```
; ==============================================================
; An Assembly Listing of the Operating System of the ZX81 ROM
; ==============================================================
; -------------------------
; Last updated: 13-DEC-2004
; -------------------------
;
; Work in progress.
; This file will cross-assemble an original version of the "Improved"
; ZX81 ROM.  The file can be modified to change the behaviour of the ROM
; when used in emulators although there is no spare space available.
;
; The documentation is incomplete and if you can find a copy
; of "The Complete Spectrum ROM Disassembly" then many routines
; such as POINTERS and most of the mathematical routines are
; similar and often identical.
;
; I've used the labels from the above book in this file and also
; some from the more elusive Complete ZX81 ROM Disassembly
; by the same publishers, Melbourne House.


#define DEFB .BYTE      ; TASM cross-assembler definitions
#define DEFW .WORD
#define EQU  .EQU


;****************************************
;** Part 1. RESTART ROUTINES AND TABLES **
;****************************************

; -----------
; THE 'START'
; -----------
; All Z80 chips start at location zero.
; At start-up the Interrupt Mode is 0, ZX computers use Interrupt Mode 1.
; Interrupts are disabled .

;; START
L0000:  OUT     ($FD),A         ; Turn off the NMI generator if this ROM is
                                ; running in ZX81 hardware. This does
nothing
                                ; if this ROM is running within an upgraded
                                ; ZX80.
        LD      BC,$7FFF        ; Set BC to the top of possible RAM.
                                ; The higher unpopulated addresses are used
for
                                ; video generation.
        JP      L03CB           ; Jump forward to RAM-CHECK.

; -------------------
; THE 'ERROR' RESTART
; -------------------
; The error restart deals immediately with an error. ZX computers execute
the
; same code in runtime as when checking syntax. If the error occurred while
; running a program then a brief report is produced. If the error occurred
; while entering a BASIC line or in input etc., then the error marker
indicates
; the exact point at which the error lies.

;; ERROR-1
```

```
L0008:  LD      HL,($4016)      ; fetch character address from CH_ADD.
        LD      ($4018),HL      ; and set the error pointer X_PTR.
        JR      L0056           ; forward to continue at ERROR-2.

; ------------------------------
; THE 'PRINT A CHARACTER' RESTART
; ------------------------------
; This restart prints the character in the accumulator using the alternate
; register set so there is no requirement to save the main registers.
; There is sufficient room available to separate a space (zero) from other
; characters as leading spaces need not be considered with a space.

;; PRINT-A
L0010:  AND     A               ; test for zero - space.
        JP      NZ,L07F1        ; jump forward if not to PRINT-CH.

        JP      L07F5           ; jump forward to PRINT-SP.

; ---

        DEFB    $FF             ; unused location.

; --------------------------------
; THE 'COLLECT A CHARACTER' RESTART
; --------------------------------
; The character addressed by the system variable CH_ADD is fetched and if
it
; is a non-space, non-cursor character it is returned else CH_ADD is
; incremented and the new addressed character tested until it is not a
space.

;; GET-CHAR
L0018:  LD      HL,($4016)      ; set HL to character address CH_ADD.
        LD      A,(HL)          ; fetch addressed character to A.

;; TEST-SP
L001C:  AND     A               ; test for space.
        RET     NZ              ; return if not a space

        NOP                     ; else trickle through
        NOP                     ; to the next routine.

; -----------------------------------
; THE 'COLLECT NEXT CHARACTER' RESTART
; -----------------------------------
; The character address in incremented and the new addressed character is
; returned if not a space, or cursor, else the process is repeated.

;; NEXT-CHAR
L0020:  CALL    L0049           ; routine CH-ADD+1 gets next immediate
                                ; character.
        JR      L001C           ; back to TEST-SP.

; ---

        DEFB    $FF, $FF, $FF   ; unused locations.

; --------------------------------------
; THE 'FLOATING POINT CALCULATOR' RESTART
; --------------------------------------
; this restart jumps to the recursive floating-point calculator.
; the ZX81's internal, FORTH-like, stack-based language.
```

```
;
; In the five remaining bytes there is, appropriately, enough room for the
; end-calc literal - the instruction which exits the calculator.

;; FP-CALC
L0028:  JP      L199D           ; jump immediately to the CALCULATE
routine.

; ---

;; end-calc
L002B:  POP     AF              ; drop the calculator return address RE-
ENTRY
        EXX                     ; switch to the other set.

        EX      (SP),HL         ; transfer H'L' to machine stack for the
                                ; return address.
                                ; when exiting recursion then the previous
                                ; pointer is transferred to H'L'.

        EXX                     ; back to main set.
        RET                     ; return.


; ------------------------------
; THE 'MAKE BC SPACES'  RESTART
; ------------------------------
; This restart is used eight times to create, in workspace, the number of
; spaces passed in the BC register.

;; BC-SPACES
L0030:  PUSH    BC              ; push number of spaces on stack.
        LD      HL,($4014)      ; fetch edit line location from E_LINE.
        PUSH    HL              ; save this value on stack.
        JP      L1488           ; jump forward to continue at RESERVE.

; ------------------------
; THE 'INTERRUPT' RESTART
; ------------------------
;   The Mode 1 Interrupt routine is concerned solely with generating the
central
;   television picture.
;   On the ZX81 interrupts are enabled only during the interrupt routine,
;   although the interrupt
;   This Interrupt Service Routine automatically disables interrupts at the
;   outset and the last interrupt in a cascade exits before the interrupts
are
;   enabled.
;   There is no DI instruction in the ZX81 ROM.
;   An maskable interrupt is triggered when bit 6 of the Z80's Refresh
register
;   changes from set to reset.
;   The Z80 will always be executing a HALT (NEWLINE) when the interrupt
occurs.
;   A HALT instruction repeatedly executes NOPS but the seven lower bits
;   of the Refresh register are incremented each time as they are when any
;   simple instruction is executed. (The lower 7 bits are incremented twice
for
;   a prefixed instruction)
;   This is controlled by the Sinclair Computer Logic Chip - manufactured
from
;   a Ferranti Uncommitted Logic Array.
```

```
;
;   When a Mode 1 Interrupt occurs the Program Counter, which is the
address in
;   the upper echo display following the NEWLINE/HALT instruction, goes on
the
;   machine stack.  193 interrupts are required to generate the last part
of
;   the 56th border line and then the 192 lines of the central TV picture
and,
;   although each interrupt interrupts the previous one, there are no stack
;   problems as the 'return address' is discarded each time.
;
;   The scan line counter in C counts down from 8 to 1 within the
generation of
;   each text line. For the first interrupt in a cascade the initial value
of
;   C is set to 1 for the last border line.
;   Timing is of the utmost importance as the RH border, horizontal retrace
;   and LH border are mostly generated in the 58 clock cycles this routine
;   takes .

;; INTERRUPT
L0038:  DEC     C               ; (4)  decrement C - the scan line counter.
        JP      NZ,L0045        ; (10/10) JUMP forward if not zero to SCAN-
LINE

        POP     HL              ; (10) point to start of next row in
display
                                ;      file.

        DEC     B               ; (4)  decrement the row counter. (4)
        RET     Z               ; (11/5) return when picture complete to
L028B
                                ;      with interrupts disabled.

        SET     3,C             ; (8)  Load the scan line counter with
eight.
                                ;      Note. LD C,$08 is 7 clock cycles
which
                                ;      is way too fast.

; ->

;; WAIT-INT
L0041:  LD      R,A             ; (9) Load R with initial rising value $DD.

        EI                      ; (4) Enable Interrupts.  [ R is now $DE ].

        JP      (HL)            ; (4) jump to the echo display file in
upper
                                ;     memory and execute characters $00 -
$3F
                                ;     as NOP instructions.  The video
hardware
                                ;     is able to read these characters and,
                                ;     with the I register is able to
convert
                                ;     the character bitmaps in this ROM
into a
                                ;     line of bytes. Eventually the
NEWLINE/HALT
```

```
                                ;     will be encountered before R reaches
$FF.
                                ;     It is however the transition from $FF
to
                                ;     $80 that triggers the next interrupt.
                                ;     [ The Refresh register is now $DF ]

; ---

;; SCAN-LINE
L0045:  POP     DE              ; (10) discard the address after NEWLINE as
the
                                ;      same text line has to be done again
                                ;      eight times.

        RET     Z               ; (5)  Harmless Nonsensical Timing.
                                ;      (condition never met)

        JR      L0041           ; (12) back to WAIT-INT

;   Note. that a computer with less than 4K or RAM will have a collapsed
;   display file and the above mechanism deals with both types of display.
;
;   With a full display, the 32 characters in the line are treated as NOPS
;   and the Refresh register rises from $E0 to $FF and, at the next
instruction
;   - HALT, the interrupt occurs.
;   With a collapsed display and an initial NEWLINE/HALT, it is the NOPs
;   generated by the HALT that cause the Refresh value to rise from $E0 to
$FF,
;   triggering an Interrupt on the next transition.
;   This works happily for all display lines between these extremes and the
;   generation of the 32 character, 1 pixel high, line will always take 128
;   clock cycles.

; ---------------------------------
; THE 'INCREMENT CH-ADD' SUBROUTINE
; ---------------------------------
; This is the subroutine that increments the character address system
variable
; and returns if it is not the cursor character. The ZX81 has an actual
; character at the cursor position rather than a pointer system variable
; as is the case with prior and subsequent ZX computers.

;; CH-ADD+1
L0049:  LD      HL,($4016)      ; fetch character address to CH_ADD.

;; TEMP-PTR1
L004C:  INC     HL              ; address next immediate location.

;; TEMP-PTR2
L004D:  LD      ($4016),HL      ; update system variable CH_ADD.

        LD      A,(HL)          ; fetch the character.
        CP      $7F             ; compare to cursor character.
        RET     NZ              ; return if not the cursor.

        JR      L004C           ; back for next character to TEMP-PTR1.

; --------------------
; THE 'ERROR-2' BRANCH
; --------------------
```

```
; This is a continuation of the error restart.
; If the error occurred in runtime then the error stack pointer will
probably
; lead to an error report being printed unless it occurred during input.
; If the error occurred when checking syntax then the error stack pointer
; will be an editing routine and the position of the error will be shown
; when the lower screen is reprinted.

;; ERROR-2
L0056:  POP     HL              ; pop the return address which points to
the
                                ; DEFB, error code, after the RST 08.
        LD      L,(HL)          ; load L with the error code. HL is not
needed
                                ; anymore.

;; ERROR-3
L0058:  LD      (IY+$00),L      ; place error code in system variable
ERR_NR
        LD      SP,($4002)      ; set the stack pointer from ERR_SP
        CALL    L0207           ; routine SLOW/FAST selects slow mode.
        JP      L14BC           ; exit to address on stack via routine SET-
MIN.

; ---

        DEFB    $FF             ; unused.

; -----------------------------------
; THE 'NON MASKABLE INTERRUPT' ROUTINE
; -----------------------------------
;   Jim Westwood's technical dodge using Non-Maskable Interrupts solved the
;   flicker problem of the ZX80 and gave the ZX81 a multi-tasking SLOW mode
;   with a steady display.  Note that the AF' register is reserved for this
;   function and its interaction with the display routines.  When counting
;   TV lines, the NMI makes no use of the main registers.
;   The circuitry for the NMI generator is contained within the SCL
(Sinclair
;   Computer Logic) chip.
;   ( It takes 32 clock cycles while incrementing towards zero ).

;; NMI
L0066:  EX      AF,AF'          ; (4) switch in the NMI's copy of the
                                ;     accumulator.
        INC     A               ; (4) increment.
        JP      M,L006D         ; (10/10) jump, if minus, to NMI-RET as
this is
                                ;     part of a test to see if the NMI
                                ;     generation is working or an
intermediate
                                ;     value for the ascending negated blank
                                ;     line counter.

        JR      Z,L006F         ; (12) forward to NMI-CONT
                                ;      when line count has incremented to
zero.

; Note. the synchronizing NMI when A increments from zero to one takes this
; 7 clock cycle route making 39 clock cycles in all.

;; NMI-RET
```

```
L006D:  EX      AF,AF'          ; (4)  switch out the incremented line
counter
                                ;      or test result $80
        RET                     ; (10) return to User application for a
while.

; ---

;   This branch is taken when the 55 (or 31) lines have been drawn.

;; NMI-CONT
L006F:  EX      AF,AF'          ; (4) restore the main accumulator.

        PUSH    AF              ; (11) *             Save Main Registers
        PUSH    BC              ; (11) **
        PUSH    DE              ; (11) ***
        PUSH    HL              ; (11) ****

;   the next set-up procedure is only really applicable when the top set of
;   blank lines have been generated.

        LD      HL,($400C)      ; (16) fetch start of Display File from
D_FILE
                                ;      points to the HALT at beginning.
        SET     7,H             ; (8) point to upper 32K 'echo display
file'

        HALT                    ; (1) HALT synchronizes with NMI.
                                ; Used with special hardware connected to
the
                                ; Z80 HALT and WAIT lines to take 1 clock
cycle.

;
-------------------------------------------------------------------------
-
;   the NMI has been generated - start counting. The cathode ray is at the
RH
;   side of the TV.
;   First the NMI servicing, similar to CALL            =  17 clock cycles.
;   Then the time taken by the NMI for zero-to-one path =  39 cycles
;   The HALT above                                      =  01 cycles.
;   The two instructions below                          =  19 cycles.
;   The code at L0281 up to and including the CALL      =  43 cycles.
;   The Called routine at L02B5                         =  24 cycles.
;   ------------------------------------                  ---
;   Total Z80 instructions                              = 143 cycles.
;
;   Meanwhile in TV world,
;   Horizontal retrace                                  =  15 cycles.
;   Left blanking border 8 character positions          =  32 cycles
;   Generation of 75% scanline from the first NEWLINE   =  96 cycles
;   ------------------------------------                  ---
;                                                         143 cycles
;
;   Since at the time the first JP (HL) is encountered to execute the echo
;   display another 8 character positions have to be put out, then the
;   Refresh register need to hold $F8. Working back and counteracting
;   the fact that every instruction increments the Refresh register then
;   the value that is loaded into R needs to be $F5.      :-)
;
;
```

```
        OUT     ($FD),A         ; (11) Stop the NMI generator.

        JP      (IX)            ; (8) forward to L0281 (after top) or L028F

; ****************
; ** KEY TABLES **
; ****************

; ------------------------------
; THE 'UNSHIFTED' CHARACTER CODES
; ------------------------------

;; K-UNSHIFT
L007E:  DEFB    $3F             ; Z
        DEFB    $3D             ; X
        DEFB    $28             ; C
        DEFB    $3B             ; V
        DEFB    $26             ; A
        DEFB    $38             ; S
        DEFB    $29             ; D
        DEFB    $2B             ; F
        DEFB    $2C             ; G
        DEFB    $36             ; Q
        DEFB    $3C             ; W
        DEFB    $2A             ; E
        DEFB    $37             ; R
        DEFB    $39             ; T
        DEFB    $1D             ; 1
        DEFB    $1E             ; 2
        DEFB    $1F             ; 3
        DEFB    $20             ; 4
        DEFB    $21             ; 5
        DEFB    $1C             ; 0
        DEFB    $25             ; 9
        DEFB    $24             ; 8
        DEFB    $23             ; 7
        DEFB    $22             ; 6
        DEFB    $35             ; P
        DEFB    $34             ; O
        DEFB    $2E             ; I
        DEFB    $3A             ; U
        DEFB    $3E             ; Y
        DEFB    $76             ; NEWLINE
        DEFB    $31             ; L
        DEFB    $30             ; K
        DEFB    $2F             ; J
        DEFB    $2D             ; H
        DEFB    $00             ; SPACE
        DEFB    $1B             ; .
        DEFB    $32             ; M
        DEFB    $33             ; N
        DEFB    $27             ; B

; ----------------------------
; THE 'SHIFTED' CHARACTER CODES
; ----------------------------


;; K-SHIFT
L00A5:  DEFB    $0E             ; :
        DEFB    $19             ; ;
        DEFB    $0F             ; ?
```

```
        DEFB    $18             ; /
        DEFB    $E3             ; STOP
        DEFB    $E1             ; LPRINT
        DEFB    $E4             ; SLOW
        DEFB    $E5             ; FAST
        DEFB    $E2             ; LLIST
        DEFB    $C0             ; ""
        DEFB    $D9             ; OR
        DEFB    $E0             ; STEP
        DEFB    $DB             ; <=
        DEFB    $DD             ; <>
        DEFB    $75             ; EDIT
        DEFB    $DA             ; AND
        DEFB    $DE             ; THEN
        DEFB    $DF             ; TO
        DEFB    $72             ; cursor-left
        DEFB    $77             ; RUBOUT
        DEFB    $74             ; GRAPHICS
        DEFB    $73             ; cursor-right
        DEFB    $70             ; cursor-up
        DEFB    $71             ; cursor-down
        DEFB    $0B             ; "
        DEFB    $11             ; )
        DEFB    $10             ; (
        DEFB    $0D             ; $
        DEFB    $DC             ; >=
        DEFB    $79             ; FUNCTION
        DEFB    $14             ; =
        DEFB    $15             ; +
        DEFB    $16             ; -
        DEFB    $D8             ; **
        DEFB    $0C             ;  £
        DEFB    $1A             ; ,
        DEFB    $12             ; >
        DEFB    $13             ; <
        DEFB    $17             ; *

; ------------------------------
; THE 'FUNCTION' CHARACTER CODES
; ------------------------------


;; K-FUNCT
L00CC:  DEFB    $CD             ; LN
        DEFB    $CE             ; EXP
        DEFB    $C1             ; AT
        DEFB    $78             ; KL
        DEFB    $CA             ; ASN
        DEFB    $CB             ; ACS
        DEFB    $CC             ; ATN
        DEFB    $D1             ; SGN
        DEFB    $D2             ; ABS
        DEFB    $C7             ; SIN
        DEFB    $C8             ; COS
        DEFB    $C9             ; TAN
        DEFB    $CF             ; INT
        DEFB    $40             ; RND
        DEFB    $78             ; KL
        DEFB    $78             ; KL
        DEFB    $78             ; KL
        DEFB    $78             ; KL
        DEFB    $78             ; KL
```

```
        DEFB    $78             ; KL
        DEFB    $78             ; KL
        DEFB    $78             ; KL
        DEFB    $78             ; KL
        DEFB    $78             ; KL
        DEFB    $C2             ; TAB
        DEFB    $D3             ; PEEK
        DEFB    $C4             ; CODE
        DEFB    $D6             ; CHR$
        DEFB    $D5             ; STR$
        DEFB    $78             ; KL
        DEFB    $D4             ; USR
        DEFB    $C6             ; LEN
        DEFB    $C5             ; VAL
        DEFB    $D0             ; SQR
        DEFB    $78             ; KL
        DEFB    $78             ; KL
        DEFB    $42             ; PI
        DEFB    $D7             ; NOT
        DEFB    $41             ; INKEY$

; -----------------------------
; THE 'GRAPHIC' CHARACTER CODES
; -----------------------------


;; K-GRAPH
L00F3:  DEFB    $08             ; graphic
        DEFB    $0A             ; graphic
        DEFB    $09             ; graphic
        DEFB    $8A             ; graphic
        DEFB    $89             ; graphic
        DEFB    $81             ; graphic
        DEFB    $82             ; graphic
        DEFB    $07             ; graphic
        DEFB    $84             ; graphic
        DEFB    $06             ; graphic
        DEFB    $01             ; graphic
        DEFB    $02             ; graphic
        DEFB    $87             ; graphic
        DEFB    $04             ; graphic
        DEFB    $05             ; graphic
        DEFB    $77             ; RUBOUT
        DEFB    $78             ; KL
        DEFB    $85             ; graphic
        DEFB    $03             ; graphic
        DEFB    $83             ; graphic
        DEFB    $8B             ; graphic
        DEFB    $91             ; inverse )
        DEFB    $90             ; inverse (
        DEFB    $8D             ; inverse $
        DEFB    $86             ; graphic
        DEFB    $78             ; KL
        DEFB    $92             ; inverse >
        DEFB    $95             ; inverse +
        DEFB    $96             ; inverse -
        DEFB    $88             ; graphic

; ------------------
; THE 'TOKEN' TABLES
; ------------------
```

```
;; TOKENS
L0111:  DEFB    $0F+$80                         ; '?'+$80
        DEFB    $0B,$0B+$80                     ; ""
        DEFB    $26,$39+$80                     ; AT
        DEFB    $39,$26,$27+$80                 ; TAB
        DEFB    $0F+$80                         ; '?'+$80
        DEFB    $28,$34,$29,$2A+$80             ; CODE
        DEFB    $3B,$26,$31+$80                 ; VAL
        DEFB    $31,$2A,$33+$80                 ; LEN
        DEFB    $38,$2E,$33+$80                 ; SIN
        DEFB    $28,$34,$38+$80                 ; COS
        DEFB    $39,$26,$33+$80                 ; TAN
        DEFB    $26,$38,$33+$80                 ; ASN
        DEFB    $26,$28,$38+$80                 ; ACS
        DEFB    $26,$39,$33+$80                 ; ATN
        DEFB    $31,$33+$80                     ; LN
        DEFB    $2A,$3D,$35+$80                 ; EXP
        DEFB    $2E,$33,$39+$80                 ; INT
        DEFB    $38,$36,$37+$80                 ; SQR
        DEFB    $38,$2C,$33+$80                 ; SGN
        DEFB    $26,$27,$38+$80                 ; ABS
        DEFB    $35,$2A,$2A,$30+$80             ; PEEK
        DEFB    $3A,$38,$37+$80                 ; USR
        DEFB    $38,$39,$37,$0D+$80             ; STR$
        DEFB    $28,$2D,$37,$0D+$80             ; CHR$
        DEFB    $33,$34,$39+$80                 ; NOT
        DEFB    $17,$17+$80                     ; **
        DEFB    $34,$37+$80                     ; OR
        DEFB    $26,$33,$29+$80                 ; AND
        DEFB    $13,$14+$80                     ; <=
        DEFB    $12,$14+$80                     ; >=
        DEFB    $13,$12+$80                     ; <>
        DEFB    $39,$2D,$2A,$33+$80             ; THEN
        DEFB    $39,$34+$80                     ; TO
        DEFB    $38,$39,$2A,$35+$80             ; STEP
        DEFB    $31,$35,$37,$2E,$33,$39+$80     ; LPRINT
        DEFB    $31,$31,$2E,$38,$39+$80         ; LLIST
        DEFB    $38,$39,$34,$35+$80             ; STOP
        DEFB    $38,$31,$34,$3C+$80             ; SLOW
        DEFB    $2B,$26,$38,$39+$80             ; FAST
        DEFB    $33,$2A,$3C+$80                 ; NEW
        DEFB    $38,$28,$37,$34,$31,$31+$80     ; SCROLL
        DEFB    $28,$34,$33,$39+$80             ; CONT
        DEFB    $29,$2E,$32+$80                 ; DIM
        DEFB    $37,$2A,$32+$80                 ; REM
        DEFB    $2B,$34,$37+$80                 ; FOR
        DEFB    $2C,$34,$39,$34+$80             ; GOTO
        DEFB    $2C,$34,$38,$3A,$27+$80         ; GOSUB
        DEFB    $2E,$33,$35,$3A,$39+$80         ; INPUT
        DEFB    $31,$34,$26,$29+$80             ; LOAD
        DEFB    $31,$2E,$38,$39+$80             ; LIST
        DEFB    $31,$2A,$39+$80                 ; LET
        DEFB    $35,$26,$3A,$38,$2A+$80         ; PAUSE
        DEFB    $33,$2A,$3D,$39+$80             ; NEXT
        DEFB    $35,$34,$30,$2A+$80             ; POKE
        DEFB    $35,$37,$2E,$33,$39+$80         ; PRINT
        DEFB    $35,$31,$34,$39+$80             ; PLOT
        DEFB    $37,$3A,$33+$80                 ; RUN
        DEFB    $38,$26,$3B,$2A+$80             ; SAVE
        DEFB    $37,$26,$33,$29+$80             ; RAND
        DEFB    $2E,$2B+$80                     ; IF
```

```
        DEFB    $28,$31,$38+$80                 ; CLS
        DEFB    $3A,$33,$35,$31,$34,$39+$80     ; UNPLOT
        DEFB    $28,$31,$2A,$26,$37+$80         ; CLEAR
        DEFB    $37,$2A,$39,$3A,$37,$33+$80     ; RETURN
        DEFB    $28,$34,$35,$3E+$80             ; COPY
        DEFB    $37,$33,$29+$80                 ; RND
        DEFB    $2E,$33,$30,$2A,$3E,$0D+$80     ; INKEY$
        DEFB    $35,$2E+$80                     ; PI


; ------------------------------
; THE 'LOAD-SAVE UPDATE' ROUTINE
; ------------------------------
;
;

;; LOAD/SAVE
L01FC:  INC     HL              ;
        EX      DE,HL           ;
        LD      HL,($4014)      ; system variable edit line E_LINE.
        SCF                     ; set carry flag
        SBC     HL,DE           ;
        EX      DE,HL           ;
        RET     NC              ; return if more bytes to load/save.

        POP     HL              ; else drop return address

; ----------------------
; THE 'DISPLAY' ROUTINES
; ----------------------
;
;

;; SLOW/FAST
L0207:  LD      HL,$403B        ; Address the system variable CDFLAG.
        LD      A,(HL)          ; Load value to the accumulator.
        RLA                     ; rotate bit 6 to position 7.
        XOR     (HL)            ; exclusive or with original bit 7.
        RLA                     ; rotate result out to carry.
        RET     NC              ; return if both bits were the same.

;   Now test if this really is a ZX81 or a ZX80 running the upgraded ROM.
;   The standard ZX80 did not have an NMI generator.

        LD      A,$7F           ; Load accumulator with %011111111
        EX      AF,AF'          ; save in AF'

        LD      B,$11           ; A counter within which an NMI should
occur
                                ; if this is a ZX81.
        OUT     ($FE),A         ; start the NMI generator.

;  Note that if this is a ZX81 then the NMI will increment AF'.

;; LOOP-11
L0216:  DJNZ    L0216           ; self loop to give the NMI a chance to
kick in.
                                ; = 16*13 clock cycles + 8 = 216 clock
cycles.

        OUT     ($FD),A         ; Turn off the NMI generator.
        EX      AF,AF'          ; bring back the AF' value.
```

```
        RLA                     ; test bit 7.
        JR      NC,L0226        ; forward, if bit 7 is still reset, to NO-
SLOW.

;   If the AF' was incremented then the NMI generator works and SLOW mode
can
;   be set.

        SET     7,(HL)          ; Indicate SLOW mode - Compute and Display.

        PUSH    AF              ; *             Save Main Registers
        PUSH    BC              ; **
        PUSH    DE              ; ***
        PUSH    HL              ; ****

        JR      L0229           ; skip forward - to DISPLAY-1.

; ---

;; NO-SLOW
L0226:  RES     6,(HL)          ; reset bit 6 of CDFLAG.
        RET                     ; return.

; -----------------------
; THE 'MAIN DISPLAY' LOOP
; -----------------------
; This routine is executed once for every frame displayed.

;; DISPLAY-1
L0229:  LD      HL,($4034)      ; fetch two-byte system variable FRAMES.
        DEC     HL              ; decrement frames counter.

;; DISPLAY-P
L022D:  LD      A,$7F           ; prepare a mask
        AND     H               ; pick up bits 6-0 of H.
        OR      L               ; and any bits of L.
        LD      A,H             ; reload A with all bits of H for PAUSE
test.

;   Note both branches must take the same time.

        JR      NZ,L0237        ; (12/7) forward if bits 14-0 are not zero
                                ; to ANOTHER

        RLA                     ; (4) test bit 15 of FRAMES.
        JR      L0239           ; (12) forward with result to OVER-NC

; ---

;; ANOTHER
L0237:  LD      B,(HL)          ; (7) Note. Harmless Nonsensical Timing
weight.
        SCF                     ; (4) Set Carry Flag.

; Note. the branch to here takes either (12)(7)(4) cyles or (7)(4)(12)
cycles.

;; OVER-NC
L0239:  LD      H,A             ; (4)  set H to zero
        LD      ($4034),HL      ; (16) update system variable FRAMES
        RET     NC              ; (11/5) return if FRAMES is in use by
PAUSE
```

```
                                ; command.

;; DISPLAY-2
L023E:  CALL    L02BB           ; routine KEYBOARD gets the key row in H
and
                                ; the column in L. Reading the ports also
starts
                                ; the TV frame synchronization pulse.
(VSYNC)

        LD      BC,($4025)      ; fetch the last key values read from
LAST_K
        LD      ($4025),HL      ; update LAST_K with new values.

        LD      A,B             ; load A with previous column - will be $FF
if
                                ; there was no key.
        ADD     A,$02           ; adding two will set carry if no previous
key.

        SBC     HL,BC           ; subtract with the carry the two key
values.

; If the same key value has been returned twice then HL will be zero.

        LD      A,($4027)       ; fetch system variable DEBOUNCE
        OR      H               ; and OR with both bytes of the difference
        OR      L               ; setting the zero flag for the upcoming
branch.

        LD      E,B             ; transfer the column value to E
        LD      B,$0B           ; and load B with eleven

        LD      HL,$403B        ; address system variable CDFLAG
        RES     0,(HL)          ; reset the rightmost bit of CDFLAG
        JR      NZ,L0264        ; skip forward if debounce/diff >0 to NO-
KEY

        BIT     7,(HL)          ; test compute and display bit of CDFLAG
        SET     0,(HL)          ; set the rightmost bit of CDFLAG.
        RET     Z               ; return if bit 7 indicated fast mode.

        DEC     B               ; (4) decrement the counter.
        NOP                     ; (4) Timing - 4 clock cycles. ??
        SCF                     ; (4) Set Carry Flag

;; NO-KEY
L0264:  LD      HL,$4027        ; sv DEBOUNCE
        CCF                     ; Complement Carry Flag
        RL      B               ; rotate left B picking up carry
                                ;  C<-76543210<-C

;; LOOP-B
L026A:  DJNZ    L026A           ; self-loop while B>0 to LOOP-B

        LD      B,(HL)          ; fetch value of DEBOUNCE to B
        LD      A,E             ; transfer column value
        CP      $FE             ;
        SBC     A,A             ;
        LD      B,$1F           ;
        OR      (HL)            ;
        AND     B               ;
```

```
        RRA                     ;
        LD      (HL),A          ;

        OUT     ($FF),A         ; end the TV frame synchronization pulse.

        LD      HL,($400C)      ; (12) set HL to the Display File from
D_FILE
        SET     7,H             ; (8) set bit 15 to address the echo
display.

        CALL    L0292           ; (17) routine DISPLAY-3 displays the top
set
                                ; of blank lines.

; ---------------------
; THE 'VIDEO-1' ROUTINE
; ---------------------

;; R-IX-1
L0281:  LD      A,R             ; (9)  Harmless Nonsensical Timing or
something
                                ;      very clever?
        LD      BC,$1901        ; (10) 25 lines, 1 scanline in first.
        LD      A,$F5           ; (7)  This value will be loaded into R and
                                ; ensures that the cycle starts at the
right
                                ; part of the display  - after 32nd
character
                                ; position.

        CALL    L02B5           ; (17) routine DISPLAY-5 completes the
current
                                ; blank line and then generates the display
of
                                ; the live picture using INT interrupts
                                ; The final interrupt returns to the next
                                ; address.

L028B:  DEC     HL              ; point HL to the last NEWLINE/HALT.

        CALL    L0292           ; routine DISPLAY-3 displays the bottom set
of
                                ; blank lines.

; ---

;; R-IX-2
L028F:  JP      L0229           ; JUMP back to DISPLAY-1

; ---------------------------------
; THE 'DISPLAY BLANK LINES' ROUTINE
; ---------------------------------
;   This subroutine is called twice (see above) to generate first the blank
;   lines at the top of the television display and then the blank lines at
the
;   bottom of the display.

;; DISPLAY-3
L0292:  POP     IX              ; pop the return address to IX register.
                                ; will be either L0281 or L028F - see
above.
```

```
        LD      C,(IY+$28)      ; load C with value of system constant
MARGIN.
        BIT     7,(IY+$3B)      ; test CDFLAG for compute and display.
        JR      Z,L02A9         ; forward, with FAST mode, to DISPLAY-4

        LD      A,C             ; move MARGIN to A  - 31d or 55d.
        NEG                     ; Negate
        INC     A               ;
        EX      AF,AF'          ; place negative count of blank lines in A'

        OUT     ($FE),A         ; enable the NMI generator.

        POP     HL              ; ****
        POP     DE              ; ***
        POP     BC              ; **
        POP     AF              ; *             Restore Main Registers

        RET                     ; return - end of interrupt.  Return is to
                                ; user's program - BASIC or machine code.
                                ; which will be interrupted by every NMI.

; ------------------------
; THE 'FAST MODE' ROUTINES
; ------------------------

;; DISPLAY-4
L02A9:  LD      A,$FC           ; (7)  load A with first R delay value
        LD      B,$01           ; (7)  one row only.

        CALL    L02B5           ; (17) routine DISPLAY-5

        DEC     HL              ; (6)  point back to the HALT.
        EX      (SP),HL         ; (19) Harmless Nonsensical Timing if
paired.
        EX      (SP),HL         ; (19) Harmless Nonsensical Timing.
        JP      (IX)            ; (8)  to L0281 or L028F

; --------------------------
; THE 'DISPLAY-5' SUBROUTINE
; --------------------------
;   This subroutine is called from SLOW mode and FAST mode to generate the
;   central TV picture. With SLOW mode the R register is incremented, with
;   each instruction, to $F7 by the time it completes.  With fast mode, the
;   final R value will be $FF and an interrupt will occur as soon as the
;   Program Counter reaches the HALT.  (24 clock cycles)

;; DISPLAY-5
L02B5:  LD      R,A             ; (9) Load R from A.    R = slow: $F5 fast:
$FC
        LD      A,$DD           ; (7) load future R value.        $F6
$FD

        EI                      ; (4) Enable Interrupts           $F7
$FE

        JP      (HL)            ; (4) jump to the echo display.   $F8
$FF

; ---------------------------------
; THE 'KEYBOARD SCANNING' SUBROUTINE
; ---------------------------------
; The keyboard is read during the vertical sync interval while no video is
```

```
; being displayed.  Reading a port with address bit 0 low i.e. $FE starts
the
; vertical sync pulse.

;; KEYBOARD
L02BB:  LD      HL,$FFFF        ; (16) prepare a buffer to take key.
        LD      BC,$FEFE        ; (20) set BC to port $FEFE. The B
register,
                                ;      with its single reset bit also acts
as
                                ;      an 8-counter.
        IN      A,(C)           ; (11) read the port - all 16 bits are put
on
                                ;      the address bus.  Start VSYNC pulse.
        OR      $01             ; (7)  set the rightmost bit so as to
ignore
                                ;      the SHIFT key.

;; EACH-LINE
L02C5:  OR      $E0             ; [7] OR %11100000
        LD      D,A             ; [4] transfer to D.
        CPL                     ; [4] complement - only bits 4-0 meaningful
now.
        CP      $01             ; [7] sets carry if A is zero.
        SBC     A,A             ; [4] $FF if $00 else zero.
        OR      B               ; [7] $FF or port FE,FD,FB....
        AND     L               ; [4] unless more than one key, L will
still be
                                ;     $FF. if more than one key is pressed
then A is
                                ;     now invalid.
        LD      L,A             ; [4] transfer to L.

; now consider the column identifier.

        LD      A,H             ; [4] will be $FF if no previous keys.
        AND     D               ; [4] 111xxxxx
        LD      H,A             ; [4] transfer A to H

; since only one key may be pressed, H will, if valid, be one of
; 11111110, 11111101, 11111011, 11110111, 11101111
; reading from the outer column, say Q, to the inner column, say T.

        RLC     B               ; [8]  rotate the 8-counter/port address.
                                ;      sets carry if more to do.
        IN      A,(C)           ; [10] read another half-row.
                                ;      all five bits this time.

        JR      C,L02C5         ; [12](7) loop back, until done, to EACH-
LINE

;   The last row read is SHIFT,Z,X,C,V  for the second time.

        RRA                     ; (4) test the shift key - carry will be
reset
                                ;     if the key is pressed.
        RL      H               ; (8) rotate left H picking up the carry
giving
                                ;     column values -
                                ;        $FD, $FB, $F7, $EF, $DF.
                                ;     or $FC, $FA, $F6, $EE, $DE if
shifted.
```

```
;   We now have H identifying the column and L identifying the row in the
;   keyboard matrix.

;   This is a good time to test if this is an American or British machine.
;   The US machine has an extra diode that causes bit 6 of a byte read from
;   a port to be reset.

        RLA                     ; (4) compensate for the shift test.
        RLA                     ; (4) rotate bit 7 out.
        RLA                     ; (4) test bit 6.

        SBC     A,A             ; (4)           $FF or $00 {USA}
        AND     $18             ; (7)           $18 or $00
        ADD     A,$1F           ; (7)           $37 or $1F

;   result is either 31 (USA) or 55 (UK) blank lines above and below the TV
;   picture.

        LD      ($4028),A       ; (13) update system variable MARGIN

        RET                     ; (10) return

; ------------------------------
; THE 'SET FAST MODE' SUBROUTINE
; ------------------------------
;
;

;; SET-FAST
L02E7:  BIT     7,(IY+$3B)      ; sv CDFLAG
        RET     Z               ;

        HALT                    ; Wait for Interrupt
        OUT     ($FD),A         ;
        RES     7,(IY+$3B)      ; sv CDFLAG
        RET                     ; return.


; --------------
; THE 'REPORT-F'
; --------------

;; REPORT-F
L02F4:  RST     08H             ; ERROR-1
        DEFB    $0E             ; Error Report: No Program Name supplied.

; --------------------------
; THE 'SAVE COMMAND' ROUTINE
; --------------------------
;
;

;; SAVE
L02F6:  CALL    L03A8           ; routine NAME
        JR      C,L02F4         ; back with null name to REPORT-F above.

        EX      DE,HL           ;
        LD      DE,$12CB        ; five seconds timing value

;; HEADER
L02FF:  CALL    L0F46           ; routine BREAK-1
```

```
        JR      NC,L0332        ; to BREAK-2

;; DELAY-1
L0304:  DJNZ    L0304           ; to DELAY-1

        DEC     DE              ;
        LD      A,D             ;
        OR      E               ;
        JR      NZ,L02FF        ; back for delay to HEADER

;; OUT-NAME
L030B:  CALL    L031E           ; routine OUT-BYTE
        BIT     7,(HL)          ; test for inverted bit.
        INC     HL              ; address next character of name.
        JR      Z,L030B         ; back if not inverted to OUT-NAME

; now start saving the system variables onwards.

        LD      HL,$4009        ; set start of area to VERSN thereby
                                ; preserving RAMTOP etc.

;; OUT-PROG
L0316:  CALL    L031E           ; routine OUT-BYTE

        CALL    L01FC           ; routine LOAD/SAVE                     >>
        JR      L0316           ; loop back to OUT-PROG

; -------------------------
; THE 'OUT-BYTE' SUBROUTINE
; -------------------------
; This subroutine outputs a byte a bit at a time to a domestic tape
recorder.

;; OUT-BYTE
L031E:  LD      E,(HL)          ; fetch byte to be saved.
        SCF                     ; set carry flag - as a marker.

;; EACH-BIT
L0320:  RL      E               ;  C < 76543210 < C
        RET     Z               ; return when the marker bit has passed
                                ; right through.                        >>

        SBC     A,A             ; $FF if set bit or $00 with no carry.
        AND     $05             ; $05               $00
        ADD     A,$04           ; $09               $04
        LD      C,A             ; transfer timer to C. a set bit has a
longer
                                ; pulse than a reset bit.

;; PULSES
L0329:  OUT     ($FF),A         ; pulse to cassette.
        LD      B,$23           ; set timing constant

;; DELAY-2
L032D:  DJNZ    L032D           ; self-loop to DELAY-2

        CALL    L0F46           ; routine BREAK-1 test for BREAK key.

;; BREAK-2
L0332:  JR      NC,L03A6        ; forward with break to REPORT-D

        LD      B,$1E           ; set timing value.
```

```
;; DELAY-3
L0336:  DJNZ    L0336           ; self-loop to DELAY-3

        DEC     C               ; decrement counter
        JR      NZ,L0329        ; loop back to PULSES

;; DELAY-4
L033B:  AND     A               ; clear carry for next bit test.
        DJNZ    L033B           ; self loop to DELAY-4 (B is zero - 256)

        JR      L0320           ; loop back to EACH-BIT

; --------------------------
; THE 'LOAD COMMAND' ROUTINE
; --------------------------
;
;

;; LOAD
L0340:  CALL    L03A8           ; routine NAME

; DE points to start of name in RAM.

        RL      D               ; pick up carry
        RRC     D               ; carry now in bit 7.

;; NEXT-PROG
L0347:  CALL    L034C           ; routine IN-BYTE
        JR      L0347           ; loop to NEXT-PROG

; ------------------------
; THE 'IN-BYTE' SUBROUTINE
; ------------------------

;; IN-BYTE
L034C:  LD      C,$01           ; prepare an eight counter 00000001.

;; NEXT-BIT
L034E:  LD      B,$00           ; set counter to 256

;; BREAK-3
L0350:  LD      A,$7F           ; read the keyboard row
        IN      A,($FE)         ; with the SPACE key.

        OUT     ($FF),A         ; output signal to screen.

        RRA                     ; test for SPACE pressed.
        JR      NC,L03A2        ; forward if so to BREAK-4

        RLA                     ; reverse above rotation
        RLA                     ; test tape bit.
        JR      C,L0385         ; forward if set to GET-BIT

        DJNZ    L0350           ; loop back to BREAK-3

        POP     AF              ; drop the return address.
        CP      D               ; ugh.

;; RESTART
L0361:  JP      NC,L03E5        ; jump forward to INITIAL if D is zero
                                ; to reset the system
```

```
                                ; if the tape signal has timed out for
example
                                ; if the tape is stopped. Not just a simple
                                ; report as some system variables will have
                                ; been overwritten.

        LD      H,D             ; else transfer the start of name
        LD      L,E             ; to the HL register

;; IN-NAME
L0366:  CALL    L034C           ; routine IN-BYTE is sort of recursion for
name
                                ; part. received byte in C.
        BIT     7,D             ; is name the null string ?
        LD      A,C             ; transfer byte to A.
        JR      NZ,L0371        ; forward with null string to MATCHING

        CP      (HL)            ; else compare with string in memory.
        JR      NZ,L0347        ; back with mis-match to NEXT-PROG
                                ; (seemingly out of subroutine but return
                                ; address has been dropped).


;; MATCHING
L0371:  INC     HL              ; address next character of name
        RLA                     ; test for inverted bit.
        JR      NC,L0366        ; back if not to IN-NAME

; the name has been matched in full.
; proceed to load the data but first increment the high byte of E_LINE,
which
; is one of the system variables to be loaded in. Since the low byte is
loaded
; before the high byte, it is possible that, at the in-between stage, a
false
; value could cause the load to end prematurely - see  LOAD/SAVE check.

        INC     (IY+$15)        ; increment system variable E_LINE_hi.
        LD      HL,$4009        ; start loading at system variable VERSN.

;; IN-PROG
L037B:  LD      D,B             ; set D to zero as indicator.
        CALL    L034C           ; routine IN-BYTE loads a byte
        LD      (HL),C          ; insert assembled byte in memory.
        CALL    L01FC           ; routine LOAD/SAVE                    >>
        JR      L037B           ; loop back to IN-PROG

; ---

; this branch assembles a full byte before exiting normally
; from the IN-BYTE subroutine.

;; GET-BIT
L0385:  PUSH    DE              ; save the
        LD      E,$94           ; timing value.

;; TRAILER
L0388:  LD      B,$1A           ; counter to twenty six.

;; COUNTER
L038A:  DEC     E               ; decrement the measuring timer.
        IN      A,($FE)         ; read the
```

```
        RLA                     ;
        BIT     7,E             ;
        LD      A,E             ;
        JR      C,L0388         ; loop back with carry to TRAILER

        DJNZ    L038A           ; to COUNTER

        POP     DE              ;
        JR      NZ,L039C        ; to BIT-DONE

        CP      $56             ;
        JR      NC,L034E        ; to NEXT-BIT

;; BIT-DONE
L039C:  CCF                     ; complement carry flag
        RL      C               ;
        JR      NC,L034E        ; to NEXT-BIT

        RET                     ; return with full byte.

; ---

; if break is pressed while loading data then perform a reset.
; if break pressed while waiting for program on tape then OK to break.

;; BREAK-4
L03A2:  LD      A,D             ; transfer indicator to A.
        AND     A               ; test for zero.
        JR      Z,L0361         ; back if so to RESTART


;; REPORT-D
L03A6:  RST     08H             ; ERROR-1
        DEFB    $0C             ; Error Report: BREAK - CONT repeats

; ------------------------------
; THE 'PROGRAM NAME' SUBROUTINE
; ------------------------------
;
;

;; NAME
L03A8:  CALL    L0F55           ; routine SCANNING
        LD      A,($4001)       ; sv FLAGS
        ADD     A,A             ;
        JP      M,L0D9A         ; to REPORT-C

        POP     HL              ;
        RET     NC              ;

        PUSH    HL              ;
        CALL    L02E7           ; routine SET-FAST
        CALL    L13F8           ; routine STK-FETCH
        LD      H,D             ;
        LD      L,E             ;
        DEC     C               ;
        RET     M               ;

        ADD     HL,BC           ;
        SET     7,(HL)          ;
        RET                     ;
```

```
; -------------------------
; THE 'NEW' COMMAND ROUTINE
; -------------------------
;
;

;; NEW
L03C3:  CALL    L02E7           ; routine SET-FAST
        LD      BC,($4004)      ; fetch value of system variable RAMTOP
        DEC     BC              ; point to last system byte.

; ----------------------
; THE 'RAM CHECK' ROUTINE
; ----------------------
;
;

;; RAM-CHECK
L03CB:  LD      H,B             ;
        LD      L,C             ;
        LD      A,$3F           ;

;; RAM-FILL
L03CF:  LD      (HL),$02        ;
        DEC     HL              ;
        CP      H               ;
        JR      NZ,L03CF        ; to RAM-FILL

;; RAM-READ
L03D5:  AND     A               ;
        SBC     HL,BC           ;
        ADD     HL,BC           ;
        INC     HL              ;
        JR      NC,L03E2        ; to SET-TOP

        DEC     (HL)            ;
        JR      Z,L03E2         ; to SET-TOP

        DEC     (HL)            ;
        JR      Z,L03D5         ; to RAM-READ

;; SET-TOP
L03E2:  LD      ($4004),HL      ; set system variable RAMTOP to first byte
                                ; above the BASIC system area.

; ---------------------------
; THE 'INITIALIZATION' ROUTINE
; ---------------------------
;
;

;; INITIAL
L03E5:  LD      HL,($4004)      ; fetch system variable RAMTOP.
        DEC     HL              ; point to last system byte.
        LD      (HL),$3E        ; make GO SUB end-marker $3E - too high for
                                ; high order byte of line number.
                                ; (was $3F on ZX80)
        DEC     HL              ; point to unimportant low-order byte.
        LD      SP,HL           ; and initialize the stack-pointer to this
                                ; location.
        DEC     HL              ; point to first location on the machine
stack
```

```
        DEC     HL              ; which will be filled by next CALL/PUSH.
        LD      ($4002),HL      ; set the error stack pointer ERR_SP to
                                ; the base of the now empty machine stack.

; Now set the I register so that the video hardware knows where to find the
; character set. This ROM only uses the character set when printing to
; the ZX Printer. The TV picture is formed by the external video hardware.
; Consider also, that this 8K ROM can be retro-fitted to the ZX80 instead
of
; its original 4K ROM so the video hardware could be on the ZX80.

        LD      A,$1E           ; address for this ROM is $1E00.
        LD      I,A             ; set I register from A.
        IM      1               ; select Z80 Interrupt Mode 1.

        LD      IY,$4000        ; set IY to the start of RAM so that the
                                ; system variables can be indexed.
        LD      (IY+$3B),$40    ; set CDFLAG 0100 0000. Bit 6 indicates
                                ; Compute nad Display required.

        LD      HL,$407D        ; The first location after System Variables
-
                                ; 16509 decimal.
        LD      ($400C),HL      ; set system variable D_FILE to this value.
        LD      B,$19           ; prepare minimal screen of 24 NEWLINEs
                                ; following an initial NEWLINE.

;; LINE
L0408:  LD      (HL),$76        ; insert NEWLINE (HALT instruction)
        INC     HL              ; point to next location.
        DJNZ    L0408           ; loop back for all twenty five to LINE

        LD      ($4010),HL      ; set system variable VARS to next location

        CALL    L149A           ; routine CLEAR sets $80 end-marker and the
                                ; dynamic memory pointers E_LINE, STKBOT
and
                                ; STKEND.

;; N/L-ONLY
L0413:  CALL    L14AD           ; routine CURSOR-IN inserts the cursor and
                                ; end-marker in the Edit Line also setting
                                ; size of lower display to two lines.

        CALL    L0207           ; routine SLOW/FAST selects COMPUTE and
DISPLAY

; ---------------------------
; THE 'BASIC LISTING' SECTION
; ---------------------------
;
;

;; UPPER
L0419:  CALL    L0A2A           ; routine CLS
        LD      HL,($400A)      ; sv E_PPC_lo
        LD      DE,($4023)      ; sv S_TOP_lo
        AND     A               ;
        SBC     HL,DE           ;
        EX      DE,HL           ;
        JR      NC,L042D        ; to ADDR-TOP
```

```
        ADD     HL,DE           ;
        LD      ($4023),HL      ; sv S_TOP_lo

;; ADDR-TOP
L042D:  CALL    L09D8           ; routine LINE-ADDR
        JR      Z,L0433         ; to LIST-TOP

        EX      DE,HL           ;

;; LIST-TOP
L0433:  CALL    L073E           ; routine LIST-PROG
        DEC     (IY+$1E)        ; sv BERG
        JR      NZ,L0472        ; to LOWER

        LD      HL,($400A)      ; sv E_PPC_lo
        CALL    L09D8           ; routine LINE-ADDR
        LD      HL,($4016)      ; sv CH_ADD_lo
        SCF                     ; Set Carry Flag
        SBC     HL,DE           ;
        LD      HL,$4023        ; sv S_TOP_lo
        JR      NC,L0457        ; to INC-LINE

        EX      DE,HL           ;
        LD      A,(HL)          ;
        INC     HL              ;
        LDI                     ;
        LD      (DE),A          ;
        JR       L0419          ; to UPPER

; ---

;; DOWN-KEY
L0454:  LD      HL,$400A        ; sv E_PPC_lo

;; INC-LINE
L0457:  LD      E,(HL)          ;
        INC     HL              ;
        LD      D,(HL)          ;
        PUSH    HL              ;
        EX      DE,HL           ;
        INC     HL              ;
        CALL    L09D8           ; routine LINE-ADDR
        CALL    L05BB           ; routine LINE-NO
        POP     HL              ;

;; KEY-INPUT
L0464:  BIT     5,(IY+$2D)      ; sv FLAGX
        JR      NZ,L0472        ; forward to LOWER

        LD      (HL),D          ;
        DEC     HL              ;
        LD      (HL),E          ;
        JR      L0419           ; to UPPER

; ---------------------------
; THE 'EDIT LINE COPY' SECTION
; ---------------------------
; This routine sets the edit line to just the cursor when
; 1) There is not enough memory to edit a BASIC line.
; 2) The edit key is used during input.
; The entry point LOWER
```

```
;; EDIT-INP
L046F:  CALL    L14AD           ; routine CURSOR-IN sets cursor only edit
line.

; ->

;; LOWER
L0472:  LD      HL,($4014)      ; fetch edit line start from E_LINE.

;; EACH-CHAR
L0475:  LD      A,(HL)          ; fetch a character from edit line.
        CP      $7E             ; compare to the number marker.
        JR      NZ,L0482        ; forward if not to END-LINE

        LD      BC,$0006        ; else six invisible bytes to be removed.
        CALL    L0A60           ; routine RECLAIM-2
        JR      L0475           ; back to EACH-CHAR

; ---

;; END-LINE
L0482:  CP      $76             ;
        INC     HL              ;
        JR      NZ,L0475        ; to EACH-CHAR

;; EDIT-LINE
L0487:  CALL    L0537           ; routine CURSOR sets cursor K or L.

;; EDIT-ROOM
L048A:  CALL    L0A1F           ; routine LINE-ENDS
        LD      HL,($4014)      ; sv E_LINE_lo
        LD      (IY+$00),$FF    ; sv ERR_NR
        CALL    L0766           ; routine COPY-LINE
        BIT     7,(IY+$00)      ; sv ERR_NR
        JR      NZ,L04C1        ; to DISPLAY-6

        LD      A,($4022)       ; sv DF_SZ
        CP      $18             ;
        JR      NC,L04C1        ; to DISPLAY-6

        INC     A               ;
        LD      ($4022),A       ; sv DF_SZ
        LD      B,A             ;
        LD      C,$01           ;
        CALL    L0918           ; routine LOC-ADDR
        LD      D,H             ;
        LD      E,L             ;
        LD      A,(HL)          ;

;; FREE-LINE
L04B1:  DEC     HL              ;
        CP      (HL)            ;
        JR      NZ,L04B1        ; to FREE-LINE

        INC     HL              ;
        EX      DE,HL           ;
        LD      A,($4005)       ; sv RAMTOP_hi
        CP      $4D             ;
        CALL    C,L0A5D         ; routine RECLAIM-1
        JR      L048A           ; to EDIT-ROOM
```

```
; --------------------------
; THE 'WAIT FOR KEY' SECTION
; --------------------------
;
;

;; DISPLAY-6
L04C1:  LD      HL,$0000        ;
        LD      ($4018),HL      ; sv X_PTR_lo

        LD      HL,$403B        ; system variable CDFLAG
        BIT     7,(HL)          ;

        CALL    Z,L0229         ; routine DISPLAY-1

;; SLOW-DISP
L04CF:  BIT     0,(HL)          ;
        JR      Z,L04CF         ; to SLOW-DISP

        LD      BC,($4025)      ; sv LAST_K
        CALL    L0F4B           ; routine DEBOUNCE
        CALL    L07BD           ; routine DECODE

        JR      NC,L0472        ; back to LOWER

; -------------------------------
; THE 'KEYBOARD DECODING' SECTION
; -------------------------------
;   The decoded key value is in E and HL points to the position in the
;   key table. D contains zero.

;; K-DECODE
L04DF:  LD      A,($4006)       ; Fetch value of system variable MODE
        DEC     A               ; test the three values together

        JP      M,L0508         ; forward, if was zero, to FETCH-2

        JR      NZ,L04F7        ; forward, if was 2, to FETCH-1

;   The original value was one and is now zero.

        LD      ($4006),A       ; update the system variable MODE

        DEC     E               ; reduce E to range $00 - $7F
        LD      A,E             ; place in A
        SUB     $27             ; subtract 39 setting carry if range 00 -
38
        JR      C,L04F2         ; forward, if so, to FUNC-BASE

        LD      E,A             ; else set E to reduced value

;; FUNC-BASE
L04F2:  LD      HL,L00CC        ; address of K-FUNCT table for function
keys.
        JR      L0505           ; forward to TABLE-ADD

; ---

;; FETCH-1
L04F7:  LD      A,(HL)          ;
        CP      $76             ;
        JR      Z,L052B         ; to K/L-KEY
```

```
        CP      $40             ;
        SET     7,A             ;
        JR      C,L051B         ; to ENTER

        LD      HL,$00C7        ; (expr reqd)

;; TABLE-ADD
L0505:  ADD     HL,DE           ;
        JR      L0515           ; to FETCH-3

; ---

;; FETCH-2
L0508:  LD      A,(HL)          ;
        BIT     2,(IY+$01)      ; sv FLAGS  - K or L mode ?
        JR      NZ,L0516        ; to TEST-CURS

        ADD     A,$C0           ;
        CP      $E6             ;
        JR      NC,L0516        ; to TEST-CURS

;; FETCH-3
L0515:  LD      A,(HL)          ;

;; TEST-CURS
L0516:  CP      $F0             ;
        JP      PE,L052D        ; to KEY-SORT

;; ENTER
L051B:  LD      E,A             ;
        CALL    L0537           ; routine CURSOR

        LD      A,E             ;
        CALL    L0526           ; routine ADD-CHAR

;; BACK-NEXT
L0523:  JP      L0472           ; back to LOWER

; ------------------------------
; THE 'ADD CHARACTER' SUBROUTINE
; ------------------------------
;
;

;; ADD-CHAR
L0526:  CALL    L099B           ; routine ONE-SPACE
        LD      (DE),A          ;
        RET                     ;

; -------------------------
; THE 'CURSOR KEYS' ROUTINE
; -------------------------
;
;

;; K/L-KEY
L052B:  LD      A,$78           ;

;; KEY-SORT
L052D:  LD      E,A             ;
        LD      HL,$0482        ; base address of ED-KEYS (exp reqd)
```

```
        ADD     HL,DE           ;
        ADD     HL,DE           ;
        LD      C,(HL)          ;
        INC     HL              ;
        LD      B,(HL)          ;
        PUSH    BC              ;

;; CURSOR
L0537:  LD      HL,($4014)      ; sv E_LINE_lo
        BIT     5,(IY+$2D)      ; sv FLAGX
        JR      NZ,L0556        ; to L-MODE

;; K-MODE
L0540:  RES     2,(IY+$01)      ; sv FLAGS  - Signal use K mode

;; TEST-CHAR
L0544:  LD      A,(HL)          ;
        CP      $7F             ;
        RET     Z               ; return

        INC     HL              ;
        CALL    L07B4           ; routine NUMBER
        JR      Z,L0544         ; to TEST-CHAR

        CP      $26             ;
        JR      C,L0544         ; to TEST-CHAR

        CP      $DE             ;
        JR      Z,L0540         ; to K-MODE

;; L-MODE
L0556:  SET     2,(IY+$01)      ; sv FLAGS  - Signal use L mode
        JR      L0544           ; to TEST-CHAR

; -------------------------
; THE 'CLEAR-ONE' SUBROUTINE
; -------------------------
;
;

;; CLEAR-ONE
L055C:  LD      BC,$0001        ;
        JP      L0A60           ; to RECLAIM-2



; ------------------------
; THE 'EDITING KEYS' TABLE
; ------------------------
;
;

;; ED-KEYS
L0562:  DEFW    L059F           ; Address: $059F; Address: UP-KEY
        DEFW    L0454           ; Address: $0454; Address: DOWN-KEY
        DEFW    L0576           ; Address: $0576; Address: LEFT-KEY
        DEFW    L057F           ; Address: $057F; Address: RIGHT-KEY
        DEFW    L05AF           ; Address: $05AF; Address: FUNCTION
        DEFW    L05C4           ; Address: $05C4; Address: EDIT-KEY
        DEFW    L060C           ; Address: $060C; Address: N/L-KEY
        DEFW    L058B           ; Address: $058B; Address: RUBOUT
        DEFW    L05AF           ; Address: $05AF; Address: FUNCTION
```

```
        DEFW    L05AF           ; Address: $05AF; Address: FUNCTION


; -------------------------
; THE 'CURSOR LEFT' ROUTINE
; -------------------------
;
;

;; LEFT-KEY
L0576:  CALL    L0593           ; routine LEFT-EDGE
        LD      A,(HL)          ;
        LD      (HL),$7F        ;
        INC     HL              ;
        JR      L0588           ; to GET-CODE

; --------------------------
; THE 'CURSOR RIGHT' ROUTINE
; --------------------------
;
;

;; RIGHT-KEY
L057F:  INC     HL              ;
        LD      A,(HL)          ;
        CP      $76             ;
        JR      Z,L059D         ; to ENDED-2

        LD      (HL),$7F        ;
        DEC     HL              ;

;; GET-CODE
L0588:  LD      (HL),A          ;

;; ENDED-1
L0589:  JR      L0523           ; to BACK-NEXT

; --------------------
; THE 'RUBOUT' ROUTINE
; --------------------
;
;

;; RUBOUT
L058B:  CALL    L0593           ; routine LEFT-EDGE
        CALL    L055C           ; routine CLEAR-ONE
        JR      L0589           ; to ENDED-1

; ------------------------
; THE 'ED-EDGE' SUBROUTINE
; ------------------------
;
;

;; LEFT-EDGE
L0593:  DEC     HL              ;
        LD      DE,($4014)      ; sv E_LINE_lo
        LD      A,(DE)          ;
        CP      $7F             ;
        RET     NZ              ;

        POP     DE              ;
```

```
;; ENDED-2
L059D:  JR      L0589           ; to ENDED-1

; ----------------------
; THE 'CURSOR UP' ROUTINE
; ----------------------
;
;

;; UP-KEY
L059F:  LD      HL,($400A)      ; sv E_PPC_lo
        CALL    L09D8           ; routine LINE-ADDR
        EX      DE,HL           ;
        CALL    L05BB           ; routine LINE-NO
        LD      HL,$400B        ; point to system variable E_PPC_hi
        JP      L0464           ; jump back to KEY-INPUT

; -------------------------
; THE 'FUNCTION KEY' ROUTINE
; -------------------------
;
;

;; FUNCTION
L05AF:  LD      A,E             ;
        AND     $07             ;
        LD      ($4006),A       ; sv MODE
        JR      L059D           ; back to ENDED-2

; ------------------------------------
; THE 'COLLECT LINE NUMBER' SUBROUTINE
; ------------------------------------
;
;

;; ZERO-DE
L05B7:  EX      DE,HL           ;
        LD      DE,L04C1 + 1    ; $04C2 - a location addressing two zeros.

; ->

;; LINE-NO
L05BB:  LD      A,(HL)          ;
        AND     $C0             ;
        JR      NZ,L05B7        ; to ZERO-DE

        LD      D,(HL)          ;
        INC     HL              ;
        LD      E,(HL)          ;
        RET                     ;

; ----------------------
; THE 'EDIT KEY' ROUTINE
; ----------------------
;
;

;; EDIT-KEY
L05C4:  CALL    L0A1F           ; routine LINE-ENDS clears lower display.

        LD      HL,L046F        ; Address: EDIT-INP
```

```
        PUSH    HL              ; ** is pushed as an error looping address.

        BIT     5,(IY+$2D)      ; test FLAGX
        RET     NZ              ; indirect jump if in input mode
                                ; to L046F, EDIT-INP (begin again).

;

        LD      HL,($4014)      ; fetch E_LINE
        LD      ($400E),HL      ; and use to update the screen cursor DF_CC

; so now RST $10 will print the line numbers to the edit line instead of
screen.
; first make sure that no newline/out of screen can occur while sprinting
the
; line numbers to the edit line.

        LD      HL,$1821        ; prepare line 0, column 0.
        LD      ($4039),HL      ; update S_POSN with these dummy values.

        LD      HL,($400A)      ; fetch current line from E_PPC may be a
                                ; non-existent line e.g. last line deleted.
        CALL    L09D8           ; routine LINE-ADDR gets address or that of
                                ; the following line.
        CALL    L05BB           ; routine LINE-NO gets line number if any
in DE
                                ; leaving HL pointing at second low byte.

        LD      A,D             ; test the line number for zero.
        OR      E               ;
        RET     Z               ; return if no line number - no program to
edit.

        DEC     HL              ; point to high byte.
        CALL    L0AA5           ; routine OUT-NO writes number to edit
line.

        INC     HL              ; point to length bytes.
        LD      C,(HL)          ; low byte to C.
        INC     HL              ;
        LD      B,(HL)          ; high byte to B.

        INC     HL              ; point to first character in line.
        LD      DE,($400E)      ; fetch display file cursor DF_CC

        LD      A,$7F           ; prepare the cursor character.
        LD      (DE),A          ; and insert in edit line.
        INC     DE              ; increment intended destination.

        PUSH    HL              ; * save start of BASIC.

        LD      HL,$001D        ; set an overhead of 29 bytes.
        ADD     HL,DE           ; add in the address of cursor.
        ADD     HL,BC           ; add the length of the line.
        SBC     HL,SP           ; subtract the stack pointer.

        POP     HL              ; * restore pointer to start of BASIC.

        RET     NC              ; return if not enough room to L046F EDIT-
INP.
                                ; the edit key appears not to work.
```

```
        LDIR                    ; else copy bytes from program to edit
line.
                                ; Note. hidden floating point forms are
also
                                ; copied to edit line.

        EX      DE,HL           ; transfer free location pointer to HL

        POP     DE              ; ** remove address EDIT-INP from stack.

        CALL    L14A6           ; routine SET-STK-B sets STKEND from HL.

        JR      L059D           ; back to ENDED-2 and after 3 more jumps
                                ; to L0472, LOWER.
                                ; Note. The LOWER routine removes the
hidden
                                ; floating-point numbers from the edit
line.

; -------------------------
; THE 'NEWLINE KEY' ROUTINE
; -------------------------
;
;

;; N/L-KEY
L060C:  CALL    L0A1F           ; routine LINE-ENDS

        LD      HL,L0472        ; prepare address: LOWER

        BIT     5,(IY+$2D)      ; sv FLAGX
        JR      NZ,L0629        ; to NOW-SCAN

        LD      HL,($4014)      ; sv E_LINE_lo
        LD      A,(HL)          ;
        CP      $FF             ;
        JR      Z,L0626         ; to STK-UPPER

        CALL    L08E2           ; routine CLEAR-PRB
        CALL    L0A2A           ; routine CLS

;; STK-UPPER
L0626:  LD      HL,L0419        ; Address: UPPER

;; NOW-SCAN
L0629:  PUSH    HL              ; push routine address (LOWER or UPPER).
        CALL    L0CBA           ; routine LINE-SCAN
        POP     HL              ;
        CALL    L0537           ; routine CURSOR
        CALL    L055C           ; routine CLEAR-ONE
        CALL    L0A73           ; routine E-LINE-NO
        JR      NZ,L064E        ; to N/L-INP

        LD      A,B             ;
        OR      C               ;
        JP      NZ,L06E0        ; to N/L-LINE

        DEC     BC              ;
        DEC     BC              ;
        LD      ($4007),BC      ; sv PPC_lo
        LD      (IY+$22),$02    ; sv DF_SZ
        LD      DE,($400C)      ; sv D_FILE_lo
```

```
        JR      L0661           ; forward to TEST-NULL

; ---

;; N/L-INP
L064E:  CP      $76             ;
        JR      Z,L0664         ; to N/L-NULL

        LD      BC,($4030)      ; sv T_ADDR_lo
        CALL    L0918           ; routine LOC-ADDR
        LD      DE,($4029)      ; sv NXTLIN_lo
        LD      (IY+$22),$02    ; sv DF_SZ

;; TEST-NULL
L0661:  RST     18H             ; GET-CHAR
        CP      $76             ;

;; N/L-NULL
L0664:  JP      Z,L0413         ; to N/L-ONLY

        LD      (IY+$01),$80    ; sv FLAGS
        EX      DE,HL           ;

;; NEXT-LINE
L066C:  LD      ($4029),HL      ; sv NXTLIN_lo
        EX      DE,HL           ;
        CALL    L004D           ; routine TEMP-PTR-2
        CALL    L0CC1           ; routine LINE-RUN
        RES     1,(IY+$01)      ; sv FLAGS  - Signal printer not in use
        LD      A,$C0           ;
        LD      (IY+$19),A      ; sv X_PTR_lo
        CALL    L14A3           ; routine X-TEMP
        RES     5,(IY+$2D)      ; sv FLAGX
        BIT     7,(IY+$00)      ; sv ERR_NR
        JR      Z,L06AE         ; to STOP-LINE

        LD      HL,($4029)      ; sv NXTLIN_lo
        AND     (HL)            ;
        JR       NZ,L06AE       ; to STOP-LINE

        LD      D,(HL)          ;
        INC     HL              ;
        LD      E,(HL)          ;
        LD      ($4007),DE      ; sv PPC_lo
        INC     HL              ;
        LD      E,(HL)          ;
        INC     HL              ;
        LD      D,(HL)          ;
        INC     HL              ;
        EX      DE,HL           ;
        ADD     HL,DE           ;
        CALL    L0F46           ; routine BREAK-1
        JR      C,L066C         ; to NEXT-LINE

        LD      HL,$4000        ; sv ERR_NR
        BIT     7,(HL)          ;
        JR      Z,L06AE         ; to STOP-LINE

        LD      (HL),$0C        ;

;; STOP-LINE
```

```
L06AE:  BIT     7,(IY+$38)      ; sv PR_CC
        CALL    Z,L0871         ; routine COPY-BUFF
        LD      BC,$0121        ;
        CALL    L0918           ; routine LOC-ADDR
        LD      A,($4000)       ; sv ERR_NR
        LD      BC,($4007)      ; sv PPC_lo
        INC     A               ;
        JR      Z,L06D1         ; to REPORT

        CP      $09             ;
        JR      NZ,L06CA        ; to CONTINUE

        INC     BC              ;

;; CONTINUE
L06CA:  LD      ($402B),BC      ; sv OLDPPC_lo
        JR      NZ,L06D1        ; to REPORT

        DEC     BC              ;

;; REPORT
L06D1:  CALL    L07EB           ; routine OUT-CODE
        LD      A,$18           ;

        RST     10H             ; PRINT-A
        CALL    L0A98           ; routine OUT-NUM
        CALL    L14AD           ; routine CURSOR-IN
        JP      L04C1           ; to DISPLAY-6

; ---

;; N/L-LINE
L06E0:  LD      ($400A),BC      ; sv E_PPC_lo
        LD      HL,($4016)      ; sv CH_ADD_lo
        EX      DE,HL           ;
        LD      HL,L0413        ; Address: N/L-ONLY
        PUSH    HL              ;
        LD      HL,($401A)      ; sv STKBOT_lo
        SBC     HL,DE           ;
        PUSH    HL              ;
        PUSH    BC              ;
        CALL    L02E7           ; routine SET-FAST
        CALL    L0A2A           ; routine CLS
        POP     HL              ;
        CALL    L09D8           ; routine LINE-ADDR
        JR      NZ,L0705        ; to COPY-OVER

        CALL    L09F2           ; routine NEXT-ONE
        CALL    L0A60           ; routine RECLAIM-2

;; COPY-OVER
L0705:  POP     BC              ;
        LD      A,C             ;
        DEC     A               ;
        OR      B               ;
        RET     Z               ;

        PUSH    BC              ;
        INC     BC              ;
        INC     BC              ;
        INC     BC              ;
        INC     BC              ;
```

```
        DEC     HL              ;
        CALL    L099E           ; routine MAKE-ROOM
        CALL    L0207           ; routine SLOW/FAST
        POP     BC              ;
        PUSH    BC              ;
        INC     DE              ;
        LD      HL,($401A)      ; sv STKBOT_lo
        DEC     HL              ;
        LDDR                    ; copy bytes
        LD      HL,($400A)      ; sv E_PPC_lo
        EX      DE,HL           ;
        POP     BC              ;
        LD      (HL),B          ;
        DEC     HL              ;
        LD      (HL),C          ;
        DEC     HL              ;
        LD      (HL),E          ;
        DEC     HL              ;
        LD      (HL),D          ;

        RET                     ; return.

; --------------------------------------
; THE 'LIST' AND 'LLIST' COMMAND ROUTINES
; --------------------------------------
;
;

;; LLIST
L072C:  SET     1,(IY+$01)      ; sv FLAGS  - signal printer in use

;; LIST
L0730:  CALL    L0EA7           ; routine FIND-INT

        LD      A,B             ; fetch high byte of user-supplied line
number.
        AND     $3F             ; and crudely limit to range 1-16383.

        LD      H,A             ;
        LD      L,C             ;
        LD      ($400A),HL      ; sv E_PPC_lo
        CALL    L09D8           ; routine LINE-ADDR

;; LIST-PROG
L073E:  LD      E,$00           ;

;; UNTIL-END
L0740:  CALL    L0745           ; routine OUT-LINE lists one line of BASIC
                                ; making an early return when the screen is
                                ; full or the end of program is reached.
>>
        JR      L0740           ; loop back to UNTIL-END

; ----------------------------------
; THE 'PRINT A BASIC LINE' SUBROUTINE
; ----------------------------------
;
;

;; OUT-LINE
L0745:  LD      BC,($400A)      ; sv E_PPC_lo
        CALL    L09EA           ; routine CP-LINES
```

```
         LD      D,$92           ;
         JR      Z,L0755         ; to TEST-END

         LD      DE,$0000        ;
         RL      E               ;

;; TEST-END
L0755:   LD      (IY+$1E),E      ; sv BERG
         LD      A,(HL)          ;
         CP      $40             ;
         POP     BC              ;
         RET     NC              ;

         PUSH    BC              ;
         CALL    L0AA5           ; routine OUT-NO
         INC     HL              ;
         LD      A,D             ;

         RST     10H             ; PRINT-A
         INC     HL              ;
         INC     HL              ;

;; COPY-LINE
L0766:   LD      ($4016),HL      ; sv CH_ADD_lo
         SET     0,(IY+$01)      ; sv FLAGS  - Suppress leading space

;; MORE-LINE
L076D:   LD      BC,($4018)      ; sv X_PTR_lo
         LD      HL,($4016)      ; sv CH_ADD_lo
         AND      A              ;
         SBC     HL,BC           ;
         JR      NZ,L077C        ; to TEST-NUM

         LD      A,$B8           ;

         RST     10H             ; PRINT-A

;; TEST-NUM
L077C:   LD      HL,($4016)      ; sv CH_ADD_lo
         LD      A,(HL)          ;
         INC     HL              ;
         CALL    L07B4           ; routine NUMBER
         LD      ($4016),HL      ; sv CH_ADD_lo
         JR      Z,L076D         ; to MORE-LINE

         CP      $7F             ;
         JR      Z,L079D         ; to OUT-CURS

         CP      $76             ;
         JR      Z,L07EE         ; to OUT-CH

         BIT     6,A             ;
         JR      Z,L079A         ; to NOT-TOKEN

         CALL    L094B           ; routine TOKENS
         JR      L076D           ; to MORE-LINE

; ---


;; NOT-TOKEN
L079A:   RST     10H             ; PRINT-A
```

```
        JR      L076D           ; to MORE-LINE

; ---

;; OUT-CURS
L079D:  LD      A,($4006)       ; Fetch value of system variable MODE
        LD      B,$AB           ; Prepare an inverse [F] for function
cursor.

        AND     A               ; Test for zero -
        JR      NZ,L07AA        ; forward if not to FLAGS-2

        LD      A,($4001)       ; Fetch system variable FLAGS.
        LD      B,$B0           ; Prepare an inverse [K] for keyword
cursor.

;; FLAGS-2
L07AA:  RRA                     ; 00000?00 -> 000000?0
        RRA                     ; 000000?0 -> 0000000?
        AND     $01             ; 0000000?    0000000x

        ADD     A,B             ; Possibly [F] -> [G]  or  [K] -> [L]

        CALL    L07F5           ; routine PRINT-SP prints character
        JR      L076D           ; back to MORE-LINE

; ------------------------
; THE 'NUMBER' SUBROUTINE
; ------------------------
;
;

;; NUMBER
L07B4:  CP      $7E             ;
        RET     NZ              ;

        INC     HL              ;
        INC     HL              ;
        INC     HL              ;
        INC     HL              ;
        INC     HL              ;
        RET                     ;

; --------------------------------
; THE 'KEYBOARD DECODE' SUBROUTINE
; --------------------------------
;
;

;; DECODE
L07BD:  LD      D,$00           ;
        SRA     B               ;
        SBC     A,A             ;
        OR      $26             ;
        LD      L,$05           ;
        SUB     L               ;

;; KEY-LINE
L07C7:  ADD     A,L             ;
        SCF                     ; Set Carry Flag
        RR      C               ;
        JR      C,L07C7         ; to KEY-LINE
```

```
         INC     C               ;
         RET      NZ             ;

         LD      C,B             ;
         DEC     L               ;
         LD      L,$01           ;
         JR      NZ,L07C7        ; to KEY-LINE

         LD      HL,$007D        ; (expr reqd)
         LD      E,A             ;
         ADD     HL,DE           ;
         SCF                     ; Set Carry Flag
         RET                     ;

; -------------------------
; THE 'PRINTING' SUBROUTINE
; -------------------------
;
;

;; LEAD-SP
L07DC:   LD      A,E             ;
         AND     A               ;
         RET     M               ;

         JR      L07F1           ; to PRINT-CH

; ---

;; OUT-DIGIT
L07E1:   XOR     A               ;

;; DIGIT-INC
L07E2:   ADD     HL,BC           ;
         INC     A               ;
         JR      C,L07E2         ; to DIGIT-INC

         SBC     HL,BC           ;
         DEC     A               ;
         JR      Z,L07DC         ; to LEAD-SP

;; OUT-CODE
L07EB:   LD      E,$1C           ;
         ADD     A,E             ;

;; OUT-CH
L07EE:   AND     A               ;
         JR      Z,L07F5         ; to PRINT-SP

;; PRINT-CH
L07F1:   RES     0,(IY+$01)      ; update FLAGS - signal leading space
permitted

;; PRINT-SP
L07F5:   EXX                     ;
         PUSH    HL              ;
         BIT     1,(IY+$01)      ; test FLAGS - is printer in use ?
         JR      NZ,L0802        ; to LPRINT-A

         CALL    L0808           ; routine ENTER-CH
         JR      L0805           ; to PRINT-EXX
```

```
; ---

;; LPRINT-A
L0802:  CALL    L0851           ; routine LPRINT-CH

;; PRINT-EXX
L0805:  POP     HL              ;
        EXX                     ;
        RET                     ;

; ---

;; ENTER-CH
L0808:  LD      D,A             ;
        LD      BC,($4039)      ; sv S_POSN_x
        LD      A,C             ;
        CP      $21             ;
        JR      Z,L082C         ; to TEST-LOW

;; TEST-N/L
L0812:  LD      A,$76           ;
        CP      D               ;
        JR      Z,L0847         ; to WRITE-N/L

        LD      HL,($400E)      ; sv DF_CC_lo
        CP      (HL)            ;
        LD      A,D             ;
        JR      NZ,L083E        ; to WRITE-CH

        DEC     C               ;
        JR      NZ,L083A        ; to EXPAND-1

        INC     HL              ;
        LD       ($400E),HL     ; sv DF_CC_lo
        LD      C,$21           ;
        DEC     B               ;
        LD      ($4039),BC      ; sv S_POSN_x

;; TEST-LOW
L082C:  LD      A,B             ;
        CP      (IY+$22)        ; sv DF_SZ
        JR      Z,L0835         ; to REPORT-5

        AND     A               ;
        JR      NZ,L0812        ; to TEST-N/L

;; REPORT-5
L0835:  LD      L,$04           ; 'No more room on screen'
        JP      L0058           ; to ERROR-3

; ---

;; EXPAND-1
L083A:  CALL    L099B           ; routine ONE-SPACE
        EX      DE,HL           ;

;; WRITE-CH
L083E:  LD      (HL),A          ;
        INC     HL              ;
        LD      ($400E),HL      ; sv DF_CC_lo
        DEC     (IY+$39)        ; sv S_POSN_x
```

```
        RET                     ;

; ---

;; WRITE-N/L
L0847:  LD      C,$21           ;
        DEC     B               ;
        SET     0,(IY+$01)      ; sv FLAGS  - Suppress leading space
        JP      L0918           ; to LOC-ADDR

; -------------------------
; THE 'LPRINT-CH' SUBROUTINE
; -------------------------
; This routine sends a character to the ZX-Printer placing the code for the
; character in the Printer Buffer.
; Note. PR-CC contains the low byte of the buffer address. The high order
byte
; is always constant.


;; LPRINT-CH
L0851:  CP      $76             ; compare to NEWLINE.
        JR      Z,L0871         ; forward if so to COPY-BUFF

        LD      C,A             ; take a copy of the character in C.
        LD      A,($4038)       ; fetch print location from PR_CC
        AND     $7F             ; ignore bit 7 to form true position.
        CP      $5C             ; compare to 33rd location

        LD      L,A             ; form low-order byte.
        LD      H,$40           ; the high-order byte is fixed.

        CALL    Z,L0871         ; routine COPY-BUFF to send full buffer to
                                ; the printer if first 32 bytes full.
                                ; (this will reset HL to start.)

        LD      (HL),C          ; place character at location.
        INC     L               ; increment - will not cross a 256
boundary.
        LD      (IY+$38),L      ; update system variable PR_CC
                                ; automatically resetting bit 7 to show
that
                                ; the buffer is not empty.
        RET                     ; return.

; -------------------------
; THE 'COPY' COMMAND ROUTINE
; -------------------------
; The full character-mapped screen is copied to the ZX-Printer.
; All twenty-four text/graphic lines are printed.

;; COPY
L0869:  LD      D,$16           ; prepare to copy twenty four text lines.
        LD      HL,($400C)      ; set HL to start of display file from
D_FILE.
        INC     HL              ;
        JR      L0876           ; forward to COPY*D

; ---

; A single character-mapped printer buffer is copied to the ZX-Printer.
```

```
;; COPY-BUFF
L0871:  LD      D,$01           ; prepare to copy a single text line.
        LD      HL,$403C        ; set HL to start of printer buffer PRBUFF.

; both paths converge here.

;; COPY*D
L0876:  CALL    L02E7           ; routine SET-FAST

        PUSH    BC              ; *** preserve BC throughout.
                                ; a pending character may be present
                                ; in C from LPRINT-CH

;; COPY-LOOP
L087A:  PUSH    HL              ; save first character of line pointer. (*)
        XOR     A               ; clear accumulator.
        LD      E,A             ; set pixel line count, range 0-7, to zero.

; this inner loop deals with each horizontal pixel line.

;; COPY-TIME
L087D:  OUT     ($FB),A         ; bit 2 reset starts the printer motor
                                ; with an inactive stylus - bit 7 reset.
        POP     HL              ; pick up first character of line pointer
(*)
                                ; on inner loop.

;; COPY-BRK
L0880:  CALL    L0F46           ; routine BREAK-1
        JR      C,L088A         ; forward with no keypress to COPY-CONT

; else A will hold 11111111 0

        RRA                     ; 0111 1111
        OUT     ($FB),A         ; stop ZX printer motor, de-activate
stylus.

;; REPORT-D2
L0888:  RST     08H             ; ERROR-1
        DEFB    $0C             ; Error Report: BREAK - CONT repeats

; ---

;; COPY-CONT
L088A:  IN      A,($FB)         ; read from printer port.
        ADD     A,A             ; test bit 6 and 7
        JP      M,L08DE         ; jump forward with no printer to COPY-END

        JR      NC,L0880        ; back if stylus not in position to COPY-
BRK

        PUSH    HL              ; save first character of line pointer (*)
        PUSH    DE              ; ** preserve character line and pixel
line.

        LD      A,D             ; text line count to A?
        CP      $02             ; sets carry if last line.
        SBC     A,A             ; now $FF if last line else zero.

; now cleverly prepare a printer control mask setting bit 2 (later moved to
1)
; of D to slow printer for the last two pixel lines ( E = 6 and 7)
```

```
         AND     E               ; and with pixel line offset 0-7
         RLCA                    ; shift to left.
         AND     E               ; and again.
         LD      D,A             ; store control mask in D.

;; COPY-NEXT
L089C:   LD      C,(HL)          ; load character from screen or buffer.
         LD      A,C             ; save a copy in C for later inverse test.
         INC     HL              ; update pointer for next time.
         CP      $76             ; is character a NEWLINE ?
         JR      Z,L08C7         ; forward, if so, to COPY-N/L

         PUSH    HL              ; * else preserve the character pointer.

         SLA     A               ; (?) multiply by two
         ADD     A,A             ; multiply by four
         ADD     A,A             ; multiply by eight

         LD      H,$0F           ; load H with half the address of character
set.
         RL      H               ; now $1E or $1F (with carry)
         ADD     A,E             ; add byte offset 0-7
         LD      L,A             ; now HL addresses character source byte

         RL      C               ; test character, setting carry if inverse.
         SBC     A,A             ; accumulator now $00 if normal, $FF if
inverse.

         XOR     (HL)            ; combine with bit pattern at end or ROM.
         LD      C,A             ; transfer the byte to C.
         LD      B,$08           ; count eight bits to output.

;; COPY-BITS
L08B5:   LD      A,D             ; fetch speed control mask from D.
         RLC     C               ; rotate a bit from output byte to carry.
         RRA                     ; pick up in bit 7, speed bit to bit 1
         LD      H,A             ; store aligned mask in H register.

;; COPY-WAIT
L08BA:   IN      A,($FB)         ; read the printer port
         RRA                     ; test for alignment signal from encoder.
         JR      NC,L08BA        ; loop if not present to COPY-WAIT

         LD      A,H             ; control byte to A.
         OUT     ($FB),A         ; and output to printer port.
         DJNZ    L08B5           ; loop for all eight bits to COPY-BITS

         POP     HL              ; * restore character pointer.
         JR      L089C           ; back for adjacent character line to COPY-
NEXT

; ---

; A NEWLINE has been encountered either following a text line or as the
; first character of the screen or printer line.

;; COPY-N/L
L08C7:   IN      A,($FB)         ; read printer port.
         RRA                     ; wait for encoder signal.
         JR      NC,L08C7        ; loop back if not to COPY-N/L
```

```
        LD      A,D             ; transfer speed mask to A.
        RRCA                    ; rotate speed bit to bit 1.
                                ; bit 7, stylus control is reset.
        OUT     ($FB),A         ; set the printer speed.

        POP     DE              ; ** restore character line and pixel line.
        INC     E               ; increment pixel line 0-7.
        BIT     3,E             ; test if value eight reached.
        JR      Z,L087D         ; back if not to COPY-TIME

; eight pixel lines, a text line have been completed.

        POP     BC              ; lose the now redundant first character
                                ; pointer
        DEC     D               ; decrease text line count.
        JR      NZ,L087A        ; back if not zero to COPY-LOOP

        LD      A,$04           ; stop the already slowed printer motor.
        OUT     ($FB),A         ; output to printer port.

;; COPY-END
L08DE:  CALL    L0207           ; routine SLOW/FAST
        POP     BC              ; *** restore preserved BC.

; -------------------------------------
; THE 'CLEAR PRINTER BUFFER' SUBROUTINE
; -------------------------------------
; This subroutine sets 32 bytes of the printer buffer to zero (space) and
; the 33rd character is set to a NEWLINE.
; This occurs after the printer buffer is sent to the printer but in
addition
; after the 24 lines of the screen are sent to the printer.
; Note. This is a logic error as the last operation does not involve the
; buffer at all. Logically one should be able to use
; 10 LPRINT "HELLO ";
; 20 COPY
; 30 LPRINT ; "WORLD"
; and expect to see the entire greeting emerge from the printer.
; Surprisingly this logic error was never discovered and although one can
argue
; if the above is a bug, the repetition of this error on the Spectrum was
most
; definitely a bug.
; Since the printer buffer is fixed at the end of the system variables, and
; the print position is in the range $3C - $5C, then bit 7 of the system
; variable is set to show the buffer is empty and automatically reset when
; the variable is updated with any print position - neat.

;; CLEAR-PRB
L08E2:  LD      HL,$405C        ; address fixed end of PRBUFF
        LD      (HL),$76        ; place a newline at last position.
        LD      B,$20           ; prepare to blank 32 preceding characters.

;; PRB-BYTES
L08E9:  DEC     HL              ; decrement address - could be DEC L.
        LD      (HL),$00        ; place a zero byte.
        DJNZ    L08E9           ; loop for all thirty-two to PRB-BYTES

        LD      A,L             ; fetch character print position.
        SET     7,A             ; signal the printer buffer is clear.
        LD      ($4038),A       ; update one-byte system variable PR_CC
        RET                     ; return.
```

```
; -------------------------
; THE 'PRINT AT' SUBROUTINE
; -------------------------
;
;

;; PRINT-AT
L08F5:  LD      A,$17           ;
        SUB     B               ;
        JR      C,L0905         ; to WRONG-VAL

;; TEST-VAL
L08FA:  CP      (IY+$22)        ; sv DF_SZ
        JP      C,L0835         ; to REPORT-5

        INC     A               ;
        LD      B,A             ;
        LD      A,$1F           ;
        SUB     C               ;

;; WRONG-VAL
L0905:  JP      C,L0EAD         ; to REPORT-B

        ADD     A,$02           ;
        LD      C,A             ;

;; SET-FIELD
L090B:  BIT     1,(IY+$01)      ; sv FLAGS  - Is printer in use
        JR      Z,L0918         ; to LOC-ADDR

        LD      A,$5D           ;
        SUB     C               ;
        LD      ($4038),A       ; sv PR_CC
        RET                     ;

; ---------------------------
; THE 'LOCATE ADDRESS' ROUTINE
; ---------------------------
;
;

;; LOC-ADDR
L0918:  LD      ($4039),BC      ; sv S_POSN_x
        LD      HL,($4010)      ; sv VARS_lo
        LD      D,C             ;
        LD      A,$22           ;
        SUB     C               ;
        LD      C,A             ;
        LD      A,$76           ;
        INC     B               ;

;; LOOK-BACK
L0927:  DEC     HL              ;
        CP      (HL)            ;
        JR      NZ,L0927        ; to LOOK-BACK

        DJNZ    L0927           ; to LOOK-BACK

        INC     HL              ;
        CPIR                    ;
        DEC     HL              ;
```

```
         LD      ($400E),HL      ; sv DF_CC_lo
         SCF                     ; Set Carry Flag
         RET     PO              ;

         DEC     D               ;
         RET     Z               ;

         PUSH    BC              ;
         CALL    L099E           ; routine MAKE-ROOM
         POP     BC              ;
         LD      B,C             ;
         LD      H,D             ;
         LD       L,E            ;

;; EXPAND-2
L0940:   LD      (HL),$00        ;
         DEC     HL              ;
         DJNZ    L0940           ; to EXPAND-2

         EX      DE,HL           ;
         INC     HL              ;
         LD      ($400E),HL      ; sv DF_CC_lo
         RET                     ;

; ------------------------------
; THE 'EXPAND TOKENS' SUBROUTINE
; ------------------------------
;
;

;; TOKENS
L094B:   PUSH    AF              ;
         CALL    L0975           ; routine TOKEN-ADD
         JR      NC,L0959        ; to ALL-CHARS

         BIT     0,(IY+$01)      ; sv FLAGS  - Leading space if set
         JR      NZ,L0959        ; to ALL-CHARS

         XOR     A               ;

         RST     10H             ; PRINT-A

;; ALL-CHARS
L0959:   LD      A,(BC)          ;
         AND     $3F             ;

         RST     10H             ; PRINT-A
         LD      A,(BC)          ;
         INC     BC              ;
         ADD     A,A             ;
         JR      NC,L0959        ; to ALL-CHARS

         POP     BC              ;
         BIT     7,B             ;
         RET     Z               ;

         CP      $1A             ;
         JR      Z,L096D         ; to TRAIL-SP

         CP      $38             ;
         RET     C               ;
```

```
;; TRAIL-SP
L096D:  XOR     A               ;
        SET     0,(IY+$01)      ; sv FLAGS  - Suppress leading space
        JP      L07F5           ; to PRINT-SP

; ---

;; TOKEN-ADD
L0975:  PUSH    HL              ;
        LD      HL,L0111        ; Address of TOKENS
        BIT     7,A             ;
        JR      Z,L097F         ; to TEST-HIGH

        AND     $3F             ;

;; TEST-HIGH
L097F:  CP      $43             ;
        JR      NC,L0993        ; to FOUND

        LD      B,A             ;
        INC     B               ;

;; WORDS
L0985:  BIT     7,(HL)          ;
        INC     HL              ;
        JR      Z,L0985         ; to WORDS

        DJNZ    L0985           ; to WORDS

        BIT     6,A             ;
        JR      NZ,L0992        ; to COMP-FLAG

        CP      $18             ;

;; COMP-FLAG
L0992:  CCF                     ; Complement Carry Flag

;; FOUND
L0993:  LD      B,H             ;
        LD       C,L            ;
        POP     HL              ;
        RET     NC              ;

        LD      A,(BC)          ;
        ADD     A,$E4           ;
        RET                     ;

; -------------------------
; THE 'ONE SPACE' SUBROUTINE
; -------------------------
;
;

;; ONE-SPACE
L099B:  LD      BC,$0001        ;

; -------------------------
; THE 'MAKE ROOM' SUBROUTINE
; -------------------------
;
;
```

```
;; MAKE-ROOM
L099E:  PUSH    HL              ;
        CALL    L0EC5           ; routine TEST-ROOM
        POP     HL              ;
        CALL    L09AD           ; routine POINTERS
        LD      HL,($401C)      ; sv STKEND_lo
        EX      DE,HL           ;
        LDDR                    ; Copy Bytes
        RET                     ;

; -------------------------
; THE 'POINTERS' SUBROUTINE
; -------------------------
;
;

;; POINTERS
L09AD:  PUSH    AF              ;
        PUSH    HL              ;
        LD      HL,$400C        ; sv D_FILE_lo
        LD      A,$09           ;

;; NEXT-PTR
L09B4:  LD      E,(HL)          ;
        INC     HL              ;
        LD      D,(HL)          ;
        EX      (SP),HL         ;
        AND     A               ;
        SBC     HL,DE           ;
        ADD     HL,DE           ;
        EX      (SP),HL         ;
        JR      NC,L09C8        ; to PTR-DONE

        PUSH    DE              ;
        EX      DE,HL           ;
        ADD     HL,BC           ;
        EX      DE,HL           ;
        LD      (HL),D          ;
        DEC     HL              ;
        LD      (HL),E          ;
        INC     HL              ;
        POP     DE              ;

;; PTR-DONE
L09C8:  INC     HL              ;
        DEC     A               ;
        JR      NZ,L09B4        ; to NEXT-PTR

        EX      DE,HL           ;
        POP     DE              ;
        POP     AF              ;
        AND     A               ;
        SBC     HL,DE           ;
        LD      B,H             ;
        LD      C,L             ;
        INC     BC              ;
        ADD     HL,DE           ;
        EX      DE,HL           ;
        RET                     ;

; ----------------------------
; THE 'LINE ADDRESS' SUBROUTINE
```

```
; ------------------------------
;
;

;; LINE-ADDR
L09D8:  PUSH    HL              ;
        LD      HL,$407D        ;
        LD      D,H             ;
        LD      E,L             ;

;; NEXT-TEST
L09DE:  POP     BC              ;
        CALL    L09EA           ; routine CP-LINES
        RET     NC              ;

        PUSH    BC              ;
        CALL     L09F2          ; routine NEXT-ONE
        EX      DE,HL           ;
        JR      L09DE           ; to NEXT-TEST

; -------------------------------------
; THE 'COMPARE LINE NUMBERS' SUBROUTINE
; -------------------------------------
;
;

;; CP-LINES
L09EA:  LD      A,(HL)          ;
        CP      B               ;
        RET     NZ              ;

        INC     HL              ;
        LD      A,(HL)          ;
        DEC     HL              ;
        CP      C               ;
        RET                     ;

; --------------------------------------
; THE 'NEXT LINE OR VARIABLE' SUBROUTINE
; --------------------------------------
;
;

;; NEXT-ONE
L09F2:  PUSH    HL              ;
        LD      A,(HL)          ;
        CP      $40             ;
        JR      C,L0A0F         ; to LINES

        BIT     5,A             ;
        JR      Z,L0A10         ; forward to NEXT-O-4

        ADD     A,A             ;
        JP      M,L0A01         ; to NEXT+FIVE

        CCF                     ; Complement Carry Flag

;; NEXT+FIVE
L0A01:  LD      BC,$0005        ;
        JR      NC,L0A08        ; to NEXT-LETT

        LD      C,$11           ;
```

```
;; NEXT-LETT
L0A08:  RLA                     ;
        INC     HL              ;
        LD      A,(HL)          ;
        JR      NC,L0A08        ; to NEXT-LETT

        JR      L0A15           ; to NEXT-ADD

; ---

;; LINES
L0A0F:  INC     HL              ;

;; NEXT-O-4
L0A10:  INC     HL              ;
        LD      C,(HL)          ;
        INC     HL              ;
        LD      B,(HL)          ;
        INC     HL              ;

;; NEXT-ADD
L0A15:  ADD     HL,BC           ;
        POP     DE              ;

; ---------------------------
; THE 'DIFFERENCE' SUBROUTINE
; ---------------------------
;
;

;; DIFFER
L0A17:  AND     A               ;
        SBC     HL,DE           ;
        LD      B,H             ;
        LD      C,L             ;
        ADD     HL,DE           ;
        EX      DE,HL           ;
        RET                     ;

; -------------------------
; THE 'LINE-ENDS' SUBROUTINE
; -------------------------
;
;

;; LINE-ENDS
L0A1F:  LD      B,(IY+$22)      ; sv DF_SZ
        PUSH    BC              ;
        CALL    L0A2C           ; routine B-LINES
        POP     BC              ;
        DEC     B               ;
        JR      L0A2C           ; to B-LINES

; ------------------------
; THE 'CLS' COMMAND ROUTINE
; ------------------------
;
;

;; CLS
L0A2A:  LD      B,$18           ;
```

```
;; B-LINES
L0A2C:  RES     1,(IY+$01)      ; sv FLAGS  - Signal printer not in use
        LD      C,$21           ;
        PUSH    BC              ;
        CALL    L0918           ; routine LOC-ADDR
        POP     BC              ;
        LD      A,($4005)       ; sv RAMTOP_hi
        CP      $4D             ;
        JR      C,L0A52         ; to COLLAPSED

        SET     7,(IY+$3A)      ; sv S_POSN_y

;; CLEAR-LOC
L0A42:  XOR     A               ; prepare a space
        CALL    L07F5           ; routine PRINT-SP prints a space
        LD      HL,($4039)      ; sv S_POSN_x
        LD      A,L             ;
        OR      H               ;
        AND     $7E             ;
        JR      NZ,L0A42        ; to CLEAR-LOC

        JP      L0918           ; to LOC-ADDR

; ---

;; COLLAPSED
L0A52:  LD      D,H             ;
        LD      E,L             ;
        DEC     HL              ;
        LD      C,B             ;
        LD      B,$00           ;
        LDIR                    ; Copy Bytes
        LD      HL,($4010)      ; sv VARS_lo

; ----------------------------
; THE 'RECLAIMING' SUBROUTINES
; ----------------------------
;
;

;; RECLAIM-1
L0A5D:  CALL    L0A17           ; routine DIFFER

;; RECLAIM-2
L0A60:  PUSH    BC              ;
        LD      A,B             ;
        CPL                     ;
        LD      B,A             ;
        LD      A,C             ;
        CPL                     ;
        LD      C,A             ;
        INC     BC              ;
        CALL    L09AD           ; routine POINTERS
        EX      DE,HL           ;
        POP     HL              ;
        ADD     HL,DE           ;
        PUSH    DE              ;
        LDIR                    ; Copy Bytes
        POP     HL              ;
        RET                     ;
```

```
; ------------------------------
; THE 'E-LINE NUMBER' SUBROUTINE
; ------------------------------
;
;

;; E-LINE-NO
L0A73:  LD      HL,($4014)      ; sv E_LINE_lo
        CALL    L004D           ; routine TEMP-PTR-2

        RST     18H             ; GET-CHAR
        BIT     5,(IY+$2D)      ; sv FLAGX
        RET     NZ              ;

        LD      HL,$405D        ; sv MEM-0-1st
        LD      ($401C),HL      ; sv STKEND_lo
        CALL    L1548           ; routine INT-TO-FP
        CALL    L158A           ; routine FP-TO-BC
        JR      C,L0A91         ; to NO-NUMBER

        LD      HL,$D8F0        ; value '-10000'
        ADD     HL,BC           ;

;; NO-NUMBER
L0A91:  JP      C,L0D9A         ; to REPORT-C

        CP      A               ;
        JP      L14BC           ; routine SET-MIN

; -------------------------------------------------
; THE 'REPORT AND LINE NUMBER' PRINTING SUBROUTINES
; -------------------------------------------------
;
;

;; OUT-NUM
L0A98:  PUSH    DE              ;
        PUSH    HL              ;
        XOR     A               ;
        BIT     7,B             ;
        JR      NZ,L0ABF        ; to UNITS

        LD       H,B            ;
        LD      L,C             ;
        LD      E,$FF           ;
        JR      L0AAD           ; to THOUSAND

; ---

;; OUT-NO
L0AA5:  PUSH    DE              ;
        LD      D,(HL)          ;
        INC     HL              ;
        LD      E,(HL)          ;
        PUSH    HL              ;
        EX      DE,HL           ;
        LD      E,$00           ; set E to leading space.

;; THOUSAND
L0AAD:  LD      BC,$FC18        ;
        CALL    L07E1           ; routine OUT-DIGIT
        LD      BC,$FF9C        ;
```

```
        CALL    L07E1           ; routine OUT-DIGIT
        LD      C,$F6           ;
        CALL    L07E1           ; routine OUT-DIGIT
        LD      A,L             ;

;; UNITS
L0ABF:  CALL    L07EB           ; routine OUT-CODE
        POP     HL              ;
        POP     DE              ;
        RET                     ;

; -------------------------
; THE 'UNSTACK-Z' SUBROUTINE
; -------------------------
; This subroutine is used to return early from a routine when checking
syntax.
; On the ZX81 the same routines that execute commands also check the syntax
; on line entry. This enables precise placement of the error marker in a
line
; that fails syntax.
; The sequence CALL SYNTAX-Z ; RET Z can be replaced by a call to this
routine
; although it has not replaced every occurrence of the above two
instructions.
; Even on the ZX-80 this routine was not fully utilized.

;; UNSTACK-Z
L0AC5:  CALL    L0DA6           ; routine SYNTAX-Z resets the ZERO flag if
                                ; checking syntax.
        POP     HL              ; drop the return address.
        RET     Z               ; return to previous calling routine if
                                ; checking syntax.

        JP      (HL)            ; else jump to the continuation address in
                                ; the calling routine as RET would have
done.

; ---------------------------
; THE 'LPRINT' COMMAND ROUTINE
; ---------------------------
;
;

;; LPRINT
L0ACB:  SET     1,(IY+$01)      ; sv FLAGS  - Signal printer in use

; -------------------------
; THE 'PRINT' COMMAND ROUTINE
; -------------------------
;
;

;; PRINT
L0ACF:  LD      A,(HL)          ;
        CP      $76             ;
        JP      Z,L0B84         ; to PRINT-END

;; PRINT-1
L0AD5:  SUB     $1A             ;
        ADC     A,$00           ;
        JR      Z,L0B44         ; to SPACING
```

```
         CP       $A7             ;
         JR       NZ,L0AFA        ; to NOT-AT


         RST      20H             ; NEXT-CHAR
         CALL     L0D92           ; routine CLASS-6
         CP       $1A             ;
         JP       NZ,L0D9A        ; to REPORT-C


         RST      20H             ; NEXT-CHAR
         CALL     L0D92           ; routine CLASS-6
         CALL     L0B4E           ; routine SYNTAX-ON

         RST      28H             ;; FP-CALC
         DEFB     $01             ;;exchange
         DEFB     $34             ;;end-calc

         CALL     L0BF5           ; routine STK-TO-BC
         CALL     L08F5           ; routine PRINT-AT
         JR       L0B37           ; to PRINT-ON

; ---

;; NOT-AT
L0AFA:   CP       $A8             ;
         JR       NZ,L0B31        ; to NOT-TAB


         RST      20H             ; NEXT-CHAR
         CALL     L0D92           ; routine CLASS-6
         CALL     L0B4E           ; routine SYNTAX-ON
         CALL     L0C02           ; routine STK-TO-A
         JP       NZ,L0EAD        ; to REPORT-B

         AND      $1F             ;
         LD       C,A             ;
         BIT      1,(IY+$01)      ; sv FLAGS  - Is printer in use
         JR       Z,L0B1E         ; to TAB-TEST

         SUB      (IY+$38)        ; sv PR_CC
         SET      7,A             ;
         ADD      A,$3C           ;
         CALL     NC,L0871        ; routine COPY-BUFF

;; TAB-TEST
L0B1E:   ADD      A,(IY+$39)      ; sv S_POSN_x
         CP       $21             ;
         LD       A,($403A)       ; sv S_POSN_y
         SBC      A,$01           ;
         CALL     L08FA           ; routine TEST-VAL
         SET      0,(IY+$01)      ; sv FLAGS  - Suppress leading space
         JR       L0B37           ; to PRINT-ON

; ---

;; NOT-TAB
L0B31:   CALL     L0F55           ; routine SCANNING
         CALL     L0B55           ; routine PRINT-STK

;; PRINT-ON
L0B37:   RST      18H             ; GET-CHAR
```

```
        SUB     $1A             ;
        ADC     A,$00           ;
        JR      Z,L0B44         ; to SPACING

        CALL    L0D1D           ; routine CHECK-END
        JP      L0B84           ;;; to PRINT-END

; ---

;; SPACING
L0B44:  CALL    NC,L0B8B        ; routine FIELD

        RST     20H             ; NEXT-CHAR
        CP      $76             ;
        RET     Z               ;

        JP      L0AD5           ;;; to PRINT-1

; ---

;; SYNTAX-ON
L0B4E:  CALL    L0DA6           ; routine SYNTAX-Z
        RET     NZ              ;

        POP     HL              ;
        JR      L0B37           ; to PRINT-ON

; ---

;; PRINT-STK
L0B55:  CALL    L0AC5           ; routine UNSTACK-Z
        BIT     6,(IY+$01)      ; sv FLAGS  - Numeric or string result?
        CALL    Z,L13F8         ; routine STK-FETCH
        JR      Z,L0B6B         ; to PR-STR-4

        JP      L15DB           ; jump forward to PRINT-FP

; ---

;; PR-STR-1
L0B64:  LD      A,$0B           ;

;; PR-STR-2
L0B66:  RST     10H             ; PRINT-A

;; PR-STR-3
L0B67:  LD      DE,($4018)      ; sv X_PTR_lo

;; PR-STR-4
L0B6B:  LD      A,B             ;
        OR      C               ;
        DEC     BC              ;
        RET     Z               ;

        LD      A,(DE)          ;
        INC     DE              ;
        LD      ($4018),DE      ; sv X_PTR_lo
        BIT      6,A            ;
        JR      Z,L0B66         ; to PR-STR-2

        CP      $C0             ;
        JR      Z,L0B64         ; to PR-STR-1
```

```
         PUSH    BC              ;
         CALL    L094B           ; routine TOKENS
         POP     BC              ;
         JR      L0B67           ; to PR-STR-3

; ---

;; PRINT-END
L0B84:   CALL    L0AC5           ; routine UNSTACK-Z
         LD      A,$76           ;

         RST     10H             ; PRINT-A
         RET                     ;

; ---

;; FIELD
L0B8B:   CALL    L0AC5           ; routine UNSTACK-Z
         SET     0,(IY+$01)      ; sv FLAGS  - Suppress leading space
         XOR     A               ;

         RST     10H             ; PRINT-A
         LD      BC,($4039)      ; sv S_POSN_x
         LD      A,C             ;
         BIT     1,(IY+$01)      ; sv FLAGS  - Is printer in use
         JR      Z,L0BA4         ; to CENTRE

         LD      A,$5D           ;
         SUB     (IY+$38)        ; sv PR_CC

;; CENTRE
L0BA4:   LD      C,$11           ;
         CP      C               ;
         JR      NC,L0BAB        ; to RIGHT

         LD      C,$01           ;

;; RIGHT
L0BAB:   CALL    L090B           ; routine SET-FIELD
         RET                     ;

; -------------------------------------
; THE 'PLOT AND UNPLOT' COMMAND ROUTINES
; -------------------------------------
;
;

;; PLOT/UNP
L0BAF:   CALL    L0BF5           ; routine STK-TO-BC
         LD      ($4036),BC      ; sv COORDS_x
         LD      A,$2B           ;
         SUB     B               ;
         JP      C,L0EAD         ; to REPORT-B

         LD      B,A             ;
         LD      A,$01           ;
         SRA     B               ;
         JR      NC,L0BC5        ; to COLUMNS

         LD      A,$04           ;
```

```
;; COLUMNS
L0BC5:  SRA     C               ;
        JR      NC,L0BCA        ; to FIND-ADDR

        RLCA                    ;

;; FIND-ADDR
L0BCA:  PUSH    AF              ;
        CALL    L08F5           ; routine PRINT-AT
        LD      A,(HL)          ;
        RLCA                    ;
        CP      $10             ;
        JR      NC,L0BDA        ; to TABLE-PTR

        RRCA                    ;
        JR      NC,L0BD9        ; to SQ-SAVED

        XOR     $8F             ;

;; SQ-SAVED
L0BD9:  LD      B,A             ;

;; TABLE-PTR
L0BDA:  LD      DE,L0C9E        ; Address: P-UNPLOT
        LD      A,($4030)       ; sv T_ADDR_lo
        SUB     E               ;
        JP      M,L0BE9         ; to PLOT

        POP     AF              ;
        CPL                     ;
        AND     B               ;
        JR      L0BEB           ; to UNPLOT

; ---

;; PLOT
L0BE9:  POP     AF              ;
        OR      B               ;

;; UNPLOT
L0BEB:  CP      $08             ;
        JR      C,L0BF1         ; to PLOT-END

        XOR     $8F             ;

;; PLOT-END
L0BF1:  EXX                     ;

        RST     10H             ; PRINT-A
        EXX                     ;
        RET                     ;

; ----------------------------
; THE 'STACK-TO-BC' SUBROUTINE
; ----------------------------
;
;

;; STK-TO-BC
L0BF5:  CALL    L0C02           ; routine STK-TO-A
        LD      B,A             ;
        PUSH    BC              ;
```

```
        CALL    L0C02           ; routine STK-TO-A
        LD      E,C             ;
        POP     BC              ;
        LD      D,C             ;
        LD      C,A             ;
        RET                     ;

; ---------------------------
; THE 'STACK-TO-A' SUBROUTINE
; ---------------------------
;
;

;; STK-TO-A
L0C02:  CALL    L15CD           ; routine FP-TO-A
        JP      C,L0EAD         ; to REPORT-B

        LD      C,$01           ;
        RET     Z               ;

        LD      C,$FF           ;
        RET                     ;

; ------------------------
; THE 'SCROLL' SUBROUTINE
; ------------------------
;
;

;; SCROLL
L0C0E:  LD      B,(IY+$22)      ; sv DF_SZ
        LD      C,$21           ;
        CALL    L0918           ; routine LOC-ADDR
        CALL    L099B           ; routine ONE-SPACE
        LD      A,(HL)          ;
        LD      (DE),A          ;
        INC     (IY+$3A)        ; sv S_POSN_y
        LD      HL,($400C)      ; sv D_FILE_lo
        INC     HL              ;
        LD      D,H             ;
        LD      E,L             ;
        CPIR                    ;
        JP      L0A5D           ; to RECLAIM-1

; -------------------
; THE 'SYNTAX' TABLES
; -------------------

; i) The Offset table

;; offset-t
L0C29:  DEFB    L0CB4 - $       ; 8B offset to; Address: P-LPRINT
        DEFB    L0CB7 - $       ; 8D offset to; Address: P-LLIST
        DEFB    L0C58 - $       ; 2D offset to; Address: P-STOP
        DEFB    L0CAB - $       ; 7F offset to; Address: P-SLOW
        DEFB    L0CAE - $       ; 81 offset to; Address: P-FAST
        DEFB    L0C77 - $       ; 49 offset to; Address: P-NEW
        DEFB    L0CA4 - $       ; 75 offset to; Address: P-SCROLL
        DEFB    L0C8F - $       ; 5F offset to; Address: P-CONT
        DEFB    L0C71 - $       ; 40 offset to; Address: P-DIM
        DEFB    L0C74 - $       ; 42 offset to; Address: P-REM
        DEFB    L0C5E - $       ; 2B offset to; Address: P-FOR
```

```
        DEFB    L0C4B - $       ; 17 offset to; Address: P-GOTO
        DEFB    L0C54 - $       ; 1F offset to; Address: P-GOSUB
        DEFB    L0C6D - $       ; 37 offset to; Address: P-INPUT
        DEFB    L0C89 - $       ; 52 offset to; Address: P-LOAD
        DEFB    L0C7D - $       ; 45 offset to; Address: P-LIST
        DEFB    L0C48 - $       ; 0F offset to; Address: P-LET
        DEFB    L0CA7 - $       ; 6D offset to; Address: P-PAUSE
        DEFB    L0C66 - $       ; 2B offset to; Address: P-NEXT
        DEFB    L0C80 - $       ; 44 offset to; Address: P-POKE
        DEFB    L0C6A - $       ; 2D offset to; Address: P-PRINT
        DEFB    L0C98 - $       ; 5A offset to; Address: P-PLOT
        DEFB    L0C7A - $       ; 3B offset to; Address: P-RUN
        DEFB    L0C8C - $       ; 4C offset to; Address: P-SAVE
        DEFB    L0C86 - $       ; 45 offset to; Address: P-RAND
        DEFB    L0C4F - $       ; 0D offset to; Address: P-IF
        DEFB    L0C95 - $       ; 52 offset to; Address: P-CLS
        DEFB    L0C9E - $       ; 5A offset to; Address: P-UNPLOT
        DEFB    L0C92 - $       ; 4D offset to; Address: P-CLEAR
        DEFB    L0C5B - $       ; 15 offset to; Address: P-RETURN
        DEFB    L0CB1 - $       ; 6A offset to; Address: P-COPY

; ii) The parameter table.


;; P-LET
L0C48:  DEFB    $01             ; Class-01 - A variable is required.
        DEFB    $14             ; Separator:  '='
        DEFB    $02             ; Class-02 - An expression, numeric or
string,
                                ; must follow.

;; P-GOTO
L0C4B:  DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0E81           ; Address: $0E81; Address: GOTO

;; P-IF
L0C4F:  DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $DE             ; Separator:  'THEN'
        DEFB    $05             ; Class-05 - Variable syntax checked
entirely
                                ; by routine.
        DEFW    L0DAB           ; Address: $0DAB; Address: IF

;; P-GOSUB
L0C54:  DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0EB5           ; Address: $0EB5; Address: GOSUB

;; P-STOP
L0C58:  DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0CDC           ; Address: $0CDC; Address: STOP

;; P-RETURN
L0C5B:  DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0ED8           ; Address: $0ED8; Address: RETURN

;; P-FOR
```

```
L0C5E:  DEFB    $04             ; Class-04 - A single character variable
must
                                ; follow.
        DEFB    $14             ; Separator:  '='
        DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $DF             ; Separator:  'TO'
        DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $05             ; Class-05 - Variable syntax checked
entirely
                                ; by routine.
        DEFW    L0DB9           ; Address: $0DB9; Address: FOR

;; P-NEXT
L0C66:  DEFB    $04             ; Class-04 - A single character variable
must
                                ; follow.
        DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0E2E           ; Address: $0E2E; Address: NEXT

;; P-PRINT
L0C6A:  DEFB    $05             ; Class-05 - Variable syntax checked
entirely
                                ; by routine.
        DEFW    L0ACF           ; Address: $0ACF; Address: PRINT

;; P-INPUT
L0C6D:  DEFB    $01             ; Class-01 - A variable is required.
        DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0EE9           ; Address: $0EE9; Address: INPUT

;; P-DIM
L0C71:  DEFB    $05             ; Class-05 - Variable syntax checked
entirely
                                ; by routine.
        DEFW    L1409           ; Address: $1409; Address: DIM

;; P-REM
L0C74:  DEFB    $05             ; Class-05 - Variable syntax checked
entirely
                                ; by routine.
        DEFW    L0D6A           ; Address: $0D6A; Address: REM

;; P-NEW
L0C77:  DEFB    $00             ; Class-00 - No further operands.
        DEFW    L03C3           ; Address: $03C3; Address: NEW

;; P-RUN
L0C7A:  DEFB    $03             ; Class-03 - A numeric expression may
follow
                                ; else default to zero.
        DEFW    L0EAF           ; Address: $0EAF; Address: RUN

;; P-LIST
L0C7D:  DEFB    $03             ; Class-03 - A numeric expression may
follow
                                ; else default to zero.
        DEFW    L0730           ; Address: $0730; Address: LIST

;; P-POKE
```

```
L0C80:  DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $1A             ; Separator:  ','
        DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0E92           ; Address: $0E92; Address: POKE

;; P-RAND
L0C86:  DEFB    $03             ; Class-03 - A numeric expression may
follow
                                ; else default to zero.
        DEFW    L0E6C           ; Address: $0E6C; Address: RAND

;; P-LOAD
L0C89:  DEFB    $05             ; Class-05 - Variable syntax checked
entirely
                                ; by routine.
        DEFW    L0340           ; Address: $0340; Address: LOAD

;; P-SAVE
L0C8C:  DEFB    $05             ; Class-05 - Variable syntax checked
entirely
                                ; by routine.
        DEFW    L02F6           ; Address: $02F6; Address: SAVE

;; P-CONT
L0C8F:  DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0E7C           ; Address: $0E7C; Address: CONT

;; P-CLEAR
L0C92:  DEFB    $00             ; Class-00 - No further operands.
        DEFW    L149A           ; Address: $149A; Address: CLEAR

;; P-CLS
L0C95:  DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0A2A           ; Address: $0A2A; Address: CLS

;; P-PLOT
L0C98:  DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $1A             ; Separator:  ','
        DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0BAF           ; Address: $0BAF; Address: PLOT/UNP

;; P-UNPLOT
L0C9E:  DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $1A             ; Separator:  ','
        DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0BAF           ; Address: $0BAF; Address: PLOT/UNP

;; P-SCROLL
L0CA4:  DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0C0E           ; Address: $0C0E; Address: SCROLL

;; P-PAUSE
```

```
L0CA7:  DEFB    $06             ; Class-06 - A numeric expression must
follow.
        DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0F32           ; Address: $0F32; Address: PAUSE

;; P-SLOW
L0CAB:  DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0F2B           ; Address: $0F2B; Address: SLOW

;; P-FAST
L0CAE:  DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0F23           ; Address: $0F23; Address: FAST

;; P-COPY
L0CB1:  DEFB    $00             ; Class-00 - No further operands.
        DEFW    L0869           ; Address: $0869; Address: COPY

;; P-LPRINT
L0CB4:  DEFB    $05             ; Class-05 - Variable syntax checked
entirely
                                ; by routine.
        DEFW    L0ACB           ; Address: $0ACB; Address: LPRINT

;; P-LLIST
L0CB7:  DEFB    $03             ; Class-03 - A numeric expression may
follow
                                ; else default to zero.
        DEFW    L072C           ; Address: $072C; Address: LLIST


; ---------------------------
; THE 'LINE SCANNING' ROUTINE
; ---------------------------
;
;

;; LINE-SCAN
L0CBA:  LD      (IY+$01),$01    ; sv FLAGS
        CALL    L0A73           ; routine E-LINE-NO

;; LINE-RUN
L0CC1:  CALL    L14BC           ; routine SET-MIN
        LD      HL,$4000        ; sv ERR_NR
        LD      (HL),$FF        ;
        LD      HL,$402D        ; sv FLAGX
        BIT     5,(HL)          ;
        JR      Z,L0CDE         ; to LINE-NULL

        CP      $E3             ; 'STOP' ?
        LD      A,(HL)          ;
        JP      NZ,L0D6F        ; to INPUT-REP

        CALL    L0DA6           ; routine SYNTAX-Z
        RET     Z               ;


        RST     08H             ; ERROR-1
        DEFB    $0C             ; Error Report: BREAK - CONT repeats


; ---------------------------
; THE 'STOP' COMMAND ROUTINE
```

```
; --------------------------
;
;

;; STOP
L0CDC:  RST     08H             ; ERROR-1
        DEFB    $08             ; Error Report: STOP statement

; ---

; the interpretation of a line continues with a check for just spaces
; followed by a carriage return.
; The IF command also branches here with a true value to execute the
; statement after the THEN but the statement can be null so
; 10 IF 1 = 1 THEN
; passes syntax (on all ZX computers).

;; LINE-NULL
L0CDE:  RST     18H             ; GET-CHAR
        LD      B,$00           ; prepare to index - early.
        CP      $76             ; compare to NEWLINE.
        RET     Z               ; return if so.

        LD      C,A             ; transfer character to C.

        RST     20H             ; NEXT-CHAR advances.
        LD      A,C             ; character to A
        SUB     $E1             ; subtract 'LPRINT' - lowest command.
        JR      C,L0D26         ; forward if less to REPORT-C2

        LD      C,A             ; reduced token to C
        LD      HL,L0C29        ; set HL to address of offset table.
        ADD     HL,BC           ; index into offset table.
        LD      C,(HL)          ; fetch offset
        ADD     HL,BC           ; index into parameter table.
        JR      L0CF7           ; to GET-PARAM

; ---

;; SCAN-LOOP
L0CF4:  LD      HL,($4030)      ; sv T_ADDR_lo

; -> Entry Point to Scanning Loop

;; GET-PARAM
L0CF7:  LD      A,(HL)          ;
        INC     HL              ;
        LD      ($4030),HL      ; sv T_ADDR_lo

        LD      BC,L0CF4        ; Address: SCAN-LOOP
        PUSH    BC              ; is pushed on machine stack.

        LD      C,A             ;
        CP      $0B             ;
        JR      NC,L0D10        ; to SEPARATOR

        LD      HL,L0D16        ; class-tbl - the address of the class
table.
        LD      B,$00           ;
        ADD     HL,BC           ;
        LD      C,(HL)          ;
        ADD     HL,BC           ;
```

```
        PUSH    HL              ;

        RST     18H             ; GET-CHAR
        RET                     ; indirect jump to class routine and
                                ; by subsequent RET to SCAN-LOOP.

; -----------------------
; THE 'SEPARATOR' ROUTINE
; -----------------------

;; SEPARATOR
L0D10:  RST     18H             ; GET-CHAR
        CP      C               ;
        JR      NZ,L0D26        ; to REPORT-C2
                                ; 'Nonsense in BASIC'

        RST     20H             ; NEXT-CHAR
        RET                     ; return


; -------------------------
; THE 'COMMAND CLASS' TABLE
; -------------------------
;

;; class-tbl
L0D16:  DEFB    L0D2D - $       ; 17 offset to; Address: CLASS-0
        DEFB    L0D3C - $       ; 25 offset to; Address: CLASS-1
        DEFB    L0D6B - $       ; 53 offset to; Address: CLASS-2
        DEFB    L0D28 - $       ; 0F offset to; Address: CLASS-3
        DEFB    L0D85 - $       ; 6B offset to; Address: CLASS-4
        DEFB    L0D2E - $       ; 13 offset to; Address: CLASS-5
        DEFB    L0D92 - $       ; 76 offset to; Address: CLASS-6


; -------------------------
; THE 'CHECK END' SUBROUTINE
; -------------------------
; Check for end of statement and that no spurious characters occur after
; a correctly parsed statement. Since only one statement is allowed on each
; line, the only character that may follow a statement is a NEWLINE.
;

;; CHECK-END
L0D1D:  CALL    L0DA6           ; routine SYNTAX-Z
        RET     NZ              ; return in runtime.

        POP     BC              ; else drop return address.

;; CHECK-2
L0D22:  LD      A,(HL)          ; fetch character.
        CP      $76             ; compare to NEWLINE.
        RET     Z               ; return if so.

;; REPORT-C2
L0D26:  JR      L0D9A           ; to REPORT-C
                                ; 'Nonsense in BASIC'

; -------------------------
; COMMAND CLASSES 03, 00, 05
; -------------------------
;
```

```
;

;; CLASS-3
L0D28:  CP      $76             ;
        CALL    L0D9C           ; routine NO-TO-STK

;; CLASS-0
L0D2D:  CP      A               ;

;; CLASS-5
L0D2E:  POP     BC              ;
        CALL    Z,L0D1D         ; routine CHECK-END
        EX      DE,HL           ;
        LD      HL,($4030)      ; sv T_ADDR_lo
        LD      C,(HL)          ;
        INC     HL              ;
        LD      B,(HL)          ;
        EX      DE,HL           ;

;; CLASS-END
L0D3A:  PUSH    BC              ;
        RET                     ;

; ------------------------------
; COMMAND CLASSES 01, 02, 04, 06
; ------------------------------
;
;

;; CLASS-1
L0D3C:  CALL    L111C           ; routine LOOK-VARS

;; CLASS-4-2
L0D3F:  LD      (IY+$2D),$00    ; sv FLAGX
        JR      NC,L0D4D        ; to SET-STK

        SET     1,(IY+$2D)      ; sv FLAGX
        JR      NZ,L0D63        ; to SET-STRLN


;; REPORT-2
L0D4B:  RST     08H             ; ERROR-1
        DEFB    $01             ; Error Report: Variable not found

; ---

;; SET-STK
L0D4D:  CALL    Z,L11A7         ; routine STK-VAR
        BIT     6,(IY+$01)      ; sv FLAGS  - Numeric or string result?
        JR      NZ,L0D63        ; to SET-STRLN

        XOR     A               ;
        CALL    L0DA6           ; routine SYNTAX-Z
        CALL    NZ,L13F8        ; routine STK-FETCH
        LD      HL,$402D        ; sv FLAGX
        OR      (HL)            ;
        LD      (HL),A          ;
        EX      DE,HL           ;

;; SET-STRLN
L0D63:  LD      ($402E),BC      ; sv STRLEN_lo
        LD      ($4012),HL      ; sv DEST-lo
```

```
; THE 'REM' COMMAND ROUTINE

;; REM
L0D6A:  RET                     ;

; ---

;; CLASS-2
L0D6B:  POP     BC              ;
        LD      A,($4001)       ; sv FLAGS

;; INPUT-REP
L0D6F:  PUSH    AF              ;
        CALL    L0F55           ; routine SCANNING
        POP     AF              ;
        LD      BC,L1321        ; Address: LET
        LD      D,(IY+$01)      ; sv FLAGS
        XOR     D               ;
        AND     $40             ;
        JR      NZ,L0D9A        ; to REPORT-C

        BIT     7,D             ;
        JR      NZ,L0D3A        ; to CLASS-END

        JR      L0D22           ; to CHECK-2

; ---

;; CLASS-4
L0D85:  CALL    L111C           ; routine LOOK-VARS
        PUSH    AF              ;
        LD      A,C             ;
        OR      $9F             ;
        INC     A               ;
        JR       NZ,L0D9A       ; to REPORT-C

        POP     AF              ;
        JR      L0D3F           ; to CLASS-4-2

; ---

;; CLASS-6
L0D92:  CALL    L0F55           ; routine SCANNING
        BIT     6,(IY+$01)      ; sv FLAGS  - Numeric or string result?
        RET     NZ              ;


;; REPORT-C
L0D9A:  RST     08H             ; ERROR-1
        DEFB    $0B             ; Error Report: Nonsense in BASIC

; --------------------------------
; THE 'NUMBER TO STACK' SUBROUTINE
; --------------------------------
;
;

;; NO-TO-STK
L0D9C:  JR      NZ,L0D92        ; back to CLASS-6 with a non-zero number.

        CALL    L0DA6           ; routine SYNTAX-Z
```

```
        RET     Z               ; return if checking syntax.

; in runtime a zero default is placed on the calculator stack.

        RST     28H             ;; FP-CALC
        DEFB    $A0             ;;stk-zero
        DEFB    $34             ;;end-calc

        RET                     ; return.

; -------------------------
; THE 'SYNTAX-Z' SUBROUTINE
; -------------------------
; This routine returns with zero flag set if checking syntax.
; Calling this routine uses three instruction bytes compared to four if the
; bit test is implemented inline.

;; SYNTAX-Z
L0DA6:  BIT     7,(IY+$01)      ; test FLAGS  - checking syntax only?
        RET                     ; return.

; ------------------------
; THE 'IF' COMMAND ROUTINE
; ------------------------
; In runtime, the class routines have evaluated the test expression and
; the result, true or false, is on the stack.

;; IF
L0DAB:  CALL    L0DA6           ; routine SYNTAX-Z
        JR      Z,L0DB6         ; forward if checking syntax to IF-END

; else delete the Boolean value on the calculator stack.

        RST     28H             ;; FP-CALC
        DEFB    $02             ;;delete
        DEFB    $34             ;;end-calc

; register DE points to exponent of floating point value.

        LD      A,(DE)          ; fetch exponent.
        AND     A               ; test for zero - FALSE.
        RET     Z               ; return if so.

;; IF-END
L0DB6:  JP      L0CDE           ; jump back to LINE-NULL

; -------------------------
; THE 'FOR' COMMAND ROUTINE
; -------------------------
;
;

;; FOR
L0DB9:  CP      $E0             ; is current character 'STEP' ?
        JR      NZ,L0DC6        ; forward if not to F-USE-ONE


        RST     20H             ; NEXT-CHAR
        CALL    L0D92           ; routine CLASS-6 stacks the number
        CALL    L0D1D           ; routine CHECK-END
        JR      L0DCC           ; forward to F-REORDER
```

```
; ---

;; F-USE-ONE
L0DC6:  CALL    L0D1D           ; routine CHECK-END

        RST     28H             ;; FP-CALC
        DEFB    $A1             ;;stk-one
        DEFB    $34             ;;end-calc



;; F-REORDER
L0DCC:  RST     28H             ;; FP-CALC      v, l, s.
        DEFB    $C0             ;;st-mem-0      v, l, s.
        DEFB    $02             ;;delete        v, l.
        DEFB    $01             ;;exchange      l, v.
        DEFB    $E0             ;;get-mem-0     l, v, s.
        DEFB    $01             ;;exchange      l, s, v.
        DEFB    $34             ;;end-calc      l, s, v.

        CALL    L1321           ; routine LET

        LD      ($401F),HL      ; set MEM to address variable.
        DEC     HL              ; point to letter.
        LD      A,(HL)          ;
        SET     7,(HL)          ;
        LD      BC,$0006        ;
        ADD     HL,BC           ;
        RLCA                    ;
        JR      C,L0DEA         ; to F-LMT-STP

        SLA     C               ;
        CALL    L099E           ; routine MAKE-ROOM
        INC     HL              ;

;; F-LMT-STP
L0DEA:  PUSH    HL              ;

        RST     28H             ;; FP-CALC
        DEFB    $02             ;;delete
        DEFB    $02             ;;delete
        DEFB    $34             ;;end-calc

        POP     HL              ;
        EX      DE,HL           ;

        LD      C,$0A           ; ten bytes to be moved.
        LDIR                    ; copy bytes

        LD      HL,($4007)      ; set HL to system variable PPC current
line.
        EX      DE,HL           ; transfer to DE, variable pointer to HL.
        INC     DE              ; loop start will be this line + 1 at
least.
        LD      (HL),E          ;
        INC     HL              ;
        LD      (HL),D          ;
        CALL    L0E5A           ; routine NEXT-LOOP considers an initial
pass.
        RET     NC              ; return if possible.

; else program continues from point following matching NEXT.
```

```
        BIT     7,(IY+$08)      ; test PPC_hi
        RET     NZ              ; return if over 32767 ???

        LD      B,(IY+$2E)      ; fetch variable name from STRLEN_lo
        RES     6,B             ; make a true letter.
        LD      HL,($4029)      ; set HL from NXTLIN

; now enter a loop to look for matching next.

;; NXTLIN-NO
L0E0E:  LD      A,(HL)          ; fetch high byte of line number.
        AND     $C0             ; mask off low bits $3F
        JR      NZ,L0E2A        ; forward at end of program to FOR-END

        PUSH    BC              ; save letter
        CALL    L09F2           ; routine NEXT-ONE finds next line.
        POP     BC              ; restore letter

        INC     HL              ; step past low byte
        INC     HL              ; past the
        INC     HL              ; line length.
        CALL    L004C           ; routine TEMP-PTR1 sets CH_ADD

        RST     18H             ; GET-CHAR
        CP      $F3             ; compare to 'NEXT'.
        EX      DE,HL           ; next line to HL.
        JR      NZ,L0E0E        ; back with no match to NXTLIN-NO

;

        EX      DE,HL           ; restore pointer.

        RST     20H             ; NEXT-CHAR advances and gets letter in A.
        EX      DE,HL           ; save pointer
        CP      B               ; compare to variable name.
        JR      NZ,L0E0E        ; back with mismatch to NXTLIN-NO

;; FOR-END
L0E2A:  LD      ($4029),HL      ; update system variable NXTLIN
        RET                     ; return.

; --------------------------
; THE 'NEXT' COMMAND ROUTINE
; --------------------------
;
;

;; NEXT
L0E2E:  BIT     1,(IY+$2D)      ; sv FLAGX
        JP      NZ,L0D4B        ; to REPORT-2

        LD      HL,($4012)      ; DEST
        BIT     7,(HL)          ;
        JR      Z,L0E58         ; to REPORT-1

        INC     HL              ;
        LD      ($401F),HL      ; sv MEM_lo

        RST     28H             ;; FP-CALC
        DEFB    $E0             ;;get-mem-0
        DEFB    $E2             ;;get-mem-2
```

```
        DEFB    $0F             ;;addition
        DEFB    $C0             ;;st-mem-0
        DEFB    $02             ;;delete
        DEFB    $34             ;;end-calc

        CALL    L0E5A           ; routine NEXT-LOOP
        RET     C               ;

        LD      HL,($401F)      ; sv MEM_lo
        LD      DE,$000F        ;
        ADD     HL,DE           ;
        LD      E,(HL)          ;
        INC     HL              ;
        LD      D,(HL)          ;
        EX      DE,HL           ;
        JR      L0E86           ; to GOTO-2

; ---


;; REPORT-1
L0E58:  RST     08H             ; ERROR-1
        DEFB    $00             ; Error Report: NEXT without FOR


; --------------------------
; THE 'NEXT-LOOP' SUBROUTINE
; --------------------------
;
;

;; NEXT-LOOP
L0E5A:  RST     28H             ;; FP-CALC
        DEFB    $E1             ;;get-mem-1
        DEFB    $E0             ;;get-mem-0
        DEFB    $E2             ;;get-mem-2
        DEFB    $32             ;;less-0
        DEFB    $00             ;;jump-true
        DEFB    $02             ;;to L0E62, LMT-V-VAL

        DEFB    $01             ;;exchange

;; LMT-V-VAL
L0E62:  DEFB    $03             ;;subtract
        DEFB    $33             ;;greater-0
        DEFB    $00             ;;jump-true
        DEFB    $04             ;;to L0E69, IMPOSS

        DEFB    $34             ;;end-calc

        AND     A               ; clear carry flag
        RET                     ; return.

; ---


;; IMPOSS
L0E69:  DEFB    $34             ;;end-calc

        SCF                     ; set carry flag
        RET                     ; return.
```

```
; -------------------------
; THE 'RAND' COMMAND ROUTINE
; -------------------------
; The keyword was 'RANDOMISE' on the ZX80, is 'RAND' here on the ZX81 and
; becomes 'RANDOMIZE' on the ZX Spectrum.
; In all invocations the procedure is the same - to set the SEED system
variable
; with a supplied integer value or to use a time-based value if no number,
or
; zero, is supplied.

;; RAND
L0E6C:  CALL    L0EA7           ; routine FIND-INT
        LD      A,B             ; test value
        OR      C               ; for zero
        JR      NZ,L0E77        ; forward if not zero to SET-SEED

        LD      BC,($4034)      ; fetch value of FRAMES system variable.

;; SET-SEED
L0E77:  LD       ($4032),BC     ; update the SEED system variable.
        RET                     ; return.

; -------------------------
; THE 'CONT' COMMAND ROUTINE
; -------------------------
; Another abbreviated command. ROM space was really tight.
; CONTINUE at the line number that was set when break was pressed.
; Sometimes the current line, sometimes the next line.

;; CONT
L0E7C:  LD      HL,($402B)      ; set HL from system variable OLDPPC
        JR      L0E86           ; forward to GOTO-2

; -------------------------
; THE 'GOTO' COMMAND ROUTINE
; -------------------------
; This token also suffered from the shortage of room and there is no space
; getween GO and TO as there is on the ZX80 and ZX Spectrum. The same also
; applies to the GOSUB keyword.

;; GOTO
L0E81:  CALL    L0EA7           ; routine FIND-INT
        LD      H,B             ;
        LD      L,C             ;

;; GOTO-2
L0E86:  LD      A,H             ;
        CP      $F0             ;
        JR      NC,L0EAD        ; to REPORT-B

        CALL    L09D8           ; routine LINE-ADDR
        LD      ($4029),HL      ; sv NXTLIN_lo
        RET                     ;

; -------------------------
; THE 'POKE' COMMAND ROUTINE
; -------------------------
;
;

;; POKE
```

```
L0E92:  CALL    L15CD           ; routine FP-TO-A
        JR      C,L0EAD         ; forward, with overflow, to REPORT-B

        JR      Z,L0E9B         ; forward, if positive, to POKE-SAVE

        NEG                     ; negate

;; POKE-SAVE
L0E9B:  PUSH    AF              ; preserve value.
        CALL    L0EA7           ; routine FIND-INT gets address in BC
                                ; invoking the error routine with overflow
                                ; or a negative number.
        POP     AF              ; restore value.

; Note. the next two instructions are legacy code from the ZX80 and
; inappropriate here.

        BIT     7,(IY+$00)      ; test ERR_NR - is it still $FF ?
        RET     Z               ; return with error.

        LD      (BC),A          ; update the address contents.
        RET                     ; return.

; ------------------------------
; THE 'FIND INTEGER' SUBROUTINE
; ------------------------------
;
;

;; FIND-INT
L0EA7:  CALL    L158A           ; routine FP-TO-BC
        JR      C,L0EAD         ; forward with overflow to REPORT-B

        RET     Z               ; return if positive (0-65535).


;; REPORT-B
L0EAD:  RST     08H             ; ERROR-1
        DEFB    $0A             ; Error Report: Integer out of range

; -------------------------
; THE 'RUN' COMMAND ROUTINE
; -------------------------
;
;

;; RUN
L0EAF:  CALL    L0E81           ; routine GOTO
        JP      L149A           ; to CLEAR

; ---------------------------
; THE 'GOSUB' COMMAND ROUTINE
; ---------------------------
;
;

;; GOSUB
L0EB5:  LD      HL,($4007)      ; sv PPC_lo
        INC     HL              ;
        EX      (SP),HL         ;
        PUSH    HL              ;
        LD      ($4002),SP      ; set the error stack pointer - ERR_SP
```

```
        CALL    L0E81           ; routine GOTO
        LD      BC,$0006        ;

; -------------------------
; THE 'TEST ROOM' SUBROUTINE
; -------------------------
;
;

;; TEST-ROOM
L0EC5:  LD      HL,($401C)      ; sv STKEND_lo
        ADD     HL,BC           ;
        JR      C,L0ED3         ; to REPORT-4

        EX      DE,HL           ;
        LD      HL,$0024        ;
        ADD     HL,DE           ;
        SBC     HL,SP           ;
        RET     C               ;

;; REPORT-4
L0ED3:  LD      L,$03           ;
        JP      L0058           ; to ERROR-3

; ----------------------------
; THE 'RETURN' COMMAND ROUTINE
; ----------------------------
;
;

;; RETURN
L0ED8:  POP     HL              ;
        EX      (SP),HL         ;
        LD      A,H             ;
        CP      $3E             ;
        JR      Z,L0EE5         ; to REPORT-7

        LD      ($4002),SP      ; sv ERR_SP_lo
        JR      L0E86           ; back to GOTO-2

; ---

;; REPORT-7
L0EE5:  EX      (SP),HL         ;
        PUSH    HL              ;

        RST     08H             ; ERROR-1
        DEFB    $06             ; Error Report: RETURN without GOSUB

; ---------------------------
; THE 'INPUT' COMMAND ROUTINE
; ---------------------------
;
;

;; INPUT
L0EE9:  BIT     7,(IY+$08)      ; sv PPC_hi
        JR      NZ,L0F21        ; to REPORT-8

        CALL    L14A3           ; routine X-TEMP
        LD      HL,$402D        ; sv FLAGX
        SET     5,(HL)          ;
```

```
        RES     6,(HL)          ;
        LD      A,($4001)       ; sv FLAGS
        AND     $40             ;
        LD      BC,$0002        ;
        JR      NZ,L0F05        ; to PROMPT

        LD      C,$04           ;

;; PROMPT
L0F05:  OR      (HL)            ;
        LD      (HL),A          ;

        RST     30H             ; BC-SPACES
        LD      (HL),$76        ;
        LD      A,C             ;
        RRCA                    ;
        RRCA                    ;
        JR      C,L0F14         ; to ENTER-CUR

        LD      A,$0B           ;
        LD      (DE),A          ;
        DEC     HL              ;
        LD      (HL),A          ;

;; ENTER-CUR
L0F14:  DEC     HL              ;
        LD      (HL),$7F        ;
        LD      HL,($4039)      ; sv S_POSN_x
        LD      ($4030),HL      ; sv T_ADDR_lo
        POP     HL              ;
        JP      L0472           ; to LOWER

; ---

;; REPORT-8
L0F21:  RST     08H             ; ERROR-1
        DEFB    $07             ; Error Report: End of file

; ---------------------------
; THE 'PAUSE' COMMAND ROUTINE
; ---------------------------
;
;

;; FAST
L0F23:  CALL    L02E7           ; routine SET-FAST
        RES     6,(IY+$3B)      ; sv CDFLAG
        RET                     ; return.

; ---------------------------
; THE 'SLOW' COMMAND ROUTINE
; ---------------------------
;
;

;; SLOW
L0F2B:  SET     6,(IY+$3B)      ; sv CDFLAG
        JP      L0207           ; to SLOW/FAST

; ---------------------------
; THE 'PAUSE' COMMAND ROUTINE
; ---------------------------
```

```
;; PAUSE
L0F32:  CALL    L0EA7           ; routine FIND-INT
        CALL    L02E7           ; routine SET-FAST
        LD      H,B             ;
        LD      L,C             ;
        CALL    L022D           ; routine DISPLAY-P

        LD      (IY+$35),$FF    ; sv FRAMES_hi

        CALL    L0207           ; routine SLOW/FAST
        JR      L0F4B           ; routine DEBOUNCE

; ----------------------
; THE 'BREAK' SUBROUTINE
; ----------------------
;
;

;; BREAK-1
L0F46:  LD      A,$7F           ; read port $7FFE - keys B,N,M,.,SPACE.
        IN      A,($FE)         ;
        RRA                     ; carry will be set if space not pressed.

; -------------------------
; THE 'DEBOUNCE' SUBROUTINE
; -------------------------
;
;

;; DEBOUNCE
L0F4B:  RES     0,(IY+$3B)      ; update system variable CDFLAG
        LD      A,$FF           ;
        LD      ($4027),A       ; update system variable DEBOUNCE
        RET                     ; return.


; -------------------------
; THE 'SCANNING' SUBROUTINE
; -------------------------
; This recursive routine is where the ZX81 gets its power. Provided there
is
; enough memory it can evaluate an expression of unlimited complexity.
; Note. there is no unary plus so, as on the ZX80, PRINT +1 gives a syntax
error.
; PRINT +1 works on the Spectrum but so too does PRINT + "STRING".

;; SCANNING
L0F55:  RST     18H             ; GET-CHAR
        LD      B,$00           ; set B register to zero.
        PUSH    BC              ; stack zero as a priority end-marker.

;; S-LOOP-1
L0F59:  CP      $40             ; compare to the 'RND' character
        JR      NZ,L0F8C        ; forward, if not, to S-TEST-PI

; ------------------
; THE 'RND' FUNCTION
; ------------------

        CALL    L0DA6           ; routine SYNTAX-Z
        JR      Z,L0F8A         ; forward if checking syntax to S-JPI-END
```

```
        LD      BC,($4032)      ; sv SEED_lo
        CALL    L1520           ; routine STACK-BC

        RST     28H             ;; FP-CALC
        DEFB    $A1             ;;stk-one
        DEFB    $0F             ;;addition
        DEFB    $30             ;;stk-data
        DEFB    $37             ;;Exponent: $87, Bytes: 1
        DEFB    $16             ;;(+00,+00,+00)
        DEFB    $04             ;;multiply
        DEFB    $30             ;;stk-data
        DEFB    $80             ;;Bytes: 3
        DEFB    $41             ;;Exponent $91
        DEFB    $00,$00,$80     ;;(+00)
        DEFB    $2E             ;;n-mod-m
        DEFB    $02             ;;delete
        DEFB    $A1             ;;stk-one
        DEFB    $03             ;;subtract
        DEFB    $2D             ;;duplicate
        DEFB    $34             ;;end-calc

        CALL    L158A           ; routine FP-TO-BC
        LD      ($4032),BC      ; update the SEED system variable.
        LD      A,(HL)          ; HL addresses the exponent of the last
value.
        AND     A               ; test for zero
        JR      Z,L0F8A         ; forward, if so, to S-JPI-END

        SUB     $10             ; else reduce exponent by sixteen
        LD      (HL),A          ; thus dividing by 65536 for last value.

;; S-JPI-END
L0F8A:  JR      L0F99           ; forward to S-PI-END

; ---

;; S-TEST-PI
L0F8C:  CP      $42             ; the 'PI' character
        JR      NZ,L0F9D        ; forward, if not, to S-TST-INK

; -------------------
; THE 'PI' EVALUATION
; -------------------

        CALL    L0DA6           ; routine SYNTAX-Z
        JR      Z,L0F99         ; forward if checking syntax to S-PI-END


        RST     28H             ;; FP-CALC
        DEFB    $A3             ;;stk-pi/2
        DEFB    $34             ;;end-calc

        INC     (HL)            ; double the exponent giving PI on the
stack.

;; S-PI-END
L0F99:  RST     20H             ; NEXT-CHAR advances character pointer.

        JP      L1083           ; jump forward to S-NUMERIC to set the flag
                                ; to signal numeric result before
advancing.
```

```
; ---

;; S-TST-INK
L0F9D:  CP      $41             ; compare to character 'INKEY$'
        JR      NZ,L0FB2        ; forward, if not, to S-ALPHANUM

; -----------------------
; THE 'INKEY$' EVALUATION
; -----------------------

        CALL    L02BB           ; routine KEYBOARD
        LD      B,H             ;
        LD      C,L             ;
        LD      D,C             ;
        INC     D               ;
        CALL    NZ,L07BD        ; routine DECODE
        LD      A,D             ;
        ADC     A,D             ;
        LD      B,D             ;
        LD      C,A             ;
        EX      DE,HL           ;
        JR      L0FED           ; forward to S-STRING

; ---

;; S-ALPHANUM
L0FB2:  CALL    L14D2           ; routine ALPHANUM
        JR      C,L1025         ; forward, if alphanumeric to S-LTR-DGT

        CP      $1B             ; is character a '.' ?
        JP      Z,L1047         ; jump forward if so to S-DECIMAL

        LD      BC,$09D8        ; prepare priority 09, operation 'subtract'
        CP      $16             ; is character unary minus '-' ?
        JR      Z,L1020         ; forward, if so, to S-PUSH-PO

        CP      $10             ; is character a '(' ?
        JR      NZ,L0FD6        ; forward if not to S-QUOTE

        CALL    L0049           ; routine CH-ADD+1 advances character
pointer.

        CALL    L0F55           ; recursively call routine SCANNING to
                                ; evaluate the sub-expression.

        CP      $11             ; is subsequent character a ')' ?
        JR      NZ,L0FFF        ; forward if not to S-RPT-C


        CALL    L0049           ; routine CH-ADD+1  advances.
        JR      L0FF8           ; relative jump to S-JP-CONT3 and then S-
CONT3

; ---

; consider a quoted string e.g. PRINT "Hooray!"
; Note. quotes are not allowed within a string.

;; S-QUOTE
L0FD6:  CP      $0B             ; is character a quote (") ?
        JR      NZ,L1002        ; forward, if not, to S-FUNCTION
```

```
        CALL    L0049           ; routine CH-ADD+1 advances
        PUSH    HL              ; * save start of string.
        JR      L0FE3           ; forward to S-QUOTE-S

; ---


;; S-Q-AGAIN
L0FE0:  CALL    L0049           ; routine CH-ADD+1

;; S-QUOTE-S
L0FE3:  CP      $0B             ; is character a '"' ?
        JR      NZ,L0FFB        ; forward if not to S-Q-NL

        POP     DE              ; * retrieve start of string
        AND     A               ; prepare to subtract.
        SBC     HL,DE           ; subtract start from current position.
        LD      B,H             ; transfer this length
        LD      C,L             ; to the BC register pair.

;; S-STRING
L0FED:  LD      HL,$4001        ; address system variable FLAGS
        RES     6,(HL)          ; signal string result
        BIT     7,(HL)          ; test if checking syntax.

        CALL    NZ,L12C3        ; in run-time routine STK-STO-$ stacks the
                                ; string descriptor - start DE, length BC.

        RST     20H             ; NEXT-CHAR advances pointer.

;; S-J-CONT-3
L0FF8:  JP      L1088           ; jump to S-CONT-3

; ---

; A string with no terminating quote has to be considered.

;; S-Q-NL
L0FFB:  CP      $76             ; compare to NEWLINE
        JR      NZ,L0FE0        ; loop back if not to S-Q-AGAIN

;; S-RPT-C
L0FFF:  JP      L0D9A           ; to REPORT-C

; ---

;; S-FUNCTION
L1002:  SUB     $C4             ; subtract 'CODE' reducing codes
                                ; CODE thru '<>' to range $00 - $XX
        JR      C,L0FFF         ; back, if less, to S-RPT-C

; test for NOT the last function in character set.

        LD      BC,$04EC        ; prepare priority $04, operation 'not'
        CP      $13             ; compare to 'NOT'  ( - CODE)
        JR      Z,L1020         ; forward, if so, to S-PUSH-PO

        JR      NC,L0FFF        ; back with anything higher to S-RPT-C

; else is a function 'CODE' thru 'CHR$'
```

```
        LD      B,$10           ; priority sixteen binds all functions to
                                ; arguments removing the need for brackets.

        ADD     A,$D9           ; add $D9 to give range $D9 thru $EB
                                ; bit 6 is set to show numeric argument.
                                ; bit 7 is set to show numeric result.

; now adjust these default argument/result indicators.

        LD      C,A             ; save code in C

        CP      $DC             ; separate 'CODE', 'VAL', 'LEN'
        JR      NC,L101A        ; skip forward if string operand to S-NO-
TO-$

        RES     6,C             ; signal string operand.

;; S-NO-TO-$
L101A:  CP      $EA             ; isolate top of range 'STR$' and 'CHR$'
        JR      C,L1020         ; skip forward with others to S-PUSH-PO

        RES     7,C             ; signal string result.

;; S-PUSH-PO
L1020:  PUSH    BC              ; push the priority/operation

        RST     20H             ; NEXT-CHAR
        JP      L0F59           ; jump back to S-LOOP-1

; ---

;; S-LTR-DGT
L1025:  CP      $26             ; compare to 'A'.
        JR      C,L1047         ; forward if less to S-DECIMAL

        CALL    L111C           ; routine LOOK-VARS
        JP      C,L0D4B         ; back if not found to REPORT-2
                                ; a variable is always 'found' when
checking
                                ; syntax.

        CALL    Z,L11A7         ; routine STK-VAR stacks string parameters
or
                                ; returns cell location if numeric.

        LD      A,($4001)       ; fetch FLAGS
        CP      $C0             ; compare to numeric result/numeric operand
        JR      C,L1087         ; forward if not numeric to S-CONT-2

        INC     HL              ; address numeric contents of variable.
        LD      DE,($401C)      ; set destination to STKEND
        CALL    L19F6           ; routine MOVE-FP stacks the five bytes
        EX      DE,HL           ; transfer new free location from DE to HL.
        LD      ($401C),HL      ; update STKEND system variable.
        JR      L1087           ; forward to S-CONT-2

; ---

; The Scanning Decimal routine is invoked when a decimal point or digit is
; found in the expression.
; When checking syntax, then the 'hidden floating point' form is placed
; after the number in the BASIC line.
```

```
; In run-time, the digits are skipped and the floating point number is
picked
; up.

;; S-DECIMAL
L1047:  CALL    L0DA6           ; routine SYNTAX-Z
        JR      NZ,L106F        ; forward in run-time to S-STK-DEC

        CALL    L14D9           ; routine DEC-TO-FP

        RST     18H             ; GET-CHAR advances HL past digits
        LD      BC,$0006        ; six locations are required.
        CALL    L099E           ; routine MAKE-ROOM
        INC     HL              ; point to first new location
        LD      (HL),$7E        ; insert the number marker 126 decimal.
        INC     HL              ; increment
        EX      DE,HL           ; transfer destination to DE.
        LD      HL,($401C)      ; set HL from STKEND which points to the
                                ; first location after the 'last value'
        LD      C,$05           ; five bytes to move.
        AND     A               ; clear carry.
        SBC     HL,BC           ; subtract five pointing to 'last value'.
        LD      ($401C),HL      ; update STKEND thereby 'deleting the
value.

        LDIR                    ; copy the five value bytes.

        EX      DE,HL           ; basic pointer to HL which may be white-
space
                                ; following the number.
        DEC     HL              ; now points to last of five bytes.
        CALL    L004C           ; routine TEMP-PTR1 advances the character
                                ; address skipping any white-space.
        JR      L1083           ; forward to S-NUMERIC
                                ; to signal a numeric result.

; ---

; In run-time the branch is here when a digit or point is encountered.

;; S-STK-DEC
L106F:  RST     20H             ; NEXT-CHAR
        CP      $7E             ; compare to 'number marker'
        JR      NZ,L106F        ; loop back until found to S-STK-DEC
                                ; skipping all the digits.

        INC     HL              ; point to first of five hidden bytes.
        LD      DE,($401C)      ; set destination from STKEND system
variable
        CALL    L19F6           ; routine MOVE-FP stacks the number.
        LD      ($401C),DE      ; update system variable STKEND.
        LD      ($4016),HL      ; update system variable CH_ADD.

;; S-NUMERIC
L1083:  SET     6,(IY+$01)      ; update FLAGS  - Signal numeric result

;; S-CONT-2
L1087:  RST     18H             ; GET-CHAR

;; S-CONT-3
L1088:  CP      $10             ; compare to opening bracket '('
        JR      NZ,L1098        ; forward if not to S-OPERTR
```

```
        BIT     6,(IY+$01)      ; test FLAGS  - Numeric or string result?
        JR      NZ,L10BC        ; forward if numeric to S-LOOP

; else is a string

        CALL    L1263           ; routine SLICING

        RST     20H             ; NEXT-CHAR
        JR      L1088           ; back to S-CONT-3

; ---

; the character is now manipulated to form an equivalent in the table of
; calculator literals. This is quite cumbersome and in the ZX Spectrum a
; simple look-up table was introduced at this point.

;; S-OPERTR
L1098:  LD      BC,$00C3        ; prepare operator 'subtract' as default.
                                ; also set B to zero for later indexing.

        CP      $12             ; is character '>' ?
        JR      C,L10BC         ; forward if less to S-LOOP as
                                ; we have reached end of meaningful
expression

        SUB     $16             ; is character '-' ?
        JR      NC,L10A7        ; forward with - * / and '**' '<>' to
SUBMLTDIV

        ADD     A,$0D           ; increase others by thirteen
                                ; $09 '>' thru $0C '+'
        JR      L10B5           ; forward to GET-PRIO

; ---

;; SUBMLTDIV
L10A7:  CP      $03             ; isolate $00 '-', $01 '*', $02 '/'
        JR      C,L10B5         ; forward if so to GET-PRIO

; else possibly originally $D8 '**' thru $DD '<>' already reduced by $16

        SUB     $C2             ; giving range $00 to $05
        JR      C,L10BC         ; forward if less to S-LOOP

        CP      $06             ; test the upper limit for nonsense also
        JR      NC,L10BC        ; forward if so to S-LOOP

        ADD     A,$03           ; increase by 3 to give combined operators
of

                                ; $00 '-'
                                ; $01 '*'
                                ; $02 '/'

                                ; $03 '**'
                                ; $04 'OR'
                                ; $05 'AND'
                                ; $06 '<='
                                ; $07 '>='
                                ; $08 '<>'
```

```
                                ; $09 '>'
                                ; $0A '<'
                                ; $0B '='
                                ; $0C '+'

;; GET-PRIO
L10B5:  ADD     A,C             ; add to default operation 'sub' ($C3)
        LD      C,A             ; and place in operator byte - C.

        LD      HL,L110F - $C3  ; theoretical base of the priorities table.
        ADD     HL,BC           ; add C ( B is zero)
        LD      B,(HL)          ; pick up the priority in B

;; S-LOOP
L10BC:  POP     DE              ; restore previous
        LD      A,D             ; load A with priority.
        CP      B               ; is present priority higher
        JR      C,L10ED         ; forward if so to S-TIGHTER

        AND     A               ; are both priorities zero
        JP      Z,L0018         ; exit if zero via GET-CHAR

        PUSH    BC              ; stack present values
        PUSH    DE              ; stack last values
        CALL    L0DA6           ; routine SYNTAX-Z
        JR      Z,L10D5         ; forward is checking syntax to S-SYNTEST

        LD      A,E             ; fetch last operation
        AND     $3F             ; mask off the indicator bits to give true
                                ; calculator literal.
        LD      B,A             ; place in the B register for BREG

; perform the single operation

        RST     28H             ;; FP-CALC
        DEFB    $37             ;;fp-calc-2
        DEFB    $34             ;;end-calc

        JR      L10DE           ; forward to S-RUNTEST

; ---

;; S-SYNTEST
L10D5:  LD      A,E             ; transfer masked operator to A
        XOR     (IY+$01)        ; XOR with FLAGS like results will reset
bit 6
        AND     $40             ; test bit 6

;; S-RPORT-C
L10DB:  JP      NZ,L0D9A        ; back to REPORT-C if results do not agree.

; ---

; in run-time impose bit 7 of the operator onto bit 6 of the FLAGS

;; S-RUNTEST
L10DE:  POP     DE              ; restore last operation.
        LD      HL,$4001        ; address system variable FLAGS
        SET     6,(HL)          ; presume a numeric result
        BIT     7,E             ; test expected result in operation
        JR      NZ,L10EA        ; forward if numeric to S-LOOPEND
```

```
        RES     6,(HL)          ; reset to signal string result

;; S-LOOPEND
L10EA:  POP     BC              ; restore present values
        JR      L10BC           ; back to S-LOOP

; ---

;; S-TIGHTER
L10ED:  PUSH    DE              ; push last values and consider these

        LD      A,C             ; get the present operator.
        BIT     6,(IY+$01)      ; test FLAGS  - Numeric or string result?
        JR      NZ,L110A        ; forward if numeric to S-NEXT

        AND     $3F             ; strip indicator bits to give clear
literal.
        ADD     A,$08           ; add eight - augmenting numeric to
equivalent
                                ; string literals.
        LD      C,A             ; place plain literal back in C.
        CP      $10             ; compare to 'AND'
        JR      NZ,L1102        ; forward if not to S-NOT-AND

        SET     6,C             ; set the numeric operand required for
'AND'
        JR      L110A           ; forward to S-NEXT

; ---

;; S-NOT-AND
L1102:  JR      C,L10DB         ; back if less than 'AND' to S-RPORT-C
                                ; Nonsense if '-', '*' etc.

        CP      $17             ; compare to 'strs-add' literal
        JR      Z,L110A         ; forward if so signaling string result

        SET     7,C             ; set bit to numeric (Boolean) for others.

;; S-NEXT
L110A:  PUSH    BC              ; stack 'present' values

        RST     20H             ; NEXT-CHAR
        JP      L0F59           ; jump back to S-LOOP-1



; -------------------------
; THE 'TABLE OF PRIORITIES'
; -------------------------
;
;

;; tbl-pri
L110F:  DEFB    $06             ;       '-'
        DEFB    $08             ;       '*'
        DEFB    $08             ;       '/'
        DEFB    $0A             ;       '**'
        DEFB    $02             ;       'OR'
        DEFB    $03             ;       'AND'
        DEFB    $05             ;       '<='
        DEFB    $05             ;       '>='
```

```
        DEFB    $05             ;       '<>'
        DEFB    $05             ;       '>'
        DEFB    $05             ;       '<'
        DEFB    $05             ;       '='
        DEFB    $06             ;       '+'


; --------------------------
; THE 'LOOK-VARS' SUBROUTINE
; --------------------------
;
;

;; LOOK-VARS
L111C:  SET     6,(IY+$01)      ; sv FLAGS  - Signal numeric result

        RST     18H             ; GET-CHAR
        CALL    L14CE           ; routine ALPHA
        JP      NC,L0D9A        ; to REPORT-C

        PUSH    HL              ;
        LD      C,A             ;

        RST     20H             ; NEXT-CHAR
        PUSH    HL              ;
        RES     5,C             ;
        CP      $10             ;
        JR      Z,L1148         ; to V-SYN/RUN

        SET     6,C             ;
        CP      $0D             ;
        JR      Z,L1143         ; forward to V-STR-VAR

        SET     5,C             ;

;; V-CHAR
L1139:  CALL    L14D2           ; routine ALPHANUM
        JR      NC,L1148        ; forward when not to V-RUN/SYN

        RES     6,C             ;

        RST     20H             ; NEXT-CHAR
        JR      L1139           ; loop back to V-CHAR

; ---

;; V-STR-VAR
L1143:  RST     20H             ; NEXT-CHAR
        RES     6,(IY+$01)      ; sv FLAGS  - Signal string result

;; V-RUN/SYN
L1148:  LD      B,C             ;
        CALL    L0DA6           ; routine SYNTAX-Z
        JR      NZ,L1156        ; forward to V-RUN

        LD      A,C             ;
        AND     $E0             ;
        SET     7,A             ;
        LD      C,A             ;
        JR      L118A           ; forward to V-SYNTAX

; ---
```

```
;; V-RUN
L1156:  LD      HL,($4010)      ; sv VARS

;; V-EACH
L1159:  LD      A,(HL)          ;
        AND     $7F             ;
        JR      Z,L1188         ; to V-80-BYTE

        CP      C               ;
        JR      NZ,L1180        ; to V-NEXT

        RLA                     ;
        ADD     A,A             ;
        JP      P,L1195         ; to V-FOUND-2

        JR      C,L1195         ; to V-FOUND-2

        POP     DE              ;
        PUSH    DE              ;
        PUSH    HL              ;

;; V-MATCHES
L116B:  INC     HL              ;

;; V-SPACES
L116C:  LD      A,(DE)          ;
        INC     DE              ;
        AND     A               ;
        JR      Z,L116C         ; back to V-SPACES

        CP      (HL)            ;
        JR      Z,L116B         ; back to V-MATCHES

        OR      $80             ;
        CP      (HL)            ;
        JR       NZ,L117F       ; forward to V-GET-PTR

        LD      A,(DE)          ;
        CALL    L14D2           ; routine ALPHANUM
        JR      NC,L1194        ; forward to V-FOUND-1

;; V-GET-PTR
L117F:  POP     HL              ;

;; V-NEXT
L1180:  PUSH    BC              ;
        CALL    L09F2           ; routine NEXT-ONE
        EX      DE,HL           ;
        POP     BC              ;
        JR      L1159           ; back to V-EACH

; ---

;; V-80-BYTE
L1188:  SET     7,B             ;

;; V-SYNTAX
L118A:  POP     DE              ;

        RST     18H             ; GET-CHAR
        CP      $10             ;
```

```
        JR      Z,L1199         ; forward to V-PASS

        SET     5,B             ;
        JR      L11A1           ; forward to V-END

; ---

;; V-FOUND-1
L1194:  POP     DE              ;

;; V-FOUND-2
L1195:  POP     DE              ;
        POP     DE              ;
        PUSH    HL              ;

        RST     18H             ; GET-CHAR

;; V-PASS
L1199:  CALL    L14D2           ; routine ALPHANUM
        JR      NC,L11A1        ; forward if not alphanumeric to V-END


        RST     20H             ; NEXT-CHAR
        JR      L1199           ; back to V-PASS

; ---

;; V-END
L11A1:  POP     HL              ;
        RL      B               ;
        BIT     6,B             ;
        RET                     ;

; ------------------------
; THE 'STK-VAR' SUBROUTINE
; ------------------------
;
;

;; STK-VAR
L11A7:  XOR     A               ;
        LD      B,A             ;
        BIT     7,C             ;
        JR      NZ,L11F8        ; forward to SV-COUNT

        BIT     7,(HL)          ;
        JR      NZ,L11BF        ; forward to SV-ARRAYS

        INC     A               ;

;; SV-SIMPLE$
L11B2:  INC     HL              ;
        LD      C,(HL)          ;
        INC     HL              ;
        LD      B,(HL)          ;
        INC     HL              ;
        EX      DE,HL           ;
        CALL    L12C3           ; routine STK-STO-$

        RST     18H             ; GET-CHAR
        JP      L125A           ; jump forward to SV-SLICE?
```

```
; ---

;; SV-ARRAYS
L11BF:  INC     HL              ;
        INC     HL              ;
        INC     HL              ;
        LD      B,(HL)          ;
        BIT     6,C             ;
        JR      Z,L11D1         ; forward to SV-PTR

        DEC     B               ;
        JR      Z,L11B2         ; forward to SV-SIMPLE$

        EX      DE,HL           ;

        RST     18H             ; GET-CHAR
        CP      $10             ;
        JR      NZ,L1231        ; forward to REPORT-3

        EX      DE,HL           ;

;; SV-PTR
L11D1:  EX      DE,HL           ;
        JR      L11F8           ; forward to SV-COUNT

; ---

;; SV-COMMA
L11D4:  PUSH    HL              ;

        RST     18H             ; GET-CHAR
        POP     HL              ;
        CP      $1A             ;
        JR      Z,L11FB         ; forward to SV-LOOP

        BIT     7,C             ;
        JR      Z,L1231         ; forward to REPORT-3

        BIT     6,C             ;
        JR      NZ,L11E9        ; forward to SV-CLOSE

        CP      $11             ;
        JR      NZ,L1223        ; forward to SV-RPT-C


        RST     20H             ; NEXT-CHAR
        RET                     ;

; ---

;; SV-CLOSE
L11E9:  CP      $11             ;
        JR      Z,L1259         ; forward to SV-DIM

        CP      $DF             ;
        JR      NZ,L1223        ; forward to SV-RPT-C


;; SV-CH-ADD
L11F1:  RST     18H             ; GET-CHAR
        DEC     HL              ;
        LD      ($4016),HL      ; sv CH_ADD
```

```
        JR      L1256           ; forward to SV-SLICE

; ---

;; SV-COUNT
L11F8:  LD      HL,$0000        ;

;; SV-LOOP
L11FB:  PUSH    HL              ;

        RST     20H             ; NEXT-CHAR
        POP     HL              ;
        LD      A,C             ;
        CP      $C0             ;
        JR      NZ,L120C        ; forward to SV-MULT


        RST     18H             ; GET-CHAR
        CP      $11             ;
        JR      Z,L1259         ; forward to SV-DIM

        CP      $DF             ;
        JR      Z,L11F1         ; back to SV-CH-ADD

;; SV-MULT
L120C:  PUSH    BC              ;
        PUSH    HL              ;
        CALL    L12FF           ; routine DE,(DE+1)
        EX      (SP),HL         ;
        EX      DE,HL           ;
        CALL    L12DD           ; routine INT-EXP1
        JR      C,L1231         ; forward to REPORT-3

        DEC     BC              ;
        CALL    L1305           ; routine GET-HL*DE
        ADD     HL,BC           ;
        POP     DE              ;
        POP     BC              ;
        DJNZ    L11D4           ; loop back to SV-COMMA

        BIT     7,C             ;

;; SV-RPT-C
L1223:  JR      NZ,L128B        ; relative jump to SL-RPT-C

        PUSH    HL              ;
        BIT     6,C             ;
        JR      NZ,L123D        ; forward to SV-ELEM$

        LD      B,D             ;
        LD      C,E             ;

        RST     18H             ; GET-CHAR
        CP      $11             ; is character a ')' ?
        JR      Z,L1233         ; skip forward to SV-NUMBER


;; REPORT-3
L1231:  RST     08H             ; ERROR-1
        DEFB    $02             ; Error Report: Subscript wrong
```

```
;; SV-NUMBER
L1233:  RST     20H             ; NEXT-CHAR
        POP     HL              ;
        LD      DE,$0005        ;
        CALL    L1305           ; routine GET-HL*DE
        ADD     HL,BC           ;
        RET                     ; return                                >>

; ---

;; SV-ELEM$
L123D:  CALL    L12FF           ; routine DE,(DE+1)
        EX      (SP),HL         ;
        CALL    L1305           ; routine GET-HL*DE
        POP     BC              ;
        ADD     HL,BC           ;
        INC     HL              ;
        LD      B,D             ;
        LD      C,E             ;
        EX      DE,HL           ;
        CALL    L12C2           ; routine STK-ST-0

        RST     18H             ; GET-CHAR
        CP      $11             ; is it ')' ?
        JR      Z,L1259         ; forward if so to SV-DIM

        CP      $1A             ; is it ',' ?
        JR      NZ,L1231        ; back if not to REPORT-3

;; SV-SLICE
L1256:  CALL    L1263           ; routine SLICING

;; SV-DIM
L1259:  RST     20H             ; NEXT-CHAR

;; SV-SLICE?
L125A:  CP      $10             ;
        JR      Z,L1256         ; back to SV-SLICE

        RES     6,(IY+$01)      ; sv FLAGS  - Signal string result
        RET                     ; return.

; ------------------------
; THE 'SLICING' SUBROUTINE
; ------------------------
;
;

;; SLICING
L1263:  CALL    L0DA6           ; routine SYNTAX-Z
        CALL    NZ,L13F8        ; routine STK-FETCH

        RST     20H             ; NEXT-CHAR
        CP      $11             ; is it ')' ?
        JR      Z,L12BE         ; forward if so to SL-STORE

        PUSH    DE              ;
        XOR     A               ;
        PUSH    AF              ;
        PUSH    BC              ;
        LD      DE,$0001        ;
```

```
         RST     18H             ; GET-CHAR
         POP     HL              ;
         CP      $DF             ; is it 'TO' ?
         JR      Z,L1292         ; forward if so to SL-SECOND

         POP     AF              ;
         CALL    L12DE           ; routine INT-EXP2
         PUSH    AF              ;
         LD      D,B             ;
         LD      E,C             ;
         PUSH    HL              ;

         RST     18H             ; GET-CHAR
         POP     HL              ;
         CP      $DF             ; is it 'TO' ?
         JR      Z,L1292         ; forward if so to SL-SECOND

         CP      $11             ;

;; SL-RPT-C
L128B:   JP      NZ,L0D9A        ; to REPORT-C

         LD      H,D             ;
         LD      L,E             ;
         JR      L12A5           ; forward to SL-DEFINE

; ---

;; SL-SECOND
L1292:   PUSH    HL              ;

         RST     20H             ; NEXT-CHAR
         POP     HL              ;
         CP      $11             ; is it ')' ?
         JR      Z,L12A5         ; forward if so to SL-DEFINE

         POP     AF              ;
         CALL    L12DE           ; routine INT-EXP2
         PUSH    AF              ;

         RST     18H             ; GET-CHAR
         LD      H,B             ;
         LD      L,C             ;
         CP      $11             ; is it ')' ?
         JR      NZ,L128B        ; back if not to SL-RPT-C

;; SL-DEFINE
L12A5:   POP     AF              ;
         EX      (SP),HL         ;
         ADD     HL,DE           ;
         DEC     HL              ;
         EX      (SP),HL         ;
         AND     A               ;
         SBC     HL,DE           ;
         LD      BC,$0000        ;
         JR      C,L12B9         ; forward to SL-OVER

         INC     HL              ;
         AND     A               ;
         JP      M,L1231         ; jump back to REPORT-3

         LD      B,H             ;
```

```
        LD      C,L             ;

;; SL-OVER
L12B9:  POP     DE              ;
        RES     6,(IY+$01)      ; sv FLAGS  - Signal string result

;; SL-STORE
L12BE:  CALL    L0DA6           ; routine SYNTAX-Z
        RET     Z               ; return if checking syntax.

; --------------------------
; THE 'STK-STORE' SUBROUTINE
; --------------------------
;
;

;; STK-ST-0
L12C2:  XOR     A               ;

;; STK-STO-$
L12C3:  PUSH    BC              ;
        CALL    L19EB           ; routine TEST-5-SP
        POP     BC              ;
        LD      HL,($401C)      ; sv STKEND
        LD      (HL),A          ;
        INC     HL              ;
        LD      (HL),E          ;
        INC     HL              ;
        LD      (HL),D          ;
        INC     HL              ;
        LD      (HL),C          ;
        INC     HL              ;
        LD      (HL),B          ;
        INC     HL              ;
        LD      ($401C),HL      ; sv STKEND
        RES     6,(IY+$01)      ; update FLAGS - signal string result
        RET                     ; return.

; -------------------------
; THE 'INT EXP' SUBROUTINES
; -------------------------
;
;

;; INT-EXP1
L12DD:  XOR     A               ;

;; INT-EXP2
L12DE:  PUSH    DE              ;
        PUSH    HL              ;
        PUSH    AF              ;
        CALL    L0D92           ; routine CLASS-6
        POP     AF              ;
        CALL    L0DA6           ; routine SYNTAX-Z
        JR      Z,L12FC         ; forward if checking syntax to I-RESTORE

        PUSH    AF              ;
        CALL    L0EA7           ; routine FIND-INT
        POP     DE              ;
        LD      A,B             ;
        OR      C               ;
        SCF                     ; Set Carry Flag
```

```
        JR      Z,L12F9         ; forward to I-CARRY

        POP     HL              ;
        PUSH    HL              ;
        AND     A               ;
        SBC     HL,BC           ;

;; I-CARRY
L12F9:  LD      A,D             ;
        SBC     A,$00           ;

;; I-RESTORE
L12FC:  POP     HL              ;
        POP     DE              ;
        RET                     ;

; -------------------------
; THE 'DE,(DE+1)' SUBROUTINE
; -------------------------
; INDEX and LOAD Z80 subroutine.
; This emulates the 6800 processor instruction LDX 1,X which loads a two-
byte
; value from memory into the register indexing it. Often these are hardly
worth
; the bother of writing as subroutines and this one doesn't save any time
or
; memory. The timing and space overheads have to be offset against the ease
of
; writing and the greater program readability from using such toolkit
routines.

;; DE,(DE+1)
L12FF:  EX      DE,HL           ; move index address into HL.
        INC     HL              ; increment to address word.
        LD      E,(HL)          ; pick up word low-order byte.
        INC     HL              ; index high-order byte and
        LD      D,(HL)          ; pick it up.
        RET                     ; return with DE = word.

; -------------------------
; THE 'GET-HL*DE' SUBROUTINE
; -------------------------
;

;; GET-HL*DE
L1305:  CALL    L0DA6           ; routine SYNTAX-Z
        RET     Z               ;

        PUSH    BC              ;
        LD      B,$10           ;
        LD      A,H             ;
        LD      C,L             ;
        LD      HL,$0000        ;

;; HL-LOOP
L1311:  ADD     HL,HL           ;
        JR      C,L131A         ; forward with carry to HL-END

        RL      C               ;
        RLA                     ;
        JR      NC,L131D        ; forward with no carry to HL-AGAIN
```

```
        ADD     HL,DE           ;

;; HL-END
L131A:  JP      C,L0ED3         ; to REPORT-4

;; HL-AGAIN
L131D:  DJNZ    L1311           ; loop back to HL-LOOP

        POP     BC              ;
        RET                     ; return.

; --------------------
; THE 'LET' SUBROUTINE
; --------------------
;
;

;; LET
L1321:  LD      HL,($4012)      ; sv DEST-lo
        BIT     1,(IY+$2D)      ; sv FLAGX
        JR      Z,L136E         ; forward to L-EXISTS

        LD      BC,$0005        ;

;; L-EACH-CH
L132D:  INC     BC              ;

; check

;; L-NO-SP
L132E:  INC     HL              ;
        LD      A,(HL)          ;
        AND     A               ;
        JR      Z,L132E         ; back to L-NO-SP

        CALL    L14D2           ; routine ALPHANUM
        JR      C,L132D         ; back to L-EACH-CH

        CP      $0D             ; is it '$' ?
        JP      Z,L13C8         ; forward if so to L-NEW$


        RST     30H             ; BC-SPACES
        PUSH    DE              ;
        LD      HL,($4012)      ; sv DEST
        DEC     DE              ;
        LD      A,C             ;
        SUB     $06             ;
        LD      B,A             ;
        LD      A,$40           ;
        JR      Z,L1359         ; forward to L-SINGLE

;; L-CHAR
L134B:  INC     HL              ;
        LD      A,(HL)          ;
        AND     A               ; is it a space ?
        JR      Z,L134B         ; back to L-CHAR

        INC     DE              ;
        LD      (DE),A          ;
        DJNZ    L134B           ; loop back to L-CHAR
```

```
         OR       $80             ;
         LD       (DE),A          ;
         LD       A,$80           ;

;; L-SINGLE
L1359:   LD       HL,($4012)      ; sv DEST-lo
         XOR      (HL)            ;
         POP      HL              ;
         CALL     L13E7           ; routine L-FIRST

;; L-NUMERIC
L1361:   PUSH     HL              ;

         RST      28H             ;; FP-CALC
         DEFB     $02             ;;delete
         DEFB     $34             ;;end-calc

         POP      HL              ;
         LD       BC,$0005        ;
         AND      A               ;
         SBC      HL,BC           ;
         JR       L13AE           ; forward to L-ENTER

; ---

;; L-EXISTS
L136E:   BIT      6,(IY+$01)      ; sv FLAGS  - Numeric or string result?
         JR       Z,L137A         ; forward to L-DELETE$

         LD       DE,$0006        ;
         ADD      HL,DE           ;
         JR       L1361           ; back to L-NUMERIC

; ---

;; L-DELETE$
L137A:   LD       HL,($4012)      ; sv DEST-lo
         LD       BC,($402E)      ; sv STRLEN_lo
         BIT      0,(IY+$2D)      ; sv FLAGX
         JR       NZ,L13B7        ; forward to L-ADD$

         LD       A,B             ;
         OR       C               ;
         RET      Z               ;

         PUSH     HL              ;

         RST      30H             ; BC-SPACES
         PUSH     DE              ;
         PUSH     BC              ;
         LD       D,H             ;
         LD       E,L             ;
         INC      HL              ;
         LD       (HL),$00        ;
         LDDR                     ; Copy Bytes
         PUSH     HL              ;
         CALL     L13F8           ; routine STK-FETCH
         POP      HL              ;
         EX       (SP),HL         ;
         AND      A               ;
         SBC      HL,BC           ;
         ADD      HL,BC           ;
```

```
        JR      NC,L13A3        ; forward to L-LENGTH

        LD      B,H             ;
        LD      C,L             ;

;; L-LENGTH
L13A3:  EX      (SP),HL         ;
        EX      DE,HL           ;
        LD      A,B             ;
        OR      C               ;
        JR      Z,L13AB         ; forward if zero to L-IN-W/S

        LDIR                    ; Copy Bytes

;; L-IN-W/S
L13AB:  POP     BC              ;
        POP     DE              ;
        POP     HL              ;

; ------------------------
; THE 'L-ENTER' SUBROUTINE
; ------------------------
;

;; L-ENTER
L13AE:  EX      DE,HL           ;
        LD      A,B             ;
        OR      C               ;
        RET     Z               ;

        PUSH    DE              ;
        LDIR                    ; Copy Bytes
        POP     HL              ;
        RET                     ; return.

; ---

;; L-ADD$
L13B7:  DEC     HL              ;
        DEC     HL              ;
        DEC     HL              ;
        LD      A,(HL)          ;
        PUSH    HL              ;
        PUSH    BC              ;

        CALL    L13CE           ; routine L-STRING

        POP     BC              ;
        POP     HL              ;
        INC     BC              ;
        INC     BC              ;
        INC     BC              ;
        JP      L0A60           ; jump back to exit via RECLAIM-2

; ---

;; L-NEW$
L13C8:  LD      A,$60           ; prepare mask %01100000
        LD      HL,($4012)      ; sv DEST-lo
        XOR     (HL)            ;

; -------------------------
```

```
; THE 'L-STRING' SUBROUTINE
; -------------------------
;

;; L-STRING
L13CE:  PUSH    AF              ;
        CALL    L13F8           ; routine STK-FETCH
        EX      DE,HL           ;
        ADD     HL,BC           ;
        PUSH    HL              ;
        INC     BC              ;
        INC     BC              ;
        INC     BC              ;

        RST     30H             ; BC-SPACES
        EX      DE,HL           ;
        POP     HL              ;
        DEC     BC              ;
        DEC     BC              ;
        PUSH    BC              ;
        LDDR                    ; Copy Bytes
        EX      DE,HL           ;
        POP     BC              ;
        DEC     BC              ;
        LD      (HL),B          ;
        DEC     HL              ;
        LD      (HL),C          ;
        POP     AF              ;

;; L-FIRST
L13E7:  PUSH    AF              ;
        CALL    L14C7           ; routine REC-V80
        POP     AF              ;
        DEC     HL              ;
        LD      (HL),A          ;
        LD      HL,($401A)      ; sv STKBOT_lo
        LD      ($4014),HL      ; sv E_LINE_lo
        DEC     HL              ;
        LD      (HL),$80        ;
        RET                     ;

; -------------------------
; THE 'STK-FETCH' SUBROUTINE
; -------------------------
; This routine fetches a five-byte value from the calculator stack
; reducing the pointer to the end of the stack by five.
; For a floating-point number the exponent is in A and the mantissa
; is the thirty-two bits EDCB.
; For strings, the start of the string is in DE and the length in BC.
; A is unused.

;; STK-FETCH
L13F8:  LD      HL,($401C)      ; load HL from system variable STKEND

        DEC     HL              ;
        LD      B,(HL)          ;
        DEC     HL              ;
        LD      C,(HL)          ;
        DEC     HL              ;
        LD      D,(HL)          ;
        DEC     HL              ;
        LD      E,(HL)          ;
```

```
        DEC     HL              ;
        LD      A,(HL)          ;

        LD      ($401C),HL      ; set system variable STKEND to lower
value.
        RET                     ; return.

; -------------------------
; THE 'DIM' COMMAND ROUTINE
; -------------------------
; An array is created and initialized to zeros which is also the space
; character on the ZX81.

;; DIM
L1409:  CALL    L111C           ; routine LOOK-VARS

;; D-RPORT-C
L140C:  JP      NZ,L0D9A        ; to REPORT-C

        CALL    L0DA6           ; routine SYNTAX-Z
        JR      NZ,L141C        ; forward to D-RUN

        RES     6,C             ;
        CALL    L11A7           ; routine STK-VAR
        CALL    L0D1D           ; routine CHECK-END

;; D-RUN
L141C:  JR      C,L1426         ; forward to D-LETTER

        PUSH    BC              ;
        CALL    L09F2           ; routine NEXT-ONE
        CALL    L0A60           ; routine RECLAIM-2
        POP     BC              ;

;; D-LETTER
L1426:  SET     7,C             ;
        LD      B,$00           ;
        PUSH    BC              ;
        LD      HL,$0001        ;
        BIT     6,C             ;
        JR      NZ,L1434        ; forward to D-SIZE

        LD      L,$05           ;

;; D-SIZE
L1434:  EX      DE,HL           ;

;; D-NO-LOOP
L1435:  RST     20H             ; NEXT-CHAR
        LD      H,$40           ;
        CALL    L12DD           ; routine INT-EXP1
        JP      C,L1231         ; jump back to REPORT-3

        POP     HL              ;
        PUSH    BC              ;
        INC     H               ;
        PUSH    HL              ;
        LD      H,B             ;
        LD      L,C             ;
        CALL    L1305           ; routine GET-HL*DE
        EX      DE,HL           ;
```

```
         RST     18H             ; GET-CHAR
         CP      $1A             ;
         JR      Z,L1435         ; back to D-NO-LOOP

         CP      $11             ; is it ')' ?
         JR      NZ,L140C        ; back if not to D-RPORT-C


         RST     20H             ; NEXT-CHAR
         POP     BC              ;
         LD      A,C             ;
         LD      L,B             ;
         LD      H,$00           ;
         INC     HL              ;
         INC     HL              ;
         ADD     HL,HL           ;
         ADD     HL,DE           ;
         JP      C,L0ED3         ; jump to REPORT-4

         PUSH    DE              ;
         PUSH    BC              ;
         PUSH    HL              ;
         LD      B,H             ;
         LD      C,L             ;
         LD      HL,($4014)      ; sv E_LINE_lo
         DEC     HL              ;
         CALL    L099E           ; routine MAKE-ROOM
         INC     HL              ;
         LD       (HL),A         ;
         POP     BC              ;
         DEC     BC              ;
         DEC     BC              ;
         DEC     BC              ;
         INC     HL              ;
         LD      (HL),C          ;
         INC     HL              ;
         LD      (HL),B          ;
         POP     AF              ;
         INC     HL              ;
         LD      (HL),A          ;
         LD      H,D             ;
         LD      L,E             ;
         DEC     DE              ;
         LD      (HL),$00        ;
         POP     BC              ;
         LDDR                    ; Copy Bytes

;; DIM-SIZES
L147F:   POP     BC              ;
         LD      (HL),B          ;
         DEC     HL              ;
         LD      (HL),C          ;
         DEC     HL              ;
         DEC     A               ;
         JR      NZ,L147F        ; back to DIM-SIZES

         RET                     ; return.

; ---------------------
; THE 'RESERVE' ROUTINE
; ---------------------
;
```

```
;

;; RESERVE
L1488:  LD      HL,($401A)      ; address STKBOT
        DEC     HL              ; now last byte of workspace
        CALL    L099E           ; routine MAKE-ROOM
        INC     HL              ;
        INC     HL              ;
        POP     BC              ;
        LD      ($4014),BC      ; sv E_LINE_lo
        POP     BC              ;
        EX      DE,HL           ;
        INC     HL              ;
        RET                     ;

; --------------------------
; THE 'CLEAR' COMMAND ROUTINE
; --------------------------
;
;

;; CLEAR
L149A:  LD      HL,($4010)      ; sv VARS_lo
        LD      (HL),$80        ;
        INC     HL              ;
        LD      ($4014),HL      ; sv E_LINE_lo

; ----------------------
; THE 'X-TEMP' SUBROUTINE
; ----------------------
;
;

;; X-TEMP
L14A3:  LD      HL,($4014)      ; sv E_LINE_lo

; ----------------------
; THE 'SET-STK' ROUTINES
; ----------------------
;
;

;; SET-STK-B
L14A6:  LD      ($401A),HL      ; sv STKBOT

;

;; SET-STK-E
L14A9:  LD      ($401C),HL      ; sv STKEND
        RET                     ;

; ----------------------
; THE 'CURSOR-IN' ROUTINE
; ----------------------
; This routine is called to set the edit line to the minimum cursor/newline
; and to set STKEND, the start of free space, at the next position.

;; CURSOR-IN
L14AD:  LD      HL,($4014)      ; fetch start of edit line from E_LINE
        LD      (HL),$7F        ; insert cursor character

        INC     HL              ; point to next location.
```

```
        LD      (HL),$76        ; insert NEWLINE character
        INC     HL              ; point to next free location.

        LD      (IY+$22),$02    ; set lower screen display file size DF_SZ

        JR      L14A6           ; exit via SET-STK-B above

; ------------------------
; THE 'SET-MIN' SUBROUTINE
; ------------------------
;
;

;; SET-MIN
L14BC:  LD      HL,$405D        ; normal location of calculator's memory
area
        LD      ($401F),HL      ; update system variable MEM
        LD      HL,($401A)      ; fetch STKBOT
        JR      L14A9           ; back to SET-STK-E


; -----------------------------------
; THE 'RECLAIM THE END-MARKER' ROUTINE
; -----------------------------------

;; REC-V80
L14C7:  LD      DE,($4014)      ; sv E_LINE_lo
        JP      L0A5D           ; to RECLAIM-1

; ----------------------
; THE 'ALPHA' SUBROUTINE
; ----------------------

;; ALPHA
L14CE:  CP      $26             ;
        JR      L14D4           ; skip forward to ALPHA-2


; -------------------------
; THE 'ALPHANUM' SUBROUTINE
; -------------------------

;; ALPHANUM
L14D2:  CP      $1C             ;


;; ALPHA-2
L14D4:  CCF                     ; Complement Carry Flag
        RET     NC              ;

        CP      $40             ;
        RET                     ;


; ----------------------------------------
; THE 'DECIMAL TO FLOATING POINT' SUBROUTINE
; ----------------------------------------
;

;; DEC-TO-FP
L14D9:  CALL    L1548           ; routine INT-TO-FP gets first part
        CP      $1B             ; is character a '.' ?
```

```
        JR      NZ,L14F5        ; forward if not to E-FORMAT


        RST     28H             ;; FP-CALC
        DEFB    $A1             ;;stk-one
        DEFB    $C0             ;;st-mem-0
        DEFB    $02             ;;delete
        DEFB    $34             ;;end-calc


;; NXT-DGT-1
L14E5:  RST     20H             ; NEXT-CHAR
        CALL    L1514           ; routine STK-DIGIT
        JR      C,L14F5         ; forward to E-FORMAT


        RST     28H             ;; FP-CALC
        DEFB    $E0             ;;get-mem-0
        DEFB    $A4             ;;stk-ten
        DEFB    $05             ;;division
        DEFB    $C0             ;;st-mem-0
        DEFB    $04             ;;multiply
        DEFB    $0F             ;;addition
        DEFB    $34             ;;end-calc

        JR      L14E5           ; loop back till exhausted to NXT-DGT-1

; ---

;; E-FORMAT
L14F5:  CP      $2A             ; is character 'E' ?
        RET     NZ              ; return if not

        LD      (IY+$5D),$FF    ; initialize sv MEM-0-1st to $FF TRUE

        RST     20H             ; NEXT-CHAR
        CP      $15             ; is character a '+' ?
        JR      Z,L1508         ; forward if so to SIGN-DONE

        CP      $16             ; is it a '-' ?
        JR      NZ,L1509        ; forward if not to ST-E-PART

        INC     (IY+$5D)        ; sv MEM-0-1st change to FALSE

;; SIGN-DONE
L1508:  RST     20H             ; NEXT-CHAR

;; ST-E-PART
L1509:  CALL    L1548           ; routine INT-TO-FP

        RST     28H             ;; FP-CALC              m, e.
        DEFB    $E0             ;;get-mem-0             m, e, (1/0)
TRUE/FALSE
        DEFB    $00             ;;jump-true
        DEFB    $02             ;;to L1511, E-POSTVE
        DEFB    $18             ;;neg                   m, -e

;; E-POSTVE
L1511:  DEFB    $38             ;;e-to-fp               x.
        DEFB    $34             ;;end-calc              x.

        RET                     ; return.
```

```
; --------------------------
; THE 'STK-DIGIT' SUBROUTINE
; --------------------------
;

;; STK-DIGIT
L1514:  CP      $1C             ;
        RET     C               ;

        CP      $26             ;
        CCF                     ; Complement Carry Flag
        RET     C               ;

        SUB     $1C             ;

; ------------------------
; THE 'STACK-A' SUBROUTINE
; ------------------------
;


;; STACK-A
L151D:  LD      C,A             ;
        LD      B,$00           ;

; --------------------------
; THE 'STACK-BC' SUBROUTINE
; --------------------------
; The ZX81 does not have an integer number format so the BC register
contents
; must be converted to their full floating-point form.

;; STACK-BC
L1520:  LD      IY,$4000        ; re-initialize the system variables
pointer.
        PUSH    BC              ; save the integer value.

; now stack zero, five zero bytes as a starting point.

        RST     28H             ;; FP-CALC
        DEFB    $A0             ;;stk-zero                    0.
        DEFB    $34             ;;end-calc

        POP     BC              ; restore integer value.

        LD      (HL),$91        ; place $91 in exponent         65536.
                                ; this is the maximum possible value

        LD      A,B             ; fetch hi-byte.
        AND     A               ; test for zero.
        JR      NZ,L1536        ; forward if not zero to STK-BC-2

        LD      (HL),A          ; else make exponent zero again
        OR      C               ; test lo-byte
        RET     Z               ; return if BC was zero - done.

; else  there has to be a set bit if only the value one.

        LD      B,C             ; save C in B.
        LD      C,(HL)          ; fetch zero to C
```

```
        LD      (HL),$89        ; make exponent $89             256.

;; STK-BC-2
L1536:  DEC     (HL)            ; decrement exponent - halving number
        SLA     C               ;  C<-76543210<-0
        RL      B               ;  C<-76543210<-C
        JR      NC,L1536        ; loop back if no carry to STK-BC-2

        SRL     B               ;  0->76543210->C
        RR      C               ;  C->76543210->C

        INC     HL              ; address first byte of mantissa
        LD      (HL),B          ; insert B
        INC     HL              ; address second byte of mantissa
        LD      (HL),C          ; insert C

        DEC     HL              ; point to the
        DEC     HL              ; exponent again
        RET                     ; return.

; ----------------------------------------
; THE 'INTEGER TO FLOATING POINT' SUBROUTINE
; ----------------------------------------
;
;

;; INT-TO-FP
L1548:  PUSH    AF              ;

        RST     28H             ;; FP-CALC
        DEFB    $A0             ;;stk-zero
        DEFB    $34             ;;end-calc

        POP     AF              ;

;; NXT-DGT-2
L154D:  CALL    L1514           ; routine STK-DIGIT
        RET     C               ;


        RST     28H             ;; FP-CALC
        DEFB    $01             ;;exchange
        DEFB    $A4             ;;stk-ten
        DEFB    $04             ;;multiply
        DEFB    $0F             ;;addition
        DEFB    $34             ;;end-calc


        RST     20H             ; NEXT-CHAR
        JR      L154D           ; to NXT-DGT-2


; ----------------------------------------
; THE 'E-FORMAT TO FLOATING POINT' SUBROUTINE
; ----------------------------------------
; (Offset $38: 'e-to-fp')
; invoked from DEC-TO-FP and PRINT-FP.
; e.g. 2.3E4 is 23000.
; This subroutine evaluates xEm where m is a positive or negative integer.
; At a simple level x is multiplied by ten for every unit of m.
; If the decimal exponent m is negative then x is divided by ten for each
unit.
```

```
; A short-cut is taken if the exponent is greater than seven and in this
; case the exponent is reduced by seven and the value is multiplied or
divided
; by ten million.
; Note. for the ZX Spectrum an even cleverer method was adopted which
involved
; shifting the bits out of the exponent so the result was achieved with six
; shifts at most. The routine below had to be completely re-written mostly
; in Z80 machine code.
; Although no longer operable, the calculator literal was retained for old
; times sake, the routine being invoked directly from a machine code CALL.
;
; On entry in the ZX81, m, the exponent, is the 'last value', and the
; floating-point decimal mantissa is beneath it.


;; e-to-fp
L155A:  RST     28H             ;; FP-CALC              x, m.
        DEFB    $2D             ;;duplicate             x, m, m.
        DEFB    $32             ;;less-0                x, m, (1/0).
        DEFB    $C0             ;;st-mem-0              x, m, (1/0).
        DEFB    $02             ;;delete                x, m.
        DEFB    $27             ;;abs                   x, +m.

;; E-LOOP
L1560:  DEFB    $A1             ;;stk-one               x, m,1.
        DEFB    $03             ;;subtract              x, m-1.
        DEFB    $2D             ;;duplicate             x, m-1,m-1.
        DEFB    $32             ;;less-0                x, m-1, (1/0).
        DEFB    $00             ;;jump-true             x, m-1.
        DEFB    $22             ;;to L1587, E-END       x, m-1.

        DEFB    $2D             ;;duplicate             x, m-1, m-1.
        DEFB    $30             ;;stk-data
        DEFB    $33             ;;Exponent: $83, Bytes: 1

        DEFB    $40             ;;(+00,+00,+00)         x, m-1, m-1, 6.
        DEFB    $03             ;;subtract              x, m-1, m-7.
        DEFB    $2D             ;;duplicate             x, m-1, m-7, m-7.
        DEFB    $32             ;;less-0                x, m-1, m-7, (1/0).
        DEFB    $00             ;;jump-true             x, m-1, m-7.
        DEFB    $0C             ;;to L157A, E-LOW

; but if exponent m is higher than 7 do a bigger chunk.
; multiplying (or dividing if negative) by 10 million - 1e7.

        DEFB    $01             ;;exchange              x, m-7, m-1.
        DEFB    $02             ;;delete                x, m-7.
        DEFB    $01             ;;exchange              m-7, x.
        DEFB    $30             ;;stk-data
        DEFB    $80             ;;Bytes: 3
        DEFB    $48             ;;Exponent $98
        DEFB    $18,$96,$80     ;;(+00)                 m-7, x, 10,000,000
(=f)
        DEFB    $2F             ;;jump
        DEFB    $04             ;;to L157D, E-CHUNK

; ---

;; E-LOW
L157A:  DEFB    $02             ;;delete                x, m-1.
        DEFB    $01             ;;exchange              m-1, x.
```

```
        DEFB    $A4             ;;stk-ten               m-1, x, 10 (=f).

;; E-CHUNK
L157D:  DEFB    $E0             ;;get-mem-0             m-1, x, f, (1/0)
        DEFB    $00             ;;jump-true             m-1, x, f
        DEFB    $04             ;;to L1583, E-DIVSN

        DEFB    $04             ;;multiply              m-1, x*f.
        DEFB    $2F             ;;jump
        DEFB    $02             ;;to L1584, E-SWAP

; ---

;; E-DIVSN
L1583:  DEFB    $05             ;;division              m-1, x/f (= new x).

;; E-SWAP
L1584:  DEFB    $01             ;;exchange              x, m-1 (= new m).
        DEFB    $2F             ;;jump                  x, m.
        DEFB    $DA             ;;to L1560, E-LOOP

; ---

;; E-END
L1587:  DEFB    $02             ;;delete                x. (-1)
        DEFB    $34             ;;end-calc              x.

        RET                     ; return.

; -------------------------------------
; THE 'FLOATING-POINT TO BC' SUBROUTINE
; -------------------------------------
; The floating-point form on the calculator stack is compressed directly
into
; the BC register rounding up if necessary.
; Valid range is 0 to 65535.4999

;; FP-TO-BC
L158A:  CALL    L13F8           ; routine STK-FETCH - exponent to A
                                ; mantissa to EDCB.
        AND     A               ; test for value zero.
        JR      NZ,L1595        ; forward if not to FPBC-NZRO

; else value is zero

        LD      B,A             ; zero to B
        LD      C,A             ; also to C
        PUSH    AF              ; save the flags on machine stack
        JR      L15C6           ; forward to FPBC-END

; ---

; EDCB  =>  BCE

;; FPBC-NZRO
L1595:  LD      B,E             ; transfer the mantissa from EDCB
        LD      E,C             ; to BCE. Bit 7 of E is the 17th bit which
        LD      C,D             ; will be significant for rounding if the
                                ; number is already normalized.

        SUB     $91             ; subtract 65536
        CCF                     ; complement carry flag
```

```
         BIT     7,B             ; test sign bit
         PUSH    AF              ; push the result

         SET     7,B             ; set the implied bit
         JR      C,L15C6         ; forward with carry from SUB/CCF to FPBC-
END
                                 ; number is too big.

         INC     A               ; increment the exponent and
         NEG                     ; negate to make range $00 - $0F

         CP      $08             ; test if one or two bytes
         JR      C,L15AF         ; forward with two to BIG-INT

         LD      E,C             ; shift mantissa
         LD      C,B             ; 8 places right
         LD      B,$00           ; insert a zero in B
         SUB     $08             ; reduce exponent by eight

;; BIG-INT
L15AF:   AND     A               ; test the exponent
         LD      D,A             ; save exponent in D.

         LD      A,E             ; fractional bits to A
         RLCA                    ; rotate most significant bit to carry for
                                 ; rounding of an already normal number.

         JR      Z,L15BC         ; forward if exponent zero to EXP-ZERO
                                 ; the number is normalized

;; FPBC-NORM
L15B5:   SRL     B               ;   0->76543210->C
         RR      C               ;   C->76543210->C

         DEC     D               ; decrement exponent

         JR      NZ,L15B5        ; loop back till zero to FPBC-NORM

;; EXP-ZERO
L15BC:   JR      NC,L15C6        ; forward without carry to NO-ROUND

         INC     BC              ; round up.
         LD      A,B             ; test result
         OR      C               ; for zero
         JR      NZ,L15C6        ; forward if not to GRE-ZERO

         POP     AF              ; restore sign flag
         SCF                     ; set carry flag to indicate overflow
         PUSH    AF              ; save combined flags again

;; FPBC-END
L15C6:   PUSH    BC              ; save BC value

; set HL and DE to calculator stack pointers.

         RST     28H             ;; FP-CALC
         DEFB    $34             ;;end-calc


         POP     BC              ; restore BC value
         POP     AF              ; restore flags
         LD      A,C             ; copy low byte to A also.
```

```
        RET                     ; return

; -------------------------------------
; THE 'FLOATING-POINT TO A' SUBROUTINE
; -------------------------------------
;
;

;; FP-TO-A
L15CD:  CALL    L158A           ; routine FP-TO-BC
        RET     C               ;

        PUSH    AF              ;
        DEC     B               ;
        INC     B               ;
        JR      Z,L15D9         ; forward if in range to FP-A-END

        POP     AF              ; fetch result
        SCF                     ; set carry flag signaling overflow
        RET                     ; return

;; FP-A-END
L15D9:  POP     AF              ;
        RET                     ;


; -----------------------------------------------
; THE 'PRINT A FLOATING-POINT NUMBER' SUBROUTINE
; -----------------------------------------------
; prints 'last value' x on calculator stack.
; There are a wide variety of formats see Chapter 4.
; e.g.
; PI            prints as       3.1415927
; .123          prints as       0.123
; .0123         prints as       .0123
; 999999999999  prints as       1000000000000
; 9876543210123 prints as       9876543200000

; Begin by isolating zero and just printing the '0' character
; for that case. For negative numbers print a leading '-' and
; then form the absolute value of x.

;; PRINT-FP
L15DB:  RST     28H             ;; FP-CALC              x.
        DEFB    $2D             ;;duplicate             x, x.
        DEFB    $32             ;;less-0                x, (1/0).
        DEFB    $00             ;;jump-true
        DEFB    $0B             ;;to L15EA, PF-NGTVE    x.

        DEFB    $2D             ;;duplicate             x, x
        DEFB    $33             ;;greater-0             x, (1/0).
        DEFB    $00             ;;jump-true
        DEFB    $0D             ;;to L15F0, PF-POSTVE   x.

        DEFB    $02             ;;delete                .
        DEFB    $34             ;;end-calc              .

        LD      A,$1C           ; load accumulator with character '0'

        RST     10H             ; PRINT-A
        RET                     ; return.                               >>
```

```
; ---

;; PF-NEGTVE
L15EA:  DEFB    $27             ; abs                   +x.
        DEFB    $34             ;;end-calc              x.

        LD      A,$16           ; load accumulator with '-'

        RST     10H             ; PRINT-A

        RST     28H             ;; FP-CALC              x.

;; PF-POSTVE
L15F0:  DEFB    $34             ;;end-calc              x.

; register HL addresses the exponent of the floating-point value.
; if positive, and point floats to left, then bit 7 is set.

        LD      A,(HL)          ; pick up the exponent byte
        CALL    L151D           ; routine STACK-A places on calculator
stack.

; now calculate roughly the number of digits, n, before the decimal point
by
; subtracting a half from true exponent and multiplying by log to
; the base 10 of 2.
; The true number could be one higher than n, the integer result.

        RST     28H             ;; FP-CALC              x, e.
        DEFB    $30             ;;stk-data
        DEFB    $78             ;;Exponent: $88, Bytes: 2
        DEFB    $00,$80         ;;(+00,+00)             x, e, 128.5.
        DEFB    $03             ;;subtract              x, e -.5.
        DEFB    $30             ;;stk-data
        DEFB    $EF             ;;Exponent: $7F, Bytes: 4
        DEFB    $1A,$20,$9A,$85 ;;                      .30103 (log10 2)
        DEFB    $04             ;;multiply              x,
        DEFB    $24             ;;int
        DEFB    $C1             ;;st-mem-1              x, n.


        DEFB    $30             ;;stk-data
        DEFB    $34             ;;Exponent: $84, Bytes: 1
        DEFB    $00             ;;(+00,+00,+00)         x, n, 8.

        DEFB    $03             ;;subtract              x, n-8.
        DEFB    $18             ;;neg                   x, 8-n.
        DEFB    $38             ;;e-to-fp               x * (10^n)

; finally the 8 or 9 digit decimal is rounded.
; a ten-digit integer can arise in the case of, say, 999999999.5
; which gives 1000000000.

        DEFB    $A2             ;;stk-half
        DEFB    $0F             ;;addition
        DEFB    $24             ;;int                   i.
        DEFB    $34             ;;end-calc

; If there were 8 digits then final rounding will take place on the
calculator
; stack above and the next two instructions insert a masked zero so that
; no further rounding occurs. If the result is a 9 digit integer then
```

```
; rounding takes place within the buffer.

        LD      HL,$406B        ; address system variable MEM-2-5th
                                ; which could be the 'ninth' digit.
        LD      (HL),$90        ; insert the value $90  10010000

; now starting from lowest digit lay down the 8, 9 or 10 digit integer
; which represents the significant portion of the number
; e.g. PI will be the nine-digit integer 314159265

        LD      B,$0A           ; count is ten digits.

;; PF-LOOP
L1615:  INC     HL              ; increase pointer

        PUSH    HL              ; preserve buffer address.
        PUSH    BC              ; preserve counter.

        RST     28H             ;; FP-CALC              i.
        DEFB    $A4             ;;stk-ten               i, 10.
        DEFB    $2E             ;;n-mod-m               i mod 10, i/10
        DEFB    $01             ;;exchange              i/10, remainder.
        DEFB    $34             ;;end-calc

        CALL    L15CD           ; routine FP-TO-A  $00-$09

        OR      $90             ; make left hand nibble 9

        POP     BC              ; restore counter
        POP     HL              ; restore buffer address.

        LD      (HL),A          ; insert masked digit in buffer.
        DJNZ    L1615           ; loop back for all ten to PF-LOOP

; the most significant digit will be last but if the number is exhausted
then
; the last one or two positions will contain zero ($90).

; e.g. for 'one' we have zero as estimate of leading digits.
; 1*10^8 100000000 as integer value
; 90 90 90 90 90   90 90 90 91 90 as buffer mem3/mem4 contents.


        INC     HL              ; advance pointer to one past buffer
        LD      BC,$0008        ; set C to 8 ( B is already zero )
        PUSH    HL              ; save pointer.

;; PF-NULL
L162C:  DEC     HL              ; decrease pointer
        LD      A,(HL)          ; fetch masked digit
        CP      $90             ; is it a leading zero ?
        JR      Z,L162C         ; loop back if so to PF-NULL

; at this point a significant digit has been found. carry is reset.

        SBC     HL,BC           ; subtract eight from the address.
        PUSH    HL              ; ** save this pointer too
        LD      A,(HL)          ; fetch addressed byte
        ADD     A,$6B           ; add $6B - forcing a round up ripple
                                ; if  $95 or over.
        PUSH    AF              ; save the carry result.
```

```
; now enter a loop to round the number. After rounding has been considered
; a zero that has arisen from rounding or that was present at that position
; originally is changed from $90 to $80.

;; PF-RND-LP
L1639:  POP     AF              ; retrieve carry from machine stack.
        INC     HL              ; increment address
        LD      A,(HL)          ; fetch new byte
        ADC     A,$00           ; add in any carry

        DAA                     ; decimal adjust accumulator
                                ; carry will ripple through the '9'

        PUSH    AF              ; save carry on machine stack.
        AND     $0F             ; isolate character 0 - 9 AND set zero flag
                                ; if zero.
        LD      (HL),A          ; place back in location.
        SET     7,(HL)          ; set bit 7 to show printable.
                                ; but not if trailing zero after decimal
point.
        JR      Z,L1639         ; back if a zero to PF-RND-LP
                                ; to consider further rounding and/or
trailing
                                ; zero identification.

        POP     AF              ; balance stack
        POP     HL              ; ** retrieve lower pointer

; now insert 6 trailing zeros which are printed if before the decimal point
; but mark the end of printing if after decimal point.
; e.g. 9876543210123 is printed as 9876543200000
; 123.456001 is printed as 123.456

        LD      B,$06           ; the count is six.

;; PF-ZERO-6
L164B:  LD      (HL),$80        ; insert a masked zero
        DEC     HL              ; decrease pointer.
        DJNZ    L164B           ; loop back for all six to PF-ZERO-6

; n-mod-m reduced the number to zero and this is now deleted from the
calculator
; stack before fetching the original estimate of leading digits.


        RST     28H             ;; FP-CALC              0.
        DEFB    $02             ;;delete                .
        DEFB    $E1             ;;get-mem-1             n.
        DEFB    $34             ;;end-calc              n.

        CALL    L15CD           ; routine FP-TO-A
        JR      Z,L165B         ; skip forward if positive to PF-POS

        NEG                     ; negate makes positive

;; PF-POS
L165B:  LD      E,A             ; transfer count of digits to E
        INC     E               ; increment twice
        INC     E               ;
        POP     HL              ; * retrieve pointer to one past buffer.

;; GET-FIRST
```

```
L165F:  DEC     HL              ; decrement address.
        DEC     E               ; decrement digit counter.
        LD      A,(HL)          ; fetch masked byte.
        AND     $0F             ; isolate right-hand nibble.
        JR      Z,L165F         ; back with leading zero to GET-FIRST

; now determine if E-format printing is needed

        LD      A,E             ; transfer now accurate number count to A.
        SUB     $05             ; subtract five
        CP      $08             ; compare with 8 as maximum digits is 13.
        JP      P,L1682         ; forward if positive to PF-E-FMT

        CP      $F6             ; test for more than four zeros after
point.
        JP      M,L1682         ; forward if so to PF-E-FMT

        ADD     A,$06           ; test for zero leading digits, e.g. 0.5
        JR      Z,L16BF         ; forward if so to PF-ZERO-1

        JP      M,L16B2         ; forward if more than one zero to PF-ZEROS

; else digits before the decimal point are to be printed

        LD      B,A             ; count of leading characters to B.

;; PF-NIB-LP
L167B:  CALL    L16D0           ; routine PF-NIBBLE
        DJNZ    L167B           ; loop back for counted numbers to PF-NIB-
LP

        JR      L16C2           ; forward to consider decimal part to PF-
DC-OUT

; ---

;; PF-E-FMT
L1682:  LD      B,E             ; count to B
        CALL    L16D0           ; routine PF-NIBBLE prints one digit.
        CALL    L16C2           ; routine PF-DC-OUT considers fractional
part.

        LD      A,$2A           ; prepare character 'E'
        RST     10H             ; PRINT-A

        LD      A,B             ; transfer exponent to A
        AND     A               ; test the sign.
        JP      P,L1698         ; forward if positive to PF-E-POS

        NEG                     ; negate the negative exponent.
        LD      B,A             ; save positive exponent in B.

        LD      A,$16           ; prepare character '-'
        JR      L169A           ; skip forward to PF-E-SIGN

; ---

;; PF-E-POS
L1698:  LD      A,$15           ; prepare character '+'

;; PF-E-SIGN
L169A:  RST     10H             ; PRINT-A
```

```
; now convert the integer exponent in B to two characters.
; it will be less than 99.

        LD      A,B             ; fetch positive exponent.
        LD      B,$FF           ; initialize left hand digit to minus one.

;; PF-E-TENS
L169E:  INC     B               ; increment ten count
        SUB     $0A             ; subtract ten from exponent
        JR      NC,L169E        ; loop back if greater than ten to PF-E-
TENS

        ADD     A,$0A           ; reverse last subtraction
        LD      C,A             ; transfer remainder to C

        LD      A,B             ; transfer ten value to A.
        AND     A               ; test for zero.
        JR      Z,L16AD         ; skip forward if so to PF-E-LOW

        CALL    L07EB           ; routine OUT-CODE prints as digit '1' -
'9'

;; PF-E-LOW
L16AD:  LD      A,C             ; low byte to A
        CALL    L07EB           ; routine OUT-CODE prints final digit of
the
                                ; exponent.
        RET                     ; return.                               >>

; ---

; this branch deals with zeros after decimal point.
; e.g.      .01 or .0000999

;; PF-ZEROS
L16B2:  NEG                     ; negate makes number positive 1 to 4.
        LD      B,A             ; zero count to B.

        LD      A,$1B           ; prepare character '.'
        RST     10H             ; PRINT-A

        LD      A,$1C           ; prepare a '0'

;; PF-ZRO-LP
L16BA:  RST     10H             ; PRINT-A
        DJNZ    L16BA           ; loop back to PF-ZRO-LP

        JR      L16C8           ; forward to PF-FRAC-LP

; ---

; there is  a need to print a leading zero e.g. 0.1 but not with .01

;; PF-ZERO-1
L16BF:  LD      A,$1C           ; prepare character '0'.
        RST     10H             ; PRINT-A

; this subroutine considers the decimal point and any trailing digits.
; if the next character is a marked zero, $80, then nothing more to print.

;; PF-DC-OUT
```

```
L16C2:  DEC     (HL)            ; decrement addressed character
        INC     (HL)            ; increment it again
        RET     PE              ; return with overflow  (was 128) >>
                                ; as no fractional part

; else there is a fractional part so print the decimal point.

        LD      A,$1B           ; prepare character '.'
        RST     10H             ; PRINT-A

; now enter a loop to print trailing digits

;; PF-FRAC-LP
L16C8:  DEC     (HL)            ; test for a marked zero.
        INC     (HL)            ;
        RET     PE              ; return when digits exhausted          >>

        CALL    L16D0           ; routine PF-NIBBLE
        JR      L16C8           ; back for all fractional digits to PF-
FRAC-LP.

; ---

; subroutine to print right-hand nibble

;; PF-NIBBLE
L16D0:  LD      A,(HL)          ; fetch addressed byte
        AND     $0F             ; mask off lower 4 bits
        CALL    L07EB           ; routine OUT-CODE
        DEC     HL              ; decrement pointer.
        RET                     ; return.


; -------------------------------
; THE 'PREPARE TO ADD' SUBROUTINE
; -------------------------------
; This routine is called twice to prepare each floating point number for
; addition, in situ, on the calculator stack.
; The exponent is picked up from the first byte which is then cleared to
act
; as a sign byte and accept any overflow.
; If the exponent is zero then the number is zero and an early return is
made.
; The now redundant sign bit of the mantissa is set and if the number is
; negative then all five bytes of the number are twos-complemented to
prepare
; the number for addition.
; On the second invocation the exponent of the first number is in B.


;; PREP-ADD
L16D8:  LD      A,(HL)          ; fetch exponent.
        LD      (HL),$00        ; make this byte zero to take any overflow
and
                                ; default to positive.
        AND     A               ; test stored exponent for zero.
        RET     Z               ; return with zero flag set if number is
zero.

        INC     HL              ; point to first byte of mantissa.
        BIT     7,(HL)          ; test the sign bit.
        SET     7,(HL)          ; set it to its implied state.
```

```
        DEC     HL              ; set pointer to first byte again.
        RET     Z               ; return if bit indicated number is
positive.>>

; if negative then all five bytes are twos complemented starting at LSB.

        PUSH    BC              ; save B register contents.
        LD      BC,$0005        ; set BC to five.
        ADD     HL,BC           ; point to location after 5th byte.
        LD      B,C             ; set the B counter to five.
        LD      C,A             ; store original exponent in C.
        SCF                     ; set carry flag so that one is added.

; now enter a loop to twos-complement the number.
; The first of the five bytes becomes $FF to denote a negative number.

;; NEG-BYTE
L16EC:  DEC     HL              ; point to first or more significant byte.
        LD      A,(HL)          ; fetch to accumulator.
        CPL                     ; complement.
        ADC     A,$00           ; add in initial carry or any subsequent
carry.
        LD      (HL),A          ; place number back.
        DJNZ    L16EC           ; loop back five times to NEG-BYTE

        LD      A,C             ; restore the exponent to accumulator.
        POP     BC              ; restore B register contents.

        RET                     ; return.

; ---------------------------------
; THE 'FETCH TWO NUMBERS' SUBROUTINE
; ---------------------------------
; This routine is used by addition, multiplication and division to fetch
; the two five-byte numbers addressed by HL and DE from the calculator
stack
; into the Z80 registers.
; The HL register may no longer point to the first of the two numbers.
; Since the 32-bit addition operation is accomplished using two Z80 16-bit
; instructions, it is important that the lower two bytes of each mantissa
are
; in one set of registers and the other bytes all in the alternate set.
;
; In: HL = highest number, DE= lowest number
;
;         : alt':   :
; Out:    :H,B-C:C,B: num1
;         :L,D-E:D-E: num2

;; FETCH-TWO
L16F7:  PUSH    HL              ; save HL
        PUSH    AF              ; save A - result sign when used from
division.

        LD      C,(HL)          ;
        INC     HL              ;
        LD      B,(HL)          ;
        LD      (HL),A          ; insert sign when used from
multiplication.
        INC     HL              ;
        LD      A,C             ; m1
        LD      C,(HL)          ;
```

```
         PUSH    BC              ; PUSH m2 m3

         INC     HL              ;
         LD      C,(HL)          ; m4
         INC     HL              ;
         LD      B,(HL)          ; m5  BC holds m5 m4

         EX      DE,HL           ; make HL point to start of second number.

         LD      D,A             ; m1
         LD      E,(HL)          ;
         PUSH    DE              ; PUSH m1 n1

         INC     HL              ;
         LD      D,(HL)          ;
         INC     HL              ;
         LD      E,(HL)          ;
         PUSH    DE              ; PUSH n2 n3

         EXX                     ; - - - - - - -

         POP     DE              ; POP n2 n3
         POP     HL              ; POP m1 n1
         POP     BC              ; POP m2 m3

         EXX                     ; - - - - - - -

         INC     HL              ;
         LD      D,(HL)          ;
         INC     HL              ;
         LD      E,(HL)          ; DE holds n4 n5

         POP     AF              ; restore saved
         POP     HL              ; registers.
         RET                     ; return.

; -----------------------------
; THE 'SHIFT ADDEND' SUBROUTINE
; -----------------------------
; The accumulator A contains the difference between the two exponents.
; This is the lowest of the two numbers to be added

;; SHIFT-FP
L171A:   AND     A               ; test difference between exponents.
         RET     Z               ; return if zero. both normal.

         CP      $21             ; compare with 33 bits.
         JR      NC,L1736        ; forward if greater than 32 to ADDEND-0

         PUSH    BC              ; preserve BC - part
         LD      B,A             ; shift counter to B.

; Now perform B right shifts on the addend  L'D'E'D E
; to bring it into line with the augend     H'B'C'C B

;; ONE-SHIFT
L1722:   EXX                     ; - - -
         SRA     L               ;    76543210->C    bit 7 unchanged.
         RR      D               ; C->76543210->C
         RR      E               ; C->76543210->C
         EXX                     ; - - -
         RR      D               ; C->76543210->C
```

```
        RR      E               ; C->76543210->C
        DJNZ    L1722           ; loop back B times to ONE-SHIFT

        POP     BC              ; restore BC
        RET     NC              ; return if last shift produced no carry.
>>

; if carry flag was set then accuracy is being lost so round up the addend.

        CALL    L1741           ; routine ADD-BACK
        RET     NZ              ; return if not FF 00 00 00 00

; this branch makes all five bytes of the addend zero and is made during
; addition when the exponents are too far apart for the addend bits to
; affect the result.

;; ADDEND-0
L1736:  EXX                     ; select alternate set for more significant
                                ; bytes.
        XOR     A               ; clear accumulator.


; this entry point (from multiplication) sets four of the bytes to zero or
if
; continuing from above, during addition, then all five bytes are set to
zero.

;; ZEROS-4/5
L1738:  LD      L,$00           ; set byte 1 to zero.
        LD      D,A             ; set byte 2 to A.
        LD      E,L             ; set byte 3 to zero.
        EXX                     ; select main set
        LD      DE,$0000        ; set lower bytes 4 and 5 to zero.
        RET                     ; return.

; -------------------------
; THE 'ADD-BACK' SUBROUTINE
; -------------------------
; Called from SHIFT-FP above during addition and after normalization from
; multiplication.
; This is really a 32-bit increment routine which sets the zero flag
according
; to the 32-bit result.
; During addition, only negative numbers like FF FF FF FF FF,
; the twos-complement version of xx 80 00 00 01 say
; will result in a full ripple FF 00 00 00 00.
; FF FF FF FF FF when shifted right is unchanged by SHIFT-FP but sets the
; carry invoking this routine.

;; ADD-BACK
L1741:  INC     E               ;
        RET     NZ              ;

        INC     D               ;
        RET     NZ              ;

        EXX                     ;
        INC     E               ;
        JR      NZ,L174A        ; forward if no overflow to ALL-ADDED

        INC     D               ;
```

```
;; ALL-ADDED
L174A:  EXX                     ;
        RET                     ; return with zero flag set for zero
mantissa.


; ---------------------------
; THE 'SUBTRACTION' OPERATION
; ---------------------------
; just switch the sign of subtrahend and do an add.

;; subtract
L174C:  LD      A,(DE)          ; fetch exponent byte of second number the
                                ; subtrahend.
        AND     A               ; test for zero
        RET     Z               ; return if zero - first number is result.

        INC     DE              ; address the first mantissa byte.
        LD      A,(DE)          ; fetch to accumulator.
        XOR     $80             ; toggle the sign bit.
        LD      (DE),A          ; place back on calculator stack.
        DEC     DE              ; point to exponent byte.
                                ; continue into addition routine.

; ------------------------
; THE 'ADDITION' OPERATION
; ------------------------
; The addition operation pulls out all the stops and uses most of the Z80's
; registers to add two floating-point numbers.
; This is a binary operation and on entry, HL points to the first number
; and DE to the second.

;; addition
L1755:  EXX                     ; - - -
        PUSH    HL              ; save the pointer to the next literal.
        EXX                     ; - - -

        PUSH    DE              ; save pointer to second number
        PUSH    HL              ; save pointer to first number - will be
the
                                ; result pointer on calculator stack.

        CALL    L16D8           ; routine PREP-ADD
        LD      B,A             ; save first exponent byte in B.
        EX      DE,HL           ; switch number pointers.
        CALL    L16D8           ; routine PREP-ADD
        LD      C,A             ; save second exponent byte in C.
        CP      B               ; compare the exponent bytes.
        JR      NC,L1769        ; forward if second higher to SHIFT-LEN

        LD      A,B             ; else higher exponent to A
        LD      B,C             ; lower exponent to B
        EX      DE,HL           ; switch the number pointers.

;; SHIFT-LEN
L1769:  PUSH    AF              ; save higher exponent
        SUB     B               ; subtract lower exponent

        CALL    L16F7           ; routine FETCH-TWO
        CALL    L171A           ; routine SHIFT-FP

        POP     AF              ; restore higher exponent.
```

```
        POP     HL              ; restore result pointer.
        LD      (HL),A          ; insert exponent byte.
        PUSH    HL              ; save result pointer again.

; now perform the 32-bit addition using two 16-bit Z80 add instructions.

        LD      L,B             ; transfer low bytes of mantissa
individually
        LD      H,C             ; to HL register

        ADD     HL,DE           ; the actual binary addition of lower bytes

; now the two higher byte pairs that are in the alternate register sets.

        EXX                     ; switch in set
        EX      DE,HL           ; transfer high mantissa bytes to HL
register.

        ADC     HL,BC           ; the actual addition of higher bytes with
                                ; any carry from first stage.

        EX      DE,HL           ; result in DE, sign bytes ($FF or $00) to
HL

; now consider the two sign bytes

        LD      A,H             ; fetch sign byte of num1

        ADC     A,L             ; add including any carry from mantissa
                                ; addition. 00 or 01 or FE or FF

        LD      L,A             ; result in L.

; possible outcomes of signs and overflow from mantissa are
;
;  H +  L + carry =  L    RRA  XOR L  RRA
; ----------------------------------------------------------
; 00 + 00         = 00    00   00
; 00 + 00 + carry = 01    00   01     carry
; FF + FF         = FE C  FF   01     carry
; FF + FF + carry = FF C  FF   00
; FF + 00         = FF    FF   00
; FF + 00 + carry = 00 C  80   80

        RRA                     ; C->76543210->C
        XOR     L               ; set bit 0 if shifting required.

        EXX                     ; switch back to main set
        EX      DE,HL           ; full mantissa result now in D'E'D E
registers.
        POP     HL              ; restore pointer to result exponent on
                                ; the calculator stack.

        RRA                     ; has overflow occurred ?
        JR      NC,L1790        ; skip forward if not to TEST-NEG

; if the addition of two positive mantissas produced overflow or if the
; addition of two negative mantissas did not then the result exponent has
to
; be incremented and the mantissa shifted one place to the right.

        LD      A,$01           ; one shift required.
```

```
        CALL    L171A           ; routine SHIFT-FP performs a single shift
                                ; rounding any lost bit
        INC     (HL)            ; increment the exponent.
        JR      Z,L17B3         ; forward to ADD-REP-6 if the exponent
                                ; wraps round from FF to zero as number is
too
                                ; big for the system.

; at this stage the exponent on the calculator stack is correct.

;; TEST-NEG
L1790:  EXX                     ; switch in the alternate set.
        LD      A,L             ; load result sign to accumulator.
        AND     $80             ; isolate bit 7 from sign byte setting zero
                                ; flag if positive.
        EXX                     ; back to main set.

        INC     HL              ; point to first byte of mantissa
        LD      (HL),A          ; insert $00 positive or $80 negative at
                                ; position on calculator stack.

        DEC     HL              ; point to exponent again.
        JR      Z,L17B9         ; forward if positive to GO-NC-MLT

; a negative number has to be twos-complemented before being placed on
stack.

        LD      A,E             ; fetch lowest (rightmost) mantissa byte.
        NEG                     ; Negate
        CCF                     ; Complement Carry Flag
        LD      E,A             ; place back in register

        LD      A,D             ; ditto
        CPL                     ;
        ADC     A,$00           ;
        LD      D,A             ;

        EXX                     ; switch to higher (leftmost) 16 bits.

        LD      A,E             ; ditto
        CPL                     ;
        ADC     A,$00           ;
        LD      E,A             ;

        LD      A,D             ; ditto
        CPL                     ;
        ADC     A,$00           ;
        JR      NC,L17B7        ; forward without overflow to END-COMPL

; else entire mantissa is now zero.  00 00 00 00

        RRA                     ; set mantissa to 80 00 00 00
        EXX                     ; switch.
        INC     (HL)            ; increment the exponent.

;; ADD-REP-6
L17B3:  JP      Z,L1880         ; jump forward if exponent now zero to
REPORT-6
                                ; 'Number too big'

        EXX                     ; switch back to alternate set.
```

```
;; END-COMPL
L17B7:  LD      D,A             ; put first byte of mantissa back in DE.
        EXX                     ; switch to main set.

;; GO-NC-MLT
L17B9:  XOR     A               ; clear carry flag and
                                ; clear accumulator so no extra bits
carried
                                ; forward as occurs in multiplication.

        JR      L1828           ; forward to common code at TEST-NORM
                                ; but should go straight to NORMALIZE.


; -----------------------------------------------
; THE 'PREPARE TO MULTIPLY OR DIVIDE' SUBROUTINE
; -----------------------------------------------
; this routine is called twice from multiplication and twice from division
; to prepare each of the two numbers for the operation.
; Initially the accumulator holds zero and after the second invocation bit
7
; of the accumulator will be the sign bit of the result.

;; PREP-M/D
L17BC:  SCF                     ; set carry flag to signal number is zero.
        DEC     (HL)            ; test exponent
        INC     (HL)            ; for zero.
        RET     Z               ; return if zero with carry flag set.

        INC     HL              ; address first mantissa byte.
        XOR     (HL)            ; exclusive or the running sign bit.
        SET     7,(HL)          ; set the implied bit.
        DEC     HL              ; point to exponent byte.
        RET                     ; return.

; ------------------------------
; THE 'MULTIPLICATION' OPERATION
; ------------------------------
;
;

;; multiply
L17C6:  XOR     A               ; reset bit 7 of running sign flag.
        CALL    L17BC           ; routine PREP-M/D
        RET     C               ; return if number is zero.
                                ; zero * anything = zero.

        EXX                     ; - - -
        PUSH    HL              ; save pointer to 'next literal'
        EXX                     ; - - -

        PUSH    DE              ; save pointer to second number

        EX      DE,HL           ; make HL address second number.

        CALL    L17BC           ; routine PREP-M/D

        EX      DE,HL           ; HL first number, DE - second number
        JR      C,L1830         ; forward with carry to ZERO-RSLT
                                ; anything * zero = zero.

        PUSH    HL              ; save pointer to first number.
```

```
        CALL    L16F7           ; routine FETCH-TWO fetches two mantissas
from
                                ; calc stack to B'C'C,B  D'E'D E
                                ; (HL will be overwritten but the result
sign
                                ; in A is inserted on the calculator stack)

        LD      A,B             ; transfer low mantissa byte of first
number
        AND     A               ; clear carry.
        SBC     HL,HL           ; a short form of LD HL,$0000 to take lower
                                ; two bytes of result. (2 program bytes)
        EXX                     ; switch in alternate set
        PUSH    HL              ; preserve HL
        SBC     HL,HL           ; set HL to zero also to take higher two
bytes
                                ; of the result and clear carry.
        EXX                     ; switch back.

        LD      B,$21           ; register B can now be used to count
thirty
                                ; three shifts.
        JR      L17F8           ; forward to loop entry point STRT-MLT

; ---

; The multiplication loop is entered at  STRT-LOOP.

;; MLT-LOOP
L17E7:  JR      NC,L17EE        ; forward if no carry to NO-ADD

                                ; else add in the multiplicand.

        ADD     HL,DE           ; add the two low bytes to result
        EXX                     ; switch to more significant bytes.
        ADC     HL,DE           ; add high bytes of multiplicand and any
carry.
        EXX                     ; switch to main set.

; in either case shift result right into B'C'C A

;; NO-ADD
L17EE:  EXX                     ; switch to alternate set
        RR      H               ; C > 76543210 > C
        RR      L               ; C > 76543210 > C
        EXX                     ;
        RR      H               ; C > 76543210 > C
        RR      L               ; C > 76543210 > C

;; STRT-MLT
L17F8:  EXX                     ; switch in alternate set.
        RR      B               ; C > 76543210 > C
        RR      C               ; C > 76543210 > C
        EXX                     ; now main set
        RR      C               ; C > 76543210 > C
        RRA                     ; C > 76543210 > C
        DJNZ    L17E7           ; loop back 33 times to MLT-LOOP

;

        EX      DE,HL           ;
```

```
        EXX                     ;
        EX      DE,HL           ;
        EXX                     ;
        POP     BC              ;
        POP     HL              ;
        LD      A,B             ;
        ADD     A,C             ;
        JR      NZ,L180E        ; forward to MAKE-EXPT

        AND     A               ;

;; MAKE-EXPT
L180E:  DEC     A               ;
        CCF                     ; Complement Carry Flag

;; DIVN-EXPT
L1810:  RLA                     ;
        CCF                     ; Complement Carry Flag
        RRA                     ;
        JP      P,L1819         ; forward to OFLW1-CLR

        JR      NC,L1880        ; forward to REPORT-6

        AND     A               ;

;; OFLW1-CLR
L1819:  INC     A               ;
        JR      NZ,L1824        ; forward to OFLW2-CLR

        JR      C,L1824         ; forward to OFLW2-CLR

        EXX                     ;
        BIT     7,D             ;
        EXX                     ;
        JR      NZ,L1880        ; forward to REPORT-6

;; OFLW2-CLR
L1824:  LD      (HL),A          ;
        EXX                     ;
        LD      A,B             ;
        EXX                     ;

; addition joins here with carry flag clear.

;; TEST-NORM
L1828:  JR      NC,L183F        ; forward to NORMALIZE

        LD      A,(HL)          ;
        AND     A               ;

;; NEAR-ZERO
L182C:  LD      A,$80           ; prepare to rescue the most significant
bit
                                ; of the mantissa if it is set.
        JR      Z,L1831         ; skip forward to SKIP-ZERO

;; ZERO-RSLT
L1830:  XOR     A               ; make mask byte zero signaling set five
                                ; bytes to zero.

;; SKIP-ZERO
L1831:  EXX                     ; switch in alternate set
```

```
        AND     D               ; isolate most significant bit (if A is
$80).

        CALL    L1738           ; routine ZEROS-4/5 sets mantissa without
                                ; affecting any flags.

        RLCA                    ; test if MSB set. bit 7 goes to bit 0.
                                ; either $00 -> $00 or $80 -> $01
        LD      (HL),A          ; make exponent $01 (lowest) or $00 zero
        JR      C,L1868         ; forward if first case to OFLOW-CLR

        INC     HL              ; address first mantissa byte on the
                                ; calculator stack.
        LD      (HL),A          ; insert a zero for the sign bit.
        DEC     HL              ; point to zero exponent
        JR      L1868           ; forward to OFLOW-CLR

; ---

; this branch is common to addition and multiplication with the mantissa
; result still in registers D'E'D E .

;; NORMALIZE
L183F:  LD      B,$20           ; a maximum of thirty-two left shifts will
be
                                ; needed.

;; SHIFT-ONE
L1841:  EXX                     ; address higher 16 bits.
        BIT     7,D             ; test the leftmost bit
        EXX                     ; address lower 16 bits.

        JR      NZ,L1859        ; forward if leftmost bit was set to NORML-
NOW

        RLCA                    ; this holds zero from addition, 33rd bit
                                ; from multiplication.

        RL      E               ; C < 76543210 < C
        RL      D               ; C < 76543210 < C

        EXX                     ; address higher 16 bits.

        RL      E               ; C < 76543210 < C
        RL      D               ; C < 76543210 < C

        EXX                     ; switch to main set.

        DEC     (HL)            ; decrement the exponent byte on the
calculator
                                ; stack.

        JR      Z,L182C         ; back if exponent becomes zero to NEAR-
ZERO
                                ; it's just possible that the last rotation
                                ; set bit 7 of D. We shall see.

        DJNZ    L1841           ; loop back to SHIFT-ONE

; if thirty-two left shifts were performed without setting the most
significant
; bit then the result is zero.
```

```
        JR      L1830           ; back to ZERO-RSLT

; ---

;; NORML-NOW
L1859:  RLA                     ; for the addition path, A is always zero.
                                ; for the mult path, ...

        JR      NC,L1868        ; forward to OFLOW-CLR

; this branch is taken only with multiplication.

        CALL    L1741           ; routine ADD-BACK

        JR      NZ,L1868        ; forward to OFLOW-CLR

        EXX                     ;
        LD      D,$80           ;
        EXX                     ;
        INC     (HL)            ;
        JR      Z,L1880         ; forward to REPORT-6

; now transfer the mantissa from the register sets to the calculator stack
; incorporating the sign bit already there.

;; OFLOW-CLR
L1868:  PUSH    HL              ; save pointer to exponent on stack.
        INC     HL              ; address first byte of mantissa which was
                                ; previously loaded with sign bit $00 or
$80.

        EXX                     ; - - -
        PUSH    DE              ; push the most significant two bytes.
        EXX                     ; - - -

        POP     BC              ; pop - true mantissa is now BCDE.

; now pick up the sign bit.

        LD      A,B             ; first mantissa byte to A
        RLA                     ; rotate out bit 7 which is set
        RL      (HL)            ; rotate sign bit on stack into carry.
        RRA                     ; rotate sign bit into bit 7 of mantissa.

; and transfer mantissa from main registers to calculator stack.

        LD      (HL),A          ;
        INC     HL              ;
        LD      (HL),C          ;
        INC     HL              ;
        LD      (HL),D          ;
        INC     HL              ;
        LD      (HL),E          ;

        POP     HL              ; restore pointer to num1 now result.
        POP     DE              ; restore pointer to num2 now STKEND.

        EXX                     ; - - -
        POP     HL              ; restore pointer to next calculator
literal.
        EXX                     ; - - -
```

```
        RET                     ; return.

; ---

;; REPORT-6
L1880:  RST     08H             ; ERROR-1
        DEFB    $05             ; Error Report: Arithmetic overflow.

; ------------------------
; THE 'DIVISION' OPERATION
; ------------------------
;   "Of all the arithmetic subroutines, division is the most complicated
and
;   the least understood.  It is particularly interesting to note that the
;   Sinclair programmer himself has made a mistake in his programming ( or
has
;   copied over someone else's mistake!) for
;   PRINT PEEK 6352 [ $18D0 ] ('unimproved' ROM, 6351 [ $18CF ] )
;   should give 218 not 225."
;   - Dr. Ian Logan, Syntax magazine Jul/Aug 1982.
;   [  i.e. the jump should be made to div-34th ]

;   First check for division by zero.

;; division
L1882:  EX      DE,HL           ; consider the second number first.
        XOR     A               ; set the running sign flag.
        CALL    L17BC           ; routine PREP-M/D
        JR      C,L1880         ; back if zero to REPORT-6
                                ; 'Arithmetic overflow'

        EX      DE,HL           ; now prepare first number and check for
zero.
        CALL    L17BC           ; routine PREP-M/D
        RET     C               ; return if zero, 0/anything is zero.

        EXX                     ; - - -
        PUSH    HL              ; save pointer to the next calculator
literal.
        EXX                     ; - - -

        PUSH    DE              ; save pointer to divisor - will be STKEND.
        PUSH    HL              ; save pointer to dividend - will be
result.

        CALL    L16F7           ; routine FETCH-TWO fetches the two numbers
                                ; into the registers H'B'C'C B
                                ;                    L'D'E'D E
        EXX                     ; - - -
        PUSH    HL              ; save the two exponents.

        LD      H,B             ; transfer the dividend to H'L'H L
        LD      L,C             ;
        EXX                     ;
        LD      H,C             ;
        LD      L,B             ;

        XOR     A               ; clear carry bit and accumulator.
        LD      B,$DF           ; count upwards from -33 decimal
        JR      L18B2           ; forward to mid-loop entry point DIV-START
```

```
; ---

;; DIV-LOOP
L18A2:  RLA                     ; multiply partial quotient by two
        RL      C               ; setting result bit from carry.
        EXX                     ;
        RL      C               ;
        RL      B               ;
        EXX                     ;

;; div-34th
L18AB:  ADD     HL,HL           ;
        EXX                     ;
        ADC     HL,HL           ;
        EXX                     ;
        JR      C,L18C2         ; forward to SUBN-ONLY

;; DIV-START
L18B2:  SBC     HL,DE           ; subtract divisor part.
        EXX                     ;
        SBC     HL,DE           ;
        EXX                     ;
        JR      NC,L18C9        ; forward if subtraction goes to NO-RSTORE

        ADD     HL,DE           ; else restore
        EXX                     ;
        ADC     HL,DE           ;
        EXX                     ;
        AND     A               ; clear carry
        JR      L18CA           ; forward to COUNT-ONE

; ---

;; SUBN-ONLY
L18C2:  AND     A               ;
        SBC     HL,DE           ;
        EXX                     ;
        SBC     HL,DE           ;
        EXX                     ;

;; NO-RSTORE
L18C9:  SCF                     ; set carry flag

;; COUNT-ONE
L18CA:  INC     B               ; increment the counter
        JP      M,L18A2         ; back while still minus to DIV-LOOP

        PUSH    AF              ;
        JR      Z,L18B2         ; back to DIV-START

; "This jump is made to the wrong place. No 34th bit will ever be obtained
; without first shifting the dividend. Hence important results like 1/10
and
; 1/1000 are not rounded up as they should be. Rounding up never occurs
when
; it depends on the 34th bit. The jump should be made to div-34th above."
; - Dr. Frank O'Hara, "The Complete Spectrum ROM Disassembly", 1983,
; published by Melbourne House.
; (Note. on the ZX81 this would be JR Z,L18AB)
;
; However if you make this change, then while (1/2=.5) will now evaluate as
; true, (.25=1/4), which did evaluate as true, no longer does.
```

```
        LD      E,A             ;
        LD      D,C             ;
        EXX                     ;
        LD      E,C             ;
        LD      D,B             ;

        POP     AF              ;
        RR      B               ;
        POP     AF              ;
        RR      B               ;

        EXX                     ;
        POP     BC              ;
        POP     HL              ;
        LD      A,B             ;
        SUB     C               ;
        JP      L1810           ; jump back to DIVN-EXPT

; -----------------------------------------------
; THE 'INTEGER TRUNCATION TOWARDS ZERO' SUBROUTINE
; -----------------------------------------------
;

;; truncate
L18E4:  LD      A,(HL)          ; fetch exponent
        CP      $81             ; compare to +1
        JR      NC,L18EF        ; forward, if 1 or more, to T-GR-ZERO

; else the number is smaller than plus or minus 1 and can be made zero.

        LD      (HL),$00        ; make exponent zero.
        LD      A,$20           ; prepare to set 32 bits of mantissa to
zero.
        JR      L18F4           ; forward to NIL-BYTES

; ---

;; T-GR-ZERO
L18EF:  SUB     $A0             ; subtract +32 from exponent
        RET     P               ; return if result is positive as all 32
bits
                                ; of the mantissa relate to the integer
part.
                                ; The floating point is somewhere to the
right
                                ; of the mantissa

        NEG                     ; else negate to form number of rightmost
bits
                                ; to be blanked.

; for instance, disregarding the sign bit, the number 3.5 is held as
; exponent $82 mantissa .11100000 00000000 00000000 00000000
; we need to set $82 - $A0 = $E2 NEG = $1E (thirty) bits to zero to form
the
; integer.
; The sign of the number is never considered as the first bit of the
mantissa
; must be part of the integer.

;; NIL-BYTES
```

```
L18F4:  PUSH    DE              ; save pointer to STKEND
        EX      DE,HL           ; HL points at STKEND
        DEC     HL              ; now at last byte of mantissa.
        LD      B,A             ; Transfer bit count to B register.
        SRL     B               ; divide by
        SRL     B               ; eight
        SRL     B               ;
        JR      Z,L1905         ; forward if zero to BITS-ZERO

; else the original count was eight or more and whole bytes can be blanked.

;; BYTE-ZERO
L1900:  LD      (HL),$00        ; set eight bits to zero.
        DEC     HL              ; point to more significant byte of
mantissa.
        DJNZ    L1900           ; loop back to BYTE-ZERO

; now consider any residual bits.

;; BITS-ZERO
L1905:  AND     $07             ; isolate the remaining bits
        JR      Z,L1912         ; forward if none to IX-END

        LD      B,A             ; transfer bit count to B counter.
        LD      A,$FF           ; form a mask 11111111

;; LESS-MASK
L190C:  SLA     A               ; 1 <- 76543210 <- o     slide mask
leftwards.
        DJNZ    L190C           ; loop back for bit count to LESS-MASK

        AND     (HL)            ; lose the unwanted rightmost bits
        LD      (HL),A          ; and place in mantissa byte.

;; IX-END
L1912:  EX      DE,HL           ; restore result pointer from DE.
        POP     DE              ; restore STKEND from stack.
        RET                     ; return.


;*********************************
;**  FLOATING-POINT CALCULATOR **
;*********************************

; As a general rule the calculator avoids using the IY register.
; Exceptions are val and str$.
; So an assembly language programmer who has disabled interrupts to use IY
; for other purposes can still use the calculator for mathematical
; purposes.


; ------------------------
; THE 'TABLE OF CONSTANTS'
; ------------------------
; The ZX81 has only floating-point number representation.
; Both the ZX80 and the ZX Spectrum have integer numbers in some form.

;; stk-zero                                              00 00 00 00 00
L1915:  DEFB    $00             ;;Bytes: 1
        DEFB    $B0             ;;Exponent $00
        DEFB    $00             ;;(+00,+00,+00)
```

```
;; stk-one                                                  81 00 00 00 00
L1918:  DEFB    $31             ;;Exponent $81, Bytes: 1
        DEFB    $00             ;;(+00,+00,+00)


;; stk-half                                                 80 00 00 00 00
L191A:  DEFB    $30             ;;Exponent: $80, Bytes: 1
        DEFB    $00             ;;(+00,+00,+00)


;; stk-pi/2                                                 81 49 0F DA A2
L191C:  DEFB    $F1             ;;Exponent: $81, Bytes: 4
        DEFB    $49,$0F,$DA,$A2 ;;

;; stk-ten                                                  84 20 00 00 00
L1921:  DEFB    $34             ;;Exponent: $84, Bytes: 1
        DEFB    $20             ;;(+00,+00,+00)


; ------------------------
; THE 'TABLE OF ADDRESSES'
; ------------------------
;
; starts with binary operations which have two operands and one result.
; three pseudo binary operations first.

;; tbl-addrs
L1923:  DEFW    L1C2F           ; $00 Address: $1C2F - jump-true
        DEFW    L1A72           ; $01 Address: $1A72 - exchange
        DEFW    L19E3           ; $02 Address: $19E3 - delete

; true binary operations.

        DEFW    L174C           ; $03 Address: $174C - subtract
        DEFW    L17C6           ; $04 Address: $176C - multiply
        DEFW    L1882           ; $05 Address: $1882 - division
        DEFW    L1DE2           ; $06 Address: $1DE2 - to-power
        DEFW    L1AED           ; $07 Address: $1AED - or

        DEFW    L1AF3           ; $08 Address: $1B03 - no-&-no
        DEFW    L1B03           ; $09 Address: $1B03 - no-l-eql
        DEFW    L1B03           ; $0A Address: $1B03 - no-gr-eql
        DEFW    L1B03           ; $0B Address: $1B03 - nos-neql
        DEFW    L1B03           ; $0C Address: $1B03 - no-grtr
        DEFW    L1B03           ; $0D Address: $1B03 - no-less
        DEFW    L1B03           ; $0E Address: $1B03 - nos-eql
        DEFW    L1755           ; $0F Address: $1755 - addition

        DEFW    L1AF8           ; $10 Address: $1AF8 - str-&-no
        DEFW    L1B03           ; $11 Address: $1B03 - str-l-eql
        DEFW    L1B03           ; $12 Address: $1B03 - str-gr-eql
        DEFW    L1B03           ; $13 Address: $1B03 - strs-neql
        DEFW    L1B03           ; $14 Address: $1B03 - str-grtr
        DEFW    L1B03           ; $15 Address: $1B03 - str-less
        DEFW    L1B03           ; $16 Address: $1B03 - strs-eql
        DEFW    L1B62           ; $17 Address: $1B62 - strs-add

; unary follow

        DEFW    L1AA0           ; $18 Address: $1AA0 - neg

        DEFW    L1C06           ; $19 Address: $1C06 - code
```

```
        DEFW    L1BA4           ; $1A Address: $1BA4 - val
        DEFW    L1C11           ; $1B Address: $1C11 - len
        DEFW    L1D49           ; $1C Address: $1D49 - sin
        DEFW    L1D3E           ; $1D Address: $1D3E - cos
        DEFW    L1D6E           ; $1E Address: $1D6E - tan
        DEFW    L1DC4           ; $1F Address: $1DC4 - asn
        DEFW    L1DD4           ; $20 Address: $1DD4 - acs
        DEFW    L1D76           ; $21 Address: $1D76 - atn
        DEFW    L1CA9           ; $22 Address: $1CA9 - ln
        DEFW    L1C5B           ; $23 Address: $1C5B - exp
        DEFW    L1C46           ; $24 Address: $1C46 - int
        DEFW    L1DDB           ; $25 Address: $1DDB - sqr
        DEFW    L1AAF           ; $26 Address: $1AAF - sgn
        DEFW    L1AAA           ; $27 Address: $1AAA - abs
        DEFW    L1ABE           ; $28 Address: $1A1B - peek
        DEFW    L1AC5           ; $29 Address: $1AC5 - usr-no
        DEFW    L1BD5           ; $2A Address: $1BD5 - str$
        DEFW    L1B8F           ; $2B Address: $1B8F - chrs
        DEFW    L1AD5           ; $2C Address: $1AD5 - not

; end of true unary

        DEFW    L19F6           ; $2D Address: $19F6 - duplicate
        DEFW    L1C37           ; $2E Address: $1C37 - n-mod-m

        DEFW    L1C23           ; $2F Address: $1C23 - jump
        DEFW    L19FC           ; $30 Address: $19FC - stk-data

        DEFW    L1C17           ; $31 Address: $1C17 - dec-jr-nz
        DEFW    L1ADB           ; $32 Address: $1ADB - less-0
        DEFW    L1ACE           ; $33 Address: $1ACE - greater-0
        DEFW    L002B           ; $34 Address: $002B - end-calc
        DEFW    L1D18           ; $35 Address: $1D18 - get-argt
        DEFW    L18E4           ; $36 Address: $18E4 - truncate
        DEFW    L19E4           ; $37 Address: $19E4 - fp-calc-2
        DEFW    L155A           ; $38 Address: $155A - e-to-fp

; the following are just the next available slots for the 128 compound
literals
; which are in range $80 - $FF.

        DEFW    L1A7F           ; $39 Address: $1A7F - series-xx    $80 -
$9F.
        DEFW    L1A51           ; $3A Address: $1A51 - stk-const-xx $A0 -
$BF.
        DEFW    L1A63           ; $3B Address: $1A63 - st-mem-xx    $C0 -
$DF.
        DEFW    L1A45           ; $3C Address: $1A45 - get-mem-xx   $E0 -
$FF.

; Aside: 3D - 7F are therefore unused calculator literals.
;        39 - 7B would be available for expansion.

; -------------------------------
; THE 'FLOATING POINT CALCULATOR'
; -------------------------------
;
;

;; CALCULATE
L199D:  CALL    L1B85           ; routine STK-PNTRS is called to set up the
                                ; calculator stack pointers for a default
```

```
                                ; unary operation. HL = last value on
stack.
                                ; DE = STKEND first location after stack.

; the calculate routine is called at this point by the series generator...

;; GEN-ENT-1
L19A0:  LD      A,B             ; fetch the Z80 B register to A
        LD      ($401E),A       ; and store value in system variable BREG.
                                ; this will be the counter for dec-jr-nz
                                ; or if used from fp-calc2 the calculator
                                ; instruction.

; ... and again later at this point

;; GEN-ENT-2
L19A4:  EXX                     ; switch sets
        EX      (SP),HL         ; and store the address of next
instruction,
                                ; the return address, in H'L'.
                                ; If this is a recursive call then the H'L'
                                ; of the previous invocation goes on stack.
                                ; c.f. end-calc.
        EXX                     ; switch back to main set.

; this is the re-entry looping point when handling a string of literals.

;; RE-ENTRY
L19A7:  LD      ($401C),DE      ; save end of stack in system variable
STKEND
        EXX                     ; switch to alt
        LD      A,(HL)          ; get next literal
        INC     HL              ; increase pointer'

; single operation jumps back to here

;; SCAN-ENT
L19AE:  PUSH    HL              ; save pointer on stack   *
        AND     A               ; now test the literal
        JP      P,L19C2         ; forward to FIRST-3D if in range $00 - $3D
                                ; anything with bit 7 set will be one of
                                ; 128 compound literals.

; compound literals have the following format.
; bit 7 set indicates compound.
; bits 6-5 the subgroup 0-3.
; bits 4-0 the embedded parameter $00 - $1F.
; The subgroup 0-3 needs to be manipulated to form the next available four
; address places after the simple literals in the address table.

        LD      D,A             ; save literal in D
        AND     $60             ; and with 01100000 to isolate subgroup
        RRCA                    ; rotate bits
        RRCA                    ; 4 places to right
        RRCA                    ; not five as we need offset * 2
        RRCA                    ; 00000xx0
        ADD     A,$72           ; add ($39 * 2) to give correct offset.
                                ; alter above if you add more literals.
        LD      L,A             ; store in L for later indexing.
        LD      A,D             ; bring back compound literal
        AND     $1F             ; use mask to isolate parameter bits
        JR      L19D0           ; forward to ENT-TABLE
```

```
; ---

; the branch was here with simple literals.

;; FIRST-3D
L19C2:  CP      $18             ; compare with first unary operations.
        JR      NC,L19CE        ; to DOUBLE-A with unary operations

; it is binary so adjust pointers.

        EXX                     ;
        LD      BC,$FFFB        ; the value -5
        LD      D,H             ; transfer HL, the last value, to DE.
        LD      E,L             ;
        ADD     HL,BC           ; subtract 5 making HL point to second
                                ; value.
        EXX                     ;

;; DOUBLE-A
L19CE:  RLCA                    ; double the literal
        LD      L,A             ; and store in L for indexing

;; ENT-TABLE
L19D0:  LD      DE,L1923        ; Address: tbl-addrs
        LD      H,$00           ; prepare to index
        ADD     HL,DE           ; add to get address of routine
        LD      E,(HL)          ; low byte to E
        INC     HL              ;
        LD      D,(HL)          ; high byte to D

        LD      HL,L19A7        ; Address: RE-ENTRY
        EX      (SP),HL         ; goes on machine stack
                                ; address of next literal goes to HL. *


        PUSH    DE              ; now the address of routine is stacked.
        EXX                     ; back to main set
                                ; avoid using IY register.
        LD      BC,($401D)      ; STKEND_hi
                                ; nothing much goes to C but BREG to B
                                ; and continue into next ret instruction
                                ; which has a dual identity


; -----------------------
; THE 'DELETE' SUBROUTINE
; -----------------------
; offset $02: 'delete'
; A simple return but when used as a calculator literal this
; deletes the last value from the calculator stack.
; On entry, as always with binary operations,
; HL=first number, DE=second number
; On exit, HL=result, DE=stkend.
; So nothing to do

;; delete
L19E3:  RET                     ; return - indirect jump if from above.

; --------------------------------
; THE 'SINGLE OPERATION' SUBROUTINE
; --------------------------------
```

```
; offset $37: 'fp-calc-2'
; this single operation is used, in the first instance, to evaluate most
; of the mathematical and string functions found in BASIC expressions.

;; fp-calc-2
L19E4:  POP     AF              ; drop return address.
        LD      A,($401E)       ; load accumulator from system variable
BREG
                                ; value will be literal eg. 'tan'
        EXX                     ; switch to alt
        JR      L19AE           ; back to SCAN-ENT
                                ; next literal will be end-calc in scanning

; ------------------------------
; THE 'TEST 5 SPACES' SUBROUTINE
; ------------------------------
; This routine is called from MOVE-FP, STK-CONST and STK-STORE to
; test that there is enough space between the calculator stack and the
; machine stack for another five-byte value. It returns with BC holding
; the value 5 ready for any subsequent LDIR.

;; TEST-5-SP
L19EB:  PUSH    DE              ; save
        PUSH    HL              ; registers
        LD      BC,$0005        ; an overhead of five bytes
        CALL    L0EC5           ; routine TEST-ROOM tests free RAM raising
                                ; an error if not.
        POP     HL              ; else restore
        POP     DE              ; registers.
        RET                     ; return with BC set at 5.


; -------------------------------------------
; THE 'MOVE A FLOATING POINT NUMBER' SUBROUTINE
; -------------------------------------------
; offset $2D: 'duplicate'
; This simple routine is a 5-byte LDIR instruction
; that incorporates a memory check.
; When used as a calculator literal it duplicates the last value on the
; calculator stack.
; Unary so on entry HL points to last value, DE to stkend

;; duplicate
;; MOVE-FP
L19F6:  CALL    L19EB           ; routine TEST-5-SP test free memory
                                ; and sets BC to 5.
        LDIR                    ; copy the five bytes.
        RET                     ; return with DE addressing new STKEND
                                ; and HL addressing new last value.

; -------------------------------
; THE 'STACK LITERALS' SUBROUTINE
; -------------------------------
; offset $30: 'stk-data'
; When a calculator subroutine needs to put a value on the calculator
; stack that is not a regular constant this routine is called with a
; variable number of following data bytes that convey to the routine
; the floating point form as succinctly as is possible.

;; stk-data
L19FC:  LD      H,D             ; transfer STKEND
        LD      L,E             ; to HL for result.
```

```
;; STK-CONST
L19FE:  CALL    L19EB           ; routine TEST-5-SP tests that room exists
                                ; and sets BC to $05.

        EXX                     ; switch to alternate set
        PUSH    HL              ; save the pointer to next literal on stack
        EXX                     ; switch back to main set

        EX      (SP),HL         ; pointer to HL, destination to stack.

        PUSH    BC              ; save BC - value 5 from test room ??.

        LD      A,(HL)          ; fetch the byte following 'stk-data'
        AND     $C0             ; isolate bits 7 and 6
        RLCA                    ; rotate
        RLCA                    ; to bits 1 and 0  range $00 - $03.
        LD      C,A             ; transfer to C
        INC     C               ; and increment to give number of bytes
                                ; to read. $01 - $04
        LD      A,(HL)          ; reload the first byte
        AND     $3F             ; mask off to give possible exponent.
        JR      NZ,L1A14        ; forward to FORM-EXP if it was possible to
                                ; include the exponent.

; else byte is just a byte count and exponent comes next.

        INC     HL              ; address next byte and
        LD      A,(HL)          ; pick up the exponent ( - $50).

;; FORM-EXP
L1A14:  ADD     A,$50           ; now add $50 to form actual exponent
        LD      (DE),A          ; and load into first destination byte.
        LD      A,$05           ; load accumulator with $05 and
        SUB     C               ; subtract C to give count of trailing
                                ; zeros plus one.
        INC     HL              ; increment source
        INC     DE              ; increment destination
        LD      B,$00           ; prepare to copy
        LDIR                    ; copy C bytes

        POP     BC              ; restore 5 counter to BC ??.

        EX      (SP),HL         ; put HL on stack as next literal pointer
                                ; and the stack value - result pointer -
                                ; to HL.

        EXX                     ; switch to alternate set.
        POP     HL              ; restore next literal pointer from stack
                                ; to H'L'.
        EXX                     ; switch back to main set.

        LD      B,A             ; zero count to B
        XOR     A               ; clear accumulator

;; STK-ZEROS
L1A27:  DEC     B               ; decrement B counter
        RET     Z               ; return if zero.          >>
                                ; DE points to new STKEND
                                ; HL to new number.

        LD      (DE),A          ; else load zero to destination
```

```
        INC     DE              ; increase destination
        JR      L1A27           ; loop back to STK-ZEROS until done.

; ------------------------------
; THE 'SKIP CONSTANTS' SUBROUTINE
; ------------------------------
; This routine traverses variable-length entries in the table of constants,
; stacking intermediate, unwanted constants onto a dummy calculator stack,
; in the first five bytes of the ZX81 ROM.

;; SKIP-CONS
L1A2D:  AND     A               ; test if initially zero.

;; SKIP-NEXT
L1A2E:  RET     Z               ; return if zero.          >>

        PUSH     AF             ; save count.
        PUSH    DE              ; and normal STKEND

        LD      DE,$0000        ; dummy value for STKEND at start of ROM
                                ; Note. not a fault but this has to be
                                ; moved elsewhere when running in RAM.
                                ;
        CALL    L19FE           ; routine STK-CONST works through variable
                                ; length records.

        POP     DE              ; restore real STKEND
        POP     AF              ; restore count
        DEC     A               ; decrease
        JR      L1A2E           ; loop back to SKIP-NEXT

; --------------------------------
; THE 'MEMORY LOCATION' SUBROUTINE
; --------------------------------
; This routine, when supplied with a base address in HL and an index in A,
; will calculate the address of the A'th entry, where each entry occupies
; five bytes. It is used for addressing floating-point numbers in the
; calculator's memory area.

;; LOC-MEM
L1A3C:  LD      C,A             ; store the original number $00-$1F.
        RLCA                    ; double.
        RLCA                    ; quadruple.
        ADD     A,C             ; now add original value to multiply by
five.

        LD      C,A             ; place the result in C.
        LD      B,$00           ; set B to 0.
        ADD     HL,BC           ; add to form address of start of number in
HL.

        RET                     ; return.

; -------------------------------------
; THE 'GET FROM MEMORY AREA' SUBROUTINE
; -------------------------------------
; offsets $E0 to $FF: 'get-mem-0', 'get-mem-1' etc.
; A holds $00-$1F offset.
; The calculator stack increases by 5 bytes.

;; get-mem-xx
L1A45:  PUSH    DE              ; save STKEND
```

```
        LD      HL,($401F)      ; MEM is base address of the memory cells.
        CALL    L1A3C           ; routine LOC-MEM so that HL = first byte
        CALL    L19F6           ; routine MOVE-FP moves 5 bytes with memory
                                ; check.
                                ; DE now points to new STKEND.
        POP     HL              ; the original STKEND is now RESULT
pointer.
        RET                     ; return.

; --------------------------------
; THE 'STACK A CONSTANT' SUBROUTINE
; --------------------------------
; offset $A0: 'stk-zero'
; offset $A1: 'stk-one'
; offset $A2: 'stk-half'
; offset $A3: 'stk-pi/2'
; offset $A4: 'stk-ten'
; This routine allows a one-byte instruction to stack up to 32 constants
; held in short form in a table of constants. In fact only 5 constants are
; required. On entry the A register holds the literal ANDed with $1F.
; It isn't very efficient and it would have been better to hold the
; numbers in full, five byte form and stack them in a similar manner
; to that which would be used later for semi-tone table values.

;; stk-const-xx
L1A51:  LD      H,D             ; save STKEND - required for result
        LD      L,E             ;
        EXX                     ; swap
        PUSH    HL              ; save pointer to next literal
        LD      HL,L1915        ; Address: stk-zero - start of table of
                                ; constants
        EXX                     ;
        CALL    L1A2D           ; routine SKIP-CONS
        CALL    L19FE           ; routine STK-CONST
        EXX                     ;
        POP     HL              ; restore pointer to next literal.
        EXX                     ;
        RET                     ; return.

; ---------------------------------------
; THE 'STORE IN A MEMORY AREA' SUBROUTINE
; ---------------------------------------
; Offsets $C0 to $DF: 'st-mem-0', 'st-mem-1' etc.
; Although 32 memory storage locations can be addressed, only six
; $C0 to $C5 are required by the ROM and only the thirty bytes (6*5)
; required for these are allocated. ZX81 programmers who wish to
; use the floating point routines from assembly language may wish to
; alter the system variable MEM to point to 160 bytes of RAM to have
; use the full range available.
; A holds derived offset $00-$1F.
; Unary so on entry HL points to last value, DE to STKEND.

;; st-mem-xx
L1A63:  PUSH    HL              ; save the result pointer.
        EX      DE,HL           ; transfer to DE.
        LD      HL,($401F)      ; fetch MEM the base of memory area.
        CALL    L1A3C           ; routine LOC-MEM sets HL to the
destination.
        EX      DE,HL           ; swap - HL is start, DE is destination.
        CALL    L19F6           ; routine MOVE-FP.
                                ; note. a short ld bc,5; ldir
                                ; the embedded memory check is not required
```

```
                                ; so these instructions would be faster!
        EX      DE,HL           ; DE = STKEND
        POP     HL              ; restore original result pointer
        RET                     ; return.

; -------------------------
; THE 'EXCHANGE' SUBROUTINE
; -------------------------
; offset $01: 'exchange'
; This routine exchanges the last two values on the calculator stack
; On entry, as always with binary operations,
; HL=first number, DE=second number
; On exit, HL=result, DE=stkend.

;; exchange
L1A72:  LD      B,$05           ; there are five bytes to be swapped

; start of loop.

;; SWAP-BYTE
L1A74:  LD      A,(DE)          ; each byte of second
        LD      C,(HL)          ; each byte of first
        EX      DE,HL           ; swap pointers
        LD      (DE),A          ; store each byte of first
        LD      (HL),C          ; store each byte of second
        INC     HL              ; advance both
        INC     DE              ; pointers.
        DJNZ    L1A74           ; loop back to SWAP-BYTE until all 5 done.

        EX      DE,HL           ; even up the exchanges
                                ; so that DE addresses STKEND.
        RET                     ; return.

; ---------------------------------
; THE 'SERIES GENERATOR' SUBROUTINE
; ---------------------------------
; offset $86: 'series-06'
; offset $88: 'series-08'
; offset $8C: 'series-0C'
; The ZX81 uses Chebyshev polynomials to generate approximations for
; SIN, ATN, LN and EXP. These are named after the Russian mathematician
; Pafnuty Chebyshev, born in 1821, who did much pioneering work on
numerical
; series. As far as calculators are concerned, Chebyshev polynomials have
an
; advantage over other series, for example the Taylor series, as they can
; reach an approximation in just six iterations for SIN, eight for EXP and
; twelve for LN and ATN. The mechanics of the routine are interesting but
; for full treatment of how these are generated with demonstrations in
; Sinclair BASIC see "The Complete Spectrum ROM Disassembly" by Dr Ian
Logan
; and Dr Frank O'Hara, published 1983 by Melbourne House.

;; series-xx
L1A7F:  LD      B,A             ; parameter $00 - $1F to B counter
        CALL    L19A0           ; routine GEN-ENT-1 is called.
                                ; A recursive call to a special entry point
                                ; in the calculator that puts the B
register
                                ; in the system variable BREG. The return
                                ; address is the next location and where
                                ; the calculator will expect its first
```

```
                                ; instruction - now pointed to by HL'.
                                ; The previous pointer to the series of
                                ; five-byte numbers goes on the machine
stack.

; The initialization phase.

        DEFB    $2D             ;;duplicate       x,x
        DEFB    $0F             ;;addition        x+x
        DEFB    $C0             ;;st-mem-0        x+x
        DEFB    $02             ;;delete          .
        DEFB    $A0             ;;stk-zero        0
        DEFB    $C2             ;;st-mem-2        0

; a loop is now entered to perform the algebraic calculation for each of
; the numbers in the series

;; G-LOOP
L1A89:  DEFB    $2D             ;;duplicate       v,v.
        DEFB    $E0             ;;get-mem-0       v,v,x+2
        DEFB    $04             ;;multiply        v,v*x+2
        DEFB    $E2             ;;get-mem-2       v,v*x+2,v
        DEFB    $C1             ;;st-mem-1
        DEFB    $03             ;;subtract
        DEFB    $34             ;;end-calc

; the previous pointer is fetched from the machine stack to H'L' where it
; addresses one of the numbers of the series following the series literal.

        CALL    L19FC           ; routine STK-DATA is called directly to
                                ; push a value and advance H'L'.
        CALL    L19A4           ; routine GEN-ENT-2 recursively re-enters
                                ; the calculator without disturbing
                                ; system variable BREG
                                ; H'L' value goes on the machine stack and
is
                                ; then loaded as usual with the next
address.

        DEFB    $0F             ;;addition
        DEFB    $01             ;;exchange
        DEFB    $C2             ;;st-mem-2
        DEFB    $02             ;;delete

        DEFB    $31             ;;dec-jr-nz
        DEFB    $EE             ;;back to L1A89, G-LOOP

; when the counted loop is complete the final subtraction yields the result
; for example SIN X.

        DEFB    $E1             ;;get-mem-1
        DEFB    $03             ;;subtract
        DEFB    $34             ;;end-calc

        RET                     ; return with H'L' pointing to location
                                ; after last number in series.

; -----------------------
; Handle unary minus (18)
; -----------------------
; Unary so on entry HL points to last value, DE to STKEND.
```

```
;; NEGATE
;; negate
L1AA0:  LD A,   (HL)            ; fetch exponent of last value on the
                                ; calculator stack.
        AND     A               ; test it.
        RET     Z               ; return if zero.

        INC     HL              ; address the byte with the sign bit.
        LD      A,(HL)          ; fetch to accumulator.
        XOR     $80             ; toggle the sign bit.
        LD      (HL),A          ; put it back.
        DEC     HL              ; point to last value again.
        RET                     ; return.

; ------------------------
; Absolute magnitude (27)
; ------------------------
; This calculator literal finds the absolute value of the last value,
; floating point, on calculator stack.

;; abs
L1AAA:  INC     HL              ; point to byte with sign bit.
        RES     7,(HL)          ; make the sign positive.
        DEC     HL              ; point to last value again.
        RET                     ; return.

; -----------
; Signum (26)
; -----------
; This routine replaces the last value on the calculator stack,
; which is in floating point form, with one if positive and with -minus one
; if negative. If it is zero then it is left as such.

;; sgn
L1AAF:  INC     HL              ; point to first byte of 4-byte mantissa.
        LD      A,(HL)          ; pick up the byte with the sign bit.
        DEC     HL              ; point to exponent.
        DEC     (HL)            ; test the exponent for
        INC     (HL)            ; the value zero.

        SCF                     ; set the carry flag.
        CALL    NZ,L1AE0        ; routine FP-0/1  replaces last value with
one
                                ; if exponent indicates the value is non-
zero.
                                ; in either case mantissa is now four
zeros.

        INC HL                  ; point to first byte of 4-byte mantissa.
        RLCA                    ; rotate original sign bit to carry.
        RR      (HL)            ; rotate the carry into sign.
        DEC HL                  ; point to last value.
        RET                     ; return.


; -------------------------
; Handle PEEK function (28)
; -------------------------
; This function returns the contents of a memory address.
; The entire address space can be peeked including the ROM.

;; peek
```

```
L1ABE:  CALL    L0EA7           ; routine FIND-INT puts address in BC.
        LD      A,(BC)          ; load contents into A register.

;; IN-PK-STK
L1AC2:  JP      L151D           ; exit via STACK-A to put value on the
                                ; calculator stack.

; ---------------
; USR number (29)
; ---------------
; The USR function followed by a number 0-65535 is the method by which
; the ZX81 invokes machine code programs. This function returns the
; contents of the BC register pair.
; Note. that STACK-BC re-initializes the IY register to $4000 if a user-
written
; program has altered it.

;; usr-no
L1AC5:  CALL    L0EA7           ; routine FIND-INT to fetch the
                                ; supplied address into BC.

        LD      HL,L1520        ; address: STACK-BC is
        PUSH    HL              ; pushed onto the machine stack.
        PUSH    BC              ; then the address of the machine code
                                ; routine.

        RET                     ; make an indirect jump to the routine
                                ; and, hopefully, to STACK-BC also.


; ------------------------
; Greater than zero ($33)
; ------------------------
; Test if the last value on the calculator stack is greater than zero.
; This routine is also called directly from the end-tests of the comparison
; routine.

;; GREATER-0
;; greater-0
L1ACE:  LD      A,(HL)          ; fetch exponent.
        AND     A               ; test it for zero.
        RET     Z               ; return if so.


        LD      A,$FF           ; prepare XOR mask for sign bit
        JR      L1ADC           ; forward to SIGN-TO-C
                                ; to put sign in carry
                                ; (carry will become set if sign is
positive)
                                ; and then overwrite location with 1 or 0
                                ; as appropriate.

; ------------------------
; Handle NOT operator ($2C)
; ------------------------
; This overwrites the last value with 1 if it was zero else with zero
; if it was any other value.
;
; e.g. NOT 0 returns 1, NOT 1 returns 0, NOT -3 returns 0.
;
; The subroutine is also called directly from the end-tests of the
comparison
```

```
; operator.

;; NOT
;; not
L1AD5:  LD      A,(HL)          ; get exponent byte.
        NEG                     ; negate - sets carry if non-zero.
        CCF                     ; complement so carry set if zero, else
reset.
        JR      L1AE0           ; forward to FP-0/1.

; -------------------
; Less than zero (32)
; -------------------
; Destructively test if last value on calculator stack is less than zero.
; Bit 7 of second byte will be set if so.

;; less-0
L1ADB:  XOR     A               ; set xor mask to zero
                                ; (carry will become set if sign is
negative).

; transfer sign of mantissa to Carry Flag.

;; SIGN-TO-C
L1ADC:  INC     HL              ; address 2nd byte.
        XOR     (HL)            ; bit 7 of HL will be set if number is
negative.
        DEC     HL              ; address 1st byte again.
        RLCA                    ; rotate bit 7 of A to carry.

; -----------
; Zero or one
; -----------
; This routine places an integer value zero or one at the addressed
location
; of calculator stack or MEM area. The value one is written if carry is set
on
; entry else zero.

;; FP-0/1
L1AE0:  PUSH    HL              ; save pointer to the first byte
        LD      B,$05           ; five bytes to do.

;; FP-loop
L1AE3:  LD      (HL),$00        ; insert a zero.
        INC     HL              ;
        DJNZ    L1AE3           ; repeat.

        POP     HL              ;
        RET     NC              ;

        LD      (HL),$81        ; make value 1
        RET                     ; return.


; ----------------------
; Handle OR operator (07)
; ----------------------
; The Boolean OR operator. eg. X OR Y
; The result is zero if both values are zero else a non-zero value.
;
; e.g.    0 OR 0  returns 0.
```

```
;         -3 OR 0  returns -3.
;          0 OR -3 returns 1.
;         -3 OR 2  returns 1.
;
; A binary operation.
; On entry HL points to first operand (X) and DE to second operand (Y).

;; or
L1AED:  LD      A,(DE)          ; fetch exponent of second number
        AND     A               ; test it.
        RET     Z               ; return if zero.

        SCF                     ; set carry flag
        JR      L1AE0           ; back to FP-0/1 to overwrite the first
operand
                                ; with the value 1.


; -----------------------------
; Handle number AND number (08)
; -----------------------------
; The Boolean AND operator.
;
; e.g.    -3 AND 2  returns -3.
;         -3 AND 0  returns 0.
;          0 and -2 returns 0.
;          0 and 0  returns 0.
;
; Compare with OR routine above.

;; no-&-no
L1AF3:  LD      A,(DE)          ; fetch exponent of second number.
        AND     A               ; test it.
        RET     NZ              ; return if not zero.

        JR      L1AE0           ; back to FP-0/1 to overwrite the first
operand
                                ; with zero for return value.

; -----------------------------
; Handle string AND number (10)
; -----------------------------
; e.g. "YOU WIN" AND SCORE>99 will return the string if condition is true
; or the null string if false.

;; str-&-no
L1AF8:  LD      A,(DE)          ; fetch exponent of second number.
        AND     A               ; test it.
        RET     NZ              ; return if number was not zero - the
string
                                ; is the result.

; if the number was zero (false) then the null string must be returned by
; altering the length of the string on the calculator stack to zero.

        PUSH    DE              ; save pointer to the now obsolete number
                                ; (which will become the new STKEND)

        DEC     DE              ; point to the 5th byte of string
descriptor.
        XOR     A               ; clear the accumulator.
        LD      (DE),A          ; place zero in high byte of length.
```

```
        DEC     DE              ; address low byte of length.
        LD      (DE),A          ; place zero there - now the null string.

        POP     DE              ; restore pointer - new STKEND.
        RET                     ; return.

; ----------------------------------
; Perform comparison ($09-$0E, $11-$16)
; ----------------------------------
; True binary operations.
;
; A single entry point is used to evaluate six numeric and six string
; comparisons. On entry, the calculator literal is in the B register and
; the two numeric values, or the two string parameters, are on the
; calculator stack.
; The individual bits of the literal are manipulated to group similar
; operations although the SUB 8 instruction does nothing useful and merely
; alters the string test bit.
; Numbers are compared by subtracting one from the other, strings are
; compared by comparing every character until a mismatch, or the end of one
; or both, is reached.
;
; Numeric Comparisons.
; --------------------
; The 'x>y' example is the easiest as it employs straight-thru logic.
; Number y is subtracted from x and the result tested for greater-0
yielding
; a final value 1 (true) or 0 (false).
; For 'x<y' the same logic is used but the two values are first swapped on
the
; calculator stack.
; For 'x=y' NOT is applied to the subtraction result yielding true if the
; difference was zero and false with anything else.
; The first three numeric comparisons are just the opposite of the last
three
; so the same processing steps are used and then a final NOT is applied.
;
; literal    Test   No  sub 8       ExOrNot  1st RRCA  exch sub  ?   End-
Tests
; =========  ====   == ======== === ======== ========  ==== ===  =  === ===
===
; no-l-eql   x<=y   09 00000001 dec 00000000 00000000  ---- x-y  ?  --- >0?
NOT
; no-gr-eql  x>=y   0A 00000010 dec 00000001 10000000c swap y-x  ?  --- >0?
NOT
; nos-neql   x<>y   0B 00000011 dec 00000010 00000001  ---- x-y  ?  NOT ---
NOT
; no-grtr    x>y    0C 00000100  -  00000100 00000010  ---- x-y  ?  --- >0?
---
; no-less    x<y    0D 00000101  -  00000101 10000010c swap y-x  ?  --- >0?
---
; nos-eql    x=y    0E 00000110  -  00000110 00000011  ---- x-y  ?  NOT ---
---
;
;                                                              comp -> C/F
;                                                              ====    ===
; str-l-eql  x$<=y$ 11 00001001 dec 00001000 00000100  ---- x$y$ 0  !or >0?
NOT
; str-gr-eql x$>=y$ 12 00001010 dec 00001001 10000100c swap y$x$ 0  !or >0?
NOT
; strs-neql  x$<>y$ 13 00001011 dec 00001010 00000101  ---- x$y$ 0  !or >0?
NOT
```

```
; str-grtr   x$>y$  14 00001100  -  00001100 00000110  ---- x$y$ 0  !or >0?
---
; str-less   x$<y$  15 00001101  -  00001101 10000110c swap y$x$ 0  !or >0?
---
; strs-eql   x$=y$  16 00001110  -  00001110 00000111  ---- x$y$ 0  !or >0?
---
;
; String comparisons are a little different in that the eql/neql carry flag
; from the 2nd RRCA is, as before, fed into the first of the end tests but
; along the way it gets modified by the comparison process. The result on
the
; stack always starts off as zero and the carry fed in determines if NOT is
; applied to it. So the only time the greater-0 test is applied is if the
; stack holds zero which is not very efficient as the test will always
yield
; zero. The most likely explanation is that there were once separate end
tests
; for numbers and strings.

;; no-l-eql,etc.
L1B03:  LD      A,B             ; transfer literal to accumulator.
        SUB     $08             ; subtract eight - which is not useful.

        BIT     2,A             ; isolate '>', '<', '='.

        JR      NZ,L1B0B        ; skip to EX-OR-NOT with these.

        DEC     A               ; else make $00-$02, $08-$0A to match bits
0-2.

;; EX-OR-NOT
L1B0B:  RRCA                    ; the first RRCA sets carry for a swap.
        JR      NC,L1B16        ; forward to NU-OR-STR with other 8 cases

; for the other 4 cases the two values on the calculator stack are
exchanged.

        PUSH    AF              ; save A and carry.
        PUSH    HL              ; save HL - pointer to first operand.
                                ; (DE points to second operand).

        CALL    L1A72           ; routine exchange swaps the two values.
                                ; (HL = second operand, DE = STKEND)

        POP     DE              ; DE = first operand
        EX      DE,HL           ; as we were.
        POP     AF              ; restore A and carry.

; Note. it would be better if the 2nd RRCA preceded the string test.
; It would save two duplicate bytes and if we also got rid of that sub 8
; at the beginning we wouldn't have to alter which bit we test.

;; NU-OR-STR
L1B16:  BIT     2,A             ; test if a string comparison.
        JR      NZ,L1B21        ; forward to STRINGS if so.

; continue with numeric comparisons.

        RRCA                    ; 2nd RRCA causes eql/neql to set carry.
        PUSH    AF              ; save A and carry

        CALL    L174C           ; routine subtract leaves result on stack.
```

```
        JR      L1B54           ; forward to END-TESTS

; ---

;; STRINGS
L1B21:  RRCA                    ; 2nd RRCA causes eql/neql to set carry.
        PUSH    AF              ; save A and carry.

        CALL    L13F8           ; routine STK-FETCH gets 2nd string params
        PUSH    DE              ; save start2 *.
        PUSH    BC              ; and the length.

        CALL    L13F8           ; routine STK-FETCH gets 1st string
                                ; parameters - start in DE, length in BC.
        POP     HL              ; restore length of second to HL.

; A loop is now entered to compare, by subtraction, each corresponding
character
; of the strings. For each successful match, the pointers are incremented
and
; the lengths decreased and the branch taken back to here. If both string
; remainders become null at the same time, then an exact match exists.

;; BYTE-COMP
L1B2C:  LD      A,H             ; test if the second string
        OR      L               ; is the null string and hold flags.

        EX      (SP),HL         ; put length2 on stack, bring start2 to HL
*.
        LD      A,B             ; hi byte of length1 to A

        JR      NZ,L1B3D        ; forward to SEC-PLUS if second not null.

        OR      C               ; test length of first string.

;; SECND-LOW
L1B33:  POP     BC              ; pop the second length off stack.
        JR      Z,L1B3A         ; forward to BOTH-NULL if first string is
also
                                ; of zero length.

; the true condition - first is longer than second (SECND-LESS)

        POP     AF              ; restore carry (set if eql/neql)
        CCF                     ; complement carry flag.
                                ; Note. equality becomes false.
                                ; Inequality is true. By swapping or
applying
                                ; a terminal 'not', all comparisons have
been
                                ; manipulated so that this is success path.
        JR      L1B50           ; forward to leave via STR-TEST

; ---
; the branch was here with a match

;; BOTH-NULL
L1B3A:  POP     AF              ; restore carry - set for eql/neql
        JR      L1B50           ; forward to STR-TEST

; ---
; the branch was here when 2nd string not null and low byte of first is yet
```

```
; to be tested.


;; SEC-PLUS
L1B3D:  OR      C               ; test the length of first string.
        JR      Z,L1B4D         ; forward to FRST-LESS if length is zero.

; both strings have at least one character left.

        LD      A,(DE)          ; fetch character of first string.
        SUB     (HL)            ; subtract with that of 2nd string.
        JR      C,L1B4D         ; forward to FRST-LESS if carry set

        JR      NZ,L1B33        ; back to SECND-LOW and then STR-TEST
                                ; if not exact match.

        DEC     BC              ; decrease length of 1st string.
        INC     DE              ; increment 1st string pointer.

        INC     HL              ; increment 2nd string pointer.
        EX      (SP),HL         ; swap with length on stack
        DEC     HL              ; decrement 2nd string length
        JR      L1B2C           ; back to BYTE-COMP

; ---
;   the false condition.

;; FRST-LESS
L1B4D:  POP     BC              ; discard length
        POP     AF              ; pop A
        AND     A               ; clear the carry for false result.

; ---
;   exact match and x$>y$ rejoin here

;; STR-TEST
L1B50:  PUSH    AF              ; save A and carry

        RST     28H             ;; FP-CALC
        DEFB    $A0             ;;stk-zero      an initial false value.
        DEFB    $34             ;;end-calc

;   both numeric and string paths converge here.

;; END-TESTS
L1B54:  POP     AF              ; pop carry  - will be set if eql/neql
        PUSH    AF              ; save it again.

        CALL    C,L1AD5         ; routine NOT sets true(1) if equal(0)
                                ; or, for strings, applies true result.
        CALL    L1ACE           ; greater-0  ??????????


        POP     AF              ; pop A
        RRCA                    ; the third RRCA - test for '<=', '>=' or
'<>'.
        CALL    NC,L1AD5        ; apply a terminal NOT if so.
        RET                     ; return.

; -------------------------
; String concatenation ($17)
; -------------------------
```

```
;   This literal combines two strings into one e.g. LET A$ = B$ + C$
;   The two parameters of the two strings to be combined are on the stack.

;; strs-add
L1B62:  CALL    L13F8           ; routine STK-FETCH fetches string
parameters
                                ; and deletes calculator stack entry.
        PUSH    DE              ; save start address.
        PUSH    BC              ; and length.

        CALL    L13F8           ; routine STK-FETCH for first string
        POP     HL              ; re-fetch first length
        PUSH    HL              ; and save again
        PUSH    DE              ; save start of second string
        PUSH    BC              ; and its length.

        ADD     HL,BC           ; add the two lengths.
        LD      B,H             ; transfer to BC
        LD      C,L             ; and create
        RST     30H             ; BC-SPACES in workspace.
                                ; DE points to start of space.

        CALL    L12C3           ; routine STK-STO-$ stores parameters
                                ; of new string updating STKEND.

        POP     BC              ; length of first
        POP     HL              ; address of start
        LD      A,B             ; test for
        OR      C               ; zero length.
        JR      Z,L1B7D         ; to OTHER-STR if null string

        LDIR                    ; copy string to workspace.

;; OTHER-STR
L1B7D:  POP     BC              ; now second length
        POP     HL              ; and start of string
        LD      A,B             ; test this one
        OR      C               ; for zero length
        JR      Z,L1B85         ; skip forward to STK-PNTRS if so as
complete.

        LDIR                    ; else copy the bytes.
                                ; and continue into next routine which
                                ; sets the calculator stack pointers.

; --------------------
; Check stack pointers
; --------------------
;   Register DE is set to STKEND and HL, the result pointer, is set to five
;   locations below this.
;   This routine is used when it is inconvenient to save these values at
the
;   time the calculator stack is manipulated due to other activity on the
;   machine stack.
;   This routine is also used to terminate the VAL routine for
;   the same reason and to initialize the calculator stack at the start of
;   the CALCULATE routine.

;; STK-PNTRS
L1B85:  LD      HL,($401C)      ; fetch STKEND value from system variable.
        LD      DE,$FFFB        ; the value -5
        PUSH    HL              ; push STKEND value.
```

```
        ADD     HL,DE           ; subtract 5 from HL.

        POP     DE              ; pop STKEND to DE.
        RET                     ; return.

; ----------------
; Handle CHR$ (2B)
; ----------------
;   This function returns a single character string that is a result of
;   converting a number in the range 0-255 to a string e.g. CHR$ 38 = "A".
;   Note. the ZX81 does not have an ASCII character set.

;; chrs
L1B8F:  CALL    L15CD           ; routine FP-TO-A puts the number in A.

        JR      C,L1BA2         ; forward to REPORT-Bd if overflow
        JR      NZ,L1BA2        ; forward to REPORT-Bd if negative

        PUSH    AF              ; save the argument.

        LD      BC,$0001        ; one space required.
        RST     30H             ; BC-SPACES makes DE point to start

        POP     AF              ; restore the number.

        LD      (DE),A          ; and store in workspace

        CALL    L12C3           ; routine STK-STO-$ stacks descriptor.

        EX      DE,HL           ; make HL point to result and DE to STKEND.
        RET                     ; return.

; ---

;; REPORT-Bd
L1BA2:  RST     08H             ; ERROR-1
        DEFB    $0A             ; Error Report: Integer out of range

; ----------------------------
; Handle VAL ($1A)
; ----------------------------
;   VAL treats the characters in a string as a numeric expression.
;       e.g. VAL "2.3" = 2.3, VAL "2+4" = 6, VAL ("2" + "4") = 24.

;; val
L1BA4:  LD      HL,($4016)      ; fetch value of system variable CH_ADD
        PUSH    HL              ; and save on the machine stack.

        CALL    L13F8           ; routine STK-FETCH fetches the string
operand
                                ; from calculator stack.

        PUSH    DE              ; save the address of the start of the
string.
        INC     BC              ; increment the length for a carriage
return.

        RST     30H             ; BC-SPACES creates the space in workspace.
        POP     HL              ; restore start of string to HL.
        LD      ($4016),DE      ; load CH_ADD with start DE in workspace.
```

```
        PUSH    DE              ; save the start in workspace
        LDIR                    ; copy string from program or variables or
                                ; workspace to the workspace area.
        EX      DE,HL           ; end of string + 1 to HL
        DEC     HL              ; decrement HL to point to end of new area.
        LD      (HL),$76        ; insert a carriage return at end.
                                ; ZX81 has a non-ASCII character set
        RES     7,(IY+$01)      ; update FLAGS  - signal checking syntax.
        CALL    L0D92           ; routine CLASS-06 - SCANNING evaluates
string
                                ; expression and checks for integer result.

        CALL    L0D22           ; routine CHECK-2 checks for carriage
return.


        POP     HL              ; restore start of string in workspace.

        LD      ($4016),HL      ; set CH_ADD to the start of the string
again.
        SET     7,(IY+$01)      ; update FLAGS  - signal running program.
        CALL    L0F55           ; routine SCANNING evaluates the string
                                ; in full leaving result on calculator
stack.

        POP     HL              ; restore saved character address in
program.
        LD      ($4016),HL      ; and reset the system variable CH_ADD.

        JR      L1B85           ; back to exit via STK-PNTRS.
                                ; resetting the calculator stack pointers
                                ; HL and DE from STKEND as it wasn't
possible
                                ; to preserve them during this routine.

; ----------------
; Handle STR$ (2A)
; ----------------
;   This function returns a string representation of a numeric argument.
;   The method used is to trick the PRINT-FP routine into thinking it
;   is writing to a collapsed display file when in fact it is writing to
;   string workspace.
;   If there is already a newline at the intended print position and the
;   column count has not been reduced to zero then the print routine
;   assumes that there is only 1K of RAM and the screen memory, like the
rest
;   of dynamic memory, expands as necessary using calls to the ONE-SPACE
;   routine. The screen is character-mapped not bit-mapped.

;; str$
L1BD5:  LD      BC,$0001        ; create an initial byte in workspace
        RST     30H             ; using BC-SPACES restart.

        LD      (HL),$76        ; place a carriage return there.

        LD      HL,($4039)      ; fetch value of S_POSN column/line
        PUSH    HL              ; and preserve on stack.

        LD      L,$FF           ; make column value high to create a
                                ; contrived buffer of length 254.
        LD      ($4039),HL      ; and store in system variable S_POSN.
```

```
        LD      HL,($400E)      ; fetch value of DF_CC
        PUSH    HL              ; and preserve on stack also.

        LD      ($400E),DE      ; now set DF_CC which normally addresses
                                ; somewhere in the display file to the
start
                                ; of workspace.
        PUSH    DE              ; save the start of new string.

        CALL    L15DB           ; routine PRINT-FP.

        POP     DE              ; retrieve start of string.

        LD      HL,($400E)      ; fetch end of string from DF_CC.
        AND     A               ; prepare for true subtraction.
        SBC     HL,DE           ; subtract to give length.

        LD      B,H             ; and transfer to the BC
        LD      C,L             ; register.

        POP     HL              ; restore original
        LD      ($400E),HL      ; DF_CC value

        POP     HL              ; restore original
        LD      ($4039),HL      ; S_POSN values.

        CALL    L12C3           ; routine STK-STO-$ stores the string
                                ; descriptor on the calculator stack.

        EX      DE,HL           ; HL = last value, DE = STKEND.
        RET                     ; return.


; -------------------
; THE 'CODE' FUNCTION
; -------------------
; (offset $19: 'code')
;   Returns the code of a character or first character of a string
;   e.g. CODE "AARDVARK" = 38  (not 65 as the ZX81 does not have an ASCII
;   character set).


;; code
L1C06:  CALL    L13F8           ; routine STK-FETCH to fetch and delete the
                                ; string parameters.
                                ; DE points to the start, BC holds the
length.
        LD      A,B             ; test length
        OR      C               ; of the string.
        JR      Z,L1C0E         ; skip to STK-CODE with zero if the null
string.

        LD      A,(DE)          ; else fetch the first character.

;; STK-CODE
L1C0E:  JP      L151D           ; jump back to STACK-A (with memory check)

; --------------------
; THE 'LEN' SUBROUTINE
; --------------------
; (offset $1b: 'len')
;   Returns the length of a string.
```

```
;   In Sinclair BASIC strings can be more than twenty thousand characters
long
;   so a sixteen-bit register is required to store the length

;; len
L1C11:  CALL    L13F8           ; routine STK-FETCH to fetch and delete the
                                ; string parameters from the calculator
stack.
                                ; register BC now holds the length of
string.

        JP      L1520           ; jump back to STACK-BC to save result on
the
                                ; calculator stack (with memory check).

; -------------------------------------
; THE 'DECREASE THE COUNTER' SUBROUTINE
; -------------------------------------
; (offset $31: 'dec-jr-nz')
;   The calculator has an instruction that decrements a single-byte
;   pseudo-register and makes consequential relative jumps just like
;   the Z80's DJNZ instruction.

;; dec-jr-nz
L1C17:  EXX                     ; switch in set that addresses code

        PUSH    HL              ; save pointer to offset byte
        LD      HL,$401E        ; address BREG in system variables
        DEC     (HL)            ; decrement it
        POP     HL              ; restore pointer

        JR      NZ,L1C24        ; to JUMP-2 if not zero

        INC     HL              ; step past the jump length.
        EXX                     ; switch in the main set.
        RET                     ; return.

;   Note. as a general rule the calculator avoids using the IY register
;   otherwise the cumbersome 4 instructions in the middle could be replaced
by
;   dec (iy+$xx) - using three instruction bytes instead of six.


; ---------------------
; THE 'JUMP' SUBROUTINE
; ---------------------
; (Offset $2F; 'jump')
;   This enables the calculator to perform relative jumps just like
;   the Z80 chip's JR instruction.
;   This is one of the few routines to be polished for the ZX Spectrum.
;   See, without looking at the ZX Spectrum ROM, if you can get rid of the
;   relative jump.

;; jump
;; JUMP
L1C23:  EXX                     ;switch in pointer set

;; JUMP-2
L1C24:  LD      E,(HL)          ; the jump byte 0-127 forward, 128-255
back.
        XOR     A               ; clear accumulator.
        BIT     7,E             ; test if negative jump
```

```
        JR      Z,L1C2B         ; skip, if positive, to JUMP-3.

        CPL                     ; else change to $FF.

;; JUMP-3
L1C2B:  LD      D,A             ; transfer to high byte.
        ADD     HL,DE           ; advance calculator pointer forward or
back.

        EXX                     ; switch out pointer set.
        RET                     ; return.

; ------------------------------
; THE 'JUMP ON TRUE' SUBROUTINE
; ------------------------------
; (Offset $00; 'jump-true')
;   This enables the calculator to perform conditional relative jumps
;   dependent on whether the last test gave a true result
;   On the ZX81, the exponent will be zero for zero or else $81 for one.

;; jump-true
L1C2F:  LD      A,(DE)          ; collect exponent byte

        AND     A               ; is result 0 or 1 ?
        JR      NZ,L1C23        ; back to JUMP if true (1).

        EXX                     ; else switch in the pointer set.
        INC     HL              ; step past the jump length.
        EXX                     ; switch in the main set.
        RET                     ; return.


; ------------------------
; THE 'MODULUS' SUBROUTINE
; ------------------------
; ( Offset $2E: 'n-mod-m' )
; ( i1, i2 -- i3, i4 )
;   The subroutine calculate N mod M where M is the positive integer, the
;   'last value' on the calculator stack and N is the integer beneath.
;   The subroutine returns the integer quotient as the last value and the
;   remainder as the value beneath.
;   e.g.    17 MOD 3 = 5 remainder 2
;   It is invoked during the calculation of a random number and also by
;   the PRINT-FP routine.

;; n-mod-m
L1C37:  RST     28H             ;; FP-CALC          17, 3.
        DEFB    $C0             ;;st-mem-0          17, 3.
        DEFB    $02             ;;delete            17.
        DEFB    $2D             ;;duplicate         17, 17.
        DEFB    $E0             ;;get-mem-0         17, 17, 3.
        DEFB    $05             ;;division          17, 17/3.
        DEFB    $24             ;;int               17, 5.
        DEFB    $E0             ;;get-mem-0         17, 5, 3.
        DEFB    $01             ;;exchange          17, 3, 5.
        DEFB    $C0             ;;st-mem-0          17, 3, 5.
        DEFB    $04             ;;multiply          17, 15.
        DEFB    $03             ;;subtract          2.
        DEFB    $E0             ;;get-mem-0         2, 5.
        DEFB    $34             ;;end-calc          2, 5.

        RET                     ; return.
```

```
; ----------------------
; THE 'INTEGER' FUNCTION
; ----------------------
; (offset $24: 'int')
;   This function returns the integer of x, which is just the same as
truncate
;   for positive numbers. The truncate literal truncates negative numbers
;   upwards so that -3.4 gives -3 whereas the BASIC INT function has to
;   truncate negative numbers down so that INT -3.4 is 4.
;   It is best to work through using, say, plus or minus 3.4 as examples.

;; int
L1C46:  RST     28H             ;; FP-CALC              x.    (= 3.4 or
-3.4).
        DEFB    $2D             ;;duplicate             x, x.
        DEFB    $32             ;;less-0                x, (1/0)
        DEFB    $00             ;;jump-true             x, (1/0)
        DEFB    $04             ;;to L1C46, X-NEG

        DEFB    $36             ;;truncate              trunc 3.4 = 3.
        DEFB    $34             ;;end-calc              3.

        RET                     ; return with + int x on stack.


;; X-NEG
L1C4E:  DEFB    $2D             ;;duplicate             -3.4, -3.4.
        DEFB    $36             ;;truncate              -3.4, -3.
        DEFB    $C0             ;;st-mem-0              -3.4, -3.
        DEFB    $03             ;;subtract              -.4
        DEFB    $E0             ;;get-mem-0             -.4, -3.
        DEFB    $01             ;;exchange              -3, -.4.
        DEFB    $2C             ;;not                   -3, (0).
        DEFB    $00             ;;jump-true             -3.
        DEFB    $03             ;;to L1C59, EXIT        -3.

        DEFB    $A1             ;;stk-one               -3, 1.
        DEFB    $03             ;;subtract              -4.

;; EXIT
L1C59:  DEFB    $34             ;;end-calc              -4.

        RET                     ; return.


; ----------------
; Exponential (23)
; ----------------
;
;

;; EXP
;; exp
L1C5B:  RST     28H             ;; FP-CALC
        DEFB    $30             ;;stk-data
        DEFB    $F1             ;;Exponent: $81, Bytes: 4
        DEFB    $38,$AA,$3B,$29 ;;
        DEFB    $04             ;;multiply
        DEFB    $2D             ;;duplicate
        DEFB    $24             ;;int
```

```
        DEFB    $C3             ;;st-mem-3
        DEFB    $03             ;;subtract
        DEFB    $2D             ;;duplicate
        DEFB    $0F             ;;addition
        DEFB    $A1             ;;stk-one
        DEFB    $03             ;;subtract
        DEFB    $88             ;;series-08
        DEFB    $13             ;;Exponent: $63, Bytes: 1
        DEFB    $36             ;;(+00,+00,+00)
        DEFB    $58             ;;Exponent: $68, Bytes: 2
        DEFB    $65,$66         ;;(+00,+00)
        DEFB    $9D             ;;Exponent: $6D, Bytes: 3
        DEFB    $78,$65,$40     ;;(+00)
        DEFB    $A2             ;;Exponent: $72, Bytes: 3
        DEFB    $60,$32,$C9     ;;(+00)
        DEFB    $E7             ;;Exponent: $77, Bytes: 4
        DEFB    $21,$F7,$AF,$24 ;;
        DEFB    $EB             ;;Exponent: $7B, Bytes: 4
        DEFB    $2F,$B0,$B0,$14 ;;
        DEFB    $EE             ;;Exponent: $7E, Bytes: 4
        DEFB    $7E,$BB,$94,$58 ;;
        DEFB    $F1             ;;Exponent: $81, Bytes: 4
        DEFB    $3A,$7E,$F8,$CF ;;
        DEFB    $E3             ;;get-mem-3
        DEFB    $34             ;;end-calc

        CALL    L15CD           ; routine FP-TO-A
        JR      NZ,L1C9B        ; to N-NEGTV

        JR      C,L1C99         ; to REPORT-6b

        ADD     A,(HL)          ;
        JR      NC,L1CA2        ; to RESULT-OK


;; REPORT-6b
L1C99:  RST     08H             ; ERROR-1
        DEFB    $05             ; Error Report: Number too big

;; N-NEGTV
L1C9B:  JR      C,L1CA4         ; to RSLT-ZERO

        SUB     (HL)            ;
        JR      NC,L1CA4        ; to RSLT-ZERO

        NEG                     ; Negate

;; RESULT-OK
L1CA2:  LD      (HL),A          ;
        RET                     ; return.


;; RSLT-ZERO
L1CA4:  RST     28H             ;; FP-CALC
        DEFB    $02             ;;delete
        DEFB    $A0             ;;stk-zero
        DEFB    $34             ;;end-calc

        RET                     ; return.


; --------------------------------
```

```
; THE 'NATURAL LOGARITHM' FUNCTION
; --------------------------------
; (offset $22: 'ln')
;   Like the ZX81 itself, 'natural' logarithms came from Scotland.
;   They were devised in 1614 by well-traveled Scotsman John Napier who
noted
;   "Nothing doth more molest and hinder calculators than the
multiplications,
;    divisions, square and cubical extractions of great numbers".
;
;   Napier's logarithms enabled the above operations to be accomplished by
;   simple addition and subtraction simplifying the navigational and
;   astronomical calculations which beset his age.
;   Napier's logarithms were quickly overtaken by logarithms to the base 10
;   devised, in conjunction with Napier, by Henry Briggs a Cambridge-
educated
;   professor of Geometry at Oxford University. These simplified the layout
;   of the tables enabling humans to easily scale calculations.
;
;   It is only recently with the introduction of pocket calculators and
;   computers like the ZX81 that natural logarithms are once more at the
fore,
;   although some computers retain logarithms to the base ten.
;   'Natural' logarithms are powers to the base 'e', which like 'pi' is a
;   naturally occurring number in branches of mathematics.
;   Like 'pi' also, 'e' is an irrational number and starts 2.718281828...
;
;   The tabular use of logarithms was that to multiply two numbers one
looked
;   up their two logarithms in the tables, added them together and then
looked
;   for the result in a table of antilogarithms to give the desired
product.
;
;   The EXP function is the BASIC equivalent of a calculator's 'antiln'
function
;   and by picking any two numbers, 1.72 and 6.89 say,
;     10 PRINT EXP ( LN 1.72 + LN 6.89 )
;   will give just the same result as
;     20 PRINT 1.72 * 6.89.
;   Division is accomplished by subtracting the two logs.
;
;   Napier also mentioned "square and cubicle extractions".
;   To raise a number to the power 3, find its 'ln', multiply by 3 and find
the
;   'antiln'.  e.g. PRINT EXP( LN 4 * 3 )  gives 64.
;   Similarly to find the n'th root divide the logarithm by 'n'.
;   The ZX81 ROM used PRINT EXP ( LN 9 / 2 ) to find the square root of the
;   number 9. The Napieran square root function is just a special case of
;   the 'to_power' function. A cube root or indeed any root/power would be
just
;   as simple.

;   First test that the argument to LN is a positive, non-zero number.

;; ln
L1CA9:  RST     28H             ;; FP-CALC
        DEFB    $2D             ;;duplicate
        DEFB    $33             ;;greater-0
        DEFB    $00             ;;jump-true
        DEFB    $04             ;;to L1CB1, VALID
```

```
         DEFB    $34             ;;end-calc


;; REPORT-Ab
L1CAF:   RST     08H             ; ERROR-1
         DEFB    $09             ; Error Report: Invalid argument

;; VALID
L1CB1:   DEFB    $A0             ;;stk-zero              Note. not
         DEFB    $02             ;;delete                necessary.
         DEFB    $34             ;;end-calc
         LD      A,(HL)          ;

         LD      (HL),$80        ;
         CALL    L151D           ; routine STACK-A

         RST     28H             ;; FP-CALC
         DEFB    $30             ;;stk-data
         DEFB    $38             ;;Exponent: $88, Bytes: 1
         DEFB    $00             ;;(+00,+00,+00)
         DEFB    $03             ;;subtract
         DEFB    $01             ;;exchange
         DEFB    $2D             ;;duplicate
         DEFB    $30             ;;stk-data
         DEFB    $F0             ;;Exponent: $80, Bytes: 4
         DEFB    $4C,$CC,$CC,$CD ;;
         DEFB    $03             ;;subtract
         DEFB    $33             ;;greater-0
         DEFB    $00             ;;jump-true
         DEFB    $08             ;;to L1CD2, GRE.8

         DEFB    $01             ;;exchange
         DEFB    $A1             ;;stk-one
         DEFB    $03             ;;subtract
         DEFB    $01             ;;exchange
         DEFB    $34             ;;end-calc

         INC     (HL)            ;

         RST     28H             ;; FP-CALC

;; GRE.8
L1CD2:   DEFB    $01             ;;exchange
         DEFB    $30             ;;stk-data
         DEFB    $F0             ;;Exponent: $80, Bytes: 4
         DEFB    $31,$72,$17,$F8 ;;
         DEFB    $04             ;;multiply
         DEFB    $01             ;;exchange
         DEFB    $A2             ;;stk-half
         DEFB    $03             ;;subtract
         DEFB    $A2             ;;stk-half
         DEFB    $03             ;;subtract
         DEFB    $2D             ;;duplicate
         DEFB    $30             ;;stk-data
         DEFB    $32             ;;Exponent: $82, Bytes: 1
         DEFB    $20             ;;(+00,+00,+00)
         DEFB    $04             ;;multiply
         DEFB    $A2             ;;stk-half
         DEFB    $03             ;;subtract
         DEFB    $8C             ;;series-0C
         DEFB    $11             ;;Exponent: $61, Bytes: 1
         DEFB    $AC             ;;(+00,+00,+00)
```

```
        DEFB    $14             ;;Exponent: $64, Bytes: 1
        DEFB    $09             ;;(+00,+00,+00)
        DEFB    $56             ;;Exponent: $66, Bytes: 2
        DEFB    $DA,$A5         ;;(+00,+00)
        DEFB    $59             ;;Exponent: $69, Bytes: 2
        DEFB    $30,$C5         ;;(+00,+00)
        DEFB    $5C             ;;Exponent: $6C, Bytes: 2
        DEFB    $90,$AA         ;;(+00,+00)
        DEFB    $9E             ;;Exponent: $6E, Bytes: 3
        DEFB    $70,$6F,$61     ;;(+00)
        DEFB    $A1             ;;Exponent: $71, Bytes: 3
        DEFB    $CB,$DA,$96     ;;(+00)
        DEFB    $A4             ;;Exponent: $74, Bytes: 3
        DEFB    $31,$9F,$B4     ;;(+00)
        DEFB    $E7             ;;Exponent: $77, Bytes: 4
        DEFB    $A0,$FE,$5C,$FC ;;
        DEFB    $EA             ;;Exponent: $7A, Bytes: 4
        DEFB    $1B,$43,$CA,$36 ;;
        DEFB    $ED             ;;Exponent: $7D, Bytes: 4
        DEFB    $A7,$9C,$7E,$5E ;;
        DEFB    $F0             ;;Exponent: $80, Bytes: 4
        DEFB    $6E,$23,$80,$93 ;;
        DEFB    $04             ;;multiply
        DEFB    $0F             ;;addition
        DEFB    $34             ;;end-calc

        RET                     ; return.

; -----------------------------
; THE 'TRIGONOMETRIC' FUNCTIONS
; -----------------------------
;   Trigonometry is rocket science. It is also used by carpenters and
pyramid
;   builders.
;   Some uses can be quite abstract but the principles can be seen in
simple
;   right-angled triangles. Triangles have some special properties -
;
;   1) The sum of the three angles is always PI radians (180 degrees).
;      Very helpful if you know two angles and wish to find the third.
;   2) In any right-angled triangle the sum of the squares of the two
shorter
;      sides is equal to the square of the longest side opposite the right-
angle.
;      Very useful if you know the length of two sides and wish to know the
;      length of the third side.
;   3) Functions sine, cosine and tangent enable one to calculate the
length
;      of an unknown side when the length of one other side and an angle is
;      known.
;   4) Functions arcsin, arccosine and arctan enable one to calculate an
unknown
;      angle when the length of two of the sides is known.

; -------------------------------
; THE 'REDUCE ARGUMENT' SUBROUTINE
; -------------------------------
; (offset $35: 'get-argt')
;
;   This routine performs two functions on the angle, in radians, that
forms
;   the argument to the sine and cosine functions.
```

```
;   First it ensures that the angle 'wraps round'. That if a ship turns
through
;   an angle of, say, 3*PI radians (540 degrees) then the net effect is to
turn
;   through an angle of PI radians (180 degrees).
;   Secondly it converts the angle in radians to a fraction of a right
angle,
;   depending within which quadrant the angle lies, with the periodicity
;   resembling that of the desired sine value.
;   The result lies in the range -1 to +1.
;
;                       90 deg.
;
;                       (pi/2)
;                II       +1        I
;                           |
;          sin+      |\    |    /|    sin+
;          cos-      | \   |   / |    cos+
;          tan-      |  \  | /   |    tan+
;                    |   \|/)   |
;   180 deg. (pi) 0 -|----+----|-- 0  (0)   0 degrees
;                    |   /|\   |
;          sin-      |  / | \   |    sin-
;          cos-      | /  |  \  |    cos+
;          tan+      |/   |   \|    tan-
;                           |
;                III      -1       IV
;                       (3pi/2)
;
;                       270 deg.


;; get-argt
L1D18:  RST     28H             ;; FP-CALC          X.
        DEFB    $30             ;;stk-data
        DEFB    $EE             ;;Exponent: $7E,
                                ;;Bytes: 4
        DEFB    $22,$F9,$83,$6E ;;                  X, 1/(2*PI)
        DEFB    $04             ;;multiply          X/(2*PI) = fraction

        DEFB    $2D             ;;duplicate
        DEFB    $A2             ;;stk-half
        DEFB    $0F             ;;addition
        DEFB    $24             ;;int

        DEFB    $03             ;;subtract          now range -.5 to .5

        DEFB    $2D             ;;duplicate
        DEFB    $0F             ;;addition          now range -1 to 1.
        DEFB    $2D             ;;duplicate
        DEFB    $0F             ;;addition          now range -2 to 2.

;   quadrant I (0 to +1) and quadrant IV (-1 to 0) are now correct.
;   quadrant II ranges +1 to +2.
;   quadrant III ranges -2 to -1.

        DEFB    $2D             ;;duplicate         Y, Y.
        DEFB    $27             ;;abs               Y, abs(Y).    range 1 to
2
        DEFB    $A1             ;;stk-one           Y, abs(Y), 1.
        DEFB    $03             ;;subtract          Y, abs(Y)-1.  range 0 to
1
```

```
        DEFB    $2D             ;;duplicate         Y, Z, Z.
        DEFB    $33             ;;greater-0         Y, Z, (1/0).

        DEFB    $C0             ;;st-mem-0          store as possible sign
                                ;;                  for cosine function.

        DEFB    $00             ;;jump-true
        DEFB    $04             ;;to L1D35, ZPLUS   with quadrants II and
III

;   else the angle lies in quadrant I or IV and value Y is already correct.

        DEFB    $02             ;;delete            Y    delete test value.
        DEFB    $34             ;;end-calc          Y.

        RET                     ; return.           with Q1 and Q4 >>>

;   The branch was here with quadrants II (0 to 1) and III (1 to 0).
;   Y will hold -2 to -1 if this is quadrant III.

;; ZPLUS
L1D35:  DEFB    $A1             ;;stk-one           Y, Z, 1
        DEFB    $03             ;;subtract          Y, Z-1.       Q3 = 0 to
-1
        DEFB    $01             ;;exchange          Z-1, Y.
        DEFB    $32             ;;less-0            Z-1, (1/0).
        DEFB    $00             ;;jump-true         Z-1.
        DEFB    $02             ;;to L1D3C, YNEG
                                ;;if angle in quadrant III

;   else angle is within quadrant II (-1 to 0)

        DEFB    $18             ;;negate            range +1 to 0


;; YNEG
L1D3C:  DEFB    $34             ;;end-calc          quadrants II and III
correct.

        RET                     ; return.


; ---------------------
; THE 'COSINE' FUNCTION
; ---------------------
; (offset $1D: 'cos')
;   Cosines are calculated as the sine of the opposite angle rectifying the
;   sign depending on the quadrant rules.
;
;
;             /|
;          h /y|
;           /  |o
;          /x  |
;         /----|
;           a
;
;   The cosine of angle x is the adjacent side (a) divided by the
hypotenuse 1.
;   However if we examine angle y then a/h is the sine of that angle.
;   Since angle x plus angle y equals a right-angle, we can find angle y by
;   subtracting angle x from pi/2.
```

```
;   However it's just as easy to reduce the argument first and subtract the
;   reduced argument from the value 1 (a reduced right-angle).
;   It's even easier to subtract 1 from the angle and rectify the sign.
;   In fact, after reducing the argument, the absolute value of the
argument
;   is used and rectified using the test result stored in mem-0 by 'get-
argt'
;   for that purpose.

;; cos
L1D3E:  RST     28H             ;; FP-CALC              angle in radians.
        DEFB    $35             ;;get-argt              X       reduce -1
to +1

        DEFB    $27             ;;abs                   ABS X   0 to 1
        DEFB    $A1             ;;stk-one               ABS X, 1.
        DEFB    $03             ;;subtract              now opposite angle
                                ;;                      though negative
sign.
        DEFB    $E0             ;;get-mem-0             fetch sign
indicator.
        DEFB    $00             ;;jump-true
        DEFB    $06             ;;fwd to L1D4B, C-ENT
                                ;;forward to common code if in QII or QIII


        DEFB    $18             ;;negate                else make positive.
        DEFB    $2F             ;;jump
        DEFB    $03             ;;fwd to L1D4B, C-ENT
                                ;;with quadrants QI and QIV

; -------------------
; THE 'SINE' FUNCTION
; -------------------
; (offset $1C: 'sin')
;   This is a fundamental transcendental function from which others such as
cos
;   and tan are directly, or indirectly, derived.
;   It uses the series generator to produce Chebyshev polynomials.
;
;
;              /|
;          1 / |
;            /  |x
;           /a  |
;          /----|
;            y
;
;   The 'get-argt' function is designed to modify the angle and its sign
;   in line with the desired sine value and afterwards it can launch
straight
;   into common code.

;; sin
L1D49:  RST     28H             ;; FP-CALC      angle in radians
        DEFB    $35             ;;get-argt      reduce - sign now correct.

;; C-ENT
L1D4B:  DEFB    $2D             ;;duplicate
        DEFB    $2D             ;;duplicate
        DEFB    $04             ;;multiply
        DEFB    $2D             ;;duplicate
```

```
        DEFB    $0F             ;;addition
        DEFB    $A1             ;;stk-one
        DEFB    $03             ;;subtract

        DEFB    $86             ;;series-06
        DEFB    $14             ;;Exponent: $64, Bytes: 1
        DEFB    $E6             ;;(+00,+00,+00)
        DEFB    $5C             ;;Exponent: $6C, Bytes: 2
        DEFB    $1F,$0B         ;;(+00,+00)
        DEFB    $A3             ;;Exponent: $73, Bytes: 3
        DEFB    $8F,$38,$EE     ;;(+00)
        DEFB    $E9             ;;Exponent: $79, Bytes: 4
        DEFB    $15,$63,$BB,$23 ;;
        DEFB    $EE             ;;Exponent: $7E, Bytes: 4
        DEFB    $92,$0D,$CD,$ED ;;
        DEFB    $F1             ;;Exponent: $81, Bytes: 4
        DEFB    $23,$5D,$1B,$EA ;;

        DEFB    $04             ;;multiply
        DEFB    $34             ;;end-calc

        RET                     ; return.


; ----------------------
; THE 'TANGENT' FUNCTION
; ----------------------
; (offset $1E: 'tan')
;
;   Evaluates tangent x as    sin(x) / cos(x).
;
;
;             /|
;          h / |
;           /  |o
;          /x  |
;         /----|
;           a
;
;   The tangent of angle x is the ratio of the length of the opposite side
;   divided by the length of the adjacent side. As the opposite length can
;   be calculates using sin(x) and the adjacent length using cos(x) then
;   the tangent can be defined in terms of the previous two functions.

;   Error 6 if the argument, in radians, is too close to one like pi/2
;   which has an infinite tangent. e.g. PRINT TAN (PI/2)  evaluates as 1/0.
;   Similarly PRINT TAN (3*PI/2), TAN (5*PI/2) etc.

;; tan
L1D6E:  RST     28H             ;; FP-CALC          x.
        DEFB    $2D             ;;duplicate         x, x.
        DEFB    $1C             ;;sin               x, sin x.
        DEFB    $01             ;;exchange          sin x, x.
        DEFB    $1D             ;;cos               sin x, cos x.
        DEFB    $05             ;;division          sin x/cos x (= tan x).
        DEFB    $34             ;;end-calc          tan x.

        RET                     ; return.

; ----------------------
; THE 'ARCTAN' FUNCTION
; ----------------------
```

```
; (Offset $21: 'atn')
;   The inverse tangent function with the result in radians.
;   This is a fundamental transcendental function from which others such as
;   asn and acs are directly, or indirectly, derived.
;   It uses the series generator to produce Chebyshev polynomials.

;; atn
L1D76:  LD      A,(HL)          ; fetch exponent
        CP      $81             ; compare to that for 'one'
        JR      C,L1D89         ; forward, if less, to SMALL

        RST     28H             ;; FP-CALC      X.
        DEFB    $A1             ;;stk-one
        DEFB    $18             ;;negate
        DEFB    $01             ;;exchange
        DEFB    $05             ;;division
        DEFB    $2D             ;;duplicate
        DEFB    $32             ;;less-0
        DEFB    $A3             ;;stk-pi/2
        DEFB    $01             ;;exchange
        DEFB    $00             ;;jump-true
        DEFB    $06             ;;to L1D8B, CASES

        DEFB    $18             ;;negate
        DEFB    $2F             ;;jump
        DEFB    $03             ;;to L1D8B, CASES

; ---

;; SMALL
L1D89:  RST     28H             ;; FP-CALC
        DEFB    $A0             ;;stk-zero

;; CASES
L1D8B:  DEFB    $01             ;;exchange
        DEFB    $2D             ;;duplicate
        DEFB    $2D             ;;duplicate
        DEFB    $04             ;;multiply
        DEFB    $2D             ;;duplicate
        DEFB    $0F             ;;addition
        DEFB    $A1             ;;stk-one
        DEFB    $03             ;;subtract

        DEFB    $8C             ;;series-0C
        DEFB    $10             ;;Exponent: $60, Bytes: 1
        DEFB    $B2             ;;(+00,+00,+00)
        DEFB    $13             ;;Exponent: $63, Bytes: 1
        DEFB    $0E             ;;(+00,+00,+00)
        DEFB    $55             ;;Exponent: $65, Bytes: 2
        DEFB    $E4,$8D         ;;(+00,+00)
        DEFB    $58             ;;Exponent: $68, Bytes: 2
        DEFB    $39,$BC         ;;(+00,+00)
        DEFB    $5B             ;;Exponent: $6B, Bytes: 2
        DEFB    $98,$FD         ;;(+00,+00)
        DEFB    $9E             ;;Exponent: $6E, Bytes: 3
        DEFB    $00,$36,$75     ;;(+00)
        DEFB    $A0             ;;Exponent: $70, Bytes: 3
        DEFB    $DB,$E8,$B4     ;;(+00)
        DEFB    $63             ;;Exponent: $73, Bytes: 2
        DEFB    $42,$C4         ;;(+00,+00)
        DEFB    $E6             ;;Exponent: $76, Bytes: 4
        DEFB    $B5,$09,$36,$BE ;;
```

```
        DEFB    $E9             ;;Exponent: $79, Bytes: 4
        DEFB    $36,$73,$1B,$5D ;;
        DEFB    $EC             ;;Exponent: $7C, Bytes: 4
        DEFB    $D8,$DE,$63,$BE ;;
        DEFB    $F0             ;;Exponent: $80, Bytes: 4
        DEFB    $61,$A1,$B3,$0C ;;

        DEFB    $04             ;;multiply
        DEFB    $0F             ;;addition
        DEFB    $34             ;;end-calc

        RET                     ; return.


; ---------------------
; THE 'ARCSIN' FUNCTION
; ---------------------
; (Offset $1F: 'asn')
;   The inverse sine function with result in radians.
;   Derived from arctan function above.
;   Error A unless the argument is between -1 and +1 inclusive.
;   Uses an adaptation of the formula asn(x) = atn(x/sqr(1-x*x))
;
;
;                 /|
;                / |
;              1/  |x
;              /a  |
;             /----|
;               y
;
;   e.g. We know the opposite side (x) and hypotenuse (1)
;   and we wish to find angle a in radians.
;   We can derive length y by Pythagoras and then use ATN instead.
;   Since y*y + x*x = 1*1 (Pythagoras Theorem) then
;   y=sqr(1-x*x)                          - no need to multiply 1 by itself.
;   So, asn(a) = atn(x/y)
;   or more fully,
;   asn(a) = atn(x/sqr(1-x*x))

;   Close but no cigar.

;   While PRINT ATN (x/SQR (1-x*x)) gives the same results as PRINT ASN x,
;   it leads to division by zero when x is 1 or -1.
;   To overcome this, 1 is added to y giving half the required angle and
the
;   result is then doubled.
;   That is, PRINT ATN (x/(SQR (1-x*x) +1)) *2
;
;
;               . /|
;            .  c/ |
;         .     /1 |x
;      . c   b /a  |
;    ---------/----|
;      1      y
;
;   By creating an isosceles triangle with two equal sides of 1, angles c
and
;   c are also equal. If b+c+d = 180 degrees and b+a = 180 degrees then
c=a/2.
;
```

```
;   A value higher than 1 gives the required error as attempting to find
the
;   square root of a negative number generates an error in Sinclair BASIC.

;; asn
L1DC4:  RST     28H             ;; FP-CALC      x.
        DEFB    $2D             ;;duplicate     x, x.
        DEFB    $2D             ;;duplicate     x, x, x.
        DEFB    $04             ;;multiply      x, x*x.
        DEFB    $A1             ;;stk-one       x, x*x, 1.
        DEFB    $03             ;;subtract      x, x*x-1.
        DEFB    $18             ;;negate        x, 1-x*x.
        DEFB    $25             ;;sqr           x, sqr(1-x*x) = y.
        DEFB    $A1             ;;stk-one       x, y, 1.
        DEFB    $0F             ;;addition      x, y+1.
        DEFB    $05             ;;division      x/y+1.
        DEFB    $21             ;;atn           a/2     (half the angle)
        DEFB    $2D             ;;duplicate     a/2, a/2.
        DEFB    $0F             ;;addition      a.
        DEFB    $34             ;;end-calc      a.

        RET                     ; return.


; ------------------------
; THE 'ARCCOS' FUNCTION
; ------------------------
; (Offset $20: 'acs')
;   The inverse cosine function with the result in radians.
;   Error A unless the argument is between -1 and +1.
;   Result in range 0 to pi.
;   Derived from asn above which is in turn derived from the preceding atn.
It
;   could have been derived directly from atn using acs(x) = atn(sqr(1-
x*x)/x).
;   However, as sine and cosine are horizontal translations of each other,
;   uses acs(x) = pi/2 - asn(x)

;   e.g. the arccosine of a known x value will give the required angle b in
;   radians.
;   We know, from above, how to calculate the angle a using asn(x).
;   Since the three angles of any triangle add up to 180 degrees, or pi
radians,
;   and the largest angle in this case is a right-angle (pi/2 radians),
then
;   we can calculate angle b as pi/2 (both angles) minus asn(x) (angle a).
;
;
;            /|
;          1 /b|
;           /  |x
;          /a  |
;         /----|
;           y

;; acs
L1DD4:  RST     28H             ;; FP-CALC      x.
        DEFB    $1F             ;;asn           asn(x).
        DEFB    $A3             ;;stk-pi/2      asn(x), pi/2.
        DEFB    $03             ;;subtract      asn(x) - pi/2.
        DEFB    $18             ;;negate        pi/2 - asn(x) = acs(x).
        DEFB    $34             ;;end-calc      acs(x)
```

```
        RET                     ; return.


; -------------------------
; THE 'SQUARE ROOT' FUNCTION
; -------------------------
; (Offset $25: 'sqr')
;   Error A if argument is negative.
;   This routine is remarkable for its brevity - 7 bytes.
;   The ZX81 code was originally 9K and various techniques had to be
;   used to shoe-horn it into an 8K Rom chip.


;; sqr
L1DDB:  RST     28H             ;; FP-CALC              x.
        DEFB    $2D             ;;duplicate             x, x.
        DEFB    $2C             ;;not                   x, 1/0
        DEFB    $00             ;;jump-true             x, (1/0).
        DEFB    $1E             ;;to L1DFD, LAST        exit if argument
zero
                                ;;                      with zero result.

;   else continue to calculate as x ** .5

        DEFB    $A2             ;;stk-half              x, .5.
        DEFB    $34             ;;end-calc              x, .5.


; ------------------------------
; THE 'EXPONENTIATION' OPERATION
; ------------------------------
; (Offset $06: 'to-power')
;   This raises the first number X to the power of the second number Y.
;   As with the ZX80,
;   0 ** 0 = 1
;   0 ** +n = 0
;   0 ** -n = arithmetic overflow.

;; to-power
L1DE2:  RST     28H             ;; FP-CALC              X,Y.
        DEFB    $01             ;;exchange              Y,X.
        DEFB    $2D             ;;duplicate             Y,X,X.
        DEFB    $2C             ;;not                   Y,X,(1/0).
        DEFB    $00             ;;jump-true
        DEFB    $07             ;;forward to L1DEE, XISO if X is zero.

;   else X is non-zero. function 'ln' will catch a negative value of X.

        DEFB    $22             ;;ln                    Y, LN X.
        DEFB    $04             ;;multiply              Y * LN X
        DEFB    $34             ;;end-calc

        JP      L1C5B           ; jump back to EXP routine.  ->

; ---

;   These routines form the three simple results when the number is zero.
;   begin by deleting the known zero to leave Y the power factor.

;; XISO
L1DEE:  DEFB    $02             ;;delete                Y.
```

```
        DEFB    $2D             ;;duplicate             Y, Y.
        DEFB    $2C             ;;not                   Y, (1/0).
        DEFB    $00             ;;jump-true
        DEFB    $09             ;;forward to L1DFB, ONE if Y is zero.

;   the power factor is not zero. If negative then an error exists.

        DEFB    $A0             ;;stk-zero              Y, 0.
        DEFB    $01             ;;exchange              0, Y.
        DEFB    $33             ;;greater-0             0, (1/0).
        DEFB    $00             ;;jump-true             0
        DEFB    $06             ;;to L1DFD, LAST        if Y was any
positive
                                ;;                      number.

;   else force division by zero thereby raising an Arithmetic overflow
error.
;   There are some one and two-byte alternatives but perhaps the most
formal
;   might have been to use end-calc; rst 08; defb 05.

        DEFB    $A1             ;;stk-one               0, 1.
        DEFB    $01             ;;exchange              1, 0.
        DEFB    $05             ;;division              1/0    >> error

; ---

;; ONE
L1DFB:  DEFB    $02             ;;delete                .
        DEFB    $A1             ;;stk-one               1.

;; LAST
L1DFD:  DEFB    $34             ;;end-calc              last value 1 or 0.

        RET                     ; return.

; ----------------------
; THE 'SPARE LOCATIONS'
; ----------------------

;; SPARE
L1DFF:  DEFB    $FF             ; That's all folks.



; ------------------------
; THE 'ZX81 CHARACTER SET'
; ------------------------

;; char-set - begins with space character.

; $00 - Character: ' '          CHR$(0)

L1E00:  DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
```

```
; $01 - Character: mosaic       CHR$(1)

        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000


; $02 - Character: mosaic       CHR$(2)

        DEFB    %00001111
        DEFB    %00001111
        DEFB    %00001111
        DEFB    %00001111
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000


; $03 - Character: mosaic       CHR$(3)

        DEFB    %11111111
        DEFB    %11111111
        DEFB    %11111111
        DEFB    %11111111
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000

; $04 - Character: mosaic       CHR$(4)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000

; $05 - Character: mosaic       CHR$(1)

        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000

; $06 - Character: mosaic       CHR$(1)

        DEFB    %00001111
        DEFB    %00001111
        DEFB    %00001111
```

```
        DEFB    %00001111
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000

; $07 - Character: mosaic       CHR$(1)

        DEFB    %11111111
        DEFB    %11111111
        DEFB    %11111111
        DEFB    %11111111
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000
        DEFB    %11110000

; $08 - Character: mosaic       CHR$(1)

        DEFB    %10101010
        DEFB    %01010101
        DEFB    %10101010
        DEFB    %01010101
        DEFB    %10101010
        DEFB    %01010101
        DEFB    %10101010
        DEFB    %01010101

; $09 - Character: mosaic       CHR$(1)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %10101010
        DEFB    %01010101
        DEFB    %10101010
        DEFB    %01010101

; $0A - Character: mosaic       CHR$(10)

        DEFB    %10101010
        DEFB    %01010101
        DEFB    %10101010
        DEFB    %01010101
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000

; $0B - Character: '"'          CHR$(11)

        DEFB    %00000000
        DEFB    %00100100
        DEFB    %00100100
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000

; $0B - Character:  £           CHR$(12)
```

```
        DEFB    %00000000
        DEFB    %000111100
        DEFB    %00100010
        DEFB    %01111000
        DEFB    %00100000
        DEFB    %00100000
        DEFB    %01111110
        DEFB    %00000000

; $0B - Character: '$'          CHR$(13)

        DEFB    %00000000
        DEFB    %00001000
        DEFB    %00111110
        DEFB    %00101000
        DEFB    %00111110
        DEFB    %00001010
        DEFB    %00111110
        DEFB    %00001000

; $0B - Character: ':'          CHR$(14)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00010000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00010000
        DEFB    %00000000

; $0B - Character: '?'          CHR$(15)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %00000100
        DEFB    %00001000
        DEFB    %00000000
        DEFB    %00001000
        DEFB    %00000000

; $10 - Character: '('          CHR$(16)

        DEFB    %00000000
        DEFB    %00000100
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00000100
        DEFB    %00000000

; $11 - Character: ')'          CHR$(17)

        DEFB    %00000000
        DEFB    %00100000
        DEFB    %00010000
        DEFB    %00010000
        DEFB    %00010000
        DEFB    %00010000
```

```
        DEFB    %00100000
        DEFB    %00000000

; $12 - Character: '>'          CHR$(18)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00010000
        DEFB    %00001000
        DEFB    %00000100
        DEFB    %00001000
        DEFB    %00010000
        DEFB    %00000000

; $13 - Character: '<'          CHR$(19)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000100
        DEFB    %00001000
        DEFB    %00010000
        DEFB    %00001000
        DEFB    %00000100
        DEFB    %00000000

; $14 - Character: '='          CHR$(20)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00111110
        DEFB    %00000000
        DEFB    %00111110
        DEFB    %00000000
        DEFB    %00000000

; $15 - Character: '+'          CHR$(21)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00111110
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00000000

; $16 - Character: '-'          CHR$(22)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00111110
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000

; $17 - Character: '*'          CHR$(23)

        DEFB    %00000000
        DEFB    %00000000
```

```
        DEFB    %00010100
        DEFB    %00001000
        DEFB    %00111110
        DEFB    %00001000
        DEFB    %00010100
        DEFB    %00000000

; $18 - Character: '/'          CHR$(24)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000010
        DEFB    %00000100
        DEFB    %00001000
        DEFB    %00010000
        DEFB    %00100000
        DEFB    %00000000

; $19 - Character: ';'          CHR$(25)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00010000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00010000
        DEFB    %00010000
        DEFB    %00100000

; $1A - Character: ','          CHR$(26)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00010000

; $1B - Character: '"'          CHR$(27)

        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00000000
        DEFB    %00011000
        DEFB    %00011000
        DEFB    %00000000

; $1C - Character: '0'          CHR$(28)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000110
        DEFB    %01001010
        DEFB    %01010010
        DEFB    %01100010
        DEFB    %00111100
        DEFB    %00000000
```

```
; $1D - Character: '1'          CHR$(29)

        DEFB    %00000000
        DEFB    %00011000
        DEFB    %00101000
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00111110
        DEFB    %00000000

; $1E - Character: '2'          CHR$(30)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %00000010
        DEFB    %00111100
        DEFB    %01000000
        DEFB    %01111110
        DEFB    %00000000

; $1F - Character: '3'          CHR$(31)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %00001100
        DEFB    %00000010
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %00000000

; $20 - Character: '4'          CHR$(32)

        DEFB    %00000000
        DEFB    %00001000
        DEFB    %00011000
        DEFB    %00101000
        DEFB    %01001000
        DEFB    %01111110
        DEFB    %00001000
        DEFB    %00000000

; $21 - Character: '5'          CHR$(33)

        DEFB    %00000000
        DEFB    %01111110
        DEFB    %01000000
        DEFB    %01111100
        DEFB    %00000010
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %00000000

; $22 - Character: '6'          CHR$(34)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000000
        DEFB    %01111100
        DEFB    %01000010
```

```
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %00000000

; $23 - Character: '7'          CHR$(35)

        DEFB    %00000000
        DEFB    %01111110
        DEFB    %00000010
        DEFB    %00000100
        DEFB    %00001000
        DEFB    %00010000
        DEFB    %00010000
        DEFB    %00000000

; $24 - Character: '8'          CHR$(36)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %00000000

; $25 - Character: '9'          CHR$(37)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %00111110
        DEFB    %00000010
        DEFB    %00111100
        DEFB    %00000000

; $26 - Character: 'A'          CHR$(38)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01111110
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %00000000

; $27 - Character: 'B'          CHR$(39)

        DEFB    %00000000
        DEFB    %01111100
        DEFB    %01000010
        DEFB    %01111100
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01111100
        DEFB    %00000000

; $28 - Character: 'C'          CHR$(40)

        DEFB    %00000000
```

```
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %01000000
        DEFB    %01000000
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %00000000

; $29 - Character: 'D'          CHR$(41)

        DEFB    %00000000
        DEFB    %01111000
        DEFB    %01000100
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000100
        DEFB    %01111000
        DEFB    %00000000

; $2A - Character: 'E'          CHR$(42)

        DEFB    %00000000
        DEFB    %01111110
        DEFB    %01000000
        DEFB    %01111100
        DEFB    %01000000
        DEFB    %01000000
        DEFB    %01111110
        DEFB    %00000000

; $2B - Character: 'F'          CHR$(43)

        DEFB    %00000000
        DEFB    %01111110
        DEFB    %01000000
        DEFB    %01111100
        DEFB    %01000000
        DEFB    %01000000
        DEFB    %01000000
        DEFB    %00000000

; $2C - Character: 'G'          CHR$(44)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %01000000
        DEFB    %01001110
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %00000000

; $2D - Character: 'H'          CHR$(45)

        DEFB    %00000000
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01111110
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %00000000
```

```
; $2E - Character: 'I'          CHR$(46)

        DEFB    %00000000
        DEFB    %00111110
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00001000
        DEFB    %00111110
        DEFB    %00000000

; $2F - Character: 'J'          CHR$(47)

        DEFB    %00000000
        DEFB    %00000010
        DEFB    %00000010
        DEFB    %00000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %00000000

; $30 - Character: 'K'          CHR$(48)

        DEFB    %00000000
        DEFB    %01000100
        DEFB    %01001000
        DEFB    %01110000
        DEFB    %01001000
        DEFB    %01000100
        DEFB    %01000010
        DEFB    %00000000

; $31 - Character: 'L'          CHR$(49)

        DEFB    %00000000
        DEFB    %01000000
        DEFB    %01000000
        DEFB    %01000000
        DEFB    %01000000
        DEFB    %01000000
        DEFB    %01111110
        DEFB    %00000000

; $32 - Character: 'M'          CHR$(50)

        DEFB    %00000000
        DEFB    %01000010
        DEFB    %01100110
        DEFB    %01011010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %00000000

; $33 - Character: 'N'          CHR$(51)

        DEFB    %00000000
        DEFB    %01000010
        DEFB    %01100010
        DEFB    %01010010
```

```
        DEFB    %01001010
        DEFB    %01000110
        DEFB    %01000010
        DEFB    %00000000

; $34 - Character: 'O'          CHR$(52)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %00000000

; $35 - Character: 'P'          CHR$(53)

        DEFB    %00000000
        DEFB    %01111100
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01111100
        DEFB    %01000000
        DEFB    %01000000
        DEFB    %00000000

; $36 - Character: 'Q'          CHR$(54)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01010010
        DEFB    %01001010
        DEFB    %00111100
        DEFB    %00000000

; $37 - Character: 'R'          CHR$(55)

        DEFB    %00000000
        DEFB    %01111100
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01111100
        DEFB    %01000100
        DEFB    %01000010
        DEFB    %00000000

; $38 - Character: 'S'          CHR$(56)

        DEFB    %00000000
        DEFB    %00111100
        DEFB    %01000000
        DEFB    %00111100
        DEFB    %00000010
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %00000000

; $39 - Character: 'T'          CHR$(57)
```

```
        DEFB    %00000000
        DEFB    %11111110
        DEFB    %00010000
        DEFB    %00010000
        DEFB    %00010000
        DEFB    %00010000
        DEFB    %00010000
        DEFB    %00000000

; $3A - Character: 'U'          CHR$(58)

        DEFB    %00000000
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %00111100
        DEFB    %00000000

; $3B - Character: 'V'          CHR$(59)

        DEFB    %00000000
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %00100100
        DEFB    %00011000
        DEFB    %00000000

; $3C - Character: 'W'          CHR$(60)

        DEFB    %00000000
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01000010
        DEFB    %01011010
        DEFB    %00100100
        DEFB    %00000000

; $3D - Character: 'X'          CHR$(61)

        DEFB    %00000000
        DEFB    %01000010
        DEFB    %00100100
        DEFB    %00011000
        DEFB    %00011000
        DEFB    %00100100
        DEFB    %01000010
        DEFB    %00000000

; $3E - Character: 'Y'          CHR$(62)

        DEFB    %00000000
        DEFB    %10000010
        DEFB    %01000100
        DEFB    %00101000
        DEFB    %00010000
        DEFB    %00010000
        DEFB    %00010000
```

```
        DEFB    %00000000

; $3F - Character: 'Z'          CHR$(63)

        DEFB    %00000000
        DEFB    %01111110
        DEFB    %00000100
        DEFB    %00001000
        DEFB    %00010000
        DEFB    %00100000
        DEFB    %01111110
        DEFB    %00000000

.END                                ;TASM assembler instruction.
```