENTS ARE CONVERTED TO  0 <= X < PI/2, AND RESULTS ARE CONVERTED.
%-------------------------------------------------------------------------------------
SIN_1        LDA  #$00           F45F   A9 00
             STA  FUNC_SGN_FLAG  F461   85 91        INITIALIZE SIGN FLAG.
             LDA  R0_3           F461   A5 13
             BPL  L01            F465   10 06        IF SIGN IS PLUS, BRANCH
             EOR  #$80           F467   49 80        IF SIGN IS MINUS, CONVERT TO ABSOLUTE.
             INC  FUNC_SGN_FLAG  F469   E6 91        SET 1 TO SIGN FLAG
             STA  R0_3           F46B   85 13
L01          LDX  #$10           F46D   A2 10
             LDY  #$78           F46F   A0 78
             JSR  MOV_FP_REG     F471   20 39 EA     MOVE ARGUMENT X,R0 TO R10.
L03          LDX  #$78           F474   A2 78
             LDY  #$14           F476   A0 14
             JSR  MOV_FP_REG     F478   20 39 EA     MOVE ARGUMENT X,R10 TO R1.
             LDA  #$80           F47B   A9 80
             STA  P6             F47D   85 80        SET HEAD ADDRESS OF TABLE $FF80 TO P6
             LDA  #$FF           F47F   A9 FF
             STA  P6+1           F481   85 81
             LDA  #$05           F483   A9 05
             LDX  #$10           F485   A2 10
             JSR  LOAD_TABLE_1   F487   20 C7 F1     LOAD 2*PI TO R0.
             JSR  SUB_FLOAT_2    F48A   20 CF EC     R0 = X - 2*PI.
             JSR  NORMALISE_1    F48D   20 D0 E2
             LDA  R0_3           F490   A5 13
             BMI  L02            F492   30 0A        THE RESULT IS MINUS, DO NOT SUBTRACT.
             LDX  #$10           F494   A2 10
             LDY  #$78           F496   A0 78
             JSR  MOV_FP_REG     F498   20 39 EA     THE RESULT IS PLUS, RESTORE RESULT TO R10.
             CLC                 F49B   18
             BCC  L03            F49C   90 D6        REPEAT TO COMPARE X AND 2*PI.
L02          LDX  #$78           F49E   A2 78
             LDY  #$14           F4A0   A0 14
             JSR  MOV_FP_REG     F4A2   20 39 EA     MOVE X,R10 TO R1.
             LDA  #$08           F4A5   A9 08
             LDX  #$10           F4A7   A2 10
             JSR  LOAD_TABLE_1   F4A9   20 C7 F1     LOAD 0.5*PI TO R0.
             JSR  SUB_FLOAT_2    F4AC   20 CF EC      R0 = X - 0.5*PI.
             LDA  R0_3           F4AF   A5 13
             BPL  L04            F4B1   10 06        IF NOT (X < 0.5*PI),BRANCH
             JSR  MACLIN_SIN     F4B3   20 CE F5     CALCULATION SINE BY MACLAURIN SERIES.
             JMP  L05            F4B6   4C 4F F5
%
L04          LDX  #$78           F4B9   A2 78
             LDY  #$14           F4BB   A0 14
             JSR  MOV_FP_REG     F4BD   20 39 EA     MOVE X,R10 TO R1.
             LDA  #$07           F4C0   A9 07
             LDX  #$10           F4C2   A2 10
             JSR  LOAD_TABLE_1   F4C4   20 C7 F1     LOAD PI TO R0.
             JSR  SUB_FLOAT_2    F4C7   20 CF EC     X - PI
             LDA  R0_3           F4CA   A5 13
             BPL  L06            F4CC   10 21        IF NOT (X < PI), BRANCH
             LDA  #$07           F4CE   A9 07
             LDX  #$14           F4D0   A2 14
             JSR  LOAD_TABLE_1   F4D2   20 C7 F1     LOAD PI TO R1.
```

```
                LDX  #$78           F4D5   A2 78
                LDY  #$10           F4D7   A0 10
                JSR  MOV_FP_REG     F4D9   20 39 EA     MOVE X,R10 TO R0.
                JSR  SUB_FLOAT_2    F4DC   20 CF EC     R0 = PI - X.
                JSR  NORMALISE_1    F4DF   20 D0 E2
                LDX  #$10           F4E2   A2 10
                LDY  #$78           F4E4   A0 78
                JSR  MOV_FP_REG     F4E6   20 39 EA    MOVE R0 TO R10
                JSR  MACLN_SIN      F4E9   20 CE F5     CALCULATION SINE BY MACLAURIN SERIES.
                JMP  L05            F4EC   4C 4F F5
%
L06             LDX  #$78           F4EE   A2 78
                LDY  #$14           F4F1   A0 14
                JSR  MOV_FP_REG     F4F3   20 39 EA     MOVE X,R10 TO R1.
                LDA  #$06           F4F6   A9 06
                LDX  #$10           F4F8   A2 10
                JSR  LOAD_TABLE_1   F4FA   20 C7 F1     LOAD 1.5*PI TO R0.
                JSR  SUB_FLOAT_2    F4FD   20 CF EC     X - 1.5*PI
                LDA  R0_3           F500   A5 13
                BPL  L07            F502   10 27        IF NOT (X < 1.5*PI), BRANCH
                LDX  #$78           F504   A2 78
                LDY  #$14           F506   A0 14
                JSR  MOV_FP_REG     F508   20 39 EA     MOVE X,R10 TO R1.
                LDA  #$07           F50B   A9 07
                LDX  #$10           F50D   A2 10
                JSR  LOAD_TABLE_1   F51F   20 C7 F1     LOAD PI TO R0.
                JSR  SUB_FLOAT_2    F512   20 CF EC     R0= X - PI
                JSR  NORMALISE_1    F515   20 D0 E2
                LDX  #$10           F518   A2 10
                LDY  #$78           F51A   A0 78
                JSR  MOV_FP_REG     F51C   20 39 EA     MOVE R0 TO R10.
                JSR  MACLN_SIN      F51F   20 CE F5     CALCULATION SINE BY MACLAURIN SERIES.
                LDA  R0_3           F522   A5 13
                EOR  #$80           F524   49 80        INVERT SIGN OF THE RESULT.
                STA  R0_3           F526   85 13
                JMP  L05            F528   4C 4F F5
%
L07             LDA  #$05           F52B   A9 05
                LDX  #$14           F52D   A2 14
                JSR  LOAD_TABLE_1   F52F   20 C7 F1     LOAD 2*PI TO R1.
                LDX  #$78           F532   A2 78
                LDY  #$10           F534   A0 10
                JSR  MOV_FP_REG     F536   20 39 EA     MOVE X,R10 TO R0.
                JSR  SUB_FLOAT_2    F539   20 CF EC     R0 = 2*PI - X.
                JSR  NORMALISE_1    F53C   20 D0 E2
                LDX  #$10           F53F   A2 10
                LDY  #$78           F541   A0 78
                JSR  MOV_FP_REG     F543   20 39 EA     MOVE R0 TO R10.
                JSR  MACLN_SIN      F546   20 CE F5     CALCULATION SINE BY MACLAURIN SERIES.
                LDA  R0_3           F549   A5 13
                EOR  #$80           F54B   49 80        INVERT SIGN OF THE RESULT.
                STA  R0_3           F54D   85 13
L05             LDA  FUNC_SGN_FLAG  F54F   A5 91        SING FLAG OF THE ARGUMENT.
                BEQ  L08            F551   F0 06
                LDA  R0_3           F553   A5 13
                EOR  #$80           F555   49 80        IF IT'S MINUS, INVERT THE RESULT SIGN.
                STA  R0_3           F557   85 13
L08             RTS                 F559   60
%
%-------------------------------------------------------------------------------------
% COSINE FUNCTION.    COS(X) = SIN(X + PI/2)
%-------------------------------------------------------------------------------------
                .ORIGIN  $F55D
%
COS_1           LDA  #$80           F55D   A9 80        INDIVIDUAL EXECUTION PROCESS, SAVE R1.
                STA  P6             F55F   85 80
                LDA  #$FF           F561   A9 FF
                STA  P6+1           F563   85 81
                LDA  #$08           F565   A9 08
                LDX  #$14           F567   A2 14
                JSR  LOAD_TABLE_1   F569   20 C7 F1     LOAD 0.5*PI TO R1.
                JSR  ADD_FLOAT_2    F56C   20 C5 EC     R0 = 0.5*PI + X.
```

```
             JSR  NORMALISE_1     F56F   20 D0 E2
             JSR  SIN_1           F572   20 5F F4      CALCULATION SINE.
             RTS                  F575   60
%
%-------------------------------------------------------------------------------------
% TANGENT FUNCTION.        TAN(X) = SIN(X)/COS(X)
%-------------------------------------------------------------------------------------
             .ORIGIN  $F57E
%
TAN_1        JMP  TAN_PATCH_1     F57E   4C B3 F5
%
             .ORIGIN  $F588
%
TAN_PATCH_2  LDA  #$80            F588   A9 80
             STA  P6              F58A   85 80
             LDA  #$FF            F58C   A9 FF
             STA  P6+1            F58E   85 81
             LDA  #$08            F590   A9 08
             LDX  #$14            F592   A2 14
             JSR  LOAD_TABLE_1    F594   20 C7 F1
             JSR  ADD_FLOAT_2     F597   20 C5 EC      R0 = 0.5*PI + X.
             JSR  NORMALISE_1     F59A   20 D0 E2
             JSR  SIN_2           F59D   20 5F F4      R0 = COS(X).
             LDX  #$70            F5A0   A2 70
             LDY  #$14            F5A2   A0 14
             JSR  MOV_FP_REG      F5A4   20 39 EA      MOVE SIN(X),R8 TO R1.
             JSR  DIV_FLOAT_1     F5A7   20 E3 EC      R0 = SIN(X) / COS(X).
             JSR  NORMALISE_1     F5AA   20 D0 E2
             RTS                  F5AD   60
%
             .ORIGIN  $F5B3
%
TAN_PATCH_1  LDX  #$10            F5B3   A2 10
             LDY  #$6C            F5B5   A0 6C
             JSR  MOV_FP_REG      F5B7   20 39 EA      MOVE ARGUMENT,R0 TO R7.
             JSR  SIN_2           F5BA   20 5F F4      R0 = SIN(X).
             LDX  #$10            F5BD   A2 10
             LDY  #$70            F5BF   A0 70
             JSR  MOV_FP_REG      F5C1   20 39 EA      MOVE SIN(X),R0 TO R8.
             LDX  #$6C            F5C4   A2 6C
             LDY  #$10            F5C6   A0 10
             JSR  MOV_FP_REG      F5C8   20 39 EA      MOVE ARGUMENT,R7 TO R0.
             JMP  TAN_PATCH_2     F5CB   4C 88 F5
%
%-------------------------------------------------------------------------------------
% SINE FUNCTION
%-------------------------------------------------------------------------------------
MACLIN_SIN   LDA  #$80            F5CE   A9 80         SET TABLE HEAD ADDRESS $FF80 TO P6
             STA  P6              F5D0   85 80
             LDA  #$FF            F5D2   A9 FF
             STA  P6+1            F5D4   85 81
             JSR  MACLAURIN_1     F5D6   20 B2 F3      MACLAURIN SERIES.
             RTS                  F5D9   60
%
%
%-------------------------------------------------------------------------------------
% COSINE FUNCTION
%-------------------------------------------------------------------------------------
FUNCTION_COS JSR  FUNC_1          F5DA   20 96 F3
             JSR  COS_1           F5DD   20 5D F5
             JSR  FUNC_2          F5E0   20 8E F3
             JMP  FUNC_RETURN     F5E3   4C 57 F4
%
%-------------------------------------------------------------------------------------
% TANGENT FUNCTION
%-------------------------------------------------------------------------------------
FUNCTION_TAN JSR  FUNC_1          F5E6   20 96 F3
             JSR  TAN_1           F5E9   20 7E F5
             JSR  FUNC_2          F5EC   20 8E F3
             JMP  FUNC_RETURN     F5EF   4C 57 F4
%
%-------------------------------------------------------------------------------------
```

```
% ARCTANGENT FUNCTION.
%      BISECTION METHOD (IF X < TAN(Y), Y = Y - PI/2^N,  ELSE Y = Y + PI/2^N)
%-------------------------------------------------------------------------------
FUNCTION_ATAN JSR  FUNC_1          F5F2   20 96 F3
              JSR  ATAN_1          F5F5   20 FE F5
              JSR  FUNC_2          F5F8   20 8E F3
              JMP  FUNC_RETURN     F5FB   4C 57 F4
%
ATAN_1        LDA  #$00            F5FE   A9 00
              STA  FUNC_SGN_FLG2   F600   85 92
              LDA  R0_3            F602   A5 13
              BPL  L01             F604   10 06
              EOR  #$80            F606   49 80
              INC  FUNC_SGN_FLG2   F608   E6 92
              STA  R0_3            F60A   85 13
L01           LDX  #$10            F60C   A2 10        MOVE ARGUMENT,R0 TO R3
              LDY  #$34            F60E   A0 34
              JSR  MOV_FP_REG      F610   20 39 EA
              LDA  #$00            F613   A9 00
              STA  P6              F615   85 80
              LDA  #$FF            F617   A9 FF
              STA  P6+1            F619   85 81        SET TABLE HEAD ADDRESS $FF00 TO P6.
              LDA  #$00            F61B   A9 00
              LDX  #$68            F61D   A2 68
              JSR  LOAD_TABLE_1    F61F   20 C7 F1     LOAD PROVISIONAL PI/4 TO R6.
              LDA  #$01            F622   A9 01
              STA  LOOP_CONTER     F624   85 90        INITIALIZE LOOP COUNTER.
L04           LDX  #$68            F626   A2 68
              LDY  #$10            F628   A0 10
              JSR  MOV_FP_REG      F62A   20 39 EA     MOVE Y,R6 TO R0.
              JSR  TAN_1           F62D   20 7E F5     R0 = TAN(Y).
              LDX  #$34            F630   A2 34
              LDY  #$14            F632   A0 14
              JSR  MOV_FP_REG      F634   20 39 EA     MOVE X,R3 TO R1.
              JSR  SUB_FLOAT_2     F637   20 CF EC     X - TAN(Y).
              LDA  R0_3            F63A   A5 13
              BPL  L02             F63C   10 22
              LDX  #$68            F63E   A2 68
              LDY  #$14            F640   A0 14
              JSR  MOV_FP_REG      F642   20 39 EA     MOVE Y,R6 TO R1.
              LDA  #$00            F645   A9 00
              STA  P6              F647   85 80
              LDA  LOOP_COUNTER    F649   A5 90
              LDX  #$10            F64B   A2 10
              JSR  LOAD_TABLE_1    F64D   20 C7 F1     LOAD K(LOOP_COUNTER) TO R0.
              JSR  SUB_FLOAT_2     F650   20 CF EC     R0 = Y - K(LOOP_COUNTER).
              JSR  NORMALISE_1     F653   20 D0 E2
              LDX  #$10            F656   A2 10
              LDY  #$68            F656   A0 68
              JSR  MOV_FP_REG      F65A   20 39 EA     MOVE R0 TO R1.
              CLC                  F65D   18
              BCC  L03             F65E   90 1F
%
L02           LDX  #$68            F660   A2 68
              LDY  #$14            F662   A0 14
              JSR  MOV_FP_REG      F664   20 39 EA     MOVE Y,R6 TO R0.
              LDA  #$00            F667   A9 00
              STA  P6              F669   85 80
              LDA  LOOP_COUNTER    F66B   A5 90
              LDX  #$10            F66D   A2 10
              JSR  LOAD_TABLE_1    F66F   20 C7 F1     LOAD K(LOOP_COUNTER) TO R0.
              JSR  ADD_FLOAT_2     F672   20 C5 EC     R0 = Y + K(LOOP_COUNTER).
              JSR  NORMALISE_1     F675   20 D0 E2
              LDX  #$10            F678   A2 10
              LDY  #$68            F67A   A0 68
              JSR  MOV_FP_REG      F67C   20 39 EA     MOVE R0 TO R1.
L03           INC  LOOP_COUNTER    F67F   E6 90
              LDA  LOOP_COUNTER    F681   A5 90
              CMP  #$10            F683   C9 10
              BNE  L04             F685   D0 9F        REPEAT UNTIL LOOP_COUNTER=15
              LDA  FUNC_SGN_FLG2   F687   A5 92
              BEQ  L05             F689   F0 06
```

```
             LDA  R0_3           F68B   A5 13
             EOR  #$80           F68D   49 80        IF ARGUMENT IS MINUS, INVERT SIGN.
             STA  R0_3           F68F   85 13
L05          RTS                 F691   60
%
%-------------------------------------------------------------------------------------
%  ARCSINE FUNCTION
%-------------------------------------------------------------------------------------
FUNCTION_ASINJSR  FUNC_1         F692   20 96 F3
             JSR  ASIN_1         F695   20 9E F6
             JSR  FUNC_2         F698   20 8E F3
             JMP  FUNC_RETURN    F69B   4C 57 F4
%
%-------------------------------------------------------------------------------------
%  ARCSINE FUNCTION
%      BISECTION METHOD (IF X < SIN(Y), Y = Y - PI/2^N,  ELSE Y = Y + PI/2^N)
%-------------------------------------------------------------------------------------
ASIN_1       LDA  #$00           F69E   A9 00
             STA  FUNC_SGN_FLG2  F6A0   85 92        INITIALIZE SIGN FLAG
             LDA  R0_3           F6A2   A5 13
             BPL  L01            F6A4   10 06
             EOR  #$80           F6A6   49 80        IF ARGUMENT IS MINUS,
             INC  FUNC_SGN_FLG2  F6A8   E6 92         INVERT SIGN OF MANTISSA
             STA  R0_3           F6AA   85 13
L01          LDX  #$10           F6AC   A2 10
             LDY  #$34           F6AE   A0 34
             JSR  MOV_FP_REG     F6B0   20 39 EA     MOVE ARGUMENT X,R0 TO R3
             LDA  #$00           F6B3   A9 00
             STA  P6             F6B5   85 80
             LDA  #$FF           F6B7   A9 FF
             STA  P6+1           F6B9   85 81        SET TABLE HEAD ADDRESS $FF00 TO P6.
             JSR  ASIN_ERR       F6BB   20 35 F7     ERROR JUDGEMENT IF X IS OVER 1.
             LDA  #$00           F6BE   A9 00
             LDX  #$68           F6C0   A2 68
             JSR  LOAD_TABLE_1   F6C2   20 C7 F1     LOAD PROVISIONAL PI/4 TO R6.
             LDA  #$01           F6C5   A9 01
             STA  LOOP_COUNTER   F6C7   85 90        INITIALIZE LOOP COUNTER.
L04          LDX  #$68           F6C9   A2 68
             LDY  #$10           F6CB   A0 10
             JSR  MOV_FP_REG     F6CD   20 39 EA     MOVE Y,R6 TO R0.
             JSR  SIN_1          F6D0   20 5F F4     R0 = SIN(Y).
             LDX  #$34           F6D3   A2 34
             LDY  #$14           F6D5   A0 14
             JSR  MOV_FP_REG     F6D7   20 39 EA     MOVE Y,R3 TO R1.
             JSR  SUB_FLOAT_2    F6DA   20 CF EC     X - SIN(Y).
             LDA  R0_3           F6DD   A5 13
             BPL  L02            F6DF   10 22
             LDX  #$68           F6E1   A2 68
             LDY  #$14           F6E3   A0 14
             JSR  MOV_FP_REG     F6E5   20 39 EA     MOVE Y,R6 TO R1.
             LDA  #$00           F6E8   A9 00
             STA  P6             F6EA   85 80
             LDA  LOOP_COUNTER   F6EC   A5 90
             LDX  #$10           F6EE   A2 10
             JSR  LOAD_TABLE_1   F6F0   20 C7 F1     LOAD K(LOOP_COUNTER) TO R0.
             JSR  SUB_FLOAT_2    F6F3   20 CF EC     R0 = Y - K(LOOP_COUNTER).
             JSR  NORMALISE_1    F6F6   20 D0 E2
             LDX  #$10           F6F9   A2 10
             LDY  #$68           F6FB   A0 68
             JSR  MOV_FP_REG     F6FD   20 39 EA     MOVE R0 TO R1.
             CLC                 F700   18
             BCC  L03            F701   90 1F
%
L02          LDX  #$68           F703   A2 68
             LDY  #$14           F705   A0 14
             JSR  MOV_FP_REG     F707   20 39 EA     MOVE Y,R6 TO R0.
             LDA  #$00           F70A   A9 00
             STA  P6             F70C   85 80
             LDA  LOOP_COUNTER   F70E   A5 90
             LDX  #$10           F710   A2 10
             JSR  LOAD_TABLE_1   F712   20 C7 F1     LOAD K(LOOP_COUNTER) TO R0.
             JSR  ADD_FLOAT_2    F715   20 C5 EC     R0 = Y + K(LOOP_COUNTER).
```

```
             JSR  NORMALISE_1    F718   20 D0 E2
             LDX  #$10           F71B   A2 10
             LDY  #$68           F71D   A0 68
             JSR  MOV_FP_REG     F71F   20 39 EA     MOVE R0 TO R1.
L03          INC  LOOP_COUNTER   F722   E6 90
             LDA  LOOP_COUNTER   F724   A5 90
             CMP  #$10           F726   C9 10
             BNE  L04            F728   D0 9F        REPEAT UNTIL LOOP_COUNTER=15
             LDA  FUNC_SGN_FLG2  F72A   A5 92
             BEQ  L05            F72C   F0 06
             LDA  R0_3           F72E   A5 13
             EOR  #$80           F730   49 80        IF ARGUMENT IS MINUS, INVERT SIGN.
             STA  R0_3           F732   85 13
L05          RTS                 F734   60
%
%-------------------------------------------------------------------------------------
% ERROR JUDGEMENT IF X IS OVER 1.
%-------------------------------------------------------------------------------------
%
ASIN_ERR     LDA  #$1F           F735   A9 1F        TABLE INDEX = 31
             LDX  #$14           F737   A2 14
             JSR  LOAD_TABLE_1   F739   20 C7 F1     LOAD 1.0 TO R1
             JSR  SUB_FLOAT_2    F73C   20 CF EC     1.0 - X
             LDA  R0_3           F73F   A5 13
             BPL  L06            F741   10 03
             JMP  ERROR_PROC     F743   4C 43 F7     IF X > 1.0, GO TO ERROR PROCESS.
L06          RTS                 F746   60
%
%-------------------------------------------------------------------------------------
% ARCCOSINE FUNCTION.
%-------------------------------------------------------------------------------------
FUNCTION_ACOSJSR  FUNC_1         F747   20 96 F3
             JSR  ACOS_1         F74A   20 53 F7
             JSR  FUNC_2         F74D   20 8E F3
             JMP  FUNC_RETURN    F750   4C 57 F4
%
%-------------------------------------------------------------------------------------
% ARCCOSINE FUNCTION.      ARCCOS(X) = PI/2 - ARCSIN(X)
%-------------------------------------------------------------------------------------
ACOS_1       JSR  ASIN_1         F753   20 9E F6     ARCSIN(X)
             LDA  #$1E           F756   A9 1E
             LDX  #$14           F758   A2 14
             JSR  LOAD_TABLE_1   F75A   20 C7 F1     LOAD 0.5*PI TO R1
             JSR  SUB_FLOAT_2    F75D   20 CF EC     R0 = 0.5*PI - ARCSIN(X)
             JSR  NORMALISE_1    F760   20 D0 E2
             RTS                 F763   60
%
%-------------------------------------------------------------------------------------
% EXPONENTIAL FUNCTION.
%-------------------------------------------------------------------------------------
FUNCTION_EXP1JSR  FUNC_1         F764   20 96 F3
             JSR  EXP_1          F767   20 70 F7
             JSR  FUNC_2         F76A   20 8E F3
             JMP  FUNC_RETURN    F76D   4C 57 F4
%
%-------------------------------------------------------------------------------------
EXP_1        JSR  SECT_EXP       F770   20 D1 F7
             NOP                 F773   EA
             NOP                 F774   EA
             NOP                 F775   EA
             NOP                 F776   EA
             NOP                 F777   EA
             NOP                 F778   EA
             RTS                 F779   60
%
%-------------------------------------------------------------------------------------
% EXPONENTIAL FUNCTION BY MACLAURIN SERIES.
%      Y = 1+X*(K8+X*(K7+X*(K6+X*(K5+X*(K4+X*(K3+X*(K2+X*(K1+X*K0)))))))).
%-------------------------------------------------------------------------------------
MACLN_2      LDX  #$10           F77A   A2 10
             LDY  #$78           F77C   A0 78
             JSR  MOV_FP_REG     F77E   20 39 EA     MOVE ARGUMENT X,R0 TO R10
```

```
                LDA  #$A4           F781   A9 A4
                STA  P6             F783   85 80
                LDA  #$FF           F785   A9 FF
                STA  P6+1           F787   85 81        SET TABLE HEAD ADDRESS $FFA4 TO P6.
                LDA  #$00           F789   A9 00
                LDX  #$14           F78B   A2 14
                JSR  LOAD_TABLE_1   F78D   20 C7 F1     LOAD K(0) TO R1.
                LDX  #$78           F790   A2 78
                LDY  #$10           F792   A0 10
                JSR  MOV_FP_REG     F794   20 39 EA     MOVE X,R10 TO R0.
                JSR  MUL_FLOAT_2    F797   20 D9 EC     R0 = K(0) * X.
                JSR  NORMALISE_1    F79A   20 D0 E2
                LDA  #$01           F79D   A9 01
                STA  LOOP_COUNTER   F79F   85 90        INITIALIZE LOOP COUNTER.
L01             LDA  LOOP_COUNTER   F7A1   A5 90
                LDX  #$14           F7A3   A2 14
                JSR  LOAD_TABLE_1   F7A5   20 C7 F1     LOAD K(LOOP_COUNTER) TO R1.
                JSR  ADD_FLOAT_2    F7A8   20 C5 EC     R0 = K(LOOP_COUNTER) + R0.
                JSR  NORMALISE_1    F7AB   20 D0 E2
                LDX  #$78           F7AE   A2 78
                LDY  #$14           F7B0   A0 14
                JSR  MOV_FP_REG     F7B2   20 39 EA     MOVE X,R10 TO R1.
                JSR  MUL_FLOAT_2    F7B5   20 D9 EC    R0 = X * (K(LOOP_COUNTER) + R0).
                JSR  NORMALISE_1    F7B8   20 D0 E2
                INC  LOOP_COUNTER   F7BB   E6 90
                LDA  LOOP_COUNTER   F7BD   A5 90
                CMP  #$09           F7BF   C9 09        REPEAT UNTIL LOOP_COUNTER=9
                BNE  L01            F7C1   D0 DE
                LDA  #$08           F7C3   A9 08
                LDX  #$14           F7C5   A2 14
                JSR  LOAD_TABLE_1   F7C7   20 C7 F1     LOAD K(8) TO R1
                JSR  ADD_FLOAT_2    F7CA   20 C5 EC     R0 = 1 + R0
                JSR  NORMALISE_1    F7CD   20 D0 E2
                RTS                 F7D0   60
%
%-------------------------------------------------------------------------------
% DECOMPOSITION PROCESS OF EXPONENTIAL FUNCTION
%-------------------------------------------------------------------------------
SECT_EXP        LDX  #$10           F7D1   A2 10
                LDY  #$70           F7D3   A0 70
                JSR  MOV_FP_REG     F7D5   20 39 EA     MOVE ARGUMENT,R0 TO R8
                LDA  #$C8           F7D8   A9 C8
                STA  P6             F7DA   85 80
                LDA  #$FF           F7DC   A9 FF
                STA  P6+1           F7DE   85 81        SET TABLE HEAD ADDRESS $FFC8 TO P6.
                LDA  #$00           F7E0   A9 00        TABLE INDEX = 0
                LDX  #$14           F7E2   A2 14
                JSR  LOAD_TABLE_1   F7E4   20 C7 F1     LOAD OVERFLOW JUDGEMENT VALUE TO R1.
                JSR  SUB_FLOAT_2    F7E7   20 CF EC
                LDA  R0_3           F7EA   A5 13
                JMP  EXP_PATCH_2    F7EC   4C B2 FB     PATCH FOR ERROR PROCESS.
%
                NOP                 F7EF   EA
                NOP                 F7F0   EA
EXP_PATCH_2R    LDA  R8_3           F7F1   A5 73
                AND  #$7F           F7F3   29 7F        REMOVE SIGN OF MANTISSA.
                CMP  #$02           F7F5   C9 02
                BEQ  L02            F7F7   F0 50        IF EXPONENT=2, BRANCH.
                CMP  #$01           F7F9   C9 01
                BEQ  L03            F7FB   F0 0F        IF EXPONENT=1, BRANCH.
                LDX  #$70           F7FD   A2 70
                LDY  #$10           F7FF   A0 10
                JSR  MOV_FP_REG     F801   20 39 EA     MOVE X,R8 TO R0
                LDA  #$A4           F804   A9 A4
                STA  P6             F806   85 80        SET TABLE HEAD ADDRESS $FFA4 TO P6.
                JSR  MACLN_2        F808   20 7A F7     R0 = EXP(X).
                RTS                 F80B   60
%
L03             LDX  #$70           F80C   A2 70        WHEN THE INTEGER PART IS ONE DIGIT.
                LDY  #$10           F80E   A0 10
                JSR  MOV_FP_REG     F810   20 39 EA     MOVE X,R8 TO R0.
                LDA  R0_2           F813   A5 12
```

```
             AND  #$0F           F815   29 0F        EXTRACT DECIMAL FRACTION OF ARGUMENT.
             STA  R0_2           F817   85 12
             JSR  NORMALISE_1    F819   20 D0 E2
             LDA  #$A4           F81C   A9 A4
             STA  P6             F81E   85 80        SET TABLE HEAD ADDRESS $FFA4 TO P6.
             JSR  MACLN_2        F81E   20 7A F7     R0 = EXP(X).
             LDA  R8_2           F823   A5 72        EXTRACT INTEGER PART OF ARGUMENT.
             LSR                 F825   4A           4 BIT RIGHT SHIFT.
             LSR                 F826   4A
             LSR                 F827   4A
             LSR                 F828   4A
             STA  P1             F829   85 42
L06          JMP  EXP_PATCH_1    F82B   4C A3 FB
%
             NOP                 F82E   EA
EXP_PATCH_1R LDA  #$07           F82F   A9 07
             STA  LOOP_COUNTER   F831   85 90        INITIALIZE LOOP COUNTER.
L05          LSR  P1             F833   46 42
             BCC  L04            F835   90 0D
             LDA  LOOP_COUNTER   F837   A5 90
             LDX  #$14           F839   A2 14
             JSR  LOAD_TABLE_1   F83B   20 C7 F1     LOAD CONSTANT TO R1.
             JSR  MUL_FLOAT_2    F83E   20 D9 EC     R0 = R1 * R0.
             JSR  NORMALISE_1    F841   20 D0 E2
L04          DEC  LOOP_COUNTER   F844   C6 90
             BNE  L05            F846   D0 EB
             RTS                 F848   60
%
L02          LDX  #$70           F849   A2 70        WHEN THE INTEGER PART IS TWO DIGIT.
             LDY  #$10           F84B   A0 10
             JSR  MOV_FP_REG     F84D   20 39 EA     MOVE X,R8 TO R0.
             LDA  R0_2           F850   A5 12
             STA  R0_0           F852   85 10        MOVE INTEGER PART TO R0 LOWER
             LDA  #$00           F854   A9 00
             STA  R0_1           F856   85 11
             STA  R0_2           F858   85 12        CLEAR R0 UPPER
             JSR  FLOAT_2_INT16  F85A   20 3B EE    CONVERT BCD TO BINARY, PUT IT TO P1.
             LDX  #$70           F85D   A2 70
             LDY  #$10           F85F   A0 10
             JSR  MOV_FP_REG     F861   20 39 EA     MOVE X,R8 TO R0.
             LDA  #$00           F864   A9 00
             STA  R0_2           F866   85 12        EXTRACT DECIMAL FRACTION OF ARGUMENT.
             JSR  NORMALISE_1    F868   20 D0 E2
             LDA  #$A4           F86B   A9 A4
             STA  P6             F86D   85 80        SET TABLE HEAD ADDRESS $FFA4 TO P6.
             JSR  MACLN_2        F86F   20 7A F7     R0 = EXP(X).
             JMP  L06            F872   4C 2B F8
%
%-------------------------------------------------------------------------------------
% NATURAL LOGARITHM FUNCTION
%-------------------------------------------------------------------------------------
FUNCTION_LN1 JSR  FUNC_1         F875   20 96 F3
             JSR  LN_1           F878   20 81 F8
             JSR  FUNC_2         F87B   20 8E F3
             JMP  FUNC_RETURN    F87E   4C 57 F4
%
%-------------------------------------------------------------------------------------
LN_1         JSR  PRE_LN_1       F881   20 50 F9
             NOP                 F884   EA
             NOP                 F885   EA
             NOP                 F886   EA
             RTS                 F887   60
%
%-------------------------------------------------------------------------------------
% NATURAL LOGARITHM FUNCTION BY MACLAURIN SERIES.
%      Z = X*(K4+X^2*(K3+X^2*(K2+X^2*(K1+K0*X^2)))).
%-------------------------------------------------------------------------------------
             .ORIGIN  $F88B
%
MACLN_3      LDX  #$10           F88B   A2 10
             LDY  #$78           F88D   A0 78
             JSR  MOV_FP_REG     F88F   20 39 EA     MOVE ARGUMENT X,R0 TO R10.
```

```
                LDX  #$10           F892   A2 10
                LDY  #$14           F894   A0 14
                JSR  MOV_FP_REG     F896   20 39 EA      MOVE X,R0 TP R1.
                JSR  MUL_FLOAT_2    F899   20 D9 EC      R0 = X * X.
                JSR  NORMALISE_1    F89C   20 D0 E2
                LDX  #$10           F89F   A2 10
                LDY  #$70           F8A1   A0 70
                JSR  MOV_FP_REG     F8A3   20 39 EA      MOVE X^2 TO R8
MACLN_4         LDA  #$80           F8A6   A9 80
                STA  P6             F8A8   85 80
                LDA  #$FE           F8AA   A9 FE
                STA  P6+1           F8AC   85 81         SET TABLE HEAD ADDRESS $FE80 TO P6.
                LDA  #$00           F8AE   A9 00         TABLE INDEX = 0
                LDX  #$14           F8B0   A2 14
                JSR  LOAD_TABLE_1   F8B2   20 C7 F1      LOAD K0 TO R1
                LDX  #$70           F8B5   A2 70
                LDY  #$10           F8B7   A0 10
                JSR  MOV_FP_REG     F8B9   20 39 EA      MOVE X^2,R8 TO R0
                JSR  MUL_FLOAT_2    F8BC   20 D9 EC      R0 = K0 * X^2.
                JSR  NORMALISE_1    F8BF   20 D0 E2
                LDA  #$01           F8C2   A9 01
                STA  LOOP_COUNTER   F8C4   85 90         INITIALIZE LOOP COUNTER.
L01             LDA  LOOP_COUNTER   F8C6   A5 90
                LDX  #$14           F8C8   A2 14
                JSR  LOAD_TABLE_1   F8CA   20 C7 F1      LOAD K(LOOP_COUNTER) TO R1.
                JSR  ADD_FLOAT_2    F8CD   20 C5 EC     R0 = K(LOOP_COUNTER) + R0.
                JSR  NORMALISE_1    F8D0   20 D0 E2
                LDX  #$70           F8D3   A2 70
                LDY  #$14           F8D5   A0 14
                JSR  MOV_FP_REG     F8D7   20 39 EA      MOVE X^2,R8 TO R1.
                JSR  MUL_FLOAT_2    F8DA   20 D9 EC      R0 = X^2 * (K(LOOP_COUNTER) + R0).
                JSR  NORMALISE_1    F8DD   20 D0 E2
                INC  LOOP_COUNTER   F8E0   E6 90
                LDA  LOOP_COUNTER   F8E2   A5 90
                CMP  #$04           F8E4   C9 04         REPEAT UNTIL LOOP_COUNTER=4
                BNE  L01            F8E6   D0 DE
                LDX  #$14           F8E8   A2 14
                JSR  LOAD_TABLE_1   F8EA   20 C7 F1      LOAD 2.0 TO R1
                JSR  ADD_FLOAT_2    F8ED   20 C5 EC      R0 = 2.0 + R0.
                JSR  NORMALISE_1    F8F0   20 D0 E2
                LDX  #$78           F8F3   A2 78
                LDY  #$14           F8F5   A0 14
                JSR  MOV_FP_REG     F8F7   20 39 EA      MOVE X,R10 TO R1.
                JSR  MUL_FLOAT_2    F8FA   20 D9 EC      R0 = X * R0.
                JSR  NORMALISE_1    F8FD   20 D0 E2
                RTS                 F900   60
%
%-------------------------------------------------------------------------------
% CONVERT Y TO X FOR NATURAL LOGARITHM FUNCTION BY MACLAURIN SERIES.
%      X = (Y - 1) / (Y + 1).
%-------------------------------------------------------------------------------
PRE_LIN_2
                LDX  #$10           F901   A2 10
                LDY  #$7C           F903   A0 7C
                JSR  MOV_FP_REG     F905   20 39 EA      MOVE ARGUMENT,R0 TO R11.
                LDA  #$80           F908   A9 80
                STA  P6             F90A   85 80
                LDA  #$FE           F90C   A9 FE
                STA  P6+1           F90E   85 81         SET TABLE HEAD ADDRESS $FE80 TO P6.
                LDA  #$05           F910   A9 05         TABLE INDEX = 5.
                LDX  #$10           F912   A2 10
                JSR  LOAD_TABLE_1   F914   20 C7 F1      LOAD 1.0 TO R1.
                LDX  #$7C           F917   A2 7C
                LDY  #$14           F919   A0 14
                JSR  MOV_FP_REG     F91B   20 39 EA      MOVE Y,R11 TO R1.
                JSR  SUB_FLOAT_2    F91E   20 CF EC      R0 = Y - 1.
                JSR  NORMALISE_1    F921   20 D0 E2
                LDX  #$10           F924   A2 10
                LDY  #$A0           F926   A0 A0
                JSR  MOV_FP_REG     F928   20 39 EA      MOVE R0 TO R12.
                LDA  #$05           F92B   A9 05         TABLE INDEX = 5.
                LDX  #$10           F92D   A2 10
```

```
              JSR  LOAD_TABLE_1    F92F   20 C7 F1      LOAD 1.0 TO R0.
              LDX  #$7C           F932   A2 7C
              LDY  #$14           F934   A0 14
              JSR  MOV_FP_REG     F936   20 39 EA      MOVE Y,R11 TO R1.
              JSR  ADD_FLOAT_2    F939   20 C5 EC      R0 = Y + 1.
              JSR  NORMALISE_1    F93C   20 D0 E2
              LDX  #$A0           F93F   A2 A0
              LDY  #$14           F941   A0 14
              JSR  MOV_FP_REG     F943   20 39 EA      MOVE R12 TO R1.
              JSR  DIV_FLOAT_1    F946   20 E3 EC      R0 = (Y - 1) / (Y + 1)
              JSR  NORMALISE_1    F949   20 D0 E2
              RTS                 F94C   60
%
%-------------------------------------------------------------------------------
% DECOMPOSITION PROCESS OF NATURAL LOGARITHM FUNCTION
%-------------------------------------------------------------------------------
              .ORIGIN  $F950
%
PRE_LN_1      LDX  #$10           F950   A2 10
              LDY  #$60           F952   A0 60
              JSR  MOV_FP_REG     F952   20 39 EA      MOVE ARGUMENT Y,R0 TO R4.
              LDA  #$80           F957   A9 80
              STA  P6             F959   85 80
              LDA  #$FE           F95B   A9 FE
              STA  P6+1           F95D   85 81         SET TABLE HEAD ADDRESS $FE80 TO P6.
              LDA  R0_2           F95F   A5 12
              BNE  L01            F961   D0 03
              JMP  ERROR_PROC     F963   4C 63 F9      IF ARGUMENT=0, GO TO ERROR PROCESS.
L01           LDA  R0_3           F966   A5 13
              BPL  L07            F968   10 03
              JMP  ERROR_PROC     F96A   4C 6A F9      IF ARGUMENT<0, GO TO ERROR PROCESS.
L07           SED                 F96D   F8           SET DECIMAL MODE.
              SEC                 F96E   38
              SBC  #$01           F96F   E9 01         EXPONENT - 1
              JMP  LN_PATCH_1     F971   4C 5B FA      SET THE EXPONENT PART TO MANTISSA OF R0.
%
L06           LDA  #$00           F974   A9 00
              STA  R0_0           F976   85 10
              STA  R0_1           F978   85 11
              LDA  #$06           F97A   A9 06
              LDX  #$14           F97C   A2 14
              JSR  LOAD_TABLE_1   F97E   20 C7 F1      LOAD LN(10) TO R1.
              JSR  MUL_FLOAT_2    F981   20 D9 EC      R0 = R1 * R0.
              JSR  NORMALISE_1    F984   20 D0 E2
              LDX  #$10           F987   A2 10
              LDY  #$6C           F989   A0 6C
              JSR  MOV_FP_REG     F98B   20 39 EA      MOVE R0 TO R7.
              LDA  R4_2           F98E   A5 62         UPPER 2 DIGIT OF MANTISSA OF ARGUMENT.
              AND  #$F0           F990   29 F0
              CMP  #$10           F992   C9 10
              BNE  L02            F994   D0 15
              LDA  #$07           F996   A9 07
              LDX  #$68           F998   A2 68         WHEN UPPER DIGIT OF MANTISSA = 1,
              JSR  LOAD_TABLE_1   F99A   20 C7 F1      MOVE K(7) TO R6.
              LDX  #$60           F99D   A2 60
              LDY  #$64           F99F   A0 64
              JSR  MOV_FP_REG     F9A1   20 39 EA      MOVE ARGUMENT Y,R4 TO R5.
              LDA  #$01           F9A4   A9 01
              STA  R5_3           F9A6   85 67         SET 1 TO EXPONENT PART OF Y.
              JMP  L03            F9A8   4C 33 FA
%
L02           LDA  R4_2           F9AB   A5 62
              AND  #$F0           F9AB   29 F0
              CMP  #$40           F9AF   C9 40         IF UPPER DIGIT OF MATISSA >= 4, BRANCH.
              BPL  L04            F9B1   10 29
              LDA  #$08           F9B3   A9 08         WHEN UPPER DIGIT OF MATISSA = 2 OR 3,
              LDX  #$68           F9B5   A2 68
              JSR  LOAD_TABLE_1   F9B7   20 C7 F1      LOAD K(8) TO R6.
              LDX  #$60           F9BA   A2 60
              LDY  #$14           F9BC   A0 14
              JSR  MOV_FP_REG     F9BE   20 39 EA      MOVE ARGUMENT Y,R4 TO R1.
              LDA  #$01           F9C1   A9 01
```

```
                STA  R1_3            F9C3   85 17        SET 1 TO EXPONENT PART OF R1.
                LDA  #$0B           F9C5   A9 0B
                LDX  #$10           F9C7   A2 10
                JSR  LOAD_TABLE_1   F9C9   20 C7 F1      LOAD K(11) TO R0.
                JSR  MUL_FLOAT_2    F9CC   20 D9 EC      R0 = R1 * R0.
                JSR  NORMALISE_1    F9CF   20 D0 E2
                LDX  #$10           F9D2   A2 10
                LDY  #$64           F9D4   A0 64
                JSR  MOV_FP_REG     F9D6   20 39 EA      MOVE R0 TO R5.
                CLC                 F9D9   18
                BCC  L03            F9DA   90 57
%
L04             LDA  R4_2           F9DC   A5 62
                AND  #$F0           F9DE   29 F0
                CMP  #$80           F9E0   C9 80         IF UPPER DIGIT OF MANTISSA >= 8, BRANCH.
                BPL  L05            F9E2   10 29
                LDA  #$09           F9E4   A9 09         WHEN UPPER DIGIT OF MANTISSA = 4,5,6,7,
                LDX  #$68           F9E6   A2 68
                JSR  LOAD_TABLE_1   F9E8   20 C7 F1      LOAD K(9) TO R6.
                LDX  #$60           F9EB   A2 60
                LDY  #$14           F9ED   A0 14
                JSR  MOV_FP_REG     F9EF   20 39 EA      MOVE ARGUMENT Y,R4 TO R1.
                LDA  #$01           F9F2   A9 01
                STA  R1_3           F9F4   85 17         SET 1 TO EXPONENT PART OF R1.
                LDA  #$0C           F9F6   A9 0C
                LDX  #$10           F9F8   A2 10
                JSR  LOAD_TABLE_1   F9FA   20 C7 F1      LOAD K(12) TO R0.
                JSR  MUL_FLOAT_2    F9FD   20 D9 EC      R0 = R1 * R0.
                JSR  NORMALISE_1    FA00   20 D0 E2
                LDX  #$10           FA03   A2 10
                LDY  #$64           FA05   A0 64
                JSR  MOV_FP_REG     FA07   20 39 EA      MOVE R0 TO R5.
                CLC                 FA0A   18
                BCC L03             FA0B   90 26
%
L05             LDA  #$0A           FA0D   A9 0A         WHEN UPPER DIGIT OF MANTISSA = 8 OR 9.
                LDX  #$68           FA0F   A2 68
                JSR  LOAD_TABLE_1   FA11   20 C7 F1      LOAD K(10) TO R6.
                LDX  #$60           FA14   A2 60
                LDY  #$14           FA16   A0 14
                JSR  MOV_FP_REG     FA18   20 39 EA      MOVE ARGUMENT Y,R4 TO R1.
                LDA  #$01           FA1B   A9 01
                STA  R1_3           FA1D   85 17         SET 1 TO EXPONENT PART OF R1.
                LDA  #$0D           FA1F   A9 0D
                LDX  #$10           FA21   A2 10
                JSR  LOAD_TABLE_1   FA23   20 C7 F1      LOAD K(13) TO R0.
                JSR  MUL_FLOAT_2    FA26   20 D9 EC      R0 = R1 * R0.
                JSR  NORMALISE_1    FA29   20 D0 E2
                LDX  #$10           FA2C   A2 10
                LDY  #$64           FA2E   A0 64
                JSR  MOV_FP_REG     FA30   20 39 EA      MOVE R0 TO R5.
L03             LDX  #$64           FA33   A2 64
                LDY  #$10           FA35   A0 10
                JSR  MOV_FP_REG     FA37   20 39 EA      MOVE R5 TO R0.
                JSR  PRE_LN_2       FA3A   20 01 F9      CONVERT Y TO X.
                JSR  MACLN_3        FA3D   20 8B F8      MACLAURIN SERIES OF NATURAL LOG.
                LDX  #$68           FA40   A2 68
                LDY  #$14           FA42   A0 14
                JSR  MOV_FP_REG     FA44   20 39 EA      MOVE R6 TO R1.
                JSR  ADD_FLOAT_2    FA47   20 C5 EC      R0 = R1 + R0.
                JSR  NORMALISE_1    FA4A   20 D0 E2
                LDX  #$6C           FA4D   A2 6C
                LDY  #$14           FA4F   A0 14
                JSR  MOV_FP_REG     FA51   20 39 EA      MOVE R7 TO R1.
                JSR  ADD_FLOAT_2    FA54   20 C5 EC      R0 = R1 + R0.
                JSR  NORMALISE_1    FA57   20 D0 E2
                RTS                 FA5A   60
%
%-------------------------------------------------------------------------------
% PATCH OF NATURAL LOGARITHM FUNCTION.
%-------------------------------------------------------------------------------
LN_PATCH_1      STA  TEMP_13        FA5B   85 56
```

```
             CMP  #$40           FA5D   C9 40
             BCC  L08            FA5F   90 37        IF EXPONENT PART IS PLUS, BRANCH
             CMP  #$99           FA61   C9 99
             BNE  L09            FA63   D0 11
             LDA  #$00           FA65   A9 00        IF EXPONENT IS 99(BCD), SET -1 TO R0.
             STA  R0_0           FA67   85 10
             STA  R0_1           FA69   85 11
             LDA  #$10           FA6B   A9 10
             STA  R0_2           FA6D   85 12
             LDA  #$81           FA6F   A9 81
             STA  R0_3           FA71   85 13
             JMP  L06            FA73   4C 74 F9
L09          SED                 FA76   F8
             SEC                 FA77   38
             LDA  #$80           FA78   A9 80        CONVERT TO ABSOLUTE VALUE BY(80 - EXPONENT)
             SBC  TEMP_13        FA7A   E5 56
             STA  R0_2           FA7C   85 12
             CMP  #$10           FA7E   C9 10        IS EXPONENT PART IS 2 DIGIT?
             BCS  L10            FA80   B0 0F        SET ABSOLUTE EXPONENT RO TO MANTISSA UPPER.
             ASL  R0_2           FA82   06 12        IF EXPONENT PART IS 1 DIGIT, SHIFT 4 BIT
             ASL  R0_2           FA84   06 12
             ASL  R0_2           FA86   06 12
             ASL  R0_2           FA88   06 12
             LDX  #$81           FA8A   A2 81        MANTISSA IS MINUS, EXPONENT IS 1 DIGIT.
             STX  R0_3           FA8C   86 13
             JMP  L06            FA8E   4C 74 F9
%
L10          LDX  #$82           FA91   A2 82        MANTISSA IS MINUS, EXPONENT IS 2 DIGITS.
             STX  R0_3           FA93   86 13
             JMP  L06            FA95   4C 74 F9
%
L08          STA  R0_2           FA98   85 12        WHEN EXPONENT PART IS PLUS,
             CMP  #$10           FA9A   C9 10
             BCS  L11            FA9C   B0 0F
             ASL  R0_2           FA9E   06 12
             ASL  R0_2           FAA0   06 12
             ASL  R0_2           FAA2   06 12
             ASL  R0_2           FAA4   06 12
             LDX  #$01           FAA6   A2 01        MANTISSA IS PLUS, EXPONENT IS 1 DIGIT.
             STX  R0_3           FAA8   86 13
             JMP  L06            FAAA   4C 74 F9
%
L11          LDX  #$02           FAAD   A2 02        MANTISSA IS PLUS, EXPONENT IS 2 DIGITS.
             STX  R0_3           FAAF   86 13
             JMP  L06            FAB1   4C 74 F9
%
%-------------------------------------------------------------------------------
% EXPONENTIATION FUNCTION.       R0 = R0 ^ R1 = EXP(R1 * LN(R0))
%-------------------------------------------------------------------------------
FUNCTION_PWR1JSR  FUNC_1         FAB4   20 96 F3
             JSR  PWR_1          FAB7   20 C0 FA
             JSR  FUNC_2         FABA   20 8E F3
             JMP  FUNC_RETURN    FABD   4C 57 F4
%
%-------------------------------------------------------------------------------
PWR_1        JSR  LN_1           FAC0   20 81 F8     R0 = LN(R0).
             LDX  #$74           FAC3   A2 74
             LDY  #$14           FAC5   A0 14
             JSR  MOV_FP_REG     FAC7   20 39 EA     MOVE R9 TO R1.
             JSR  MUL_FLOAT_1    FACA   20 D9 EC     R0 = R1 * R0.
             JSR  NORMALISE_1    FACD   20 D0 E2
             JSR  EXP_1          FAD0   20 70 F7     R0 = EXP(R1 * LN(R0)).
             RTS                 FAD3   60
%
%-------------------------------------------------------------------------------
% HYPERBOLIC SINE FUNCTION.      SINH(X) = 0.5 * (EXP(X) - EXP(-X)).
%-------------------------------------------------------------------------------
FUNCTION_HSINJSR  FUNC_1         FAD4   20 96 F3
             JSR  HSIN_1         FAD7   20 E0 FA
             JSR  FUNC_2         FADA   20 8E F3
             JMP  FUNC_RETRUN    FADD   4C 57 F4
%
```

```
%-------------------------------------------------------------------------------------
HSIN_1        LDX  #$10           FAE0   A2 10
              LDY  #$6C           FAE2   A0 6C
              JSR  MOV_FP_REG     FAE4   20 39 EA      MOVE ARGUMENT X,R0 TO R7.
              JSR  EXP_1          FAE7   20 70 F7      R0 = EXP(X).
              LDX  #$10           FAEA   A2 10
              LDY  #$68           FAEC   A0 68
              JSR  MOV_FP_REG     FAEE   20 39 EA      MOVE R0 TO R6.
              LDX  #$6C           FAF1   A2 6C
              LDY  #$10           FAF3   A0 10
              JSR  MOV_FP_REG     FAF5   20 39 EA      MOVE ARGUMENT X,R7 TO R0.
              LDA  R0_3           FAF8   A5 13
              EOR  #$80           FAFA   49 80         INVERT SIGN OF MANTISSA.
              STA  R0_3           FAFC   85 13
              JSR  EXP_1          FAFE   20 70 F7      R0 = EXP(-X).
              LDX  #$68           FB01   A2 68
              LDY  #$14           FB03   A0 14
              JSR  MOV_FP_REG     FB05   20 39 EA      MOVE R6 TO R1.
              JSR  SUB_FLOAT_2    FB08   20 CF EC      R0 = EXP(X) - EXP(-X).
              JSR  NORMALISE_1    FB0B   20 D0 E2
              LDA  #$00           FB0E   A9 00         SET 0.5 TO R1
              STA  R1_0           FB10   85 14
              STA  R1_1           FB12   85 15
              STA  R1_3           FB14   85 17
              LDA  #$50           FB16   A9 50
              STA  R1_2           FB18   85 16
              JSR  MUL_FLOAT_2    FB1A   20 D9 EC      R0 = 0.5 * (EXP(X) - EXP(-X)).
              JSR  NORMALISE_1    FB1D   20 D0 E2
              RTS                 FB20   60
%
%-------------------------------------------------------------------------------------
FUNCTION_HCOS JSR  FUNC_1         FB21   20 96 F3
              JSR  HCOS_1         FB24   20 2D FB
              JSR  FUNC_2         FB27   20 8E F3
              JMP  FUNC_RETURN    FB2A   4C 57 F4
%
%-------------------------------------------------------------------------------------
% HYPERBOLIC COSINE FUNCTION.               COSH(X) = 0.5 * (EXP(X) + EXP(-X)).
%-------------------------------------------------------------------------------------
HCOS_1        LDX  #$10           FB2D   A2 10
              LDY  #$6C           FB2F   A0 6C
              JSR  MOV_FP_REG     FB31   20 39 EA      MOVE ARGUMENT X,R0 TO R7.
              JSR  EXP_1          FB34   20 70 F7      R0 = EXP(X).
              LDX  #$10           FB37   A2 10
              LDY  #$68           FB39   A0 68
              JSR  MOV_FP_REG     FB3B   20 39 EA      MOVE R0 TO R6.
              LDX  #$6C           FB3E   A2 6C
              LDY  #$10           FB40   A0 10
              JSR  MOV_FP_REG     FB42   20 39 EA      MOVE ARGUMENT X,R7 TO R0.
              LDA  R0_3           FB45   A5 13
              EOR  #$80           FB47   49 80         INVERT SIGN OF MANTISSA.
              STA  R0_3           FB49   85 13
              JSR  EXP_1          FB4B   20 70 F7      R0 = EXP(-X).
              LDX  #$68           FB4E   A2 68
              LDY  #$14           FB50   A0 14
              JSR  MOV_FP_REG     FB52   20 39 EA      MOVE R6 TO R1.
              JSR  ADD_FLOAT_2    FB55   20 C5 EC      R0 = EXP(X) + EXP(-X).
              JSR  NORMALISE_1    FB58   20 D0 E2
              LDA  #$00           FB5B   A9 00         SET 0.5 TO R1
              STA  R1_0           FB5D   85 14
              STA  R1_1           FB5F   85 15
              STA  R1_3           FB61   85 17
              LDA  #$50           FB63   A9 50
              STA  R1_2           FB65   85 16
              JSR  MUL_FLOAT_2    FB67   20 D9 EC      R0 = 0.5 * (EXP(X) + EXP(-X)).
              JSR  NORMALISE_1    FB6A   20 D0 E2
              RTS                 FB6D   60
%
%-------------------------------------------------------------------------------------
% HYPABOLIC TANGENT FUNCTION.               TANH(X) = SINH(X) / COSH(X).
%-------------------------------------------------------------------------------------
FUNCTION_HTAN JSR  FUNC_1         FB6E   20 96 F3
```

```
             JSR  HTAN_1          FB71   20 7A FB
             JSR  FUNC_2          FB74   20 8E F3
             JMP  FUNC_RETURN     FB77   4C 57 F4
%
%-------------------------------------------------------------------------------------
HTAN_1       LDX  #$10            FB7A   A2 10
             LDY  #$7C            FB7C   A0 7C
             JSR  MOV_FP_REG      FB7E   20 39 EA      MOVE ARGUMENT X,R0 TO R11.
             JSR  HSIN_1          FB81   20 E0 FA      R0 = SINH(X).
             LDX  #$10            FB84   A2 10
             LDY  #$64            FB86   A0 64
             JSR  MOV_FP_REG      FB88   20 39 EA      MOVE R0 TO R5.
             LDX  #$7C            FB8B   A2 7C
             LDY  #$10            FB8D   A0 10
             JSR  MOV_FP_REG      FB8F   20 39 EA      MOVE ARGUMENT X,R11 TO R0.
             JSR  HCOS_1          FB92   20 2D FB      R0 = COSH(X).
             LDX  #$64            FB95   A2 64
             LDY  #$14            FB97   A0 14
             JSR  MOV_FP_REG      FB99   20 39 EA      MOVE R5 TO R1.
             JSR  DIV_FLOAT_1     FB9C   20 E3 EC      R0 = SINH(X) / COSH(X).
             JSR  NORMALISE_1     FB9F   20 D0 E2
             RTS                  FBA2   60
%
%-------------------------------------------------------------------------------------
% PATCH FOR EXPONENTIAL FUNCTION.
%      SWITCH THE TABLE BY THE SIGN OF THE ARGUMENT.
%-------------------------------------------------------------------------------------
EXP_PATCH_1  LDA  R8_3            FBA3   A5 73
             BMI  L07             FBA5   30 04         IF MANTISSA SIGN IS MINUS, BRANCH
             LDA  #$C8            FBA7   A9 C8         IF MANTISSA SIGN IS PLUS,
             BNE  L08             FBA9   D0 02          SET $FFC8 TO TABLE HEAD ADDRESS.
L07          LDA  #$40            FBAB   A9 40         IF MANTISSA SIGN IS MINUS,
L08          STA  P6              FBAD   85 80          SET $FF40 TO TABLE HEAD ADDRESS.
             JMP  EXP_PATCH_1R    FBAF   4C 2F F8
%
%-------------------------------------------------------------------------------------
% PATCH FOR EXPONENTIAL FUNCTION.
%      OVERFLOW HANDLING WHEN ARGUMENTS ARE POSITIVE AND NEGATIVE.
%-------------------------------------------------------------------------------------
EXP_PATCH_2  BPL  L01              FBB2   10 03         POSITIVE OVERFLOW IF X > K(0).
             JMP  ERROR_PROC      FBB4   4C B4 FB
L01          LDX  #$70            FBB7   A2 70
             LDY  #$10            FBB9   A0 10
             JSR  MOV_FP_REG      FBBB   20 39 EA      MOVE X,R8 TO R0.
             LDA  #$40            FBBE   A9 40
             STA  P6              FBC0   85 80         SET TABLE HEAD ADDRESS $FF40 TO P6.
             LDA  #$00            FBC2   A9 00         INDEX = 0.
             LDX  #$14            FBC4   A2 14
             JSR  LOAD_TABLE_1    FBC6   20 C7 F1      LOAD NEGATIVE OVERFLOW JUDGEMENT VALUE C1.
             JSR  SUB_FLOAT_2     FBC9   20 CF EC      C1 - X
             LDA  R0_3            FBCC   A5 13
             BMI  L09             FBCE   30 0B
             LDA  #$00            FBD0   A9 00         IF X < C1, RESULT IS ZERO.
             STA  R0_0            FBD2   85 10
             STA  R0_1            FBD4   85 11
             STA  R0_2            FBD6   85 12
             STA  R0_3            FBD8   85 13
             RTS                  FBDA   60
%
L09          JMP  EXP_PATCH_2R    FBDB   4C F1 F7
%
%-------------------------------------------------------------------------------------
% PATCH FOR SQUARE ROOT FUNCTION.
%      SAVE R1, AND MAKE IT TO EXECUTE INDIVIDUAL.
%-------------------------------------------------------------------------------------
FUNCTION_SQRTJSR  FUNC_1          FBDE   20 96 F3
             JSR  SQRT_1          FBE1   20 FF F0
             JSR  FUNC_2          FBE4   20 8E F3
             JMP  FUNC_RETURN     FBE7   4C 57 F4
%
%-------------------------------------------------------------------------------------
% ABSOLUTE VALUE FUNCTION.
```

```
%-------------------------------------------------------------------------------------------
FUNCTION_ABS JSR  FUNC_1          FBEA   20 96 F3
             JSR  ABS_1           FBED   20 F6 FB
             JSR  FUNC_2          FBF0   20 8E F3
             JMP  FUNC_RETURN     FBF3   4C 57 F4
%
%-------------------------------------------------------------------------------------------
ABS_1        LDA  R0_3            FBF6   A5 13
             AND  #$7F            FBF8   29 7F
             STA  R0_3            FBFA   85 13
             RTS                  FBFC   60
%
%-------------------------------------------------------------------------------------------
% INTEGER VALUE FUNCTION
%-------------------------------------------------------------------------------------------
FUNCTION_INT JSR  FUNC_1          FBFD   20 96 F3
             JSR  INT_1           FC00   20 09 FC
             JSR  FUNC_2          FC03   20 8E F3
             JMP  FUNC_RETURN     FC06   4C 57 F4
%
%-------------------------------------------------------------------------------------------
INT_1        LDA  R0_3            FC09   A5 13
             AND  #$7F            FC0B   29 7F
             BEQ  L03             FC0D   F0 04        IF EXPONENT PART = 0, BRANCH.
             CMP  #$40            FC0F   C9 40
             BCC  L01             FC11   90 0B        IF EXPONENT PART <= 39, BRANCH.
L03          LDA  #$00            FC13   A9 00
             STA  R0_2            FC15   85 12        SET 0 TO R0.
             STA  R0_1            FC17   85 11
             STA  R0_0            FC19   85 10
             NOP                  FC1B   EA
             NOP                  FC1C   EA
             RTS                  FC1D   60
%
L01          CMP  #$01            FC1E   C9 01        WHEN EXPONENT PART = 1,
             BNE  L02             FC20   D0 0D         SET 0 TO LOWER 5 DIGIT.
             LDA  R0_2            FC22   A5 12
             AND  #$F0            FC24   29 F0
             STA  R0_2            FC26   85 12
L05          LDA  #$00            FC28   A9 00
             STA  R0_1            FC2A   85 11
             STA  R0_0            FC2C   85 10
             RTS                  FC2E   60
%
L02          CMP  #$02            FC2F   C9 02        WHEN EXPONENT PART = 2,
             BNE  L04             FC31   D0 03         SET 0 TO LOWER 4 DIGIT.
             CLC                  FC33   18
             BCC  L05             FC34   90 F2
%
L04          CMP  #$03            FC36   C9 03        WHEN EXPONENT PART = 3,
             BNE  L06             FC38   D0 0B         SET 0 TO LOWER 3 DIGIT.
             LDA  R0_1            FC3A   A5 11
             AND  #$F0            FC3C   29 F0
             STA  R0_1            FC3E   85 11
L08          LDA  #$00            FC40   A9 00
             STA  R0_0            FC42   85 10
             RTS                  FC44   60
%
L06          CMP  #$04            FC45   C9 04        WHEN EXPONENT PART = 4,
             BNE  L07             FC47   D0 03         SET 0 TO LOWER 2 DIGIT.
             CLC                  FC49   18
             BCC  L08             FC4A   90 F4
%
L07          CMP  #$05            FC4C   C9 05        WHEN EXPONENT PART = 5,
             BNE  L09             FC4E   D0 06         SET 0 TO LOWER 1 DIGIT.
             LDA  R0_0            FC50   A5 10         OTHERWISE, NOTHING WILL BE DONE.
             AND  #$F0            FC52   29 F0
             STA  R0_0            FC54   85 10
L09          RTS                  FC56   60
%
%-------------------------------------------------------------------------------------------
% PATCH TO PREVENT OUTPUT SPACE CODE IN FORMAT ;;     (V.1.3.1  AUG/06/2022)
```

```
%-------------------------------------------------------------------------------------------
ENDLINE_PR_5A JSR  DISP_BUFFER_2  FC57   20 E6 ED     DISPLAY BUFFER WITH NO SPACE CODE
              JMP  ENDLINE_PROC_5 FC5A   4C AB F0
%
%-------------------------------------------------------------------------------------------
%---------------------------------  TABLES   -----------------------------------------
%-------------------------------------------------------------------------------------------
%
%----------------  MACLAURIN SERIES COEFFICIENT OF LN FUNCTION   ------------------
%
              .ORIGIN  $FE80
%                                                          INDEX    DECIMAL VALUE
                                  FE80   22 22 22 00        0      2.22222E-1  (2/9)
                                  FE84   14 57 28 00        1      2.85714E-1  (2/7)
                                  FE88   00 00 40 00        2      4.00000E-1  (2/5)
                                  FE8C   67 66 66 00        3      6.66667E-1  (2/3)
                                  FE90   00 00 20 01        4      2.00000E0   (2/1)
                                  FE94   00 00 10 01        5      1.00000E0   (1)
%
%--------------------  DECOMPOSITION CONSTANT OF LN FUNCTION   --------------------
%
              .ORIGIN  $FE98
%                                                          INDEX    DECIMAL VALUE
                                  FE98   59 02 23 01        6      2.30259E0  (LN(10))
                                  FE9C   00 00 00 00        7      0          (LN(1))
                                  FEA0   47 31 69 00        8      6.93147E-1 (LN(2))
                                  FEA4   29 86 13 01        9      1.38629E0   (LN(4))
                                  FEA8   44 79 20 01        10     2.07944E0   (LN(8))
                                  FEAC   00 00 50 00        11     0.5        (1/2)
                                  EFB0   00 00 25 00        12     0.25        (1/4)
                                  FEB4   00 50 12 00        13     0.125       (1/8)
%
%-----------  BISECTION METHOD COMPARISON VALUES OF ATAN FUNCTION   ----------------
%
              .ORIGIN  $FF00
%
%                                                          INDEX    DECIMAL VALUE
                                  FF00   98 53 78 00        0      7.85398E-1 (PI/4)
                                  FF04   99 26 39 00        1      3.92699E-1 (PI/8)
                                  FF08   50 63 19 00        2      1.96350E-1 (PI/16)
                                  FF0C   48 17 98 79        3      9.81748E-2 (PI/32)
                                  FF10   74 08 49 79        4      4.90874E-2 (PI/64)
                                  FF14   37 54 24 79        5      2.45437E-2 (PI/128)
                                  FF18   18 27 12 79        6      1.22718E-2 (PI/256)
                                  FF1C   92 35 61 78        7      6.13592E-3 (PI/512)
                                  FF20   96 67 30 78        8      3.06796E-3 (PI/1024)
                                  FF24   98 33 15 78        9      1.53398E-3 (PI/2^11)
                                  FF28   90 69 76 77        10     7.66990E-4 (PI/2^12)
                                  FF2C   95 34 38 77        11     3.83495E-4 (PI/2^13)
                                  FF30   48 17 19 77        12     1.91748E-4 (PI/2^14)
                                  FF34   38 87 95 76        13     9.58738E-5 (PI/2^15)
                                  FF38   69 93 47 76        14     4.79369E-5 (PI/2^16)
                                  FF3C   84 96 23 76        15     2.39684E-5 (PI/2^17)
%
%----------------  ARGUMENT DECOMPOSITION CONSTANT OF EXP FUNCTION   ------------------
%
              .ORIGIN  $FF40
%                                                          INDEX    DECIMAL VALUE
                                  FF40   34 10 92 82        0      -92.1034 (EXP=-9.99999E39)
                                  FF44   81 03 16 53        1      1.60381E-28  (EXP(-64))
                                  FF48   42 66 12 67        2      1.26642E-14  (EXP(-32))
                                  FF4C   35 25 11 74        3      1.12535E-7   (EXP(-16))
                                  FF50   63 54 33 77        4      3.35463E-4   (EXP(-8))
                                  FF54   56 31 18 79        5      1.83156E-2   (EXP(-4))
                                  FF58   35 53 13 00        6      1.35335E-1   (EXP(-2))
                                  FF5C   79 78 36 00        7      3.67879E-1   (EXP(-1))
%
%-----------  BISECTION METHOD COMPARISON VALUES OF ATAN FUNCTION   ----------------
%
              .ORIGIN  $FF78
%
                                  FF78   80 70 15 01        30     1.57080    (PI/2)
```

```
                                FF7C   00 00 10 01         31     1.00000
%
%----------------  MACLAURIN SERIES COEFFICIENT OF SIN FUNCTION   ------------------
%
             .ORIGIN  $FF80
%                                                       INDEX    DECIMAL VALUE
                                FF80   73 55 27 75         0     2.75573E-6
                                FF84   13 84 19 77         1     1.98413E-4
                                FF88   33 33 83 78         2     8.33333E-3
                                FF8C   67 66 16 00         3     1.66667E-1
                                FF90   00 00 10 01         4     1E0
                                FF94   19 83 62 01         5     6.28319 (2*PI)
                                FF98   39 12 47 01         6     4.71239 (1.5*PI)
                                FF9C   59 41 31 01         7     3.14159 (PI)
                                FFA0   80 70 15 01         8     1.57080 (0.5*PI)
%
%----------------  MACLAURIN SERIES COEFFICIENT OF EXP FUNCTION   ------------------
%
             .ORIGIN  $FFA4
%                                                       INDEX    DECIMAL VALUE
                                FFA4   73 55 27 75         0     2.75573E-6 (1/9!)
                                FFA8   16 80 24 76         1     2.48016E-5 (1/8!)
                                FFAC   13 84 19 77         2     1.98413E-4 (1/7!)
                                FFB0   89 88 13 78         3     1.38889E-3 (1/6!)
                                FFB4   33 33 83 78         4     8.33333E-3 (1/5!)
                                FFB8   67 66 41 79         5     4.16667E-2 (1/4!)
                                FFBC   67 66 16 00         6     1.66667E-1 (1/3!)
                                FFC0   00 00 50 00         7     0.5        (1/2!)
                                FFC4   00 00 10 01         8     1.0        (1/1!)
%
%----------------------  DECOMPOSITION CONSTANT OF EXP FUNCTION   ----------------------
%
             .ORIGIN  $FFC8
%                                                       INDEX    DECIMAL VALUE
                                FFC8   84 32 89 02         0     89.3284 (EXP=6.23485E38)
                                FFCC   15 35 62 28         1     6.23515E27  (EXP(64))
                                FFD0   30 96 78 14         2     7.89630E13  (EXP(32))
                                FFD4   11 86 88 07         3     8.88611E6   (EXP(16))
                                FFD8   96 80 29 04         4     2.98096E3   (EXP(8))
                                FFDC   82 59 54 02         5     5.45982E1   (EXP(4))
                                FFE0   06 89 73 01         6     7.38906E0   (EXP(2))
                                FFE4   28 18 27 01         7     2.71828E0   (EXP(1))
%
% END OF PROGRAM
%-------------------------------------------------------------------------------------
%-------------------------------------------------------------------------------------
% 2023/03/11
```