```
; Disassembly of the file "C:\lab\if1-2.rom"
;
; CPU Type: Z80
;
; Created with dZ80 1.50
;
; on Sunday, 28 of April 2002 at 12:35 PM
;

; ------------------------
; last updated 14-JAN-2004
; ------------------------

#define DEFB .BYTE
#define DEFW .WORD
#define DEFM .TEXT
#define EQU  .EQU
#define ORG  .ORG

        ORG     $0000

; --------------------------------------
; FLAGS3 System Variable - IY+$7C ($5CB6)
; --------------------------------------
; Bit 0 - set when executing an extended command.
; Bit 1 - set during CRT-VARS and CLEAR #, CLOSE etc.
; Bit 2 - settable by User to force the ERR_SP routine to handle errors.
; Bit 3 - set when networking.
; Bit 4 - set during LOAD and MOVE
; Bit 5 - set during SAVE
; Bit 6 - set during MERGE
; Bit 7 - set during VERIFY
;
; Note. before initialization of FLAGS_3, this is considered to be the first
; byte of channels and so PEEK 23734 gives 244 decimal (%11110100) the high
; order byte of the Main ROM address PRINT-OUT - $09F4.
;
; ----------------------------------------

; --------------------------------
; THE 'RETURN TO MAIN ROM' ROUTINE
; --------------------------------
;   The system is initialized by the Main ROM so this address is accessed
;   solely by a RST 00H instruction.  It is used from five locations to return
;   to the Main ROM.

;; MAIN-ROM
L0000:  POP     HL              ; discard the return address in this ROM.
        LD      (IY+$7C),$00    ; reset all the bits of FLAGS_3.
        JP      L0700           ; jump forward to UNPAGE address.

; -------------------
; THE 'START' ROUTINE
; -------------------
;   An instruction fetch on address $0008 pages in this ROM.
;   The three-byte instruction at this location must exist on both sides of
```

```
;   the looking-glass.  The value fetched is immediately discarded.
;   It follows that this restart should never be invoked from this ROM.

;; ST-SHADOW
L0008:  LD      HL,($5C5D)      ; fetch character address from CH_ADD.
        POP     HL              ; pop return address to HL register.
        PUSH    HL              ; and save again on machine stack.

        JP      L009A           ; jump forward to continue at START-2.

; ----------------------------
; THE 'CALL A MAIN ROM' ROUTINE
; ----------------------------
;   Call an address in the main ROM. The address follows the restart so this
;   is as convenient and as brief as a CALL instruction.
;   The SBRT routine within the system variables area reads
;
;   L5CB9         LD      HL,value
;   L5C5C         CALL    addr
;   L5C5F         LD      (L5CB9+1),HL
;   L5CC2         RET
;
;   By immediately placing the current value of HL in the subroutine, then
;   all registers before the call are as they were before the RST
;   instruction.  The value of HL after the call is stored immediately in
;   this now redundant location so that, after this ROM is paged back in,
;   the registers, after the RST instruction has executed, are as they were
;   immediately after the CALL.
;   see START-2.

;; CALBAS
L0010:  LD      ($5CBA),HL      ; insert the current value of HL in the
                                ; Z80 code to be picked up later.

        POP     HL              ; drop the return address - the location
                                ; of address to be called.
        PUSH    DE              ; preserve the DE register contents.

        JR      L0081           ; forward to continue at CALBAS-2.

        DEFB    $FF             ; unused.

; ---------------------------------------------
; THE 'TEST IF SYNTAX IS BEING CHECKED' ROUTINE
; ---------------------------------------------
;   On the ZX80, testing the syntax flag was done with the 4-byte
;   instruction that tests the System Variable FLAGS.  On the ZX81 and
;   ZX Spectrum, a call to SYNTAX-Z reduced the invocation to a three-byte
;   CALL.   Here it is reduced to a one-byte restart.

;; CHKSYNTAX
L0018:  BIT     7,(IY+$01)      ; test most significant bit of FLAGS
        RET                     ; return the result.
                                ; (Z = Syntax, NZ = Run-time)

        DEFB    $FF             ; unused.
        DEFB    $FF             ; unused.
        DEFB    $FF             ; unused.
```

```
; -------------------------
; THE 'SHADOW-ERROR' ROUTINE
; -------------------------
;   This is similar to the Main ROM error handler and the following byte
;   indicates the type of error and in runtime the message that should be
;   printed.  If checking syntax then the error pointer is set before a
;   return is made to the Main ROM.

;; SH-ERR
L0020:  RST     18H             ; checking syntax ?
        JR      Z,L0068         ; forward, if so, to ST-ERROR

        JR      L003A           ; forward, in run-time, to TEST-SP,
                                ; and then REP-MSG

        DEFB    $FF             ; unused.
        DEFB    $FF             ; unused.
        DEFB    $FF             ; unused.

; ----------------------------------
; THE 'MAIN ROM ERROR RESTART' ROUTINE
; ----------------------------------
;   This restart invokes the error handler of the Main 16K ROM.  The required
;   error number is usually first placed in the System Variable ERR_NR.  In
;   some cases the error code is already present and this restart is used when
;   the error situations handled by this ROM have been eliminated.
;   Since the exit from this point is by manipulating the stack, the return
;   address is of no importance as that route is never taken.  There are also
;   three conditional jumps back to this point.

;; ROMERR
L0028:  RES     3,(IY+$02)      ; update TV_FLAG - signal no change in mode.
        JR      L0040           ; forward to RMERR-2.

        DEFB    $FF             ; unused.
        DEFB    $FF             ; unused.

; ------------------------------------------------
; THE 'CREATE NEW SYSTEM VARIABLES RESTART' ROUTINE
; ------------------------------------------------
;    This restart is used the first time that that the ROM is paged in to
;    create the System Variables.  This will be either by an instruction
;    fetch on $0008 or $1708.

;; NEWVARS
L0030:  JP      L01F7           ; jump to CRT-VARS

        DEFB    $FF             ; unused.
        DEFB    $FF             ; unused.
        DEFB    $FF             ; unused.
        DEFB    $FF             ; unused.
        DEFB    $FF             ; unused.

; -------------------------------
; THE 'MASKABLE INTERRUPT' ROUTINE
; -------------------------------
;   There is no service routine but should the routine be called either
;   directly or by straying into a RST $38 instruction, then interrupts are
;   enabled.
```

```
;; INT-SERV
L0038:  EI                      ; Enable Interrupts
        RET                     ; return.


; ------------------------
; THE 'TEST SYSTEM' BRANCH
; ------------------------
;   This branch allows the user to trap errors before this ROM is used to print
;   the error report.

;; TEST-SP
L003A:  CALL    L0077           ; routine CHECK-SP
                                ; usually returns.
        JP      L0260           ; jump to REP-MSG

; ---------------------------
; THE 'MAIN ROM ERROR' ROUTINE
; ---------------------------
;   a continuation of RST 28H.
;   This ROM has inserted a Main ROM error code into ERR_NR and the routine in
;   the Main ROM is now invoked.
;   First a check is made to see if the user wishes to trap errors using a
;   custom routine in ERR_SP.  This will be used in the syntax path anyway.
;

;; RMERR-2
L0040:  RST     18H             ; checking syntax ?
        JR      Z,L0068         ; forward, if so, to ST-ERROR.

        CALL    L0077           ; routine CHECK-SP allows the user to trap
                                ; run-time errors at this point but normally
                                ; returns here.

        CALL    L17B7           ; routine RCL-T-CH reclaims any temporary
                                ; channels and stops all microdrive motors.

        BIT     1,(IY+$7C)      ; test FLAGS_3.
        JR      Z,L0068         ; forward, if executing CLOSE, to ST-ERROR.

        BIT     4,(IY+$7C)      ; test FLAGS_3 - loading filename 'run' ?
        JR      Z,L0068         ; forward, if not, to ST-ERROR.

;   As a security measure, the file 'run' can not be hacked.

        LD      A,(IY+$00)      ; fetch error number from the System Variable
                                ; ERR_NR.
        CP      $14             ; is it "CODE error" ?
        JR      NZ,L0068        ; forward, if not, to ST-ERROR.

;   The user has pressed BREAK while trying to load the program 'run'.

        LD      HL,$0000        ; cause a system reset.
        PUSH    HL              ; place address zero on machine stack.
        RST     00H             ; switch to MAIN-ROM.

;
```

```
        DEFB    $FF             ; unused
        DEFB    $FF             ; unused
        DEFB    $FF             ; unused
        DEFB    $FF             ; unused
        DEFB    $FF             ; unused

; -----------------------------------
; THE 'NON-MASKABLE INTERRUPT' ROUTINE
; -----------------------------------
;   There is no NMI functionality.

;; NMINT-SRV
L0066:  RETN                    ; return to previous interrupt state.


; -------------------------
; THE 'SYNTAX ERROR' ROUTINE
; -------------------------
;   An error has occurred during syntax checking so the position must be
;   highlighted when a return is made to the Editor in the Main ROM.

;; ST-ERROR
L0068:  LD      HL,($5C5D)      ; fetch character address from CH_ADD.
        LD      ($5C5F),HL      ; set X_PTR to same to position error cursor.

        LD      SP,($5C3D)      ; set the Stack Pointer from ERR_SP.

        LD      HL,$16C5        ; prepare address of main SET-STK.
        PUSH    HL              ; push on the machine stack.

        RST     00H             ; switch to MAIN-ROM where SET-STK will clean up
                                ; the work areas before returning to the Error
                                ; Routine obtained from ERR_SP.

; --------------------------------------
; THE 'CHECK ERROR STACK POINTER' ROUTINE
; --------------------------------------
;   This allows the user's software to trap any errors at this point by setting
;   the otherwise unused bit 2 of FLAGS_3 after inserting a custom error
;   handler in the System Variable ERR_SP.
;   Both Shadow ROM situations and Main ROM situations can be trapped and the
;   routine is called from BOTH RST 20H and RST 28H.

;; CHECK-SP
L0077:  BIT     2,(IY+$7C)      ; test FLAGS_3 has the user set up a custom
                                ; error handler in Main RAM ?
        RET     Z               ; return if not.

;   Otherwise the user, or the third party software, has set up a custom routine
;   in the system variable ERR_SP and set bit 2 of FLAGS_3 so that it is invoked
;   at this point.

        LD      SP,($5C3D)      ; set stack pointer from ERR_SP.
        RST     00H             ; switch to MAIN-ROM.

; ---------------------
; THE 'CALBAS-2' ROUTINE
; ---------------------
;   A continuation of the code at $0010.
```

```
;   Continue by picking up the address to be called, located after the RST
;   instruction and placing after the CALL instruction in the SBRT sequence.

;; CALBAS-2
L0081:  LD      E,(HL)          ; fetch low byte of called address
        INC     HL              ; advance pointer.
        LD      D,(HL)          ; fetch high byte.

        LD      ($5CBD),DE      ; place in the Z80 code SBRT
        INC     HL              ; increment pointer.

        EX      (SP),HL         ; transfer continuation address to machine
                                ; stack - and the stack value (was DE) to HL.

        EX      DE,HL           ; original DE value now restored.

        LD      HL,$0000        ; signal CALBAS routine in use.
        PUSH    HL              ; place on stack.

        LD      HL,$0008        ; address of main ERROR restart
        PUSH    HL              ; place on stack

        LD      HL,$5CB9        ; address of calling SBRT subroutine.
        PUSH    HL              ; place on stack.

        JP      L0700           ; jump to UNPAGE

; ---------------------
; THE 'CONTROL' ROUTINE
; ---------------------
;   A continuation of code at L0008. The return address has been dropped off
;   the machine stack into HL.
;
;   First see if this ROM was paged in as a result of the $0008 address
;   stacked during the CALBAS routine. (see above)

;; START-2
L009A:  PUSH    AF              ; preserve accumulator and status flags.

        LD      A,H             ; test HL for zero - the CALBAS
        OR      L               ; indicator value.
        JR      NZ,L00A5        ; forward, if not, to START-3.

        POP     AF              ; restore accumulator and flags.

        POP     HL              ; discard address stacked by RST 08.
        LD      HL,($5CBA)      ; pick up post-CALL HL value from SBRT.
        RET                     ; return.

;-------------------------------------------------------------------------------
;   Now consider that the address $0008 may have been an input or output
;   routine that precedes the letter of one of the new channels.  These
;   paging addresses ensure that this ROM is paged in so that the real
;   input/output addresses can be read from the locations after the
;   channel's letter. In this case, the return address is towards the end
;   of the CALL-SUB routine in the Main ROM, i.e.
;   L15FB         CALL    $162C   ; routine CALL-JUMP (a JP (HL) instr.)
;   L15FE         POP     HL      ; return address
;-------------------------------------------------------------------------------
```

```
;; START-3
L00A5:  PUSH    DE              ; preserve DE.
        LD      DE,$15FE        ; test against possible return address 0x15FE
        SBC     HL,DE           ; subtract (carry is clear)
        POP     DE              ; restore DE.
        JR      NZ,L00BC        ; forward with no match to START-4.

;   This ROM has been paged by an attempt to use a stream.

        POP     AF              ; restore accumulator.

        LD      HL,L0700        ; stack the address UNPAGE to switch to
        PUSH    HL              ; the Main ROM afterwards.

        LD      HL,$0004        ; the shadow routine is 4 bytes forward
        ADD     HL,DE           ; adjust input/output address pointer.
        LD      E,(HL)          ; pick up low-order byte of I/O routine.
        INC     HL              ; bump pointer.
        LD      D,(HL)          ; pick up high-order byte of routine.
        EX      DE,HL           ; transfer I/O address to HL.

        JP      (HL)            ; jump to routine and then to UNPAGE

; ---

;   By elimination, the address $0008 has been reached as a result of a
;   RST 08 instruction in the Main ROM.  This may be the very first time
;   that this ROM has been paged in after startup or NEW.

;; START-4
L00BC:  RST     30H             ; create new system variables if first time.

        LD      A,$01           ; %00000001
        OUT     ($F7),A         ;

        LD      A,$EE           ; %11101110
        OUT     ($EF),A         ;

        POP     AF              ; temporarily drop the accumulator.
        POP     HL              ; fetch address of error code/hook code to HL.
        PUSH    AF              ; save accumulator again.

;   Note. the address of the code could be anywhere in the 64K address space
;   but it is not in this ROM.  Luckily in the Main ROM at $007B is the
;   sequence ld a,(hl) ; ret which will fetch the unknown error code from
;   the known address.

        RST     10H             ; CALBAS
        DEFW    $007B           ; main TEMP-PTR3

        LD      ($5C3A),A       ; place the error code in sysvar ERR_NR

;   The error code at this stage is one less than actual code.

        CP      $FF             ; is it 'OK'
        JR      NZ,L00E9        ; forward, if not, to TEST-CODE

        BIT     1,(IY+$7C)      ; test FLAGS_3  - first time ?
```

```
        JR      Z,L00E7         ; forward, if not, to NREPORT-2
                                ; 'Program finished'

        BIT     7,(IY+$0C)      ; test PPC_hi - a direct command ?
        JR      Z,L00E7         ; forward, if not, to NREPORT-2

        LD      HL,($5C59)      ; use E_LINE to address the first character of
                                ; the edit buffer.
        LD      A,(HL)          ; searching for RUN without whitespace.

        CP      $F7             ; is character the token 'RUN' ?
        JP      Z,L0A99         ; jump forward, if so, to LOAD-RUN

;; NREPORT-2
L00E7:  RST     20H             ; Shadow Error Restart
        DEFB    $FF             ; 'Program finished'

; ---

;   Continue to consider the error code.  This may have occurred after the
;   Error RESTART in the Main ROM - range $00 (NEXT without FOR) to
;   $1A (Tape Loading Error) or a RESTART in RAM which could also include
;   the Hook Codes.

;; TEST-CODE
L00E9:  SUB     $1B             ; subtract lowest Hook Code (PAUSE)
        JP      NC,L1E71        ; jump, if same or higher, to HOOK-CODE

        CP      $F0             ; was it $0B 'Nonsense in basic'
        JR      Z,L00FB         ; forward to COPYCHADD

        CP      $F3             ; was it $0D 'Invalid file name'
        JR      Z,L00FB         ; forward to COPYCHADD

        CP      $FC             ; was it $17 'Invalid stream'
        JP      NZ,L0028        ; jump, if not, to ROMERR

;   If one of the above three reports, then this is possibly an extended
;   command and further investigation is required. A number of situations
;   may apply. The error could have occurred -
;
;   1) In INPUT - just pass control back to Main ROM. This is just a normal
;      Nonsense in BASIC and will not be due to anything new.
;   2) While already investigating an error. Too much - just use Main ROM.
;   3) While entering a new or modified line and syntax failed.
;   4) While running the program and an error was encountered.
;
;   The character address CH_ADD is not much use as that is the place
;   after the command where the standard ROM encountered an error.
;   It will be required by the Main ROM if control is passed back so, in
;   order that the Main ROM parsing routines can be used, make a copy of the
;   error character position. We will have to work forward from the
;   beginning of the line if checking syntax or from the start of the
;   program in run-time so that the errant command can be found. It may also
;   be necessary to remove hidden characters from the BASIC line.

;; COPYCHADD
L00FB:  LD      HL,($5C5D)      ; fetch character address from CH_ADD and
        LD      ($5CCB),HL      ; store in shadow system variable CHADD_
```

```
        POP     AF              ; restore accumulator.

        BIT     5,(IY+$37)      ; test FLAGX - in INPUT mode ?

        JP      NZ,L0028        ; jump back, if so, to ROMERR

;   Continue if in Editing or Run-time Mode.

        BIT     0,(IY+$7C)      ; test FLAGS_3 - already extended command ?
        JP      NZ,L0028        ; jump, if so, to ROMERR

;   else signal - handling an extended command - so that such a double error
;   can be trapped.

        SET     0,(IY+$7C)      ; update FLAGS_3 - signal executing an
                                ; extended command.

        RST     18H             ; checking syntax ?

        JR      NZ,L011B        ; skip forward, if not, to RUNTIME

        LD      (IY+$0C),$FF    ; set bit 7 of PPC_hi to indicate a line
                                ; entry situation.

;    In both cases, load B with the statement number where the error was
;    encountered.  Previous validated statements are not to be disturbed.

;; RUNTIME
L011B:  LD      B,(IY+$0D)      ; load B with statement number from SUBPPC
        LD      C,$00           ; and set C to zero for a quotes flag.

        BIT     7,(IY+$0C)      ; test PPC_hi - line entry ?
        JR      Z,L0130         ; forward, if not, to PROG-LINE

;   An edit line may have a line number at start and whitespace. We need to
;   set CH_ADD at the first command.

        PUSH    BC              ; save BC

        RST     10H             ; CALBAS
        DEFW    $19FB           ; main E-LINE-NO fetches any line number to
                                ; BC, setting CH_ADD at the command token.

        POP     BC              ; restore BC - discarding line number.

        RST     10H             ; CALBAS
        DEFW    $0018           ; main GET-CHAR gets first command of the
                                ; first statement of the errant line.

        JR      L016F           ; forward to statement loop - S-STAT to find
                                ; the errant statement.

; ---

;; PROG-LINE
L0130:  LD      HL,($5C53)      ; set pointer to start of program from PROG.

;; SC-L-LOOP
```

```
L0133:  LD      A,($5C46)       ; fetch high byte of errant line from PPC_hi
        CP      (HL)            ; compare with tested high byte.
        JR      NC,L013B        ; forward, if errant line higher or same,
                                ; to TEST-LOW

; else, unusually, the current line is not there so let Main ROM handle.

;; NREPORT-1
L0139:  RST     20H             ; Shadow Error Restart
        DEFB    $00             ; Nonsense in BASIC

; ---

;; TEST-LOW
L013B:  INC     HL              ; increment program pointer to address low byte.
        JR      NZ,L0144        ; forward, if high bytes not same, to LINE-LEN

        LD      A,($5C45)       ; fetch low byte of current line from PPC_lo
        CP      (HL)            ; compare to addressed byte.
        JR      C,L0139         ; back, if not in program area, to NREPORT-1


;; LINE-LEN
L0144:  INC     HL              ; increment program
        LD      E,(HL)          ; pointer and
        INC     HL              ; pick up the
        LD      D,(HL)          ; length of the BASIC line
        INC     HL              ; resting at the first character.

        JR      Z,L016F         ; forward, if line numbers matched, to S-STAT
                                ; the mid-entry point of the statement loop.

        ADD     HL,DE           ; else add length to current address.
        JR      L0133           ; loop back to SC-L-LOOP

; --------------------
; THE 'STATEMENT LOOP'
; --------------------
;    Entered at mid-point S-STAT with statement counter in B and a quotes
;    counter, C, set at an even zero.

;; SKIP-NUM
L014E:  LD      DE,$0006        ; a hidden floating point number has six bytes.
        ADD     HL,DE           ; add to skip to next character.

; -> The Looping Point.

;; EACH-ST
L0152:  LD      A,(HL)          ; fetch addressed BASIC character.
        CP      $0E             ; is it the hidden number indicator ?
        JR      Z,L014E         ; back to SKIP-NUM to ignore.

        INC     HL              ; else increase pointer.

        CP      $22             ; is it quotes character '"' ?
        JR      NZ,L015D        ; skip forward, if not, to CHKEND

        DEC     C               ; decrement quotes counter.
```

```
;; CHKEND
L015D:  CP      $3A             ; is character ':' ?
        JR      Z,L0165         ; skip forward to CHKEVEN

        CP      $CB             ; is character 'THEN' ?
        JR      NZ,L0169        ; skip forward to CHKEND-L


;; CHKEVEN
L0165:  BIT     0,C             ; are quotes balanced ?
        JR      Z,L016F         ; forward, if so, to S-STAT
                                ; for next statement.

;   A carriage return must not appear within quotes.

;; CHKEND-L
L0169:  CP      $0D             ; carriage return ?
        JR      NZ,L0152        ; back, if not, to EACH-ST

        JR      L0139           ; back to NREPORT-1
                                ; 'Nonsense in BASIC'


;    The Statement Loop Entry Point -->

;; S-STAT
L016F:  DJNZ    L0152           ; decrement statement counter and loop back
                                ; to EACH-ST.

;    The errant statement has been located and CH_ADD is set to start.

        DEC     HL              ; point to start or ':'

        LD      ($5C5D),HL      ; set the Main ROM system variable CH_ADD

        RST     18H             ; checking syntax ?

        JR      NZ,L01AA        ; forward, if not, to CL-WORK

        BIT     7,(IY+$0C)      ; test PPC_hi - is it an Edit Line ?
        JP      Z,L01F0         ; jump forward, if not, to ERR-6.

        DEC     HL              ; prepare to enter loop below.

        LD      C,$00           ; ??

;   It is well to reflect on what has been achieved up to this point.  At
;   each statement, the first attempt at validation is made by the Main ROM.
;   Then if that should encounter something not to its liking, this ROM has
;   a bash.  There could be ten or more statements before this one and each
;   will have been validated by the Main ROM or by this routine.  As part of
;   that validation process, when a number is parsed, then the integer or
;   floating point form of the number is inserted after the digits, rendered
;   invisible by a CHR$(14).
;
;   Once a statement has passed validation by either ROM, then it is not
;   undone.  If, say, the Main ROM has failed on the third statement of
;
;   10 PRINT "Hi :" : LET vat = 15 : OPEN# 7, "T" : LET tax = cost * (vat/100)
```

```
;
;   then it will have already inserted six bytes after the '7' before raising
;   the error 'Invalid stream'.  This ROM has located the separator before
;   the command but needs to remove the hidden numbers before parsing the
;   statement as the latter process will put them back in and we can't
;   double up.  The easiest way to do this is to search for hidden numbers
;   right to the end of the line.  There won't be any after this statement
;   but stopping at a CHR$(13) is easier than considering end of statement
;   markers in quotes.  It seems that this neat solution was not arrived at
;   immediately and the instruction, above, sets C to the quotes flag again
;   and it is needlessly preserved on the stack.
;
;   The end-user is oblivious to this elegant toing and froing between ROMS
;   and the unseen error code generation and cancellation. All that is
;   apparent is that when the RETURN key is pressed, the line simply enters
;   the program.

;; RCLM-NUM
L0182:  INC     HL              ; increment character pointer
        LD      A,(HL)          ; fetch the character.

        CP      $0E             ; is it the number marker ?
        JR      NZ,L01A5        ; forward, if not, to NEXTNUM

        PUSH    BC              ; preserve BC (zero)

        LD      BC,$0006        ; six bytes to reclaim.

        RST     10H             ; CALBAS
        DEFW    $19E8           ; main RECLAIM-2

        PUSH    HL              ; preserve character pointer.

        LD      DE,($5CCB)      ; fetch error pointer from CHADD_
        AND     A               ; prepare for true subtraction.
        SBC     HL,DE           ; test if character position less than error.
        JR      NC,L01A3        ; forward, if not, to NXT-1

        EX      DE,HL           ; transfer CHADD_ value to HL.
        LD      BC,$0006        ;
        AND     A               ;
        SBC     HL,BC           ; reduce by six.
        LD      ($5CCB),HL      ; store back in system variable CHADD_

;; NXT-1
L01A3:  POP     HL              ; restore character pointer.
        POP     BC              ; and restore BC (zero)

;; NEXTNUM
L01A5:  LD      A,(HL)          ; fetch character.
        CP      $0D             ; carriage return ?
        JR      NZ,L0182        ; loop back, if not, to RCLM-NUM

;  The run-time path rejoins here

;; CL-WORK
L01AA:  RST     10H             ; CALBAS
        DEFW    $16BF           ; main SET-WORK
```

```
        CALL    L0255           ; routine RES-VARS sets new system variables
                                ; from that following CHADD_ to that preceding
                                ; COPIES to the value $FF.

        RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR advances CH_ADD and fetches
                                ; the command character.

        SUB     $CE             ; reduce tokens - why?

        CP      $01             ; 'CAT' ?
        JP      Z,L0486         ; jump to CAT-SYN

        CP      $02             ; 'FORMAT' ?
        JP      Z,L04B4         ; jump to FRMT-SYN

        CP      $03             ; 'MOVE' ?
        JP      Z,L053D         ; jump to MOVE-SYN

        CP      $04             ; 'ERASE' ?
        JP      Z,L0531         ; jump to ERASE-SYN

        CP      $05             ; 'OPEN #' ?
        JP      Z,L04ED         ; jump to OPEN-SYN

        CP      $2A             ; 'SAVE' ?
        JP      Z,L082F         ; jump to SAVE-SYN

        CP      $21             ; 'LOAD' ?
        JP      Z,L0898         ; jump to LOAD-SYN

        CP      $08             ; 'VERIFY' ?
        JP      Z,L08A2         ; jump to VERIF-SYN

        CP      $07             ; 'MERGE' ?
        JP      Z,L08AC         ; jump to MRG-SYN

        CP      $2D             ; 'CLS' ?
        JP      Z,L0559         ; jump to CLS#-SYN

        CP      $2F             ; 'CLEAR' ?
        JP      Z,L057F         ; jump to CLR#-SYN

;   If none of the new extended commands then load HL from the VECTOR
;   system variable which normally points to the error routine below.
;   However the user, or a third party software publisher, may have
;   altered the vector to point to their own extended BASIC routines.

;; ERR-V
L01EC:  LD      HL,($5CB7)      ; fetch address from system variable VECTOR
        JP      (HL)            ; jump to address.

; ---

;; ERR-6
L01F0:  LD      HL,($5CCB)      ; fetch original character address from
                                ; CHADD_
        LD      ($5C5D),HL      ; and place in standard CH_ADD
        RST     28H             ; Error Main ROM.
```

```
; ----------------------------------------
; THE 'CREATE NEW SYSTEM VARIABLES' ROUTINE
; ----------------------------------------
;   A continuation of the restart code at $0030. A check is made to see if
;   the 58 variables already exist and the stack is set up to create the
;   room using the main ROM routine. If there isn't 58 free bytes available
;   then an 'Out of memory' report is generated by the Main ROM.

;; CRT-VARS
L01F7:  LD      HL,($5C4F)      ; system variable CHANS normally   $5CB6.
        LD      DE,$A349        ; add test value                   $A349.
                                ;                                   ----
        ADD     HL,DE           ; add - if uninitialized will give $FFFF.
        JR      C,L023D         ; forward, if higher, to VAR-EXIST

        LD      HL,L0224        ; prepare address of DEFAULT routine
        PUSH    HL              ; push on machine stack

        LD      HL,($5C63)      ; use system variable STKBOT
        LD      ($5C65),HL      ; to set system variable STKEND

        LD      HL,$5C92        ; use system variable MEMBOT
        LD      ($5C68),HL      ; to set system variable MEM

        LD      HL,$5CB5        ; the last standard system variable.
                                ; P-RAMT_hi - the location before new area.
        LD      BC,L003A        ; 58 bytes to allocate.

;   Now call MAKE-ROOM in the Main ROM by placing a sequence of addresses
;   on the machine stack as it is not possible to use the CALBAS routine yet.

        LD      DE,$0000        ; indicator - signals Main ROM has been used.
        PUSH    DE              ; stack word.

        LD      E,$08           ; form address $0008 in Main ROM.
        PUSH    DE              ; stack word.

        LD      DE,$1655        ; the Main ROM address MAKE-ROOM.
        PUSH    DE              ; stack word.

;   The machine stack now has the hierarchy DEFAULT; $0000; ERROR-1;
;   MAKE-ROOM which will be handled in reverse order.

        JP      L0700           ; jump to UNPAGE.

;   After creating room and paging this ROM back in, 'return' to the next
;   address which was the first in the sequence pushed on machine stack
;   earlier.

;; DEFAULT
L0224:  LD      HL,L0242        ; default system variable values.
        LD      BC,$0013        ; nineteen bytes to move.
        LD      DE,$5CB6        ; old CHANS area, new sysvar FLAGS_3.
        LDIR                    ; copy the bytes.

;   Note. So far the value in the accumulator, which may be the number of a
;   stream to close, has not been altered.  This next instruction is worded
;   wrongly and
```

```
;
;   OPEN #7,"s" : CLOSE #7
;
;   may not work.
;   The fix would be to use 'ld hl $5cef ; ld (hl), $01' (5 bytes)
;   or even 'dec h ; ld ($5cee),hl' (4 bytes)
;   The next pair of instructions would have been better if executed using
;   the HL register pair also.

        LD      A,$01           ; set accumulator to 1.
        LD      ($5CEF),A       ; set system variable COPIES.

        LD      (IY+$77),$50    ; set NMI_ADD_hi to eighty.
        LD      (IY+$76),$00    ; set NMI_ADD_lo to zero.

        RET                     ; return.

; ---

;   The extended System Variables already exist.

;; VAR-EXIST
L023D:  RES     1,(IY+$7C)      ; reset indicator in FLAGS_3.
        RET                     ; return.


; -----------------------------------------
; THE 'SYSTEM VARIABLES DEFAULT VALUES' TABLE
; -----------------------------------------
;   These are the initial values of the first section of the extended System
;   Variables that are copied, once only, to a newly opened area following
;   the standard 48K Spectrum System Variables.  The memory area that was at
;   this location (CHANS) is moved upwards to make room.
;   The first new location (which was the first byte of CHANS) is now
;   FLAGS_3, accessible by the IY register, and normally zero when the Main
;   ROM becomes active again.  Bit 1 is set when a CLEAR# is active and also
;   by the copy itself.

;; SV-DEFS
L0242:  DEFB    $02             ; FLAGS3 (with bit 1 already set).
        DEFW    $01F0           ; VECTOR

        LD      HL,$0000        ; SBRT located at $5CB9
        CALL    $0000           ;
        LD      ($5CBA),HL      ;
        RET                     ;

        DEFW    $000C           ; BAUD
        DEFB    $01             ; NTSTAT
        DEFB    $00             ; IOBORD - black.
        DEFW    $0000           ; SER_FL


; ---------------------------------------
; THE 'RESET NEW SYSTEM VARIABLES' ROUTINE
; ---------------------------------------
;   The central area is filled with $FF bytes.
;   This occurs whenever a new extended command is invoked.
```

```
;; RES-VARS
L0255:  LD      HL,$5CCD        ; set pointer to NTRESP - start of area.
        LD      B,$22           ; thirty four bytes to fill.

;; EACH-VAR
L025A:  LD      (HL),$FF        ; insert a default $FF value.
        INC     HL              ; bump the pointer.
        DJNZ    L025A           ; loop back to EACH-VAR.

        RET                     ; return.

; -----------------------------------
; THE 'SHADOW REPORT PRINTING' ROUTINE
; -----------------------------------
;   This routine prints the error reports of the Shadow ROM.
;   These relate to the code that follows a RST 20H restart.  The error code
;   is not printed as it would conflict with Main ROM reports.  The text of
;   the message is printed and then the Main ROM routine is used to print a
;   comma and then the line number and statement.  For example,
;   Program finished, 0:1
;   The code is similar to that at MAIN-4 in the Main ROM.  Some improvements
;   have been made but at least one slight error has been replicated.

;; REP-MSG
L0260:  LD      (IY+$7C),$00    ; clear FLAGS_3 in preparation for leaving
                                ; this ROM.

        EI                      ; Enable Interrupts.

        HALT                    ; wait for the first interrupt.

        CALL    L17B7           ; routine RCL-T-CH reclaims any temporary
                                ; channels and stops any running drive motor.

        RES     5,(IY+$01)      ; update FLAGS - 'Ready for new key'.
        BIT     1,(IY+$30)      ; test FLAGS2 - is printer buffer empty ?
        JR      Z,L0276         ; forward, if so, to FETCH-ERR

        RST     10H             ; CALBAS - call a Base ROM routine.
        DEFW    $0ECD           ; main routine - COPY-BUFF
                                ; Note. the programmer has neglected to
                                ; set bit 1 of FLAGS first.

;; FETCH-ERR
L0276:  POP     HL              ; drop the return address - after RST.
        LD      A,(HL)          ; fetch the error code.
        LD      (IY+$00),A      ; place in system variable ERR_NR.
        INC     A               ; increment setting zero if was $FF.
        PUSH    AF              ; save actual code and status flags.

        LD      HL,$0000        ; prepare to blank some system variables.
        LD      (IY+$37),H      ; clear all the bits of FLAGX.
        LD      (IY+$26),H      ; blank X_PTR_hi to suppress error marker.
        LD      ($5C0B),HL      ; blank DEFADD to signal that no defined
                                ; function is being evaluated.

        INC     L               ; select offset of 1 (explicit in main ROM ).
        LD      ($5C16),HL      ; update STRMS_00 - inputs from keyboard.
```

```
        RST     10H             ; CALBAS
        DEFW    $16B0           ; main SET-MIN clears workspace etc.

        RES     5,(IY+$37)      ; update FLAGX - signal in EDIT mode
                                ; not INPUT mode.
                                ; Note. all the bits were reset earlier.

        RST     10H             ; CALBAS
        DEFW    $0D6E           ; main CLS-LOWER

        SET     5,(IY+$02)      ; update TV_FLAG - signal lower screen
                                ; requires clearing.
        RES     3,(IY+$02)      ; update TV_FLAG - no change in mode.

        POP     AF              ; restore the incremented error code.
        LD      HL,L02BF        ; start search at REP-MSGS table below.
        LD      B,$04           ; roughly ensure that BC does not limit
                                ; search area as code must be found.
        CPIR                    ; search for code $00 - $17 skipping
                                ; all ASCII text.

;   At this point HL addresses first character of message.

;; PR-REP-LP
L02A7:  LD      A,(HL)          ; fetch each character in turn.
        CP      $20             ; compare to space.
        JR      C,L02B4         ; forward if less to END-PR-MS

        PUSH    HL              ; save the character pointer
        RST     10H             ; CALBAS
        DEFW    $0010           ; main PRINT-A

        POP     HL              ; restore pointer
        INC     HL              ; and increment.
        JR      L02A7           ; loop back to PR-REP-LP

; ---

;; END-PR-MS
L02B4:  LD      SP,($5C3D)      ; set machine stack pointer from ERR_SP
        INC     SP              ; prepare to overwrite the MAIN-4
        INC     SP              ; address $1303.
        LD      HL,$1349        ; substitute with the part that prints
                                ; the comma and line statement.
        PUSH    HL              ; push address to base of stack.
        RST     00H             ; return to MAIN-ROM.

;   Note. at this stage we have, say, "Program finished" on the screen and
;   the Main ROM routine at $1349 will complete the ", 0:1" part looping
;   back to MAIN-2 to put $1303 on the stack again.

; -----------------------------------
; THE 'SHADOW REPORT MESSAGES' ROUTINE
; -----------------------------------
;   These are the Shadow Error Reports.  Note. that the never used
;   "Header mismatch error" has been largely reclaimed.  Each error code,
;   which must be less than a space, serves to delimit the preceding text.
;   The final delimiter might just as well be $18.
```

```
;; REP-MSGS
L02BF   DEFB    $00
        DEFM    "Program finished"
        DEFB    $01
        DEFM    "Nonsense in BASIC"     ; Duplicate of a Main ROM error
        DEFB    $02
        DEFM    "Invalid stream number"
        DEFB    $03
        DEFM    "Invalid device expression"
        DEFB    $04
        DEFM    "Invalid name"
        DEFB    $05
        DEFM    "Invalid drive number"
        DEFB    $06
        DEFM    "Invalid station number"
        DEFB    $07
        DEFM    "Missing name"
        DEFB    $08
        DEFM    "Missing station number"
        DEFB    $09
        DEFM    "Missing drive number"
        DEFB    $0A
        DEFM    "Missing baud rate"
        DEFB    $0B
        DEFM    "er mismatch e"         ; Note. remnants of unused text.
        DEFB    $0C
        DEFM    "Stream already open"
        DEFB    $0D
        DEFM    "Writing to a 'read' file"
        DEFB    $0E
        DEFM    "Reading a 'write' file"
        DEFB    $0F
        DEFM    "Drive 'write' protected"
        DEFB    $10
        DEFM    "Microdrive full"
        DEFB    $11
        DEFM    "Microdrive not present"
        DEFB    $12
        DEFM    "File not found"
        DEFB    $13
        DEFM    "Hook code error"       ; not listed in manual.
        DEFB    $14
        DEFM    "CODE error"
        DEFB    $15
        DEFM    "MERGE error"
        DEFB    $16
        DEFM    "Verification has failed"
        DEFB    $17
        DEFM    "Wrong file type"
        DEFB    $18                     ; end-marker

; *****************************************
; **  T H E   S Y N T A X   R O U T I N E S  **
; *****************************************

; -------------------------------
; THE 'CAT COMMAND SYNTAX' ROUTINE
; -------------------------------
;   e.g. CAT 3
```

```
;   Without the syntax tables of the Main ROM, checking syntax is quite
;   laborious.  Although the Main ROM allowed CAT without a parameter, a
;   single expression in the range 1 - 8 is now required.  By default, CAT
;   outputs to the upper screen but output may be directed to any stream in
;   the range 0 to 15 decimal.  The subroutines used to evaluate the numeric
;   expressions use the SCANNING routine, in Main ROM, which inserts the
;   hidden five-byte numbers after any numeric arguments.

;; CAT-SYN
L0486:  LD      HL,$5CD8        ; address system variable S_STR1.
        LD      (HL),$02        ; default to stream 2 the screen.

        RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR

        CP      $0D             ; carriage return ?
        JR      Z,L0494         ; forward, if so, to MISSING-D

        CP      $3A             ; is character ':' ?

;; MISSING-D
L0494:  JP      Z,L0683         ; jump if no parameter to NREPORT-9

        CP      $23             ; is character '#' ?
        JR      NZ,L04A6        ; forward to CAT-SCRN

;   Output is directed at a specific stream.

        CALL    L064E           ; routine EXPT-STRM checks for number in range.
        CALL    L05B1           ; routine SEPARATOR checks for ',' or ';'.
        JR      NZ,L04B2        ; forward, if not present, to OREPORT-1
                                ; 'Nonsense in BASIC'

        RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR

;; CAT-SCRN
L04A6:  CALL    L061E           ; routine EXPT-NUM
        CALL    L05B7           ; routine ST-END
        CALL    L066D           ; routine CHECK-M-2 checks that drive is in
                                ; range 1 - 8.

        JP      L1AB5           ; jump forward to CAT-RUN

; ---

;; OREPORT-1
L04B2:  RST     20H             ; Shadow Error Restart
        DEFB    $00             ; Nonsense in BASIC

; ----------------------------------
; THE 'FORMAT COMMAND SYNTAX' ROUTINE
; ----------------------------------
;   e.g.

;; FRMT-SYN
L04B4:  CALL    L05F2           ; routine EXPT-SPEC
        CALL    L05B1           ; routine SEPARATOR
        JR      NZ,L04BF        ; forward to NO-FOR-M
```

```
        CALL    L062F           ; routine EXPT-NAME

;; NO-FOR-M
L04BF:  CALL    L05B7           ; routine ST-END
        LD      A,($5CD9)       ; sv L_STR1 device letter.
        CP      $54             ; is character  "T" ?
        JR      Z,L04CD         ; forward to FOR-B-T

        CP      $42             ; is character  "B" ?
        JR      NZ,L04D3        ; forward to NOT-FOR-B


;; FOR-B-T
L04CD:  CALL    L06B0           ; routine TEST-BAUD
        JP      L0ACD           ; jump to SET-BAUD

;; NOT-FOR-B
L04D3:  CP      $4E             ; is character "N" ?
        JR      NZ,L04E7        ; forward to FOR-M

        CALL    L068F           ; routine TEST-STAT
        LD      A,($5CD6)       ; sv D_STR1 drive number
        AND     A
        JP      Z,L069F         ; jump to NREPORT-6
        LD      ($5CC5),A       ; sv NTSTAT
        JP      L05C1           ; jump to END1

;; FOR-M
L04E7:  CALL    L0685           ; routine TEST-MNAM
        JP      L1ABA           ; jump to FOR-RUN

; ---------------------------------
; THE 'OPEN COMMAND SYNTAX' ROUTINE
; ---------------------------------
;

;; OPEN-SYN
L04ED:  CALL    L064E           ; routine EXPT-STRM
        CALL    L05B1           ; routine SEPARATOR
        JR      NZ,L04B2        ; back to OREPORT-1
                                ; 'Nonsense in BASIC'

        CALL    L05F2           ; routine EXPT-SPEC
        CALL    L05B1           ; routine SEPARATOR
        JR      NZ,L0500        ; forward to NOT-OP-M

        CALL    L062F           ; routine EXPT-NAME

;; NOT-OP-M
L0500:  CALL    L05B7           ; routine ST-END
        LD      A,($5CD8)       ; sv D_STR1

        RST     10H             ; CALBAS
        DEFW    $1727           ; main STR-DATA1

        LD      HL,$0011        ;
        AND     A               ;
        SBC     HL,BC           ;
```

```
        JR      C,L052F         ; forward to NREPORT-C

        LD      A,($5CD9)       ; sv L_STR1 device letter.
        CP      $54             ; "T" ?
        JR      Z,L051C         ; forward to OPEN-RS

        CP      $42             ; "B" ?
        JR      NZ,L051F        ; forward to NOT-OP-B


;; OPEN-RS
L051C:  JP      L0B4E           ; jump to OP-RSCHAN

; ---

;; NOT-OP-B
L051F:  CP      $4E             ; is character "N" ?
        JR      NZ,L0529        ; forward to OP-M-C

        CALL    L068F           ; routine TEST-STAT
        JP      L0F40           ; jump to OPEN-N-ST

; ---

;; OP-M-C
L0529:  CALL    L0685           ; routine TEST-MNAM
        JP      L1ABF           ; jump to OP-RUN

; ---

;; NREPORT-C
L052F:  RST     20H             ; Shadow Error Restart
        DEFB    $0B             ; Stream already open

; ---------------------------------
; THE 'ERASE COMMAND SYNTAX' ROUTINE
; ---------------------------------
;

;; ERASE-SYN
L0531:  CALL    L06A3           ; routine EXPT-EXPR
        CALL    L05B7           ; routine ST-END
        CALL    L0685           ; routine TEST-MNAM
        JP      L1AAB           ; jump to ERASE-RUN

; ---------------------------------
; THE 'MOVE COMMAND SYNTAX' ROUTINE
; ---------------------------------
;

;; MOVE-SYN
L053D:  CALL    L06B9           ; routine EXPT-EXP1
        CALL    L059F           ; routine EX-D-STR
        RST     10H             ; CALBAS
        DEFW    $0018           ; main GET-CHAR

        CP      $CC             ; 'TO' ?
        JR      NZ,L0584        ; forward to NONSENSE
```

```
        CALL    L06B9           ; routine EXPT-EXP1
        CALL    L059F           ; routine EX-D-STR
        RST     10H             ; CALBAS
        DEFW    $0018           ; main GET-CHAR

        CALL    L05B7           ; routine ST-END
        JP      L1AB0           ; jump to MOVE-RUN

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

;; CLS#-SYN
L0559:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR

        CP      $23             ; is the character '#' ?
        JR      NZ,L0584        ; forward, if not, to NONSENSE

        RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR

        CALL    L05B7           ; routine ST-END

        LD      HL,L0038        ; prepare a zero and black ink on white paper.
        LD      ($5C8D),HL      ; set system variables ATTR_P and MASK_P.
        LD      ($5C8F),HL      ; set system variables ATTR_T and MASK_T.
                                ; Note. not really necessary as done by CLS.
        LD      (IY+$0E),L      ; set system variable BORDCR to colour scheme.
        LD      (IY+$57),H      ; set system variable P_FLAG to zero.

        LD      A,$07           ; load A with white.
        OUT     ($FE),A         ; directly change border colour.

        RST     10H             ; CALBAS
        DEFW    $0D6B           ; main CLS clears screen and sets colours.

        JP      L05C1           ; jump forward to END1.

; ---------------------------
; THE 'CLEAR# COMMAND' ROUTINE
; ---------------------------
;

;; CLR#-SYN
L057F:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR


        CP      $23             ; '#' ?

;; NONSENSE
L0584:  JP      NZ,L04B2        ; jump to OREPORT-1
                                ; 'Nonsense in BASIC'

        RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR
```

```
        CALL    L05B7           ; routine ST-END


        XOR     A               ;

;; ALL-STRMS
L058E:  PUSH    AF              ;
        SET     1,(IY+$7C)      ; sv FLAGS_3
        CALL    L1718           ; routine CLOSE
        POP     AF              ;
        INC     A               ;
        CP      $10             ;
        JR      C,L058E         ; back to ALL-STRMS

        JP      L05C1           ; jump to END1


; -----------------------------------------------------
; THE 'EXCHANGE FILE SPECIFIERS DSTRI AND STR2' ROUTINE
; -----------------------------------------------------
;   This routine is used by the MOVE routines to bring one of the two 8-byte
;   file specifiers into context.  There were two similar routines in the
;   first Interface 1 ROM and this, the most efficient, has survived.

;; EX-D-STR
L059F:  LD      HL,$5CD6        ; sv D_STR1. drive number
        LD      DE,$5CDE        ; sv D_STR2.
        LD      B,$08           ; eight bytes to swap.

;; ALL-BYTES
L05A7:  LD      A,(DE)          ; fetch byte 1.
        LD      C,(HL)          ; fetch byte 2.
        LD      (HL),A          ; place byte 1.
        LD      A,C             ; byte 2 to accumulator.
        LD      (DE),A          ; place byte 2.

        INC     HL              ; increment the
        INC     DE              ; two pointers.
        DJNZ    L05A7           ; loop back, for all eight, to ALL-BYTES.

        RET                     ; return.


; ----------------------
; THE 'SEPARATOR' ROUTINE
; ----------------------
;   This routine returns with zero flag set if the current character is
;   either a comma or semi-colon.

;; SEPARATOR
L05B1:  CP      $2C             ; is character ',' ?
        RET     Z               ; return with zero set if so.

        CP      $3B             ; is character ';' ?
        RET                     ; return.

; ----------------------------
; THE 'END OF STATEMENT' ROUTINE
```

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

;; ST-END
L05B7:  CP      $0D             ; is character carriage return ?
        JR      Z,L05BF         ; forward, if so, to TEST-RET

        CP      $3A             ; is character a ':' ?
        JR      NZ,L0584        ; back, if not, to NONSENSE


;; TEST-RET
L05BF:  RST     18H             ; checking syntax ?
        RET     NZ              ; return if not.


; ----------------------------------------
; THE 'RETURN TO THE MAIN INTERPRETER' ROUTINE
; ----------------------------------------
;

;; END1
L05C1:  LD      SP,($5C3D)      ; sv ERR_SP
        LD      (IY+$00),$FF    ; sv ERR_NR

        LD      HL,$1BF4        ; Main ROM address STMT-NEXT

        RST     18H             ; checking syntax ?
        JR      Z,L05E0         ; forward, if so, to RETAD-SYN

        LD      A,$7F           ;
        IN      A,($FE)         ;
        RRA                     ;
        JR      C,L05DD         ; forward to RETAD-RUN

        LD      A,$FE           ;
        IN      A,($FE)         ;
        RRA                     ;
        JR      NC,L05E2        ; forward to BREAK-PGM


;; RETAD-RUN
L05DD:  LD      HL,$1B7D        ; Main ROM address STMT-R-1

;; RETAD-SYN
L05E0:  PUSH    HL              ;
        RST     00H             ; to MAIN-ROM

; ---

;; BREAK-PGM
L05E2:  LD      (IY+$00),$14    ; insert error code in system variable ERR_NR.
        RST     28H             ; Error Main ROM
                                ; 'BREAK into program'

; ----------------------------------------
; THE 'EVALUATE STRING EXPRESSION' ROUTINE
; ----------------------------------------
;
```

```
;; EXPT-STR
L05E7:  RST     10H             ; CALBAS
        DEFW    $1C8C           ; main EXPT-EXP
        RST     18H             ; checking syntax ?
        RET     Z

        PUSH    AF
        RST     10H             ; CALBAS
        DEFW    $2BF1           ; main STK-FETCH
        POP     AF
        RET


; ----------------------------------------
; THE 'EVALUATE CHANNEL EXPRESSION' ROUTINE
; ----------------------------------------
;

;; EXPT-SPEC
L05F2:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR


;; EXP-SPEC2
L05F5   CALL    L05E7           ; routine EXPT-STR evaluates a string e.g. "m"
                                ; start in DE, length in BC.

; one of the main tenets of Sinclair BASIC is that a value can be replaced
; by an expression of the same type at any time, so this routine must allow
; something like "tomato"(3) as well as the more conventional "m" specifier.
; Only in runtime when the expression is evaluated can a single character be
; insisted upon.

        JR      Z,L060C         ; forward, if checking syntax, to TEST-NEXT.

        PUSH    AF              ; save following character.

        LD      A,C             ; in runtime check
        DEC     A               ; immediately for
        OR      B               ; a single character.

        JR      NZ,L062D        ; forward, if not, to NREPORT-3
                                ; 'Invalid device expression'

        LD      A,(DE)          ; fetch the addressed character.

        RST     10H             ; CALBAS
        DEFW    $2C8D           ; main ALPHA

        JR      NC,L062D        ; forward, if not alphabetic, to NREPORT-3

        AND     $DF             ; convert to uppercase with 'AND %11011111'

        LD      ($5CD9),A       ; place in system variable L_STR1 device letter.

        POP     AF              ; restore the following character.

;; TEST-NEXT
```

```
L060C:  CP      $0D             ; test for carriage return.
        RET     Z               ; return if so.

        CP      $3A             ; is character ':' ?
        RET     Z               ; return if so.

        CP      $A5             ; RND
        RET     NC              ; return with a token??

        CALL    L05B1           ; routine SEPARATOR tests for both ';' and ','.

        JP      NZ,L04B2        ; jump back, if not, to OREPORT-1
                                ; 'Nonsense in BASIC'

        RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR

; ------------------------------------------------
; THE 'EVALUATE NUMERIC DRIVE EXPRESSION' ROUTINE
; ------------------------------------------------
;   This routine is called once only to evaluate the numeric expression
;   following a 'CAT' command token or is used from above to check a numeric
;   expression following for example "M"; .

;; EXPT-NUM
L061E:  RST     10H             ; CALBAS
        DEFW    $1C82           ; main EXPT-1NUM
        RST     18H             ; checking syntax ?
        RET     Z               ; return if checking syntax.

        PUSH    AF              ; save NZ not syntax flag

        RST     10H             ; CALBAS
        DEFW    $1E99           ; main FIND-INT2

        LD      ($5CD6),BC      ; set system variable D_STR1 drive number

        POP     AF              ; restore NZ not syntax flag

        RET                     ; return.

; ---

;; NREPORT-3
L062D:  RST     20H             ; Shadow Error Restart
        DEFB    $02             ; 'Invalid device expression'

; -------------------------------
; THE 'EVALUATE FILENAME' ROUTINE
; -------------------------------
;

;; EXPT-NAME
L062F:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR


        CALL    L05E7           ; routine EXPT-STR
        RET     Z
```

```
        PUSH    AF
        LD      A,C
        OR      B
        JR      Z,L064C         ; forward to NREPORT-4

        LD      HL,$000A
        SBC     HL,BC
        JR      C,L064C         ; forward to NREPORT-4

        LD      ($5CDA),BC      ; sv N_STR1
        LD      ($5CDC),DE      ; sv D_STR1
        POP     AF
        RET

; ---

;; NREPORT-4
L064C:  RST     20H             ; Shadow Error Restart
        DEFB    $03             ; Invalid name

; -----------------------------------
; THE 'EVALUATE STREAM NUMBER' ROUTINE
; -----------------------------------
;

;; EXPT-STRM
L064E:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR


        RST     10H             ; CALBAS
        DEFW    $1C82           ; main EXPT-1NUM
        RST     18H             ; checking syntax ?
        RET     Z               ;

        PUSH    AF              ;
        RST     10H             ; CALBAS
        DEFW    $1E94           ; main FIND-INT1
        CP      $10             ;
        JR      NC,L0663        ; forward to NREPORT-2

        LD      ($5CD8),A       ; sv D_STR1
        POP     AF              ;
        RET                     ;

; ---

;; NREPORT-2
L0663:  RST     20H             ; Shadow Error Restart
        DEFB    $01             ; Invalid stream number


; -----------------------------------
; THE 'CHECK "M" PARAMETERS' ROUTINE
; -----------------------------------
; called once from TEST-MNAM

;; CHECK-M
```

```
L0665:  LD      A,($5CD9)       ; fetch system variable L_STR1 device letter.
        CP      $4D             ; is character "M" ?
        JP      NZ,L062D        ; jump back, if not, to NREPORT-3
                                ; Error: 'Invalid device expression'.

;; CHECK-M-2
L066D:  LD      DE,($5CD6)      ; fetch system variable D_STR1 drive number.
        LD      A,E             ; test for
        OR      D               ; zero.
        JR      Z,L0681         ; forward, if so, to NREPORT-5
                                ; 'Invalid drive number'

        INC     DE              ; also test that
        LD      A,E             ; location does not hold
        OR      D               ; the default $FFFF value.
        JR      Z,L0683         ; forward, if so, to NREPORT-9
                                ; 'Missing drive number'.

        DEC     DE              ; restore to initial value.
        LD      HL,L0008        ; and test that
        SBC     HL,DE           ; drive is in range 1 - 8.
        RET     NC              ; return if so.


;; NREPORT-5
L0681:  RST     20H             ; Shadow Error Restart
        DEFB    $04             ; Invalid drive number
; ---

;; NREPORT-9
L0683:  RST     20H             ; Shadow Error Restart
        DEFB    $08             ; Missing drive number

; ----------------------------------------------
; THE 'CHECK "M" PARAMETERS AND FILENAME' ROUTINE
; ----------------------------------------------
; This routine checks that the device expression is "M", that the drive is in
; the range 1 - 8 and that the filename is not null.

;; TEST-MNAM
L0685:  CALL    L0665           ; routine CHECK-M checks for "M" and valid
                                ; drive number.

        LD      A,($5CDB)       ; load A with D_STR1 the high byte of length
                                ; of filename.
        AND     A               ; test for zero.
        RET     Z               ; return if so.

; else system default $FF.

        RST     20H             ; Shadow Error Restart
        DEFB    $06             ; Missing name


; ---------------------------------
; THE 'CHECK STATION NUMBER' ROUTINE
; ---------------------------------
;
```

```
;; TEST-STAT
L068F:  LD      DE,($5CD6)      ; sv D_STR1 drive number
        INC     DE
        LD      A,E
        OR      D
        JR      Z,L06A1         ; forward to NREPORT-8

        DEC     DE
        LD      HL,L0040
        SBC     HL,DE
        RET     NC


;; NREPORT-6
L069F:  RST     20H             ; Shadow Error Restart
        DEFB    $05             ; Invalid station number

;; NREPORT-8
L06A1:  RST     20H             ; Shadow Error Restart
        DEFB    $07             ; Missing station number

; ----------------------------------
; THE 'EVALUATE "X";N;"NAME"' ROUTINE
; ----------------------------------
;

;; EXPT-EXPR
L06A3:  CALL    L05F2           ; routine EXPT-SPEC
        CALL    L05B1           ; routine SEPARATOR
        JP      NZ,L04B2        ; jump to OREPORT-1
                                ; 'Nonsense in BASIC'

        CALL    L062F           ; routine EXPT-NAME
        RET                     ; return...


; ----------------------------
; THE 'CHECK BAUD RATE' ROUTINE
; ----------------------------
;

;; TEST-BAUD
L06B0:  LD      HL,($5CD6)      ; sv D_STR1 drive number
        INC     HL
        LD      A,L
        OR      H
        RET     NZ

        RST     20H             ; Shadow Error Restart
        DEFB    $09             ; Missing baud rate

; ----------------------------------------
; THE 'EVALUATE STREAM OR EXPRESSION' ROUTINE
; ----------------------------------------
;

;; EXPT-EXP1
L06B9:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR
```

```
        CP      $23             ; is character '#' ?
        JP      Z,L064E         ; jump to EXPT-STRM

        CALL    L05F5           ; routine EXP-SPEC2
        CALL    L05B1           ; routine SEPARATOR
        JR      NZ,L06CC        ; forward to ENDHERE

        CALL    L062F           ; routine EXPT-NAME

;; ENDHERE
L06CC:  RST     18H             ; checking syntax ?
        RET     Z

        LD      A,($5CD9)       ; sv L_STR1 device letter.
        CP      $54             ; is character "T" ?
        RET     Z               ;

        CP      $42             ; is character "B" ?
        RET     Z               ;

        CP      $4E             ; is character "N" ?
        JP      Z,L068F         ; jump, if so, to TEST-STAT

        JP      L0685           ; jump to TEST-MNAM

; ---

        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
```

```
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF

; --------------------
; THE 'UNPAGE' ROUTINE
; --------------------
;

;; UNPAGE
L0700:  RET


; --------------------------------
; THE 'EVALUATE PARAMETERS' ROUTINE
; --------------------------------
;

;; EXPT-PRMS
L0701:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR


        CP      $2A             ; is character '*'
        JR      NZ,L073C        ; forward, if not, to OREP-1-2

        RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR


        CALL    L05F5           ; routine EXP-SPEC2
        CALL    L05B1           ; routine SEPARATOR
        JR      NZ,L0716        ; forward to NO-NAME

        CALL    L062F           ; routine EXPT-NAME

;; NO-NAME
L0716:  PUSH    AF
        LD      A,($5CD9)       ; sv L_STR1 device letter.

        CP      $4E             ; is character "N" ?
        JR      NZ,L0722        ; forward, if not, to NOT-NET

        SET     3,(IY+$7C)      ; update FLAGS_3 signal networking.

;; NOT-NET
L0722:  POP     AF
        CP      $0D             ; is character carriage return ?
        JR      Z,L0750         ; forward to END-EXPT

        CP      $3A             ; is character ':' ?
        JR      Z,L0750         ; forward to END-EXPT

        CP      $AA             ; is character the token 'SCREEN$' ?
        JR      Z,L0771         ; forward to SCREEN$

        CP      $AF             ; is character the token 'CODE' ?
        JR      Z,L0789         ; forward to CODE
```

```
        CP      $CA             ; is character the token 'LINE' ?
        JR      Z,L073E         ; forward to LINE

        CP      $E4             ; is character the token 'DATA' ?
        JP      Z,L07D2         ; jump to DATA

;; OREP-1-2
L073C:  RST     20H             ; Shadow Error Restart
        DEFB    $00             ; Nonsense in BASIC

; ---

;; LINE
L073E:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR

        RST     10H             ; CALBAS
        DEFW    $1C82           ; main EXPT-1NUM

        CALL    L05B7           ; routine ST-END

        RST     10H             ; CALBAS
        DEFW    $1E99           ; main FIND-INT2

        LD      ($5CED),BC      ; sv HD_11
        JR      L0753           ; forward to PROG

; ---

;; END-EXPT
L0750:  CALL    L05B7           ; routine ST-END

; the 'PROGRAM' SUBROUTINE is used when loading 'run'.

;; PROG
L0753:  XOR     A               ;
        LD      ($5CE6),A       ; sv HD_00
        LD      HL,($5C59)      ; sv E_LINE
        LD      DE,($5C53)      ; sv PROG
        LD      ($5CE9),DE      ; sv HD_0D
        SCF                     ;
        SBC     HL,DE           ;
        LD      ($5CE7),HL      ; sv HD_0B
        LD      HL,($5C4B)      ; sv VARS
        SBC     HL,DE           ;
        LD      ($5CEB),HL      ; sv HD_0F
        RET

; ---

;; SCREEN$
L0771:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR

        CALL    L05B7           ; routine ST-END
        LD      HL,$1B00
        LD      ($5CE7),HL      ; sv HD_0B
        LD      HL,$4000
        LD      ($5CE9),HL      ; sv HD_0D
```

```
        LD      A,$03
        LD      ($5CE6),A       ; sv HD_00
        RET

; ---

;; CODE
L0789:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR

        CP      $0D             ; is character a carriage return ?
        JR      Z,L079A         ; forward to DEFLT-0

        CP      $3A             ; is character a ':' ?
        JR      NZ,L079F        ; forward to PAR-1

        BIT     5,(IY+$7C)      ; sv FLAGS_3
        JR      NZ,L073C        ; back to OREP-1-2


;; DEFLT-0
L079A:  RST     10H             ; CALBAS
        DEFW    $1CE6           ; main USE-ZERO
        JR      L07A7           ; forward to TEST-SAVE

; ---

;; PAR-1
L079F:  RST     10H             ; CALBAS
        DEFW    $1C82           ; main EXPT-1NUM
        CALL    L05B1           ; routine SEPARATOR
        JR      Z,L07B2         ; forward to PAR-2


;; TEST-SAVE
L07A7:  BIT     5,(IY+$7C)      ; sv FLAGS_3
        JR      NZ,L073C        ; back to OREP-1-2

        RST     10H             ; CALBAS
        DEFW    $1CE6           ; main USE-ZERO
        JR      L07B8           ; forward to END-CODE

; ---

;; PAR-2
L07B2:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR


        RST     10H             ; CALBAS
        DEFW    $1C82           ; main EXPT-1NUM

;; END-CODE
L07B8:  RST     10H             ; CALBAS
        DEFW    $0018           ; main GET-CHAR

        CALL    L05B7           ; routine ST-END

        RST     10H             ; CALBAS
```

```
        DEFW    $1E99           ; main FIND-INT2
        LD      ($5CE7),BC      ; sv HD_0B

        RST     10H             ; CALBAS
        DEFW    $1E99           ; main FIND-INT2
        LD      ($5CE9),BC      ; sv HD_0D

        LD      A,$03
        LD      ($5CE6),A       ; sv HD_00
        RET                     ; return.

; ---
;
; ---


;; DATA
L07D2:  BIT     6,(IY+$7C)      ; sv FLAGS_3
        JR      Z,L07DA         ; forward to NO-M-ARR

        RST     20H             ; Shadow Error Restart
        DEFB    $14             ; MERGE error

;; NO-M-ARR
L07DA:  RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR


        RST     10H             ; CALBAS
        DEFW    $28B2           ; main LOOK-VARS

        SET     7,C
        JR      NC,L07F2        ; forward to EXISTING

        LD      HL,$0000
        BIT     4,(IY+$7C)      ; sv FLAGS_3
        JR      NZ,L080E        ; forward to LD-DATA

        LD      (IY+$00),$01    ; sv ERR_NR to '2 Variable not found'
        RST     28H             ; Error Main ROM

; ---

;; EXISTING
L07F2:  JR      Z,L07F6         ; forward to G-TYPE


;; NONS-BSC
L07F4:  RST     20H             ; Shadow Error Restart
        DEFB    $00             ; Nonsense in BASIC

; ---

;; G-TYPE
L07F6:  RST     18H             ; checking syntax ?
        JR      Z,L081C         ; forward to END-DATA

        BIT     5,(IY+$7C)      ; sv FLAGS_3
        JR      Z,L0803         ; forward to VR-DATA
```

```
        BIT     7,(HL)
        JR      Z,L07F4         ; back to NONS-BSC


;; VR-DATA
L0803:  INC     HL
        LD      A,(HL)
        LD      ($5CE7),A       ; sv HD_0B
        INC     HL
        LD      A,(HL)
        LD      ($5CE8),A       ; sv HD_0B_hi
        INC     HL

;; LD-DATA
L080E:  LD      A,C
        LD      ($5CEB),A       ; sv HD_0F
        LD      A,$01
        BIT     6,C
        JR      Z,L0819         ; forward to NUM-ARR

        INC     A

;; NUM-ARR
L0819:  LD      ($5CE6),A       ; sv HD_00

;; END-DATA
L081C:  EX      DE,HL
        RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR


        CP      $29             ; is character ')' ?
        JR      NZ,L07F4        ; back to NONS-BSC

        RST     10H             ; CALBAS
        DEFW    $0020           ; main NEXT-CHAR


        CALL    L05B7           ; routine ST-END
        LD      ($5CE9),DE      ; sv HD_0D
        RET                     ; return.


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

;; SAVE-SYN
L082F:  SET     5,(IY+$7C)      ; sv FLAGS_3
        CALL    L0701           ; routine EXPT-PRMS

        LD      A,($5CD9)       ; sv L_STR1 device letter.

        CP      $42             ; is character 'B' ?
        JR      Z,L084F         ; forward to SA-HEADER

        CP      $4E             ; is character 'N' ?
```

```
        JR      NZ,L0849        ; forward to SAVE-M

        CALL    L068F           ; routine TEST-STAT
        CALL    L0F46           ; routine OP-TEMP-N
        JR      L084F           ; forward to SA-HEADER

; ---

;; SAVE-M
L0849:  CALL    L0685           ; routine TEST-MNAM
        JP      L1AC4           ; jump to SAVE-RUN

; ---

;; SA-HEADER
L084F:  LD      B,$09
        LD      HL,$5CE6        ; sv HD_00

;; HD-LOOP
L0854:  CALL    L0884           ; routine SA-BYTE
        INC     HL
        DJNZ    L0854           ; back to HD-LOOP

        LD      HL,($5CE9)      ; sv HD_0D
        BIT     3,(IY+$7C)      ; sv FLAGS_3
        JR      Z,L086E         ; forward to SA-BLOCK

        LD      A,($5CE6)       ; sv HD_00
        CP      $03             ; compare with three - type CODE
        JR      NC,L086E        ; forward to SA-BLOCK

        LD      DE,$0114        ;
        ADD     HL,DE           ;

;; SA-BLOCK
L086E:  LD      BC,($5CE7)      ; sv HD_0B

;; SA-BLK-LP
L0872:  LD      A,C             ;
        OR      B               ;
        JR      Z,L0881         ; forward to S-BLK-END

        PUSH    IX              ;;;

        CALL    L0884           ; routine SA-BYTE

        POP     IX              ;;;

        DEC     BC              ;
        INC     HL              ;
        JR      L0872           ; back to SA-BLK-LP

; ---

;; S-BLK-END
L0881:  JP      L098C           ; jump to TST-MR-M

; -------------------------------------------------
; THE 'SAVE A BYTE TO NETWORK OR RS232 LINK' ROUTINE
```

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

;; SA-BYTE
L0884:  PUSH    HL              ;
        PUSH    BC              ;
        BIT     3,(IY+$7C)      ; sv FLAGS_3
        LD      A,(HL)          ;
        JR      NZ,L0892        ; forward to SA-NET

        CALL    L0D07           ; routine BCHAN-OUT
        JR      L0895           ; forward to SA-B-END

; ---

;; SA-NET
L0892:  CALL    L0E09           ; routine NCHAN-OUT

;; SA-B-END
L0895:  POP     BC              ;
        POP     HL              ;
        RET                     ;


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

;; LOAD-SYN
L0898:  SET     4,(IY+$7C)      ; sv FLAGS_3
        CALL    L0701           ; routine EXPT-PRMS
        JP      L08B3           ; jump to LD-VF-MR

; ---------------------------------
; THE 'VERIFY COMMAND SYNTAX' ROUTINE
; ---------------------------------
;

;; VERIF-SYN
L08A2:  SET     7,(IY+$7C)      ; sv FLAGS_3
        CALL    L0701           ; routine EXPT-PRMS
        JP      L08B3           ; jump to LD-VF-MR

; ---------------------------------
; THE 'MERGE COMMAND SYNTAX' ROUTINE
; ---------------------------------
;

;; MRG-SYN
L08AC:  SET     6,(IY+$7C)      ; sv FLAGS_3
        CALL    L0701           ; routine EXPT-PRMS

; -----------------------------------------
; THE 'LOAD-VERIFY-MERGE COMMANDS' ROUTINE
; -----------------------------------------
;

;; LD-VF-MR
```

```
L08B3:  LD      HL,$5CE6        ; set source to HD_00
        LD      DE,$5CDE        ; set destination to D_STR2
        LD      BC,$0007        ; seven bytes to copy.
        LDIR                    ; copy type, start, length, length of program.

        LD      A,($5CD9)       ; sv L_STR1 device letter.

        CP      $4E             ; "N" ?

        JR      Z,L08D1         ; forward to TS-L-NET

        CP      $42             ; "B" ?

        JR      Z,L08D7         ; forward to TS-L-RS

; proceed with Microdrive device.

        CALL    L0685           ; routine TEST-MNAM return without error if
                                ; device is "M" and drive and filename are OK.

        CALL    L1971           ; routine F-M-HEAD loads the header type
                                ; record for the above filename and populates
                                ; the locations HD_00 to HD_11.

        JR      L08F6           ; forward to TEST-TYPE which tests that file
                                ; types agree and then loads rest of records.

; ---

;; TS-L-NET
L08D1:  CALL    L068F           ; routine TEST-STAT
        CALL    L0F46           ; routine OP-TEMP-N

;; TS-L-RS
L08D7:  LD      HL,$5CE6        ; sv HD_00
        LD      B,$09           ;

;; LD-HEADER
L08DC:  PUSH    HL
        PUSH    BC
        BIT     3,(IY+$7C)      ; sv FLAGS_3
        JR      Z,L08EB         ; forward to LD-HD-RS


;; LD-HD-NET
L08E4:  CALL    L0DAF           ; routine NCHAN-IN
        JR      NC,L08E4        ; back to LD-HD-NET

        JR      L08F0           ; forward to LD-HDR-2

; ---

;; LD-HD-RS
L08EB:  CALL    L0B88           ; routine BCHAN-IN
        JR      NC,L08EB        ; back to LD-HD-RS


;; LD-HDR-2
L08F0:  POP     BC
```

```
        POP     HL
        LD      (HL),A
        INC     HL
        DJNZ    L08DC           ; back to LD-HEADER

; -->

;; TEST-TYPE
L08F6:  LD      A,($5CDE)       ; sv D_STR2
        LD      B,A
        LD      A,($5CE6)       ; sv HD_00
        CP      B
        JR      NZ,L0906        ; forward to NREPORT-N

        CP      $03             ; compare with three - type CODE
        JR      Z,L0915         ; forward to T-M-CODE

        JR      C,L0908         ; forward to TST-MERGE

;; NREPORT-N
L0906:  RST     20H             ; Shadow Error Restart
        DEFB    $16             ; Wrong file type

; ---

;; TST-MERGE
L0908:  BIT     6,(IY+$7C)      ; sv FLAGS_3
        JR      NZ,L096B        ; forward to MERGE-BLK

        BIT     7,(IY+$7C)      ; sv FLAGS_3
        JP      Z,L09A7         ; jump to LD-PR-AR

;; T-M-CODE
L0915:  BIT     6,(IY+$7C)      ; sv FLAGS_3
        JR      Z,L091D         ; forward to LD-BLOCK

        RST     20H             ; Shadow Error Restart
        DEFB    $14             ; MERGE error

; ---

;; LD-BLOCK
L091D:  LD      HL,($5CDF)      ; sv D_STR2 (+1) length of data
        LD      DE,($5CE7)      ; sv HD_0B
        LD      A,H             ;
        OR      L               ;
        JR      Z,L0936         ; forward to LD-BLK-2

        SBC     HL,DE           ;
        JR      NC,L0936        ; forward to LD-BLK-2

        BIT     4,(IY+$7C)      ; sv FLAGS_3
        JR      Z,L0934         ; forward to NREPORT-L

        RST     20H             ; Shadow Error Restart
        DEFB    $13             ; Code Error

; ---
```

```
;; NREPORT-L
L0934:  RST     20H             ; Shadow Error Restart
        DEFB    $15             ; Verification has failed

; ---

;; LD-BLK-2
L0936:  LD      HL,($5CE1)      ; sv L_STR2
        LD      A,(IX+$04)      ; channel letter
        CP      $CD             ; 'M' +$80 ?
        JR      NZ,L0945        ; forward to LD-BLK-3

        LD      HL,($5CE4)      ; sv D_STR2             ********
        JR      L0956           ; forward to LD-BLK-4

; ---

;; LD-BLK-3
L0945:  BIT     3,(IY+$7C)      ; sv FLAGS_3
        JR      Z,L0956         ; forward to LD-BLK-4

        LD      A,($5CE6)       ; sv HD_00
        CP      $03             ; compare with three - type CODE
        JR      Z,L0956         ; forward to LD-BLK-4

        LD      BC,$0114        ;
        ADD     HL,BC           ;

;; LD-BLK-4
L0956:  LD      A,H             ;
        OR      L               ;
        JR      NZ,L095D        ; forward to LD-BLK-5

        LD      HL,($5CE9)      ; sv HD_0D

;; LD-BLK-5
L095D:  LD      A,($5CE6)       ; sv HD_00
        AND     A               ;
        JR      NZ,L0966        ; forward to LD-NO-PGM

        LD      HL,($5C53)      ; sv PROG

;; LD-NO-PGM
L0966:  CALL    L0A60           ; routine LV-ANY
        JR      L098C           ; forward to TST-MR-M

; ---


;; MERGE-BLK
L096B:  LD      A,($5CEE)       ; sv HD_11_hi
        AND     $C0             ;
        JR      NZ,L0977        ; forward to NO-AUTOST

        CALL    L17B7           ; routine RCL-T-CH


        RST     20H             ; Shadow Error Restart
        DEFB    $14             ; MERGE error
```

```
; ---

;; NO-AUTOST
L0977:  LD      BC,($5CE7)      ; sv HD_0B
        PUSH    BC              ;
        INC     BC              ;

        RST     10H             ; CALBAS
        DEFW    $0030           ; main BC-SPACES

        LD      (HL),$80        ;
        EX      DE,HL           ;
        POP     DE              ;
        PUSH    HL              ;
        CALL    L0A60           ; routine LV-ANY
        POP     HL              ;

        RST     10H             ; CALBAS
        DEFW    $08CE           ; main ME-CTRLX

; ---

;; TST-MR-M
L098C:  LD      A,(IX+$04)      ; channel letter
        CP      $CD             ; 'M' + $80 ?
        JR      NZ,L0998        ; forward to TST-MR-N

        CALL    L138E           ; routine CLOSE-M2
        JR      L09A4           ; forward to MERGE-END

; ---

;; TST-MR-N
L0998:  BIT     3,(IY+$7C)      ; sv FLAGS_3
        JR      Z,L09A4         ; forward to MERGE-END

        CALL    L0FAE           ; routine SEND-NEOF
        CALL    L17B7           ; routine RCL-T-CH

;; MERGE-END
L09A4:  JP      L05C1           ; jump to END1

; ---

;; LD-PR-AR
L09A7:  LD      DE,($5CE7)      ; sv HD_0B
        LD      HL,($5CE1)      ; sv L_STR2
        PUSH    HL              ;
        LD      A,H             ;
        OR      L               ;
        JR      NZ,L09B9        ; forward to LD-PROG

        INC     DE              ;
        INC     DE              ;
        INC     DE              ;
        EX      DE,HL           ;
        JR      L09C2           ; forward to TST-SPACE
```

```
; ---


;; LD-PROG
L09B9:  LD      HL,($5CDF)      ; sv D_STR2 (+1) length of data
        EX      DE,HL           ;
        SCF                     ;
        SBC     HL,DE           ;
        JR      C,L09CB         ; forward to TST-TYPE


;; TST-SPACE
L09C2:  LD      DE,$0005        ;
        ADD     HL,DE           ;
        LD      B,H             ;
        LD      C,L             ;

        RST     10H             ; CALBAS
        DEFW    $1F05           ; main TEST-ROOM

;   Note. that before the above call, interrupts are disabled and the motor
;   of the microdrive is running.  If there should be insufficient room,
;   then the processor stops at the HALT instruction at address $1303
;   (MAIN-4), in the main ROM, while trying to output the "Out of Memory"
;   report.  This could be corrected by replacing the above 3 bytes to a
;   call to a 6-byte subroutine which carries out the same instructions
;   between an EI/DI pair.  In the production of the "Out of Memory" report
;   this ROM will be paged again by the instruction fetch at 0008. The
;   motors are stopped at START-4 and then Control will then pass to the
;   other ROM to execute the "LD A,(HL)", then back to this ROM to eliminate
;   the "OK" message before a final switch to the Main ROM for the actual
;   message text.

;; TST-TYPE
L09CB:  POP     HL
        LD      A,($5CE6)       ; sv HD_00
        AND     A
        JR      Z,L0A19         ; forward to SET-PROG

        LD      A,H
        OR      L
        JR      Z,L09F7         ; forward to CRT-NEW

        LD      A,(IX+$04)      ; channel letter
        CP      $CD             ; is character an inverted "M" ?
        JR      NZ,L09E2        ; forward to T-LD-NET

        LD      HL,($5CE4)      ; sv D_STR2
        JR      L09EC           ; forward to RCLM-OLD

; ---

;; T-LD-NET
L09E2:  BIT     3,(IY+$7C)      ; sv FLAGS_3
        JR      Z,L09EC         ; forward to RCLM-OLD

        LD      DE,$0114
        ADD     HL,DE
```

```
;; RCLM-OLD
L09EC:  DEC     HL
        LD      B,(HL)
        DEC     HL
        LD      C,(HL)
        DEC     HL
        INC     BC
        INC     BC
        INC     BC
        RST     10H             ; CALBAS
        DEFW    $19E8           ; main RECLAIM-2

;; CRT-NEW
L09F7:  LD      HL,($5C59)      ; sv E_LINE
        DEC     HL
        LD      BC,($5CE7)      ; sv HD_0B
        PUSH    BC
        INC     BC
        INC     BC
        INC     BC
        LD      A,($5CE3)       ; sv D_STR2
        PUSH    AF
        RST     10H             ; CALBAS
        DEFW    $1655           ; main MAKE-ROOM
        INC     HL
        POP     AF
        LD      (HL),A
        POP     DE
        INC     HL
        LD      (HL),E
        INC     HL
        LD      (HL),D
        INC     HL

;; END-LD-PR
L0A13:  CALL    L0A60           ; routine LV-ANY

        JP      L098C           ; jump to TST-MR-M

; ---

;; SET-PROG
L0A19:  RES     1,(IY+$7C)      ; sv FLAGS_3
        LD      DE,($5C53)      ; sv PROG
        LD      HL,($5C59)      ; sv E_LINE
        DEC     HL
        RST     10H             ; CALBAS
        DEFW    $19E5           ; main RECLAIM-1
        LD      BC,($5CE7)      ; sv HD_0B
        LD      HL,($5C53)      ; sv PROG
        RST     10H             ; CALBAS
        DEFW    $1655           ; main MAKE-ROOM
        INC     HL
        LD      BC,($5CEB)      ; sv HD_0F
        ADD     HL,BC
        LD      ($5C4B),HL      ; sv VARS
        LD      A,($5CEE)       ; sv HD_11_hi
        LD      H,A
        AND     $C0
```

```
        JR      NZ,L0A52        ; forward to NO-AUTO

        SET     1,(IY+$7C)      ; sv FLAGS_3
        LD      A,($5CED)       ; sv HD_11
        LD      L,A
        LD      ($5C42),HL      ; sv NEWPPC
        LD      (IY+$0A),$00    ; sv NSPPC

;; NO-AUTO
L0A52:  LD      HL,($5C53)      ; sv PROG
        LD      DE,($5CE7)      ; sv HD_0B
        DEC     HL
        LD      ($5C57),HL      ; sv DATADD
        INC     HL
        JR      L0A13           ; back to END-LD-PR


; ---------------------------
; THE 'LOAD OR VERIFY' ROUTINE
; ---------------------------
;   This routine is able to either LOAD or VERIFY a block of bytes, from any
;   of the three possible binary sources, A Microdrive cartridge, the Binary
;   "B" RS232 channel or the Network "N" channel.
;   The block could be a program, code bytes or an array and the first
;   receiving location is in HL and the length in DE.

;; LV-ANY
L0A60:  LD      A,D             ; test the length
        OR      E               ; for zero.
        RET     Z               ; return if so.

        LD      A,(IX+$04)      ; fetch channel letter
        CP      $CD             ; is letter "M" + $80 ?

        JR      NZ,L0A6E        ; forward, if not, to LV-BN to load from
                                ; the B channel or network.

; else is a temporary "M" channel so load or verify and then return.

        CALL    L199A           ; routine LV-MCH loads or verifies a block
                                ; of code from microdrive.

        RET                     ; return after called routine.

; ---

; Load or Verify from B channel or Network.

;; LV-BN
L0A6E:  PUSH    HL              ; save address.
        PUSH    DE              ; save byte count.
        BIT     3,(IY+$7C)      ; test FLAGS_3 - using network ?
        JR      Z,L0A7D         ; forward, if not, to LV-B

; Load or Verify from "N" channel.

;; LV-N
L0A76:  CALL    L0DAF           ; routine NCHAN-IN
        JR      NC,L0A76        ; back to LV-N
```

```
        JR      L0A82           ; forward to LV-BN-E

; ---

; Load or Verify from "B" channel.

;; LV-B
L0A7D:  CALL    L0B88           ; routine BCHAN-IN
        JR      NC,L0A7D        ; back to LV-B

; Load or Verify "B","N" end test.

;; LV-BN-E
L0A82:  POP     DE              ; restore code length.
        DEC     DE              ; and decrement.
        POP     HL              ; restore load address.

        BIT     7,(IY+$7C)      ; test FLAGS_3 - verify operation.
        JR      NZ,L0A8E        ; forward, if so missing load, to VR-BN

        LD      (HL),A          ; load the byte into memory.
        JR      L0A93           ; forward to LVBN-END

; ---

; Verify "B" or "N" bytes.

;; VR-BN
L0A8E:  CP      (HL)            ; compare the received byte with the byte in
                                ; memory.
        JR      Z,L0A93         ; forward, with match, to LVBN-END.

        RST     20H             ; Shadow Error Restart
        DEFB    $15             ; 'Verification has failed'

; ---

; Load or Verify "B","N" end.

;; LVBN-END
L0A93:  INC     HL              ; increment the address.
        LD      A,E             ; test the byte
        OR      D               ; counter for zero.
        JR      NZ,L0A6E        ; back, if not, to LV-BN

        RET                     ; return.


; --------------------------------
; THE 'LOAD "RUN" PROGRAM' ROUTINE
; --------------------------------
;

;; LOAD-RUN
L0A99:  LD      BC,$0001        ; set drive to one.
        LD      ($5CD6),BC      ; update D_STR1 drive number.

        LD      BC,$0003        ; length of "run" is three.
```

```
        LD      ($5CDA),BC      ; update N_STR1 length of filename.

        LD      BC,L0ACA        ; addr: NAME-RUN (below)
        LD      ($5CDC),BC      ; update A_STR1 - address of filename.

        SET     4,(IY+$7C)      ; update FLAGS_3 signal a LOAD operation.

        CALL    L0753           ; routine PROG sets up the first seven header
                                ; bytes for a program.

        LD      HL,$5CE6        ; set start to HD_00
        LD      DE,$5CDE        ; set destination to D_STR2
        LD      BC,$0009        ; nine bytes are copied.
                                ; Note. should be seven but is mostly harmless.

        LDIR                    ; block copy.

        SET     7,(IY+$0A)      ; update Main NSPPC - signal no jump to be made.

        CALL    L1971           ; routine F-M-HEAD loads the header type
                                ; record for the 'run' file and populates
                                ; the nine locations HD_00 to HD_11.

        JP      L08F6           ; jump back to TEST-TYPE to test that type is
                                ; 'program' and load the rest.

; ---

;; NAME-RUN
L0ACA:  DEFM    "run"           ; the filename "run"

; ****************************************
; **  T H E   R S 2 3 2   R O U T I N E S  **
; ****************************************

; ----------------------------------------
; THE 'SET "BAUD" SYSTEM VARIABLE' ROUTINE
; ----------------------------------------
;

;; SET-BAUD
L0ACD:  LD      BC,($5CD6)      ; sv D_STR1 drive number
        LD      HL,L0AF3        ; RS-CONSTS

;; NXT-ENTRY
L0AD4:  LD      E,(HL)
        INC     HL
        LD      D,(HL)
        INC     HL
        EX      DE,HL
        LD      A,H
        CP      $4B             ;
        JR      NC,L0AE8        ; forward to END-SET

        AND     A
        SBC     HL,BC
        JR      NC,L0AE8        ; forward to END-SET

        EX      DE,HL
```

```
        INC     HL
        INC     HL
        JR      L0AD4           ; loop back to NXT-ENTRY

; ---

;; END-SET
L0AE8:  EX      DE,HL
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        LD      ($5CC3),DE      ; sv BAUD
        JP      L05C1           ; jump to END1

; -----------------------------------
; THE 'RS232 TIMING CONSTANTS' ROUTINE
; -----------------------------------
;

;; RS-CONSTS
L0AF3:  DEFW    $0032           ;
        DEFW    $0A82           ;
        DEFW    $006E           ;
        DEFW    $04C5           ;
        DEFW    $012C           ;
        DEFW    $01BE           ;
        DEFW    $0258           ;
        DEFW    $00DE           ;
        DEFW    $04B0           ;
        DEFW    $006E           ;
        DEFW    $0960           ;
        DEFW    $0036           ;
        DEFW    $12C0           ;
        DEFW    $001A           ;
        DEFW    $2580           ;
        DEFW    $000C           ;
        DEFW    $4B00           ;
        DEFW    $0005           ;


; ---------------------------------------------
; THE 'OPEN RS232 CHANNEL IN CHANS AREA' ROUTINE
; ---------------------------------------------
;

;; OP-RS-CH
L0B17:  LD      HL,($5C53)      ; use system variable PROG to address the
                                ; location following the Channels area.
        DEC     HL              ; step back to the end-marker.
        LD      BC,$000B        ; eleven bytes of room required.
        PUSH    BC              ; save bytes

        RST     10H             ; CALBAS
        DEFW    $1655           ; main routine MAKE-ROOM opens up the space.
                                ; register HL points to location before room.

        POP     BC              ; bring back the eleven bytes.

        PUSH    DE              ; save DE briefly
```

```
        CALL    L1A82           ; routine REST-N-AD adjusts the dynamic memory
                                ; pointers to filenames in D_STR1 and D_STR2.
        POP     DE              ; restore DE.

        LD      HL,L0B76 - 1    ; last byte of T-Channel info.
        LD      BC,$000B        ; eleven bytes to copy.
        LDDR                    ; block copy downwards.

        INC     DE              ;
        LD      A,($5CD9)       ; sv L_STR1 device letter.
        CP      $42             ; is it "B" ?
        RET     NZ              ; return as must be "T".

; but if this is to be a binary channel then overwrite the letter and the output
; and input routines.

        PUSH    DE              ;

        LD      HL,$0004        ;
        ADD     HL,DE           ;

        LD      (HL),$42        ; 'B'
        INC     HL              ;

        LD      DE,L0D07        ; address B-CHAN-OUT
        LD      (HL),E          ;
        INC     HL              ;
        LD      (HL),D          ;
        INC     HL              ;

        LD      DE,L0B7C        ; address B-INPUT
        LD      (HL),E          ;
        INC     HL              ;
        LD      (HL),D          ;

        POP     DE              ;
        RET                     ; return.


; ----------------------------------------
; THE 'ATTACH CHANNEL TO A STREAM' ROUTINE
; ----------------------------------------
;

;; OP-RSCHAN
L0B4E:  CALL    L0B17           ; routine OP-RS-CH

;; OP-STREAM
L0B51:  LD      HL,($5C4F)      ; sv CHANS
        DEC     HL
        EX      DE,HL
        AND     A
        SBC     HL,DE
        EX      DE,HL
        LD      HL,$5C16        ; sv STRMS_00
        LD      A,($5CD8)       ; sv D_STR1
        RLCA
        LD      C,A
        LD      B,$00
```

```
        ADD     HL,BC
        LD      (HL),E
        INC     HL
        LD      (HL),D
        JP      L05C1           ; jump to END1

; ---------------------
; THE '"T" CHANNEL DATA'
; ---------------------
; the eleven-byte "T" CHANNEL descriptor.

;; TCHAN-DAT
L0B6B:  DEFW    $0008           ; main ERROR-1
        DEFW    $0008           ; main ERROR-1
        DEFB    $54             ; character "T"
        DEFW    L0C3A           ; TCHAN-OUT
        DEFW    L0B76           ; T-INPUT
        DEFW    $000B           ; channel length - 11 bytes.

; ------------------------------
; THE '"T" CHANNEL INPUT' ROUTINE
; ------------------------------
;

;; T-INPUT
L0B76:  LD      HL,L0B82        ; address of routine TCHAN-IN
        JP      L0D5A           ; jump to CALL-INP

; ------------------------------
; THE '"B" CHANNEL INPUT' ROUTINE
; ------------------------------
;

;; B-INPUT
L0B7C:  LD      HL,L0B88        ; address of routine BCHAN-IN
        JP      L0D5A           ; jump to CALL-INP

; --------------------------------------
; THE '"T" CHANNEL INPUT SERVICE' ROUTINE
; --------------------------------------
;

;; TCHAN-IN
L0B82:  CALL    L0B88           ; routine BCHAN-IN
        RES     7,A
        RET


; --------------------------------------
; THE '"B" CHANNEL INPUT SERVICE' ROUTINE
; --------------------------------------
; (Hook Code: $1D)

;; BCHAN-IN
L0B88:  LD      HL,$5CC7        ; sv SER_FL
        LD      A,(HL)          ;
        AND     A               ;
        JR      Z,L0B95         ; forward to REC-BYTE
```

```
        LD      (HL),$00        ;
        INC     HL              ;
        LD      A,(HL)          ;
        SCF                     ;

        RET                     ; Return.

; ---

;; REC-BYTE
L0B95:  CALL    L163E           ; routine TEST-BRK

        DI                      ; Disable Interrupts

        LD      A,($5CC6)       ; sv IOBORD
        OUT     ($FE),A

        LD      DE,($5CC3)      ; sv BAUD
        LD      HL,$0320        ; 800d.
        LD      B,D             ;
        LD      C,E             ;
        SRL     B               ;
        RR      C               ;
        LD      A,$FE           ;
        OUT     ($EF),A         ;

;; READ-RS
L0BAF:  IN      A,($F7)         ; bit 7 is input serial data (txdata)
        RLCA                    ;
        JR      NC,L0BC3        ; forward to TST-AGAIN

        IN      A,($F7)         ;
        RLCA                    ;
        JR      NC,L0BC3        ; forward to TST-AGAIN

        IN      A,($F7)         ;
        RLCA                    ;
        JR      NC,L0BC3        ; forward to TST-AGAIN

        IN      A,($F7)         ;
        RLCA                    ;
        JR      C,L0BCF         ; forward to START-BIT


;; TST-AGAIN
L0BC3:  DEC     HL              ;
        LD      A,H             ;
        OR      L               ;
        JR      NZ,L0BAF        ; back to READ-RS

        PUSH    AF              ;
        LD      A,$EE           ;
        OUT     ($EF),A         ;
        JR      L0BEE           ; forward to WAIT-1

; ---

;; START-BIT
L0BCF:  LD      H,B             ; Load HL with halved BAUD value.
```

```
        LD      L,C             ;

        LD      B,$80           ; Load B with the start bit.

        DEC     HL              ; Reduce the counter by the time for the four
        DEC     HL              ; tests.
        DEC     HL              ;

;; SERIAL-IN
L0BD6:  ADD     HL,DE           ; Add the BAUD value.
        NOP                     ; (4) a timing value.

;; BD-DELAY
L0BD8:  DEC     HL              ; (6) Delay for 26 * BAUD
        LD      A,H             ; (4)
        OR      L               ; (4)
        JR      NZ,L0BD8        ; (12) back to BD-DELAY

        ADD     A,$00           ; (7) wait
        IN      A,($F7)         ; Read a bit.
        RLCA                    ; Rotate bit 7 to carry.
        RR      B               ; pick up carry in B
        JR      NC,L0BD6        ; back , if no start bit, to SERIAL-IN

        LD      A,$EE           ; Send CTS line low  (comms data 0 also)
        OUT     ($EF),A         ; send to serial port

        LD      A,B             ; Transfer the received byte to A.
        CPL                     ; Complement.
        SCF                     ; Set Carry to signal success.
        PUSH    AF              ; (*) push the success flag.

;   The success and failure (time out) paths converge here with the HL register
;   holding zero.

;; WAIT-1
L0BEE:  ADD     HL,DE           ; (11) transfer DE (BAUD) to HL.

;; WAIT-2
L0BEF:  DEC     HL              ; ( 6) delay for stop bit.
        LD      A,L             ; ( 4)
        OR      H               ; ( 4)
        JR      NZ,L0BEF        ; (12/7) back to WAIT-2

;   Register HL is now zero again.

        ADD     HL,DE           ; HL = 0 + BAUD
        ADD     HL,DE           ; HL = 2 * BAUD
        ADD     HL,DE           ; HL = 3 * BAUD

;   The device at the other end of the cable (not a Spectrum) may send a
;   second byte even though CTS (Clear To Send) is low.

;; T-FURTHER
L0BF7:  DEC     HL              ; Decrement counter.
        LD      A,L             ; Test for
        OR      H               ; zero.
        JR      Z,L0C34         ; forward, if no second byte, to END-RS-IN
```

```
        IN      A,($F7)         ; Read TXdata.
        RLCA                    ; test the bit read.
        JR      NC,L0BF7        ; back, if none,  to T-FURTHER

;   As with first byte, TXdata must be high four four tests.

        IN      A,($F7)         ;
        RLCA                    ;
        JR      NC,L0BF7        ; back to T-FURTHER

        IN      A,($F7)         ;
        RLCA                    ;
        JR      NC,L0BF7        ; back to T-FURTHER

        IN      A,($F7)         ;
        RLCA                    ;
        JR      NC,L0BF7        ; back to T-FURTHER


;   A second byte is on its way and is received exactly as before.

        LD      H,D             ;
        LD      L,E             ;
        SRL     H               ;
        RR      L               ;
        LD      B,$80           ;
        DEC     HL              ;
        DEC     HL              ;
        DEC     HL              ;

;; SER-IN-2
L0C1B:  ADD     HL,DE           ;
        NOP                     ; timing.

;; BD-DELAY2
L0C1D:  DEC     HL              ;
        LD      A,H             ;
        OR      L               ;
        JR      NZ,L0C1D        ; back to BD-DELAY2

        ADD     A,$00           ;
        IN      A,($F7)         ;
        RLCA                    ;
        RR      B               ;
        JR      NC,L0C1B        ; back to SER-IN-2

;  The start bit has been pushed out and B contains the second received byte.

        LD      HL,$5CC7        ; Address the SER_FL System Variable.

        LD      (HL),$01        ; Signal there is a byte in the next location.
        INC     HL              ; Address that location.
        LD      A,B             ; Transfer the byte to A.
        CPL                     ; Complement
        LD      (HL),A          ; and insert in the second byte of serial flag.

;; END-RS-IN
L0C34:  CALL    L0D4D           ; routine BORD-REST
```

```
        POP     AF              ; ( either 0 and NC or the first received byte
                                ;   and the carry flag set )

        EI                      ; Enable Interrupts

        RET                     ; Return.

; -------------------------------
; THE '"T" CHANNEL OUTPUT' ROUTINE
; -------------------------------
;   The text channel output routine is able to list programs and, when
;   printing, takes correct action with TAB values etc.

;; TCHAN-OUT
L0C3A:  CP      $A5             ; 'RND' - first token
        JR      C,L0C44         ; forward, if less, to NOT-TOKEN

        SUB     $A5             ; reduce to range $00-5A
        RST     10H             ; CALBAS
        DEFW    $0C10           ; main PO-TOKENS
        RET                     ; return.

; ---

;; NOT-TOKEN
L0C44:  LD      HL,$5C3B        ; Address the FLAGS System Variable.
        RES     0,(HL)          ; update FLAGS - allow for leading space.
        CP      $20             ; compare to space
        JR      NZ,L0C4F        ; forward, if not, to NOT-LEAD

        SET     0,(HL)          ; update FLAGS - signal suppress leading space.

;; NOT-LEAD
L0C4F:  CP      $7F             ; compare to copyright symbol. (DEL in ASCII)
        JR      C,L0C55         ; forward, if less, to NOT-GRAPH

        LD      A,$3F           ; output CHR$(127) and graphics as '?'

;; NOT-GRAPH
L0C55:  CP      $20             ; compare against space.
        JR      C,L0C70         ; forward to CTRL-CD

        PUSH    AF              ; Preserve character.

        INC     (IY+$76)        ; Increment width         NMI_ADD_lo
        LD      A,($5CB1)       ; Load A with limit from  NMI_ADD_hi
        CP      (IY+$76)        ; Compare to width        NMI_ADD_lo

        JR      NC,L0C6C        ; forward, if less or equal, to EMIT-CH

        CALL    L0C74           ; routine TAB-SETZ emits CR/LF.

        LD      (IY+$76),$01    ; Set width to one        NMI_ADD_lo

;; EMIT-CH
L0C6C:  POP     AF              ; Restore the unprinted character.

        JP      L0D07           ; jump to BCHAN-OUT
```

```
; ---

;; CTRL-CD
L0C70:  CP      $0D             ; carriage return ?
        JR      NZ,L0C82        ; forward to NOT-CR


;; TAB-SETZ
L0C74:  LD      (IY+$76),$00    ; sv NMI_ADD_lo
        LD      A,$0D           ; output a CR carriage return.
        CALL    L0D07           ; routine BCHAN-OUT
        LD      A,$0A           ; output a LF line feed.
        JP      L0D07           ; jump to BCHAN-OUT

; ---

;; NOT-CR
L0C82:  CP      $06             ;
        JR      NZ,L0CA5        ; forward to NOT-CMM

        LD      BC,($5CB0)      ; sv NMI_ADD
        LD      E,$00           ;

;; SPC-COUNT
L0C8C:  INC     E
        INC     C
        LD      A,C
        CP      B
        JR      Z,L0C9A         ; forward to CMM-LP2


;; CMM-LOOP
L0C92:  SUB     $08
        JR      Z,L0C9A         ; forward to CMM-LP2

        JR      NC,L0C92        ; back to CMM-LOOP

        JR      L0C8C           ; back to SPC-COUNT


;; CMM-LP2
L0C9A:  PUSH    DE
        LD      A,$20
        CALL    L0C3A           ; routine TCHAN-OUT
        POP     DE
        DEC     E
        RET     Z

        JR      L0C9A           ; back to CMM-LP2

; ---

;; NOT-CMM
L0CA5:  CP      $16
        JR      Z,L0CB5         ; forward to TAB-PROC

        CP      $17
        JR      Z,L0CB5         ; forward to TAB-PROC
```

```
        CP      $10
        RET     C

        LD      DE,$0CD0
        JR      L0CB8           ; forward to STORE-COD

; ---

;; TAB-PROC
L0CB5:  LD      DE,L0CC8        ; addr: TAB-SERV

;; STORE-COD
L0CB8:  LD      ($5C0E),A       ; sv TVDATA

;; ALTER-OUT
L0CBB:  LD      HL,($5C51)      ; sv CURCHL
        PUSH    DE
        LD      DE,$0005
        ADD     HL,DE
        POP     DE
        LD      (HL),E
        INC     HL
        LD      (HL),D
        RET

; ---

;; TAB-SERV
L0CC8:  LD      DE,L0CD0        ; addr: TAB-SERV2
        LD      ($5C0F),A       ; sv TVDATA
        JR      L0CBB           ; back to ALTER-OUT

; ---

;; TAB-SERV2
L0CD0:  LD      DE,L0C3A        ; addr: TCHAN-OUT
        CALL    L0CBB           ; routine ALTER-OUT
        LD      D,A
        LD      A,($5C0E)       ; sv TVDATA
        CP      $16             ; AT control code ?
        JR      Z,L0CE6         ; forward to TST-WIDTH

        CP      $17             ; TAB control code ?
        CCF
        RET     NZ

        LD      A,($5C0F)       ; sv TVDATA
        LD      D,A

;; TST-WIDTH
L0CE6:  LD      A,($5CB1)       ; sv NMI_ADD
        CP      D
        JR      Z,L0CEE         ; forward to TAB-MOD

        JR      NC,L0CF4        ; forward to TABZERO


;; TAB-MOD
L0CEE:  LD      B,A
```

```
        LD      A,D
        SUB     B
        LD      D,A
        JR      L0CE6           ; back to TST-WIDTH

; ---

;; TABZERO
L0CF4:  LD      A,D
        OR      A
        JP      Z,L0C74         ; jump to TAB-SETZ

;; TABLOOP
L0CF9:  LD      A,($5CB0)       ; sv NMI_ADD_lo
        CP      D               ;
        RET     Z               ;

        PUSH    DE              ;
        LD      A,$20           ;
        CALL    L0C3A           ; routine TCHAN-OUT
        POP     DE              ;
        JR      L0CF9           ; back to TABLOOP


; --------------------------------
; THE '"B" CHANNEL OUTPUT' ROUTINE
; --------------------------------
; (Hook Code: $1E)
;   The bits of a byte are sent inverted. They are fixed in length and heralded
;   by a start bit and followed by two stop bits.

;; BCHAN-OUT
L0D07:  LD      B,$0B           ; Set bit count to eleven - 1 + 8 + 2.

        CPL                     ; Invert the bits of the character.
        LD      C,A             ; Copy the character to C.

        LD      A,($5CC6)       ; Load A from System Variable IOBORD
        OUT     ($FE),A         ; Change the border colour.

        LD      A,$EF           ; Set to %11101111
        OUT     ($EF),A         ; Make CTS (Clear to Send) low.

        CPL                     ; reset bit 0 (other bits of no importance)
        OUT     ($F7),A         ; Make RXdata low. %00010000

        LD      HL,($5CC3)      ; Fetch value from BAUD System Variable.
        LD      D,H             ; Copy BAUD value to DE for count.
        LD      E,L             ;

;; BD-DEL-1
L0D1C:  DEC     DE              ; ( 6) Wait 26 * BAUD cycles
        LD      A,D             ; ( 4)
        OR      E               ; ( 4)
        JR      NZ,L0D1C        ; (12) back to BD-DEL-1

;; TEST-DTR
L0D21:  CALL    L163E           ; routine TEST-BRK allows user to stop.
        IN      A,($EF)         ; Read the communication port.
```

```
        AND     $08             ; isolate DTR (Data Terminal Ready) bit.
        JR      Z,L0D21         ; back, until DTR found high, to TEST-DTR

        SCF                     ; Set carry flag as the start bit.
        DI                      ; Disable Interrupts.

;; SER-OUT-L
L0D2C:  ADC     A,$00           ; Set bit 0            76543210 <- C
        OUT     ($F7),A         ; Send RXdata the start bit.

        LD      D,H             ; Transfer the BAUD value to DE for count.
        LD      E,L             ;

;; BD-DEL-2
L0D32:  DEC     DE              ; ( 6) Wait for 26 * BAUD
        LD      A,D             ; ( 4)
        OR      E               ; ( 4)
        JR      NZ,L0D32        ; (12) back to BD-DEL-2

        DEC     DE              ; ( 6)
        XOR     A               ; ( 4) clear rxdata bit
        SRL     C               ;      shift a bit of output byte to carry.
        DJNZ    L0D2C           ; back for 11 bits to SER-OUT-L

;   Note the last two bits will have been sent reset as C is exhausted.

        EI                      ; Enable Interrupts.

        LD      A,$01           ; Set RXdata

        LD      C,$EF           ; prepare port address.
        LD      B,$EE           ; prepare value %11101110
        OUT     ($F7),A         ; Send RXdata high.
        OUT     (C),B           ; Send CTS and comms data low - switch off RS232

;; BD-DEL-3
L0D48:  DEC     HL              ; ( 6) The final 26 * BAUD delay
        LD      A,L             ; ( 4)
        OR      H               ; ( 4)
        JR      NZ,L0D48        ; (12) back to BD-DEL-3


; ---------------------------------
; THE 'BORDER COLOUR RESTORE' ROUTINE
; ---------------------------------
;

;; BORD-REST
L0D4D:  PUSH    AF              ; Preserve the accumulator value throughout.

        LD      A,($5C48)       ; Fetch System Variable BORDCR
        AND     $38             ; Mask off the paper bits.
        RRCA                    ; Rotate to the range 0 - 7
        RRCA                    ;
        RRCA                    ;
        OUT     ($FE),A         ; Change the border colour.

        POP     AF              ; Restore accumulator and flags.
```

```
        RET                     ; Return.


; ---------------------
; THE 'CALL-INP' ROUTINE
; ---------------------
;   If the extended streams e.g. #7 are being used for input then this ROM
;   will be paged in as a result of the $0008 address in the normal INPUT
;   channel position. Since 'INPUT #7' or 'INKEYS #7' could have been used
;   it is the purpose of this routine to determine which has been used.
;   Note also that 'MOVE #7 TO #2' could also invoke this routine and that MOVE
;   operations are further differentiated in the INKEY$ branch.

;; CALL-INP
L0D5A:  RES     3,(IY+$02)      ; update TV_FLAG - The mode is to be considered
                                ; unchanged.
                                ; Note. this should have been done by the Main
                                ; ROM before entering the EDITOR.

        PUSH    HL              ; (*) Preserve HL the address of the actual
                                ; service routine - either NCHAN_IN, MCHAN_IN,
                                ; BCHAN_IN ot T_CHAN_IN.

        LD      HL,($5C3D)      ; Fetch address of Error Stack Pointer ERR_SP

        LD      E,(HL)          ; Extract the address of the error handler
        INC     HL              ; If INPUT is being used this will be
        LD      D,(HL)          ; address $107F in the Main ROM.

        AND     A               ; Prepare to subtract.

        LD      HL,$107F        ; address of ED-ERROR in the Main ROM

        SBC     HL,DE           ; subtract from test value.

        JR      NZ,L0D98        ; forward if not in EDITOR to INKEY$

;   continue to handle INPUT from a stream.

        POP     HL              ; (*) POP service routine to HL e.g. NCHAN_IN

        LD      SP,($5C3D)      ; set Stack Pointer from System Variable ERR_SP
        POP     DE              ; discard the known ED-ERROR address $107F.
        POP     DE              ; POP the next value in hierarchy - MAIN-4
                                ; (usually).
        LD      ($5C3D),DE      ; and set the system variable ERR_SP

;; IN-AGAIN.
L0D78:  PUSH    HL              ; Push the address of the service routine
                                ; e.g. NCHAN_IN on the machine stack.

        LD      DE,L0D7E        ; addr: IN-AG-RET (below)
        PUSH    DE              ; push this address

        JP      (HL)            ; jump to the service routine either MCHAN_IN,
                                ; NCHAN_IN, BCHAN_IN or TCHAN_IN and then return
                                ; to the next address IN-AG-RET.

; ---
```

```
;; IN-AG-RET
L0D7E   JR      C,L0D8A         ; forward with acceptable codes to ACC-CODE

        JR      Z,L0D87         ; forward with time-out to NO-READ

;   Otherwise Iris has closed her channel or the microdrive file was exhausted.

;; OREPORT-8
L0D82:  LD      (IY+$00),$07    ; set ERR_NR to  '8 End of file'
        RST     28H             ; Error Main ROM.

; ---

;; NO-READ
L0D87:  POP     HL              ; Retrieve the address of teh service routine
                                ; and try again as always for INPUT.
        JR      L0D78           ; back to IN-AGAIN.

; ---

;; ACC-CODE
L0D8A:  CP      $0D             ; Is the acceptable code ENTER?
        JR      Z,L0D94         ; forward, if so, to END-INPUT

        RST     10H             ; CALBAS - Call the Base ROM.
        DEFW    $0F85           ; main ADD-CHRX
                                ; A special entry point within ADD-CHAR to add
                                ; the character to WORKSPACE.

        POP     HL              ; Retrieve the address of the saved service
                                ; routine.
        JR      L0D78           ; back for another character to IN-AGAIN.

; ---

;; END-INPUT
L0D94:  POP     HL              ; Discard the service routine.
        JP      L0700           ; jump to UNPAGE

; -------------------
; THE 'INKEY$' BRANCH
; -------------------

;; INKEY$
L0D98:  POP     HL              ; (*) POP service routine to HL e.g. NCHAN_IN
        LD      DE,L0D9E        ; ret addr. INK-RET (below)
        PUSH    DE              ; push this address for the return address.

        JP      (HL)            ; jump to the service routine either MCHAN_IN,
                                ; NCHAN_IN, BCHAN_IN or TCHAN_IN and then return
                                ; to the next address IN-AG-RET.

; ---

;; INK-RET
L0D9E   RET     C               ; Return with acceptable character.

        RET     Z               ; Return with no character.
```

```
        BIT     4,(IY+$7C)      ; sv FLAGS_3               MOVE?
        JR      Z,L0D82         ; back to OREPORT-8

        OR      $01             ;
        RET                     ; return with zero and carry reset.


; *********************************************
; **  T H E   N E T W O R K   R O U T I N E S  **
; *********************************************

; ------------------------------
; THE '"N" CHANNEL INPUT' ROUTINE
; ------------------------------
;

;; N-INPUT
L0DA9:  LD      HL,L0DAF        ; Address: NCHAN-IN
        JP      L0D5A           ; jump to CALL-INP

; --------------------------------------
; THE '"N" CHANNEL INPUT SERVICE' ROUTINE
; --------------------------------------
;

;; NCHAN-IN
L0DAF:  LD      IX,($5C51)      ; sv CURCHL
        LD      A,(IX+$10)      ; NCOBL
        AND     A
        JR      Z,L0DBB         ; forward to TEST-BUFF

        RST     20H             ; Shadow Error Restart
        DEFB    $0D             ; Reading a 'write' file

; ---

;; TEST-BUFF
L0DBB:  LD      A,(IX+$14)      ; NCIBL
        AND     A
        JR      Z,L0DD5         ; forward to TST-N-EOF

        LD      E,(IX+$13)      ; NCCUR
        DEC     A
        SUB     E
        JR      C,L0DD5         ; forward to TST-N-EOF

        LD      D,$00
        INC     E
        LD      (IX+$13),E      ; NCCUR
        ADD     IX,DE
        LD      A,(IX+$14)      ;
        SCF
        RET

; ---

;; TST-N-EOF
L0DD5:  LD      A,(IX+$0F)      ; NCTYPE
```

```
        AND     A
        JR      Z,L0DDC         ; forward to GET-N-BUF

        RET

; ---

;; GET-N-BUF
L0DDC:  LD      A,($5CC6)       ; sv IOBORD
        OUT     ($FE),A
        DI

;; TRY-AGAIN
L0DE2:  CALL    L0FD3           ; routine WT-SC-E
        JR      NC,L0DFC        ; forward to TIME-OUT

        CALL    L0EB5           ; routine GET-NBLK
        JR      NZ,L0DFC        ; forward to TIME-OUT

        EI

        CALL    L0D4D           ; routine BORD-REST

        LD      (IX+$13),$00    ; NCCUR
        LD      A,($5CD2)       ; sv NTTYPE
        LD      (IX+$0F),A      ; NCTYPE
        JR      L0DBB           ; back to TEST-BUFF

; ---

;; TIME-OUT
L0DFC:  LD      A,(IX+$0B)      ; NCIRIS
        AND     A               ;
        JR      Z,L0DE2         ; back to TRY-AGAIN

        EI                      ;
        CALL    L0D4D           ; routine BORD-REST
        AND     $00             ;
        RET                     ;


; --------------------------------
; THE '"N" CHANNEL OUTPUT' ROUTINE
; --------------------------------
;

;; NCHAN-OUT
L0E09:  LD      IX,($5C51)      ; sv CURCHL
        LD      B,A
        LD      A,(IX+$14)      ; NCIBL
        AND     A
        LD      A,B
        JR      Z,L0E17         ; forward to TEST-OUT

        RST     20H             ; Shadow Error Restart
        DEFB    $0C             ; Writing to a 'read' file

;; TEST-OUT
L0E17:  LD      E,(IX+$10)      ; NCOBL
```

```
        INC     E               ;
        JR      NZ,L0E25        ; forward to ST-BF-LEN

        PUSH    AF              ;
        XOR     A               ;
        CALL    L0E48           ; routine S-PACK-1
        POP     AF              ;
        LD      E,$01           ;

;; ST-BF-LEN
L0E25:  LD      (IX+$10),E      ; NCOBL
        LD      D,$00           ;
        ADD     IX,DE           ;
        LD      (IX+$14),A      ; NCIBL
        RET                     ;


; -----------------------
; THE 'OUT-BLK-N' ROUTINE
; -----------------------
;

;; OUT-BLK-N
L0E30:  CALL    L1082           ; routine OUTPAK
        LD      A,(IX+$0B)      ; NCIRIS
        AND     A               ;
        RET     Z               ;

        LD      HL,$5CCD        ; sv NTRESP
        LD      (HL),$00        ;
        LD      E,$01           ;
        CALL    L104F           ; routine INPAK
        RET     NZ              ;

        LD      A,($5CCD)       ; sv NTRESP
        DEC     A               ;
        RET                     ;


; ----------------------
; THE 'S-PACK-1' ROUTINE
; ----------------------
;

;; S-PACK-1
L0E48:  CALL    L0E4F           ; routine SEND-PACK
        RET     NZ              ;

        JP      L0EAC           ; jump to BR-DELAY

; -----------------------
; THE 'SEND-PACK' ROUTINE
; -----------------------
; (Hook Code: $30)

;; SEND-PACK
L0E4F:  LD      (IX+$0F),A      ; NCTYPE
        LD      B,(IX+$10)      ; NCOBL
        LD      A,($5CC6)       ; sv IOBORD
```

```
        OUT     ($FE),A         ;

        PUSH    IX              ;
        POP     DE              ;

        LD      HL,$0015        ;
        ADD     HL,DE           ;
        XOR     A               ;

;; CHKS1
L0E62:  ADD     A,(HL)          ;
        INC     HL              ;
        DJNZ    L0E62           ; back to CHKS1

        LD      (IX+$11),A      ; NCDCS
        LD      HL,$000B        ;
        ADD     HL,DE           ;
        PUSH    HL              ;
        LD      B,$07           ;
        XOR     A               ;

;; CHKS2
L0E71:  ADD     A,(HL)          ;
        INC     HL              ;
        DJNZ    L0E71           ; back to CHKS2

        LD      (HL),A          ;
        DI                      ;

;; SENDSCOUT
L0E77:  CALL    L101E           ; routine SEND-SC
        POP     HL              ;
        PUSH    HL              ;
        LD      E,$08           ;
        CALL    L0E30           ; routine OUT-BLK-N
        JR      NZ,L0E77        ; back to SENDSCOUT

        PUSH    IX              ;
        POP     HL              ;

        LD      DE,$0015        ;
        ADD     HL,DE           ;
        LD      E,(IX+$10)      ; NCOBL
        LD      A,E             ;
        AND     A               ;
        JR      Z,L0E9A         ; forward to INC-BLKN

        LD      B,$20           ;

;; SP-DL-1
L0E93:  DJNZ    L0E93           ; back to SP-DL-1

        CALL    L0E30           ; routine OUT-BLK-N
        JR      NZ,L0E77        ; back to SENDSCOUT


;; INC-BLKN
L0E9A:  INC     (IX+$0D)        ; NCNUMB
        JR      NZ,L0EA2        ; forward to SP-N-END
```

```
        INC     (IX+$0E)        ; NCNUMB_hi

;; SP-N-END
L0EA2:  POP     HL              ;
        CALL    L0D4D           ; routine BORD-REST
        EI                      ;
        LD      A,(IX+$0B)      ; NCIRIS
        AND     A               ;
        RET                     ;


; ----------------------
; THE 'BR-DELAY' ROUTINE
; ----------------------
;

;; BR-DELAY
L0EAC:  LD      DE,$1500        ;

;; DL-LOOP
L0EAF:  DEC     DE              ;
        LD      A,E             ;
        OR      D               ;
        JR      NZ,L0EAF        ; back to DL-LOOP

        RET                     ;


; -----------------------------------------
; THE 'HEADER AND DATA BLOCK RECEIVING' ROUTINE
; -----------------------------------------
;

;; GET-NBLK
L0EB5:  LD      HL,$5CCE        ; sv NTDEST
        LD      E,$08
        CALL    L104F           ; routine INPAK
        RET     NZ

        LD      HL,$5CCE        ; sv NTDEST
        XOR     A
        LD      B,$07

;; CHKS3
L0EC4:  ADD     A,(HL)
        INC     HL
        DJNZ    L0EC4           ; back to CHKS3

        CP      (HL)
        RET     NZ

        LD      A,($5CCE)       ; sv NTDEST
        AND     A
        JR      Z,L0EDD         ; forward to BRCAST

        CP      (IX+$0C)        ; NCSELF
        RET     NZ
```

```
        LD      A,($5CCF)       ; sv NTSRCE
        CP      (IX+$0B)        ; NCIRIS
        RET     NZ

        JR      L0EE2           ; forward to TEST-BLKN

; ---

;; BRCAST
L0EDD:  LD      A,(IX+$0B)      ; NCIRIS
        OR      A
        RET     NZ


;; TEST-BLKN
L0EE2:  LD      HL,($5CD0)      ; sv NTNUMB
        LD      E,(IX+$0D)      ; NCNUMB_lo
        LD      D,(IX+$0E)      ; NCNUMB_hi
        AND     A
        SBC     HL,DE
        JR      Z,L0F02         ; forward to GET-NBUFF

        DEC     HL
        LD      A,H
        OR      L
        RET     NZ

;   Note. The return status of the next routine should really be checked.

        CALL    L0F02           ; routine GET-NBUFF

;   Note. The DEC instruction does not affect the carry flag.

        DEC     (IX+$0D)        ; NCNUMB_lo
        JR      NC,L0EFF        ; forward, with no carry, to GETNB-END !!

        DEC     (IX+$0E)        ; NCNUMB_hi

;; GETNB-END
L0EFF:  OR      $01
        RET


;; GET-NBUFF
L0F02:  LD      A,($5CCE)       ; sv NTDEST
        OR      A
        CALL    NZ,L107B        ; routine SEND-RESP
        LD      A,($5CD3)       ; sv NTLEN
        AND     A
        JR      Z,L0F30         ; forward to STORE-LEN

        PUSH    IX
        POP     HL

        LD      DE,$0015
        ADD     HL,DE
        PUSH    HL
        LD      E,A
        CALL    L104F           ; routine INPAK
```

```
        POP     HL
        RET     NZ

        LD      A,($5CD3)       ; sv NTLEN
        LD      B,A
        LD      A,($5CD4)       ; sv NTDCS

;; CHKS4
L0F24:  SUB     (HL)
        INC     HL
        DJNZ    L0F24           ; back to CHKS4

        RET     NZ

        LD      A,($5CCE)       ; sv NTDEST
        AND     A
        CALL    NZ,L107B        ; routine SEND-RESP

;; STORE-LEN
L0F30:  LD      A,($5CD3)       ; sv NTLEN
        LD      (IX+$14),A      ; NCIBL
        INC     (IX+$0D)        ; NCNUMB_lo
        JR      NZ,L0F3E        ; forward to GETBF-END

        INC     (IX+$0E)        ; NCNUMB_hi

;; GETBF-END
L0F3E:  CP      A
        RET


; -------------------------------------
; THE 'OPEN "N" CHANNEL COMMAND' ROUTINE
; -------------------------------------
;

;; OPEN-N-ST
L0F40:  CALL    L0F52           ; routine OP-PERM-N
        JP      L0B51           ; jump to OP-STREAM

; -----------------------------------------
; THE 'OPEN TEMPORARY "N" CHANNEL' ROUTINE
; -----------------------------------------
; (Hook Code: $2D)
;

;; OP-TEMP-N
L0F46:  CALL    L0F52           ; routine OP-PERM-N
        LD      IX,($5C51)      ; sv CURCHL
        SET     7,(IX+$04)      ; channel letter
        RET


; -----------------------------------------
; THE 'OPEN PERMANENT "N" CHANNEL' ROUTINE
; -----------------------------------------
;

;; OP-PERM-N
```

```
L0F52:  LD      HL,($5C53)      ; sv PROG
        DEC     HL

        LD      BC,$0114
        PUSH    BC
        PUSH    HL
        PUSH    BC
        LD      HL,($5C65)      ; sv STKEND
        ADD     HL,BC
        JP      C,L0F9E         ; jump to OUTMEM

        LD      BC,$0050
        ADD     HL,BC
        JP      C,L0F9E         ; jump to OUTMEM

        SBC     HL,SP
        JP      NC,L0F9E        ; jump to OUTMEM

        POP     BC
        POP     HL

        RST     10H             ; CALBAS
        DEFW    $1655           ; main MAKE-ROOM

        INC     HL
        POP     BC
        CALL    L1A82           ; routine REST-N-AD
        LD      ($5C51),HL      ; sv CURCHL

        EX      DE,HL           ;
        LD      HL,L0FA3        ; NCHAN-DAT
        LD      BC,$000B        ; eleven bytes.
        LDIR                    ;

        LD      A,($5CD6)       ; sv D_STR1 drive number
        LD      (DE),A
        INC     DE
        LD      A,($5CC5)       ; sv NTSTAT
        LD      (DE),A
        INC     DE
        XOR     A
        LD      (DE),A
        LD      H,D
        LD      L,E
        INC     DE
        LD      BC,$0106
        LDIR
        LD      DE,($5C51)      ; sv CURCHL
        RET

; ---


;; OUTMEM
L0F9E:  LD      (IY+$00),$03    ; sv ERR_NR
        RST     28H             ; Error Main ROM


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

```
; THE '"N" CHANNEL DATA' ROUTINE
; ------------------------------
;

;; NCHAN_DAT
L0FA3:  DEFW    $0008           ; main ERROR-1
        DEFW    $0008           ; main ERROR-1
        DEFB    $4E             ; character "N"
        DEFW    L0E09           ; NCHAN-OUT
        DEFW    L0DA9           ; N-INPUT
        DEFW    $0114           ; length


; --------------------------------------
; THE 'SEND EOF BLOCK TO NETWORK' ROUTINE
; --------------------------------------
;

;; SEND-NEOF
L0FAE:  LD      IX,($5C51)      ; sv CURCHL
        LD      A,(IX+$10)      ; NCOBL
        AND     A               ;
        RET     Z               ;

        LD      A,$01           ;
        JP      L0E48           ; jump to S-PACK-1

; --------------------------
; THE 'NETWORK STATE' ROUTINE
; --------------------------
;

;; NET-STATE
L0FBC:  LD      A,R             ;
        OR      $C0             ;
        LD      B,A             ;
        CALL    L0FC7           ; routine CHK-REST
        JR      C,L0FBC         ; back to NET-STATE

        RET                     ;


; --------------------------
; THE 'CHECK-RESTING' ROUTINE
; --------------------------
;

;; CHK-REST
L0FC7:  CALL    L163E           ; routine TEST-BRK

;; MAKESURE
L0FCA:  PUSH    BC              ;
        POP     BC              ;
        IN      A,($F7)         ;
        RRCA                    ;
        RET     C               ;

        DJNZ    L0FCA           ; back to MAKESURE
```

```
        RET                     ;


; ------------------------
; THE 'WAIT-SCOUT' ROUTINE
; ------------------------
;

;; WT-SC-E
L0FD3:  CALL    L163E           ; routine TEST-BRK
        LD      HL,$01C2        ;

;; CLAIMED
L0FD9:  LD      B,$80           ;
        CALL    L0FC7           ; routine CHK-REST
        JR      NC,L0FED        ; forward to WT-SYNC

        DEC     HL              ;
        DEC     HL              ;
        LD      A,H             ;
        OR      L               ;
        JR      NZ,L0FD9        ; back to CLAIMED

        LD      A,(IX+$0B)      ; NCIRIS
        AND     A               ;
        JR      Z,L0FD9         ; back to CLAIMED

        RET                     ;


;; WT-SYNC
L0FED:  IN      A,($F7)         ;
        RRCA                    ;
        JR      C,L1013         ; forward to SCOUT-END

        LD      A,$7F           ;
        IN      A,($FE)         ;
        OR      $FE             ;
        IN      A,($FE)         ;
        RRA                     ;
        CALL    NC,L163E        ; routine TEST-BRK
        DEC     HL              ;
        LD      A,H             ;
        OR      L               ;
        JR      NZ,L0FED        ; back to WT-SYNC

        LD      A,(IX+$0B)      ; NCIRIS
        AND     A               ;
        JR      Z,L0FED         ; back to WT-SYNC

        RET                     ;


; -------------------------------------
; THE 'BREAK INTO I/O OPERATION' ROUTINE
; -------------------------------------
; Note. an obsolete duplicate.

;; E-READ-N
```

```
L100A:  EI                      ;
        CALL    L0D4D           ; routine BORD-REST
        LD      (IY+$00),$14    ; sv ERR_NR
        RST     28H             ; Error Main ROM

; ---------------------
; THE 'SCOUT END' BRANCH
; ---------------------
;

;; SCOUT-END
L1013:  LD      L,$09

;; LP-SCOUT
L1015:  DEC     L
        SCF
        RET     Z

        LD      B,$0E

;; DELAY-SC
L101A:  DJNZ    L101A           ; back to DELAY-SC

        JR      L1015           ; back to LP-SCOUT


; ------------------------
; THE 'SEND-SCOUT' ROUTINE
; ------------------------
;

;; SEND-SC
L101E:  CALL    L0FBC           ; routine NET-STATE
        LD      C,$F7
        LD      HL,$0009
        LD      A,($5CC5)       ; sv NTSTAT
        LD      E,A
        IN      A,($F7)
        RRCA
        JR      C,L101E         ; back to SEND-SC


;; ALL-BITS
L102F:  OUT     (C),H
        LD      D,H
        LD      H,$00
        RLC     E
        RL      H


        LD      B,$08

;; S-SC-DEL
L103A:  DJNZ    L103A           ; back to S-SC-DEL

        IN      A,($F7)
        AND     $01
        CP      D
        JR      Z,L101E         ; back to SEND-SC
```

```
        DEC     L
        JR      NZ,L102F        ; back to ALL-BITS

        LD      A,$01
        OUT     ($F7),A
        LD      B,$0E

;; END-S-DEL
L104C:  DJNZ    L104C           ; back to END-S-DEL

        RET


; -------------------
; THE 'INPAK' ROUTINE
; -------------------
;

;; INPAK
L104F:  LD      B,$FF

;; N-ACTIVE
L1051:  IN      A,($F7)
        RRA
        JR      C,L105A         ; forward to INPAK-2

        DJNZ    L1051           ; back to N-ACTIVE

        INC     B
        RET

; ---

;; INPAK-2
L105A:  LD      B,E

;; INPAK-L
L105B:  LD      E,$80
        LD      A,$CE
        OUT     ($EF),A
        NOP
        NOP
        INC     IX
        DEC     IX
        INC     IX
        DEC     IX

;; UNTIL-MK
L106B:  LD      A,$00
        IN      A,($F7)
        RRA
        RR      E
        JP      NC,L106B        ; jump to UNTIL-MK
        LD      (HL),E
        INC     HL
        DJNZ    L105B           ; back to INPAK-L

        CP      A
```

```
        RET


; -------------------------------
; THE 'SEND RESPONSE BYTE' ROUTINE
; -------------------------------
;

;; SEND-RESP
L107B:  LD      A,$01
        LD      HL,$5CCD        ; sv NTRESP
        LD      (HL),A
        LD      E,A

; --------------------
; THE 'OUTPAK' ROUTINE
; --------------------
;

;; OUTPAK
L1082:  XOR     A
        OUT     ($F7),A
        LD      B,$04

;; DEL-0-1
L1087:  DJNZ    L1087           ; back to DEL-0-1


;; OUTPAK-L
L1089:  LD      A,(HL)
        CPL
        SCF
        RLA
        LD      B,$0A

;; UNT-MARK
L108F:  OUT     ($F7),A
        RRA
        AND     A
        DEC     B
        LD      D,$00
        JP      NZ,L108F        ; jump to UNT-MARK
        INC     HL
        DEC     E
        PUSH    HL
        POP     HL
        JP      NZ,L1089        ; jump to OUTPAK-L
        LD      A,$01
        OUT     ($F7),A
        RET

; *************************************************
; **  T H E   M I C R O D R I V E   R O U T I N E S  **
; *************************************************
; The shadow ROM uses the alternate HL register solely in connection with the
; microdrive maps. This does not conflict with the Main ROM use in the
; calculator.  When used as a Hook Codes, then the calculator is implicitly in
; use by the user and so HL' should be preserved throughout.
```

```
; ----------------------------------------
; THE 'SET A TEMPORARY "M" CHANNEL' ROUTINE
; ----------------------------------------
; (Hook Code: $2B)
; This routine is used to create all microdrive channels. The routine that
; creates a permanent channel (as used by a print file) uses this routine and
; then converts the temporary channel to a permanent one.
; Temporary channels are created by LOAD, SAVE, CAT etc. and last just as long
; as required. They are deleted before returning to the Main ROM by the next
; routine DEL-M-BUF.

;; SET-T-MCH
L10A5:  EXX                     ; exx
        LD      HL,$0000        ; set HL' to zero as the default no-map-exists
                                ; condition.
        EXX                     ; exx

        LD      IX,($5C4F)      ; set IX from system variable CHANS.
        LD      DE,$0014        ; skip over the twenty bytes of the standard
        ADD     IX,DE           ; channels to point to the next or end-marker.

; now enter a search of existing "M" channels to see if any use the same drive.

;; CHK-LOOP
L10B3:  LD      A,(IX+$00)      ; fetch the next byte.
        CP      $80             ; compare to end-marker.
        JR      Z,L10F1         ; forward, if so, to CHAN-SPC.

        LD      A,(IX+$04)      ; fetch the letter of the extended channel.
        AND     $7F             ; reset bit 7.
        CP      $4D             ; is it character 'M' ?
        JR      NZ,L10E7        ; forward, if not, to NEXT-CHAN.

; an existing Microdrive Channel has been found.

        LD      A,($5CD6)       ; fetch drive number from system variable D_STR1
        CP      (IX+$19)        ; compare to CHDRIV the drive associated with
                                ; this channel.
        JR      NZ,L10E7        ; forward, if not the same, to NEXT-CHAN.

; a Microdrive Channel has been found that matches the current drive.
; It will not be necessary to create a new map for the temporary channel.

        EXX                     ; -
        LD      L,(IX+$1A)      ; load address of the associated microdrive.
        LD      H,(IX+$1B)      ; map into the HL' register.
        EXX                     ; -

        LD      BC,($5CDA)      ; load BC with length of filename from N_STR1.
        LD      HL,($5CDC)      ; load HL with address of filename.

        CALL    L1403           ; routine CHK-NAME checks name in channel
                                ; against name addressed by HL.

        JR      NZ,L10E7        ; forward, with name mismatch, to NEXT-CHAN.

        BIT     0,(IX+$18)      ; test CHFLAG.
        JR      Z,L10E7         ; forward to NEXT-CHAN.
```

```
        RST     20H             ; Shadow Error Restart.
        DEFB    $0D             ; Reading a 'write' file.

;; NEXT-CHAN
L10E7:  LD      E,(IX+$09)      ; fetch length of channel.
        LD      D,(IX+$0A)      ; to the DE register pair.
        ADD     IX,DE           ; add to point to the following location.
        JR      L10B3           ; loop back to CHK-LOOP until end-marker found.

; ---

; Now create the space for the channel.

;; CHAN-SPC
L10F1:  LD      HL,($5C53)      ; set pointer from system variable PROG.
        DEC     HL              ; now points to channels end-marker (as does IX)

        PUSH    HL              ; * save a copy of new location.
        LD      BC,$0253        ; set amount of bytes required.

;   Note. interrupts are disabled so on the original shadow ROM, which launched
;   straight into the MAKE-ROOM routine, the system hung if there was
;   insufficient free memory, at the HALT instruction in the Main error report.
;   The solution here is to perform the same checks that will be performed by
;   the Main MAKE-ROOM routine.

        PUSH    HL              ; save first location
        PUSH    BC              ; and amount while free memory is checked.

        LD      HL,($5C65)      ; fetch start of free memory from STKEND
        ADD     HL,BC           ; add bytes required producing carry if
                                ; result is higher than 65535
        JP      C,L119A         ; jump, if so, to OUTMEM2

        LD      BC,$0050        ; now allow for overhead of eighty bytes.
        ADD     HL,BC           ; and perform same test.
        JP      C,L119A         ; jump, if too high, to OUTMEM2

        SBC     HL,SP           ; finally test that result is less than the
                                ; stack pointer at the other side of free memory.
        JP      NC,L119A        ; jump, if higher, to OUTMEM2.

        POP     BC              ; restore the new room
        POP     HL              ; parameters.

; now call the MAKE-ROOM routine in the certain knowledge that nothing can
; go wrong.

        RST     10H             ; CALBAS
        DEFW    $1655           ; main MAKE-ROOM

        POP     DE              ; * restore pointer to first new location.
        PUSH    DE              ; * and save on machine stack again.

        LD      HL,L14B1        ; the default "M" CHANNEL DATA.
        LD      BC,$0019        ; twenty five bytes to copy including blank
        LDIR                    ; filename to start of new channel.

        LD      A,($5CD6)       ; fetch drive number from D_STR1.
```

```
        LD      (IX+$19),A      ; insert at CHDRIV.

        LD      BC,$0253        ; set BC to amount of room that was created.

        PUSH    IX              ; move start of channel
        POP     HL              ; to HL register.

        CALL    L1A82           ; routine REST-N-AD corrects filename pointers
                                ; leaving DE at first filename D_STR1.

        EX      DE,HL           ; transfer filename pointer to HL.

        LD      BC,($5CDA)      ; set BC to length of filename from N_STR1.

        BIT     7,B             ; test for the default $FF bytes.
        JR      NZ,L1143        ; forward, with no name, to TEST-MAP

; now enter a loop to transfer the filename to CHNAME, counting BC down to zero.
; The filename could be in ROM with 'run' or more usually in string workspace
; with its parameters on the calculator stack as with
; LOAD * "m";1;"crapgame"
; SAVE * "M";7; CHR$0 + "secret".


;; T-CH-NAME
L1135:  LD      A,B             ; check length
        OR      C               ; for zero.
        JR      Z,L1143         ; forward, if so, to TEST-MAP.

        LD      A,(HL)          ; fetch character of filename.
        LD      (IX+$0E),A      ; transfer to same position in CHNAME.

        INC     HL              ; increment
        INC     IX              ; both pointers.
        DEC     BC              ; decrement length.
        JR      L1135           ; loop back to T-CH-NAME.

; ---

;; TEST-MAP
L1143:  POP     IX              ; * restore pointer to first location of channel.

        EXX                     ; exchange set - no need now to keep balanced.
        LD      A,H             ; test map address for zero .
        OR      L               ; indicating that this drive has no map.
        JR      NZ,L1168        ; forward, if map exists, to ST-MAP-AD.

; a microdrive map is now created for this drive.

        LD      HL,($5C4F)      ; set pointer from system variable CHANS.
        PUSH    HL              ; save this pointer to the new area.
        DEC     HL              ; set HL to location before new room.
        LD      BC,$0020        ; thirty two bytes are required.

        RST     10H             ; CALBAS
        DEFW    $1655           ; main MAKE-ROOM.

; now handle dynamic pointers outside the control of the Main ROM
```

```
        POP     HL              ; restore pointer to first location.
        LD      BC,$0020        ; thirty two bytes were created.
        ADD     IX,BC           ; channel was moved up so adjust that pointer.

        CALL    L1A82           ; routine REST-N-AD corrects filename pointers.

; fill map with $FF bytes

        LD      A,$FF           ; the fill byte.
        LD      B,$20           ; thirty two locations.
        PUSH    HL              ; save map address pointer.

;; FILL-MAP
L1163:  LD      (HL),A          ; insert the byte
        INC     HL              ; next location.
        DJNZ    L1163           ; loop back to FILL-MAP

        POP     HL              ; restore address.

;; ST-MAP-AD
L1168:  LD      (IX+$1A),L      ; place map address in
        LD      (IX+$1B),H      ; channel at CHMAP.

; now make DE point to IX+$19 the header preamble and copy ROM preamble bytes.

        PUSH    IX              ; push start of channel
        POP     HL              ; pop to HL

        LD      DE,$001C        ; the offset is $1C
        ADD     HL,DE           ; add to point to start of header preamble.
        EX      DE,HL           ; transfer this destination to DE.

        LD      HL,L14CA        ; point HL to PREAMBLE data in this ROM.
        LD      BC,$000C        ; twelve bytes to copy to channel.
        LDIR                    ; in they go.

; now use the same technique to copy the same 12 bytes of ROM preamble
; to IX+$37, the data block preamble in the channel.
; A little long-winded as the destination only requires adjustment.

        PUSH    IX              ;
        POP     HL              ;
        LD      DE,$0037        ;
        LD      BC,$000C        ;

        ADD     HL,DE           ;
        EX      DE,HL           ;
        LD      HL,L14CA        ; the PREAMBLE data.
        LDIR                    ;

; now form the offset from CHANS to this channel for a return value to be
; inserted in the STRMS area.

        PUSH    IX              ; transfer
        POP     HL              ; pointer.

        LD      DE,($5C4F)      ; fetch start of CHANS area from CHANS
        OR      A               ; clear carry for subtraction.
        SBC     HL,DE           ; the true offset.
```

```
        INC     HL              ; add one as the offset is to second location.

        RET                     ; return.               >>>

; ---

;; OUTMEM2
L119A:  LD      (IY+$00),$03    ; set ERR_NR for '4 Out of memory'
        RST     28H             ; Error Main ROM

; --------------------------------
; THE 'RECLAIM "M" CHANNEL' ROUTINE
; --------------------------------
; (Hook Code: $2C)
; This routine is used to reclaim a temporary "M" channel such as that created
; by the routine above and to reclaim a permanent "M" channel by the CLOSE
; command routines.

;; DEL-M-BUF
L119F:  LD      L,(IX+$1A)      ; fetch map address.
        LD      H,(IX+$1B)      ; from CHMAP.
        PUSH    HL              ; and save.
        LD      A,(IX+$19)      ; fetch drive number from CHDRIV.
        PUSH    AF              ; and save also.

        PUSH    IX              ; transfer channel base address
        POP     HL              ; to the HL register pair.

        LD      BC,$0253        ; set BC to bytes to reclaim.

        RST     10H             ; CALBAS
        DEFW    $19E8           ; main RECLAIM-2 reclaims the channel.

        PUSH    IX              ; transfer channel
        POP     HL              ; base address again.

        LD      DE,($5C4F)      ; set DE to start of channels from CHANS
        OR      A               ; clear carry.
        SBC     HL,DE           ; subtract to form the offset.
        INC     HL              ; add 1 as points to second byte.

        LD      BC,$0253        ; set the number of bytes reclaimed.

        CALL    L1444           ; routine REST-STRM corrects all stream offsets
                                ; in the standard systems variables area
                                ; reducing them if they followed the deleted
                                ; channel.

        POP     AF              ; restore drive number
        POP     HL              ; and old map address.

; now consider deleting the map if it was used only by the reclaimed channel.

        LD      B,A             ; transfer drive to B
        LD      IX,($5C4F)      ; set IX from CHANS
        LD      DE,$0014        ; prepare to step over the twenty standard bytes
        ADD     IX,DE           ; to address next channel or end-marker.
```

```
;; TEST-MCHL
L11D0:  LD      A,(IX+$00)      ; fetch current byte.
        CP      $80             ; compare to end-marker.
        JR      Z,L11EF         ; forward, with match, to RCLM-MAP

        LD      A,(IX+$04)      ; fetch the channel letter.
        AND     $7F             ; cancel any inverted bit.
        CP      $4D             ; is character "M" ?
        JR      NZ,L11E5        ; forward, if not, to NXTCHAN

        LD      A,(IX+$19)      ; fetch this channel drive number.
        CP      B               ; compare to that of deleted channel.
        RET     Z               ; return with match - the microdrive map is
                                ; still in use.                              >>

; else continue search.

;; NXTCHAN
L11E5:  LD      E,(IX+$09)      ; fetch length of channel
        LD      D,(IX+$0A)      ; to DE register.
        ADD     IX,DE           ; add to address next channel.
        JR      L11D0           ; loop back to TEST-MCHL

; ---

; the branch was here when the end-marker was encountered without finding a
; channel that uses the map.

;; RCLM-MAP
L11EF:  LD      BC,$0020        ; thirty two bytes to reclaim.
        PUSH    HL              ; save pointer to start.
        PUSH    BC              ; save the 32 bytes.

        RST     10H             ; CALBAS
        DEFW    $19E8           ; main RECLAIM-2 reclaims the microdrive map.

        POP     BC              ; restore 32 counter.
        POP     HL              ; restore map address.

        CALL    L1476           ; routine REST-MAP adjusts all channel map
                                ; addresses.

        RET                     ; return.

; ------------------------------
; THE '"M" CHANNEL INPUT' ROUTINE
; ------------------------------
;

;; M-INPUT
L11FD:  LD      IX,($5C51)      ; sv CURCHL
        LD      HL,L1207        ; addr: MCHAN-IN
        JP      L0D5A           ; jump to CALL-INP

; --------------------------------------
; THE '"M" CHANNEL INPUT SERVICE' ROUTINE
; --------------------------------------
;
```

```
;; MCHAN-IN
L1207:  BIT     0,(IX+$18)      ; test CHFLAG
        JR      Z,L120F         ; forward, if reset, to TEST-M-BF

;; rwf-err
L120D:  RST     20H             ; Shadow Error Restart
        DEFB    $0D             ; Reading a 'write' file

; ---

;; TEST-M-BF
L120F:  LD      E,(IX+$0B)      ; load DE with the offset from CHDATA of the
        LD      D,(IX+$0C)      ; next byte to be received from CHBYTE.

        LD      L,(IX+$45)      ; load HL with the number of data bytes
        LD      H,(IX+$46)      ; in CHDATA from RECLEN.

        SCF                     ; set carry to include
        SBC     HL,DE           ; subtract the two relative positions.
        JR      C,L1233         ; forward to CHK-M-EOF

        INC     DE              ; else increment pointer.
        LD      (IX+$0B),E      ; store back
        LD      (IX+$0C),D      ; in CHBYTE.
        DEC     DE              ; decrement pointer.

        PUSH    IX              ; save start of channel.
        ADD     IX,DE           ; add the offset within CHDATA first.
        LD      A,(IX+$52)      ; now apply offset of CHDATA from start of
                                ; channel to character.
        POP     IX              ; restore channel start.
        SCF                     ; set carry flag.
        RET                     ; return.

; ---

;; CHK-M-EOF
L1233:  BIT     1,(IX+$43)      ; bit 1 of RECFLG is set if this is the last
                                ; record in this file.
        JR      Z,L123D         ; forward, if not EOF, to NEW-BUFF.

        XOR     A               ; set accumulator to zero.
        ADD     A,$0D           ; add to carriage return clearing the
                                ; carry flag and resetting the zero flag.
        RET                     ; return.

; ---

;; NEW-BUFF
L123D:  LD      DE,$0000        ; set next byte offset to zero.
        LD      (IX+$0B),E      ; and update the
        LD      (IX+$0C),D      ; pointer CHBYTE.

        INC     (IX+$0D)        ; increment record number CHREC.
        CALL    L1252           ; routine GET-RECD gets the record specified
                                ; by CHREQ matching filename CHNAME from the
                                ; cartridge in the drive CHDRIV which is
                                ; started.
```

```
        XOR     A               ; signal stop all motors.
        CALL    L1532           ; routine SEL-DRIVE.

        JR      L120F           ; back to TEST-M-BF.

; -------------------------
; THE 'GET A RECORD' ROUTINE
; -------------------------
;   This routine is used to read a specific record from a PRINT type file.
;   It is called twice -
;   1) From the "M" input routine when the current record is exhausted and the
;      next record is to be read in.
;   2) From Hook Code $27 READ-RANDOM.

;; GET-RECD
L1252:  LD      A,(IX+$19)      ; get drive number from CHDRIV.
        CALL    L1532           ; routine SEL-DRIVE starts the motor.

; ->

;; GET-R-2
L1258:  LD      BC,$04FB        ; set sector counter to 1275 = 255*5
        LD      ($5CC9),BC      ; update system variable SECTOR

;; GET-R-LP
L125F:  CALL    L1280           ; routine G-HD-RC reads in the next header and
                                ; matching record to pass the tape head.

        JR      C,L1279         ; forward, with name mismatch, to NXT-SCT

        JR      Z,L1279         ; forward, if not in use, to NXT-SCT

        LD      A,(IX+$44)      ; fetch the record number 0-n from RECNUM
        CP      (IX+$0D)        ; compare with that required in CHREC
        JR      NZ,L1279        ; forward, if no number match, to NXT-SCT

        PUSH    IX              ; transfer address of Microdrive channel
        POP     HL              ; from the IX to HL registers.

        LD      DE,$0052        ; offset to CHDATA
        ADD     HL,DE           ; add to form address of start of 512 byte data
        CALL    L142B           ; routine CHKS-BUFF
        RET     Z               ; return if checksums match.


;; NXT-SCT
L1279:  CALL    L13F7           ; routine DEC-SECT
        JR      NZ,L125F        ; loop back, if not zero, to GET-R-LP

; else produce the Error Report.

        RST     20H             ; Shadow Error Restart
        DEFB    $11             ; File not found


; --------------------------------------
; THE 'GET HEADER AND DATA BLOCK' ROUTINE
; --------------------------------------
; This routine fetches at random a header and matching record and sets the
```

```
; flags to indicate three possible outcomes.
;
; Zero flag set - record is not in use.
; Carry flag set - name does not match required
; Both flags reset - the name matches required.

;; G-HD-RC
L1280:  CALL    L13A9           ; routine GET-M-HD2 reads in and checksums
                                ; the next 14 byte header to pass tape heads.

        LD      DE,$001B        ; prepare the offset from header to RECFLG and
        ADD     HL,DE           ; add to address the start of 528 byte RECORD

        CALL    L15EB           ; routine GET-M-BUF reads in the record
                                ; descriptor and data.
                                ; register HL addresses RECFLG

        CALL    L1426           ; routine CHKS-HD-R checksums the 14 bytes
                                ; of the record descriptor.

        JR      NZ,L12B1        ; forward, with error, to G-REC-ERR

        BIT     0,(IX+$43)      ; check RECFLG - should be reset.
        JR      NZ,L12B1        ; forward, if not, to G-REC-ERR

; now test descriptor for an unused record.

        LD      A,(IX+$43)      ; load A with RECFLG - bit 1 indicates EOF
        OR      (IX+$46)        ; combine with RECLEN_hi bit 1 set if full.
        AND     $02             ; test for either full record or EOF.
        RET     Z               ; return if not with zero set and carry reset
                                ; signaling that record is unused.

; the record is a contender for a header record.

        PUSH    IX              ; transfer start of channel
        POP     HL              ; to the HL register pair.

        LD      DE,$0047        ; offset to 10 characters of filename.
        ADD     HL,DE           ; add so HL addresses the start of RECNAM.
        LD      BC,$000A        ; ten bytes to compare against required CHNAME.

        CALL    L1403           ; routine CHK-NAME

        JR      NZ,L12B1        ; forward, with name mismatch, to G-REC-ERR

; else set flags to signal success before returning.

        LD      A,$FF           ; prepare to reset zero flag
        OR      A               ; also reset carry
        RET                     ; return with zero reset and carry reset.

; ---

; else set carry to signal names do not match.

;; G-REC-ERR
L12B1:  SCF                     ; set carry flag to signal failure and
                                ; instigate another search.
```

```
        RET                     ; return with zero reset and carry set.


; -------------------------------
; THE '"M" CHANNEL OUTPUT' ROUTINE
; -------------------------------
; labeled MWRCH in source code.

;; MCHAN-OUT
L12B3:  LD      IX,$FFFA
        ADD     IX,DE
        BIT     0,(IX+$18)      ; ???? CHFLAG
        JR      NZ,L12C1        ; forward to NOREAD

        RST     20H             ; Shadow Error Restart
        DEFB    $0C             ; Writing to a 'read' file

;; NOREAD
L12C1:  LD      E,(IX+$0B)      ; CHBYTE


        LD      D,(IX+$0C)      ; CHBYTE_hi
        PUSH    IX
        ADD     IX,DE
        LD      (IX+$52),A      ; indexed
        POP     IX
        INC     DE
        LD      (IX+$0B),E      ; CHBYTE
        LD      (IX+$0C),D      ; CHBYTE_hi
        BIT     1,D             ; is CHBYTE the maximum $0200 ?
        RET     Z               ; return if not.


; ----------------------------------------
; THE 'WRITE RECORD ONTO MICRODRIVE' ROUTINE
; ----------------------------------------
; (Hook Code: $26)
;

;; WR-RECD
L12DA:  LD      A,(IX+$19)      ; fetch drive number.
        CALL    L1532           ; routine SEL-DRIVE

        LD      BC,$32C8        ; set BC to 13000 decimal
        CALL    L1652           ; routine DELAY-BC

        CALL    L12EE           ; routine WRITE-PRC

        XOR     A               ; signal stop motor
        CALL    L1532           ; routine SEL-DRIVE

        RET                     ; return.

; ----------------------------
; THE 'WRITE RECORD' SUBROUTINE
; ----------------------------
;
;
```

```
;; WRITE-PRC
L12EE:  CALL    L1349           ; routine CHK-FULL.

        JR      NZ,L12FC        ; forward, if not, to NOFULL.

        CALL    L119F           ; routine DEL-M-BUF reclaims the buffer.

        XOR     A               ; set accumulator to zero.
        CALL    L1532           ; routine SEL-DRIVE stops the motor.

        RST     20H             ; Shadow Error Restart.
        DEFB    $0F             ; 'Microdrive full'

; ---

;; NOFULL
L12FC:  PUSH    IX              ; save the pointer to channel base.
        LD      B,$0A           ; count ten characters.

;; CP-NAME
L1300:  LD      A,(IX+$0E)      ; copy a character of CHNAME
        LD      (IX+$47),A      ; to RECNAM
        INC     IX              ; increment the index pointer.
        DJNZ    L1300           ; loop back for all ten characters to CP-NAME

        POP     IX              ; restore base of "M" channel.

        LD      C,(IX+$0B)      ; fetch  CHBYTE_lo
        LD      (IX+$45),C      ; update RECLEN_lo

        LD      A,(IX+$0C)      ; fetch  CHBYTE_hi
        LD      (IX+$46),A      ; update RECLEN_hi

        LD      A,(IX+$0D)      ; fetch  CHREC
        LD      (IX+$44),A      ; update RECNUM

        RES     0,(IX+$43)      ; reset RECFLG indicating a record.

        PUSH    IX              ; transfer channel base address
        POP     HL              ; to the HL register.

        LD      DE,$0043        ; prepare offset to point to RECFLG
        ADD     HL,DE           ; and add to address the record descriptor.

        CALL    L1426           ; routine CHKS-HD-R checksums the 14 bytes.

        LD      DE,$000F        ; add extra offset to CHDATA
        ADD     HL,DE           ; the 512 bytes of data.

        CALL    L142B           ; routine CHKS-BUFF checksums the buffer.

        PUSH    IX              ; Note. this code is redundant and erroneous.
        POP     HL              ; the three registers are set up properly
        LD      DE,$0047        ; in the next routine.

        CALL    L135A           ; routine SEND-BLK writes block to microdrive
                                ; cartridge as indicated my microdrive map
                                ; which is updated.
```

```
; now prepare channel for next record. accumulator could be used to set CHBYTE.

        LD      DE,$0000        ; set DE to zero.
        LD      (IX+$0B),E      ; set CHBYTE_lo to zero
        LD      (IX+$0C),D      ; set CHBYTE_hi to zero
        INC     (IX+$0D)        ; increment the record counter CHREC

        RET                     ; return.


; ---------------------
; THE 'CHK-FULL' ROUTINE
; ---------------------
; Check the thirty two bytes of a microdrive map for a reset bit.

;; CHK-FULL
L1349:  LD      L,(IX+$1A)      ; load the address of the microdrive map
        LD      H,(IX+$1B)      ; from CHMAP to HL.
        LD      B,$20           ; set counter to thirty two.

;; NXT-B-MAP
L1351:  LD      A,(HL)          ; fetch each byte in turn.
        CP      $FF             ; compare to the all-full indicator.
        RET     NZ              ; return if there is a spare sector       >>

        INC     HL              ; next address.
        DJNZ    L1351           ; loop back to NXT-B-MAP

        XOR     A               ; set the zero flag for failure.
        RET                     ; return.


; ---------------------
; THE 'SEND-BLK' ROUTINE
; ---------------------
;   This important routine is called from the FORMAT routine and the WRITE-PRC
;   routine to write the record to the cartridge at the next available free
;   sector as indicated by the microdrive map.

;; SEND-BLK
L135A:  PUSH    IX              ; transfer the channel
        POP     HL              ; address to HL.

        LD      DE,$0037        ; offset to data preamble.
        ADD     HL,DE           ; add to address using HL
        PUSH    HL              ; save pointer to data block

; now enter a loop to find the header of an available record on microdrive.
; This SEND-BLK routine is only called when there is known to be a record
; available on the tape.

;; FAILED
L1362:  CALL    L13A9           ; routine GET-M-HD2 gets any old header.
        CALL    L13C4           ; routine CHECK-MAP checks if sector is free
                                ; on the microdrive map.
        JR      NZ,L1362        ; back, if not, to FAILED.

; A usable sector has been found on the drive. HL addresses byte within map.
```

```
        EX      (SP),HL         ; map address to stack, bring back data pointer.
        PUSH    BC              ; preserve B the map byte mask.

        IN      A,($EF)         ; test the drive.
        AND     $01             ; examine 'write protect' bit.
        JR      NZ,L1374        ; forward, if not protected, to NO-PRT.

        RST     20H             ; Shadow Error Restart.
        DEFB    $0E             ; Drive 'write' protected

;; NO-PRT
L1374:  LD      A,$E6           ;      xx100110
        OUT     ($EF),A         ; enable writing.

        LD      BC,$0168        ; a delay value of 360 decimal.
        CALL    L1652           ; routine DELAY-BC pauses briefly as the
                                ; record now approaches the tape heads.

        CALL    L15B3           ; routine OUT-M-BUF writes descriptor and
                                ; data buffer.

        LD      A,$EE           ;       xx101110
        OUT     ($EF),A         ; disable writing.

        POP     BC              ; restore the map bit.
        POP     HL              ; and the address of the byte within microdrive
                                ; map.
        LD      A,B             ; transfer masked bit to A.
        OR      (HL)            ; combine with status of other 7 sectors.
        LD      (HL),A          ; update the map to show this sector is now
                                ; used.

        RET                     ; return.


; ------------------------
; THE 'CLOSE FILE' ROUTINE
; ------------------------
; Note. The first entry point is not used.

;; close-m
L138B:  PUSH    HL              ;
        POP     IX              ;

; (Hook Code: $23)

;; CLOSE-M2
L138E:  BIT     0,(IX+$18)      ; CHFLAG
        JR      Z,L139B         ; forward to NOEMP

        SET     1,(IX+$43)      ; RECFLG
        CALL    L12DA           ; routine WR-RECD

;; NOEMP
L139B:  XOR     A               ;
        CALL    L1532           ; routine SEL-DRIVE
        CALL    L119F           ; routine DEL-M-BUF
        RET                     ; return after subroutine.
```

```
; ----------------------------------------
; THE 'MAIN ERROR RESTART EMULATION' ROUTINE
; ----------------------------------------

;; ERR-RS
L13A3:  POP     HL              ;
        LD      A,(HL)          ;
        LD      ($5C3A),A       ; sv ERR_NR
        RST     28H             ; Error Main ROM

; ----------------------------------------
; THE 'FETCH HEADER FROM MICRODRIVE' ROUTINE
; ----------------------------------------
;   This routine fetches the next valid 14-byte header to pass the tape heads
;   ensuring that it is a header as opposed to a record descriptor.

;; GET-M-HD2
L13A9:  PUSH    IX              ; transfer start of channel
        POP     HL              ; to the HL register pair.

        LD      DE,$0028        ; offset to HDFLAG
        ADD     HL,DE           ; add to form first receiving location.

        CALL    L15E2           ; routine GET-M-HD reads 15 bytes from
                                ; microdrive - last is a checksum byte.

        CALL    L1426           ; routine CHKS-HD-R checksums the bytes.

        JR      NZ,L13A9        ; back, with error, to GET-M-HD2

        BIT     0,(IX+$28)      ; test HDFLAG should be set.
        JR      Z,L13A9         ; back, if not a header, to GET-M-HD2

        RET                     ; return - with HL addressing start of header.


; --------------------------------
; THE 'CHECK MAP BIT STATE' ROUTINE
; --------------------------------
;

;; CHK-MAP-2
L13BF:  LD      E,(IX+$44)      ; pick up record from RECNUM
        JR      L13C7           ; forward to ENTRY

; ---

;; CHECK-MAP
L13C4:  LD      E,(IX+$29)      ; pick up sector from HDNUMB

; ->

;; ENTRY
L13C7:  LD      L,(IX+$1A)      ; fetch address of associated
        LD      H,(IX+$1B)      ; microdrive map from CHMAP

; the pseudo-map routine enters here with a temporary map address.

;; ENTRY-2
```

```
L13CD:  XOR     A               ; clear accumulator is one way to
        LD      D,A             ; clear D in preparation for addition.
        LD      A,E             ; transfer sector to A.
        AND     $07             ; and mask off lower 8 bits for later

        SRL     E               ; returning to E,
        SRL     E               ; divide the
        SRL     E               ; sector or record by eight.
        ADD     HL,DE           ; add to map base to give address of map bit.
        LD      B,A             ; now load sector mod 8 to B and
        INC     B               ; increment to form counter 1 - 8.

        XOR     A               ; clear A
        SCF                     ; and set carry bit ready to rotate in.

;; ROTATE
L13DD:  RLA                     ; rotate left A
        DJNZ    L13DD           ; back, while counter not zero, to ROTATE

        LD      B,A             ; return sector bit in B.
        AND     (HL)            ; AND accumulator with map sector byte.
        RET                     ; return - Z = free, NZ = occupied.


; ---------------------------------
; THE 'RESET BIT IN MAP AREA' ROUTINE
; ---------------------------------
;   This routine is called when opening a channel and by FORMAT, CAT and ERASE
;   to mark a map bit representing a sector as available.

;; RES-B-MAP
L13E3:  CALL    L13C4           ; routine CHECK-MAP fetches bit mask for map
                                ; location addressed by HL into B register.

        LD      A,B             ; fetch sector mask with one bit set.
        CPL                     ; complement - seven bits set and one bit reset.
        AND     (HL)            ; combine with other sector bits.
        LD      (HL),A          ; and update map byte resetting the bit.
        RET                     ; return.


; ----------------------------------------
; THE 'CHECK 'PSEUDO-MAP' BIT STATE' ROUTINE
; ----------------------------------------
;

;; TEST-PMAP
L13EB:  PUSH    IX              ;
        POP     HL              ;

        LD      DE,$0052        ;
        ADD     HL,DE           ;
        LD      E,(IX+$29)      ; HDNUMB
        JR      L13CD           ; back to ENTRY-2


; ---------------------------------
; THE 'DECREASE SECTOR COUNTER' ROUTINE
; ---------------------------------
```

```
;

;; DEC-SECT
L13F7:  LD      BC,($5CC9)      ; sv SECTOR
        DEC     BC              ;
        LD      ($5CC9),BC      ; sv SECTOR
        LD      A,B             ;
        OR      C               ;
        RET                     ;


; ------------------------
; THE 'CHECK-NAME' ROUTINE
; ------------------------
;

;; CHK-NAME
L1403:  PUSH    IX              ; preserve original channel base address.

        LD      B,$0A           ;

;; ALL-CHARS
L1407:  LD      A,(HL)          ;
        CP      (IX+$0E)        ; CHNAME
        JR      NZ,L1423        ; forward to CKNAM-END

        INC     HL              ;
        INC     IX              ;
        DEC     B               ;
        DEC     C               ;
        JR      NZ,L1407        ; back to ALL-CHARS

        LD      A,B             ;
        OR      A               ;
        JR      Z,L1423         ; forward to CKNAM-END


;; ALLCHR-2
L1418:  LD      A,(IX+$0E)      ; CHNAME
        CP      $20             ;
        JR      NZ,L1423        ; forward to CKNAM-END

        INC     IX              ;
        DJNZ    L1418           ; back to ALLCHR-2


;; CKNAM-END
L1423:  POP     IX              ;
        RET                     ;


; ---------------------------------------
; THE 'CALCULATE/COMPARE CHECKSUMS' ROUTINE
; ---------------------------------------
; Used for microdrive channels only.
; While the two checksums within a Network buffer are simple 8-bit sums of
; the data, the algorithm used for the microdrive channels is a little more
; sophisticated as it avoids the formation of the result $FF. While across the
; network a byte is as good as its neighbour, with microdrives the value $FF
```

```
; might arise as a result of a failed read.
; The same routine is used both to prepare the checksum prior to saving and to
; calculate and compare the checksum after reading.
; The first entry point is used for the 14 bytes of HDCHK and DESCHK
; and the second entry point is used for the 512 bytes of DCHK.

;; CHKS-HD-R
L1426:  LD      BC,$000E        ; fourteen bytes
        JR      L142E           ; forward to CHKS-ALL

; ---
; ->

;; CHKS-BUFF
L142B:  LD      BC,$0200        ; 512 bytes.

; common code.

;; CHKS-ALL
L142E:  PUSH    HL              ; save pointer to first address.
        LD      E,$00           ; initialize checksum to zero

;; NXT-BYTE
L1431:  LD      A,E             ; fetch running sum
        ADD     A,(HL)          ; add to current location.
        INC     HL              ; point to next location.


        ADC     A,$01           ; avoid the value $FF.
        JR      Z,L1439         ; forward to STCHK

        DEC     A               ; decrement.

;; STCHK
L1439:  LD      E,A             ; update the 8-bit sum.

        DEC     BC              ; reduce counter
        LD      A,B             ; and check
        OR      C               ; for zero.
        JR      NZ,L1431        ; back, if not, to NXT-BYTE

        LD      A,E             ; fetch running sum
        CP      (HL)            ; compare to checksum contents
        LD      (HL),A          ; before inserting the byte.

        POP     HL              ; restore pointer to first address.
        RET                     ; return - with zero flag set if sums agree.


; --------------------------------
; THE 'RESTORE STREAM DATA' ROUTINE
; --------------------------------
; When a channel is deleted, then the streams that point to channels beyond
; that one have to have their offsets reduced by the deleted amount.
; Also a stream that exactly matches the offset to the deleted channel, and
; there could be several, will have its entry set to zero.
; On entry, HL = offset, BC = $0253

;; REST-STRM
```

```
L1444:  PUSH    HL              ; save the offset
        LD      A,$10           ; maximum streams + 1
        LD      HL,$5C16        ; the start of the user streams area STRMS_00

;; NXT-STRM
L144A:  LD      ($5C5F),HL      ; save stream pointer temporarily in X_PTR

        LD      E,(HL)          ; fetch low byte of offset.
        INC     HL              ; bump address.
        LD      D,(HL)          ; fetch high byte of streams offset.

        POP     HL              ; retrieve the
        PUSH    HL              ; supplied offset.

        OR      A               ; clear carry.
        SBC     HL,DE           ; subtract looking for an exact match
        JR      NZ,L145C        ; forward, if not, to NOTRIGHT

        LD      DE,$0000        ; else set displacement to zero.
        JR      L1463           ; forward to STO-DISP to close the stream.

; ---

;; NOTRIGHT
L145C:  JR      NC,L1469        ; forward, if entry lower, to UPD-POINT ->

; else this stream entry is to be reduced by $0253 bytes.

        EX      DE,HL           ; streams offset to HL
        OR      A               ; clear carry
        SBC     HL,BC           ; reduce by 595 decimal bytes
        EX      DE,HL           ; transfer reduced entry to DE.

;; STO-DISP
L1463:  LD      HL,($5C5F)      ; fetch stream address from X_PTR
        LD      (HL),E          ; and insert
        INC     HL              ; the updated
        LD      (HL),D          ; offset.

; ->

;; UPD-POINT
L1469:  LD      HL,($5C5F)      ; fetch stream address from X_PTR.
        INC     HL              ; bump - each stream entry
        INC     HL              ; is two bytes.
        DEC     A               ; decrement the loop counter.
        JR      NZ,L144A        ; back, if not zero, to NXT-STRM

; else clean up and return.

        LD      ($5C5F),A       ; set X_PTR_hi to zero resting value.
        POP     HL              ; balance stack.
        RET                     ; return.


; ---------------------------------
; THE 'RESTORE MAP ADDRESSES' ROUTINE
; ---------------------------------
; When a microdrive map is reclaimed, then all the addresses of the microdrive
```

```
; maps in the "M" channels are examined and if higher than the deleted map, the
; address is reduced by thirty two bytes.
; On entry, HL = map address, BC = $0020.

;; REST-MAP
L1476:  LD      BC,$0020        ; set BC to thirty two. Already done.
        LD      IX,($5C4F)      ; load IX from system variable CHANS.
        LD      DE,$0014        ; there are 20 bytes of the standard 4 channels
        ADD     IX,DE           ; add to skip these.

; now enter a loop.

;; LCHAN
L1482:  LD      A,(IX+$00)      ; fetch first byte.
        CP      $80             ; is it the channels area end-marker ?
        RET     Z               ; return if so - all maps adjusted.       >>

        PUSH    HL              ; save map address.
        LD      A,(IX+$04)      ; fetch channel letter.
        AND     $7F             ; reset bit 7.
        CP      $4D             ; compare to "M"
        JR      NZ,L14A6        ; forward, if not, to LPEND

; a microdrive channel has been found so compare the address of the map.

        LD      E,(IX+$1A)      ; fetch address of the microdrive
        LD      D,(IX+$1B)      ; map for this channel from CHMAP.
        SBC     HL,DE           ; subtract from that of deleted map.
        JR      NC,L14A6        ; forward, if is lower, to LPEND

; address of this microdrive map is higher than the one deleted.

        EX      DE,HL           ; transfer address to HL.
        OR      A               ; clear carry.
        SBC     HL,BC           ; subtract thirty two.
        LD      (IX+$1A),L      ; and place back
        LD      (IX+$1B),H      ; in CHMAP.

;; LPEND
L14A6:  POP     HL              ; restore address of deleted map.
        LD      E,(IX+$09)      ; fetch length of channel
        LD      D,(IX+$0A)      ; to DE.
        ADD     IX,DE           ; add to address next channel.
        JR      L1482           ; loop back to LCHAN.

; ------------------------------
; THE '"M" CHANNEL DEFAULT' DATA
; ------------------------------
;

;; MCH-DAT
L14B1:  DEFW    $0008           ; main ERROR-1
        DEFW    $0008           ; main ERROR-1
        DEFB    $CD             ; inverted "M" character
        DEFW    L12B3           ; MCHAN-OUT
        DEFW    L11FD           ; M-INPUT
        DEFW    $0253           ; length
        DEFW    $0000           ;
        DEFB    $00             ;
```

```
        DEFM    "          "    ; 10 spaces
        DEFB    $FF             ; CHFLAG

; -------------------
; THE 'PREAMBLE DATA'
; -------------------
;   The PREAMBLE consists of twelve distinctive bytes that are saved to a
;   microdrive cartridge before the data.  They are not read back but allow
;   the ULA of the microdrive to recognize the start of a saved data block.

;; PREAMBLE
L14CA:  DEFB    $00, $00, $00
        DEFB    $00, $00, $00
        DEFB    $00, $00, $00
        DEFB    $00, $FF, $FF

; ------------------------------
; THE 'NOT-USED TOOLKIT' ROUTINES
; ------------------------------
; The following four routines are for debugging
; purposes during development.

; ----------------------
; THE 'DISP-HEX' ROUTINE
; ----------------------
;   display a byte as two hex characters.

;; DISP-HEX
L14D6:  PUSH    AF              ;
        RRA                     ;
        RRA                     ;
        RRA                     ;
        RRA                     ;
        CALL    L14DF           ; routine DISP-NIB
        POP     AF              ;

;; DISP-NIB
L14DF:  AND     $0F             ;

        CP      $0A             ;

        JR      C,L14E7         ; forward to CONV-1

        ADD     A,$07           ;

;; CONV-1
L14E7:  ADD     A,$30           ;

        CALL    L14F8           ; routine DISP-CH
        RET                     ;



; ----------------------
; THE 'DISP-HEX2' ROUTINE
; ----------------------
;   display a byte in hexadecimal followed by a space

;; DISP-HEX2
L14ED:  PUSH    AF              ;
```

```
        CALL    L14D6           ; routine DISP-HEX
        LD      A,$20           ;
        CALL    L14F8           ; routine DISP-CH
        POP     AF              ;
        RET                     ;


; --------------------
; THE 'DISP-CH' ROUTINE
; --------------------
;

;; DISP-CH
L14F8:  PUSH    HL              ;
        PUSH    DE              ;
        PUSH    BC              ;
        PUSH    AF              ;
        EXX                     ;
        PUSH    HL              ;
        PUSH    DE              ;
        PUSH    BC              ;
        PUSH    AF              ;
        LD      HL,($5C51)      ; sv CURCHL
        PUSH    HL              ;
        PUSH    AF              ;
        LD      A,$02           ;
        RST     10H             ; CALBAS
        DEFW    $1601           ; main CHAN-OPEN
        POP     AF              ;
        RST     10H             ; CALBAS
        DEFW    $0010           ; main PRINT-A


        POP     HL              ;
        LD      ($5C51),HL      ; sv CURCHL
        POP     AF              ;
        POP     BC              ;
        POP     DE              ;
        POP     HL              ;
        EXX                     ;
        POP     AF              ;
        POP     BC              ;
        POP     DE              ;
        POP     HL              ;
        RET                     ;


; ----------------------
; THE 'HEX-LINE' ROUTINE
; ----------------------
;   The Master routine which displays ten bytes of memory, addressed by HL,
;   in Hexadecimal followed by a CR.  The thirty output characters sit
;   comfortably within the 32 character display of the Spectrum.

;; HEX-LINE
L151D:  PUSH    HL              ;
        PUSH    BC              ;
        PUSH    AF              ;
        LD      B,$0A           ;
```

```
;; HEX-LINE2
L1522:  LD      A,(HL)          ;
        CALL    L14ED           ; routine DISP-HEX2
        INC     HL              ;
        DJNZ    L1522           ; back to HEX-LINE2

        LD      A,$0D           ;
        CALL    L14F8           ; routine DISP-CH
        POP     AF              ;
        POP     BC              ;
        POP     HL              ;
        RET                     ; return.


; -------------------------------
; THE 'SELECT DRIVE MOTOR' ROUTINE
; -------------------------------
; (Hook Code: $21)
;   This important routine is called on over twenty occasions to activate a
;   microdrive whose number is in the accumulator, or with a parameter of
;   zero, to stop all motors.  It is the sole means of controlling the real
;   or virtual bank of eight microdrives.
;   It is called with interrupts disabled and this condition should be in
;   force when the Hook Code is used.

;; SEL-DRIVE
L1532:  PUSH    HL              ; preserve the original HL value throughout.

        CP      $00             ; is the parameter zero ?
        JR      NZ,L153D        ; forward, if not, to TURN-ON.

;   The requirement is to ensure that all eight drives are switched off.

        CALL    L1565           ; routine SW-MOTOR with A holding zero.

        EI                      ; Enable Interrupts.

        POP     HL              ; restore original HL value.
        RET                     ; return.                                  >>


; --------------------
; THE 'TURN ON' BRANCH
; --------------------
;   This route turns on a drive in the range 1 - 8.  If the Hook Code has
;   been erroneously invoked with a higher value, then this will be treated
;   in much the same way as with zero.  See later.

;; TURN-ON
L153D:  DI                      ; Disable Interrupts.

        CALL    L1565           ; routine SW-MOTOR

        LD      HL,$1388        ; prepare decimal 5,000 delay value.

;; TON-DELAY
L1544:  DEC     HL              ; a simple
        LD      A,H             ; delay loop to
```

```
        OR      L               ; let things settle down.
        JR      NZ,L1544        ; back, if not zero, to TON-DELAY

        LD      HL,$1388        ; load with five thousand again.

; Now enter another 5000 loop testing for break and searching for a GAP on
; the tape at each iteration.

;; REPTEST
L154C:  LD      B,$06           ; six consecutive reads required to register
                                ; as a gap.

;; CHK-PRES
L154E:  CALL    L163E           ; routine TEST-BRK allows the user to stop.

        IN      A,($EF)         ; read the microdrive port.
        AND     $04             ; test for the gap bit
        JR      NZ,L155B        ; forward, if not, to NOPRES

        DJNZ    L154E           ; loop back six times to CHK-PRES

; A gap has been found - a formatted cartridge is in the drive.

        POP     HL              ; restore original HL value.
        RET                     ; return with motor running, interrupts
                                ; disabled.                                >>

; -------------------
; THE 'NO GAP' BRANCH
; -------------------
;   If no gap signal found on drive so far then continue counting down from
;   5000 and looping back to test for six gaps.

;; NOPRES
L155B:  DEC     HL              ; decrement the counter
        LD      A,H             ; test for
        OR      L               ; zero.
        JR      NZ,L154C        ; back, if not, to REPTEST

        CALL    L1532           ; routine SEL-DRIVE with accumulator zero
                                ; stops the drive motor.

        RST     20H             ; Shadow Error Restart
        DEFB    $10             ; 'Microdrive not present'

; ----------------------------
; THE 'SWITCH MOTOR' SUBROUTINE
; ----------------------------
;   The main developer of the microdrives and acknowledged co-inventor was
;   the late Ben Cheese, 14-Jul-1954 - 15-Jan-2001.
;
;   This ROM software always handles the switching of microdrives as if
;   there were eight drives connected.  There is no short cut to directly
;   switch on a drive and they must be handled as an array of eight devices.
;   Each microdrive includes a D-flip flop, capable of holding logic state
;   one or zero.  When the flip-flop is set at logic one then the
;   recording/playback device is switched on.
;
;   The first microdrive has the D-input terminal of the flip-flop connected
```

```
;   to the comms data line of the Interface 1 and the clock-input terminal
;   connected to the clock-output terminal of Interface 1.  Subsequent
;   microdrives have the D-input terminal connected to the Q-output terminal
;   of the next innermost drive/flip-flop and the CLOCK-input terminal
;   connected to the CLOCK-input terminal of the same adjacent
;   drive/flip-flop.
;
;   The eight microdrives thus behave as a shift register allowing a logic 1
;   condition, originating at the Interface 1 control device, to be loaded
;   into the first flip-flop by a single clock pulse and to be shifted out
;   to the appropriate flip-flop by a series of further clock pulses.
;
;   As eight pulses will be required, then the logic state of drive eight is
;   considered first and drive one is the last to be considered.
;
;   By negating the drive number and adding nine, the routine below begins
;   by effecting this reversal and, by converting zero to nine, it ensures
;   that eight logic zeros are shifted out for this case and for the case
;   of any out-of-range parameter, which can arise in the case of a User
;   experimenting with Hook Codes.
;   The limit of eight microdrives is set in the routine below and not in
;   hardware.
;
;   As Ben pointed out on his patent from which some of these details are
;   taken, "it will be appreciated that the control device may be used to
;   select associated devices other than recording/playback devices and that
;   any number of associated devices may be accommodated by use of the
;   technique described."

;; SW-MOTOR
L1565:  PUSH    DE              ; preserve the original DE value throughout.

        LD      DE,$0100        ; load DE with the constants logic one and
                                ; logic zero.

        NEG                     ; negate the supplied drive number 0 - n
        ADD     A,$09           ; add 9 so that 0 = 9, -1 = 8, -8 = 1, -10 = -1
        LD      C,A             ; place the reversed parameter in C.
        LD      B,$08           ; set clock shift counter to eight.

;; ALL-MOTRS
L1570:  DEC     C               ; decrement the drive selector.
        JR      NZ,L1586        ; forward, if not in position, to OFF-MOTOR.

; The time has come to send out a signal to start this drive.

        LD      A,D             ; select logic one.
        OUT     ($F7),A         ; output to data port.

        LD      A,$EE           ; select comms clock 1, comms data 0
        OUT     ($EF),A         ; output to D-flip flop.

        CALL    L15A2           ; routine DEL-S-1 holds for a millisecond.

        LD      A,$EC           ; select comms clock 0, comms data 0
        OUT     ($EF),A         ; output to D-flip flops.

        CALL    L15A2           ; routine DEL-S-1 holds for a millisecond.
```

```
        JR      L1597           ; forward to NXT-MOTOR

; ---

;; OFF-MOTOR
L1586:  LD      A,$EF           ; select comms clock 1, comms data 1
        OUT     ($EF),A         ; output to D-flip flop.

        LD      A,E             ; select logic 0.
        OUT     ($F7),A         ; output to data port.

        CALL    L15A2           ; routine DEL-S-1 holds for a millisecond.

        LD      A,$ED           ; select comms clock 0, comms data 1
        OUT     ($EF),A         ; output to microdrive port.

        CALL    L15A2           ; routine DEL-S-1 holds for a millisecond.

;; NXT-MOTOR
L1597:  DJNZ    L1570           ; back, for all eight drives, to ALL-MOTRS.

        LD      A,D             ; select logic one.
        OUT     ($F7),A         ; output to data port.
        LD      A,$EE           ; select comms clock 1, comms data 0.
        OUT     ($EF),A         ; output to microdrive port.

        POP     DE              ; restore original DE value.
        RET                     ; return.


; ---------------------------------
; THE '1 MILLISECOND DELAY' ROUTINE
; ---------------------------------
;   This subroutine is used to time the transitions of the Delay-flip-flops
;   used, above, to control the array of microdrives attached to Interface 1.
;   Delay flip flops become unstable if transitions are too close together
;   and this routine provides a 1 millisecond delay between clock pulses.

;; DEL-S-1
L15A2:  PUSH    BC              ; preserve counters.
        PUSH    AF              ;

        LD      BC,$0087        ; 135 decimal.
        CALL    L1652           ; routine DELAY-BC

        POP     AF              ;
        POP     BC              ; restore counters

        RET                     ; return.


; ------------------------------------------
; THE 'SEND HEADER BLOCK TO MICRODRIVE' ROUTINE
; ------------------------------------------
;   Routine is called once from the FORMAT routine.

;; OUT-M-HD
L15AD:  PUSH    HL              ;
        LD      DE,$001E        ; 30 bytes.
```

```
        JR      L15B7           ; forward to OUT-M-BLK ->

; ----------------------------------------
; THE 'SEND DATA BLOCK TO MICRODRIVE' ROUTINE
; ----------------------------------------
;

;; OUT-M-BUF
L15B3:  PUSH    HL              ;
        LD      DE,$021F        ; 543 bytes.

; -> Common code.

;; OUT-M-BLK
L15B7:  IN      A,($EF)         ;
        AND     $01             ; isolate write prot. bit.
        JR      NZ,L15BF        ; forward to NOT-PROT

        RST     20H             ; Shadow Error Restart
        DEFB    $0E             ; Drive 'write' protected

; ---

;; NOT-PROT
L15BF:  LD      A,($5CC6)       ; sv IOBORD
        OUT     ($FE),A         ;
        LD      A,$E2           ;
        OUT     ($EF),A         ;
        INC     D               ;
        LD      A,D             ;
        LD      B,E             ;
        LD      C,$E7           ;

        NOP                     ;
        NOP                     ;
        NOP                     ;

;; OUT-M-BYT
L15D0:  OTIR                    ;
        DEC     A               ;
        JR      NZ,L15D0        ; back to OUT-M-BYT

        LD      A,$E6           ;
        OUT     ($EF),A         ;
        CALL    L0D4D           ; routine BORD-REST
        POP     HL              ;
        RET                     ; return.

; ----------------------------
; THE 'SIGNAL ERROR' EXIT POINT
; ----------------------------
;   This exit point is used  twice from the next routines when the required
;   header or record block is not found within the requisite time.

;; SIGN-ERR
L15DE:  POP     BC              ; balance the stack.
        POP     HL              ; first byte of destination.
        INC     (HL)            ; increment RECFLG or HDFLAG.
        RET                     ; return.
```

```
; ------------------------------------------------
; THE 'RECEIVE BLOCK FROM MICRODRIVE HEADER' ROUTINE
; ------------------------------------------------
;

;; GET-M-HD
L15E2:  PUSH    HL              ; save destination
        LD      DE,$000F        ; set fifteen bytes to load.
        LD      HL,$0000        ; set large delay when waiting for a header.
        JR      L15F2           ; forward to GET-M-BLK

; ------------------------------------------------
; THE 'RECEIVE BLOCK FROM MICRODRIVE RECORD' ROUTINE
; ------------------------------------------------
;

;; GET-M-BUF
L15EB:  PUSH    HL              ; save destination.
        LD      DE,$0210        ; set 528d bytes to load.
        LD      HL,$01F4        ; set delay counter to 500d.

; -->

;; GET-M-BLK
L15F2:  LD      B,E             ; load B register for first INIR load.
        LD      C,D             ; load C register with count of further loads.
        INC     C               ; adjust to count down to zero.
        PUSH    BC              ; save the INIR counters.

;

;; CHK-AGAIN
L15F6:  LD      B,$08           ; set gap counter to eight.

        DEC     HL              ;
        LD      A,H             ;
        OR      L               ;

        JR      Z,L15DE         ; back to SIGN-ERR

;; CHKLOOP
L15FD:  CALL    L163E           ; routine TEST-BRK

        IN      A,($EF)         ;
        AND     $04             ; isolate gap bit.
        JR      Z,L15F6         ; back to CHK-AGAIN

        DJNZ    L15FD           ; back to CHKLOOP

;; CHK-AG-2
L1608:  LD      B,$06           ;
        DEC     HL              ;
        LD      A,H             ;
        OR      L               ;
        JR      Z,L15DE         ; back to SIGN-ERR
```

```
;; CHK-LP-2
L160F:  CALL    L163E           ; routine TEST-BRK
        IN      A,($EF)         ;
        AND     $04             ; isolate gap bit.
        JR      NZ,L1608        ; back to CHK-AG-2

        DJNZ    L160F           ; back to CHK-LP-2

        LD      A,$EE           ;
        OUT     ($EF),A         ;

        LD      B,$3C           ; set count 60 decimal.

;; DR-READY
L1620:  IN      A,($EF)         ;
        AND     $02             ; isolate sync bit.
        JR      Z,L162A         ; forward to READY-RE

        DJNZ    L1620           ; back to DR-READY

        JR      L15F6           ; back to CHK-AGAIN

; ---

;; READY-RE
L162A:  POP     BC              ; retrieve counters from the stack.
        POP     HL              ; retrieve the destination
        PUSH    HL              ; and stack again.
        CALL    L163E           ; routine TEST-BRK.
        LD      A,C             ; transfer repeat counter to A.
        LD      C,$E7           ; set port to $E7.

;   Now the INIR (INput to memory Increment and Repeat) instruction is used.

;; IN-M-BLK
L1633:  INIR                    ; read B bytes from port C to destination HL.

;   B (zero) will now count 256 bytes if first block was not the total.

        DEC     A               ; decrement repeat counter.
        JR      NZ,L1633        ; back, if not zero, to IN-M-BLK

;   All bytes, 15 or 528 have now been read.

        LD      A,$EE           ;
        OUT     ($EF),A         ;

        POP     HL              ; restore pointer to first byte.
        RET                     ; return.

; ----------------------
; THE 'TEST-BRK' ROUTINE
; ----------------------
;   Note. used more consistently in this ROM.

;; TEST-BRK
L163E:  LD      A,$7F           ; read port $7FFE - keys B, N, M, SYM, SPACE.
        IN      A,($FE)         ;
        RRA                     ; test for SPACE key.
```

```
        RET     C               ; return if not pressed.

        LD      A,$FE           ; read port $FEFE - keys SHIFT, Z, X, C, V.
        IN      A,($FE)         ;
        RRA                     ; test for SHIFT key.
        RET     C               ; return if not pressed.

        CALL    L0D4D           ; routine BORD-REST.

        LD      (IY+$00),$14    ; set ERR_NR to main 'L BREAK into program'
        RST     28H             ; invoke the Main ROM error routine.

; ---------------------
; THE 'DELAY-BC' ROUTINE
; ---------------------
;

;; DELAY-BC
L1652:  PUSH    AF              ;

;; DELAY-BC1
L1653:  DEC     BC              ;
        LD      A,B             ;
        OR      C               ;
        JR      NZ,L1653        ; back to DELAY-BC1

        POP     AF              ;
        RET                     ;

; ------------------------
; THE 'READ BLOCK' ROUTINE
; ------------------------
;   Note. new in this ROM.
;   Used by format routine.

;; READ-BLK
L165A:  PUSH    HL
        PUSH    BC

;; RDLOOP1
L165C:  LD      B,$08

;; RDLOOP2
L165E:  CALL    L163E           ; routine TEST-BRK

        IN      A,($EF)
        AND     $04             ; isolate gap bit.

        JR      Z,L165C         ; back to RDLOOP1

        DJNZ    L165E           ; back to RDLOOP2


;; RDLOOP3
L1669:  LD      B,$06

;; RDLOOP4
L166B:  CALL    L163E           ; routine TEST-BRK
```

```
        IN      A,($EF)
        AND     $04             ; isolate gap bit.

        JR      NZ,L1669        ; back to RDLOOP3

        DJNZ    L166B           ; back to RDLOOP4

        LD      A,$EE
        OUT     ($EF),A

        LD      B,$3C           ; set counter to 60d.

;; SYNC-RD
L167C:  IN      A,($EF)
        AND     $02             ; isolate sync bit.
        JR      Z,L1686         ; forward to READY-R2

        DJNZ    L167C           ; back to SYNC-RD

        JR      L165C           ; back to RDLOOP1

; ---


;; READY-R2
L1686:  POP     BC
        POP     HL
        PUSH    HL
        CALL    L163E           ; routine TEST-BRK

        LD      C,$E7           ; port
        LD      E,$FC           ; required test byte
        LD      B,$0F           ; initial counter.
        LD      D,$64           ; final counter.
        INIR

;; RD-BYT-1
L1696:  IN      A,(C)
        CP      E
        JR      NZ,L16AD        ; forward to ENDRD

        DJNZ    L1696           ; back to RD-BYT-1


;; RD-BYT-2
L169D:  IN      A,(C)
        CP      E
        JR      NZ,L16AD        ; forward to ENDRD

        DJNZ    L169D           ; back to RD-BYT-2

        LD      B,D             ; final counter is $64

;; RD-BYT-3
L16A5:  IN      A,(C)
        CP      E
        JR      NZ,L16AD        ; forward to ENDRD

        DJNZ    L16A5           ; back to RD-BYT-3
```

```
        XOR     A               ; set zero flag to signal successful read

;; ENDRD
L16AD:  POP     HL
        RET                     ; return.

; ------------------------
; THE 'WRITE BLOCK' ROUTINE
; ------------------------
;   Note. new in this ROM.
;   Called once from the FORMAT routine.

;; WR-BLK
L16AF:  PUSH    HL              ; preserve HL throughout.

        LD      A,($5CC6)       ; fetch the value of IOBORD
        OUT     ($FE),A         ; and change the border colour.

        LD      A,$E2
        OUT     ($EF),A         ; enable writing

        LD      E,$66
        LD      C,$E7
        LD      B,$1B
        LD      A,$FC           ; test byte written
        NOP                     ;
        OTIR                    ;

;; WR-BYT-1
L16C4:  OUT     (C),A
        DJNZ    L16C4           ; back to WR-BYT-1


;; WR-BYT-2
L16C8:  OUT     (C),A
        DJNZ    L16C8           ; back to WR-BYT-2

        LD      B,E             ; load counter with $66

;; WR-BYT-3
L16CD:  OUT     (C),A
        DJNZ    L16CD           ; back to WR-BYT-3

        LD      A,$E6
        OUT     ($EF),A

        CALL    L0D4D           ; routine BORD-REST

        POP     HL              ; restore initial HL value.

        RET                     ; return.

; --------------------
; THE 'UNUSED' SECTION
; --------------------
;   Contains copyright holder and initials of the main programmer.  The rest
;   is set to $FF.  This section is situated before the fixed-position CLOSE
;   rectification routine.
```

```
        DEFB    $7F             ; copyright  ©
        DEFM    " 1983 Sinclair"
        DEFM    " Research Ltd"
        DEFM    " MJB "         ; Martin Brennan

        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF
        DEFB    $FF

; --------------------------
; THE 'CLOSE STREAM' ROUTINE
; --------------------------
;   Note. An instruction fetch on main address L1708 pages in this ROM.

;; CLOSE-CH
L1708:  INC     HL              ; Same instruction as in Main ROM

        RST     30H             ; Create the new system variables

;   Note. If extra System Variables were created then the accumulator will
;   now hold the value 1 from the setting of COPIES.
;   A command like       OPEN #7,"s" : CLOSE #7      will not work.
;   If the system variables were already created then it is possible to
;   derive the stream from the accumulator by reversing the operations
;   that were performed on the value in the main ROM.


        SRL     A               ; Shift Right Logical halves the value
        SUB     $03             ; Subtraction produced the original stream.

        RES     1,(IY+$7C)      ; set FLAGS_3 to indicate a CLOSE operation.

        CALL    L1718           ; routine CLOSE (below)

        JP      L05C1           ; jump back to normal command exit at END1

; ---------------------------
; THE 'CLOSE COMMAND' ROUTINE
; ---------------------------
;   This command is called from above and also from ALL-STRMS as part of the
;   CLEAR # command execution.

;; CLOSE
L1718:  RST     10H             ; CALBAS
        DEFW    $1727           ; main STR-DATA1

;   Now perform the check that should have taken place in the original ROM.
```

```
        LD      A,C             ; Test offset for zero.
        OR      B               ;
        RET     Z               ; Return if the stream is already closed.

        PUSH    BC              ;
        PUSH    HL              ;

        LD      HL,($5C4F)      ; sv CHANS
        DEC     HL              ;
        ADD     HL,BC           ;
        EX      (SP),HL         ;
        RST     10H             ; CALBAS
        DEFW    $16EB           ; main CLOSEX
        LD      HL,($5C4F)      ; sv CHANS
        LD      DE,$0014        ;
        ADD     HL,DE           ;

        POP     DE              ;
        SCF                     ;
        SBC     HL,DE           ;
        POP     BC              ;


        RET     NC              ;

        PUSH    BC              ;
        PUSH    DE              ;
        EX      DE,HL           ;
        LD      ($5C51),HL      ; sv CURCHL
        INC     HL              ;


        INC     HL              ;
        INC     HL              ;
        INC     HL              ;

        LD      A,(HL)          ; fetch the letter.

;   Now mark the channel as temporary so that if anything goes wrong, such
;   as the user pressing BREAK, then the channel can be reclaimed by CLEAR #.

L1741:  SET     7,(HL)          ; As suggested by Andrew Pennell 1983.

        LD      DE,$0005        ;
        ADD     HL,DE           ;
        LD      E,(HL)          ;
        INC     HL              ;
        LD      D,(HL)          ;
        PUSH    DE              ;
        CP      $54             ; compare to "T"
        JR      NZ,L175C        ; forward to CL-N-CH

        BIT     1,(IY+$7C)      ; sv FLAGS_3
        JR      NZ,L177D        ; forward to RCLM-CH

        LD      A,$0D
        CALL    L0D07           ; routine BCHAN-OUT
        JR      L177D           ; forward to RCLM-CH
```

```
; ---

;; CL-N-CH
L175C:  CP      $4E             ; character "N" ?
        JR      NZ,L176B        ; forward to CL-M-CH

        BIT     1,(IY+$7C)      ; sv FLAGS_3
        JR      NZ,L177D        ; forward to RCLM-CH

        CALL    L0FAE           ; routine SEND-NEOF
        JR      L177D           ; forward to RCLM-CH

; ---


;; CL-M-CH
L176B:  CP      $4D             ; character "M"
        JR      NZ,L177D        ; forward to RCLM-CH

        POP     DE              ;
        POP     IX              ;
        POP     DE              ;
        BIT     1,(IY+$7C)      ; sv FLAGS_3
        JP      Z,L138E         ; jump to CLOSE-M2

        JP      L119F           ; jump to DEL-M-BUF

; ---

;; RCLM-CH
L177D:  POP     BC              ;
        POP     HL              ;
        PUSH    BC              ;
        RST     10H             ; CALBAS
        DEFW    $19E8           ; main RECLAIM-2
        XOR     A               ;
        LD      HL,$5C16        ; sv STRMS_00

;; UPD-STRM
L1787:  LD      E,(HL)          ;
        INC     HL              ;
        LD      D,(HL)          ;
        DEC     HL              ;
        LD      ($5C5F),HL      ; sv X_PTR
        POP     BC              ;
        POP     HL              ;
        PUSH    HL              ;
        PUSH    BC              ;
        AND     A               ;
        SBC     HL,DE           ;
        JR      NC,L17A2        ; forward to UPD-NXT-S

        EX      DE,HL           ;
        AND     A               ;
        SBC     HL,BC           ;
        EX      DE,HL           ;
        LD      HL,($5C5F)      ; sv X_PTR
        LD      (HL),E          ;
        INC     HL              ;
```

```
        LD      (HL),D          ;

;; UPD-NXT-S
L17A2:  LD      HL,($5C5F)      ; sv X_PTR
        INC     HL              ;
        INC     HL              ;
        INC     A               ;
        CP      $10             ;
        JR      C,L1787         ; back to UPD-STRM

        LD      (IY+$26),$00    ; sv X_PTR_hi
        POP     HL              ;
        POP     HL              ;
        RES     1,(IY+$7C)      ; sv FLAGS_3
        RET                     ; return.

; ----------------------------------------
; THE 'RECLAIM TEMPORARY CHANNELS' ROUTINE
; ----------------------------------------
;

;; RCL-T-CH
L17B7:  LD      IX,($5C4F)      ; sv CHANS
        LD      DE,$0014
        ADD     IX,DE

;; EX-CHANS
L17C0:  LD      A,(IX+$00)      ; first character of channel
        CP      $80             ; is it the end-marker ?
        JR      NZ,L17D0        ; forward to CHK-TEMPM

        LD      A,$EE
        OUT     ($EF),A

        XOR     A
        JP      L1532           ; jump to SEL-DRIVE

; ---

        RET                     ; unused - the above JP was probably once a CALL.

; ---

;; CHK-TEMPM
L17D0:  LD      A,(IX+$04)      ; channel letter
        CP      $CD             ; is it an inverted "M" ?
        JR      NZ,L17DC        ; forward to CHK-TEMPN

        CALL    L119F           ; routine DEL-M-BUF
        JR      L17B7           ; back to RCL-T-CH


;; CHK-TEMPN
L17DC:  CP      $CE             ; is channel letter an inverted "N" ?
        JR      NZ,L17EB        ; forward to PT-N-CHAN

        LD      BC,$0114
        PUSH    IX
        POP     HL
```

```
        RST     10H             ; CALBAS
        DEFW    $19E8           ; main RECLAIM-2
        JR      L17B7           ; back to RCL-T-CH


;; PT-N-CHAN
L17EB:  LD      E,(IX+$09)      ; length of
        LD      D,(IX+$0A)      ; channel
        ADD     IX,DE
        JR      L17C0           ; back to EX-CHANS


; -------------------------
; THE 'MOVE COMMAND' ROUTINE
; -------------------------
;

;; MOVE
L17F5:  SET     4,(IY+$7C)      ; update FLAGS_3 to indicate a MOVE is in
                                ; progress - see INKEY$

        CALL    L1859           ; routine OP-STRM
        LD      HL,($5C4F)      ; sv CHANS
        PUSH    HL
        CALL    L059F           ; routine EX-D-STR
        CALL    L1859           ; routine OP-STRM
        CALL    L059F           ; routine EX-D-STR
        POP     DE
        LD      HL,($5C4F)      ; sv CHANS
        OR      A
        SBC     HL,DE
        LD      DE,($5CDA)      ; sv N_STR1
        ADD     HL,DE
        LD      ($5CDA),HL      ; sv N_STR1

;; M-AGAIN
L1818:  LD      HL,($5CDA)      ; sv N_STR1
        LD      ($5C51),HL      ; sv CURCHL

;; I-AGAIN
L181E:  RST     10H             ; CALBAS
        DEFW    $15E6           ; main INPUT-AD
        JR      C,L1827         ; forward to MOVE-OUT

        JR      Z,L181E         ; back to I-AGAIN

        JR      L1832           ; forward to MOVE-EOF


;; MOVE-OUT
L1827:  LD      HL,($5CE2)      ; sv D_STR2
        LD      ($5C51),HL      ; sv CURCHL
        RST     10H             ; CALBAS
        DEFW    $0010           ; main PRINT-A

        JR      L1818           ; back to M-AGAIN

;; MOVE-EOF
L1832:  RES     4,(IY+$7C)      ; sv FLAGS_3
```

```
        LD      HL,($5C4F)      ; sv CHANS
        PUSH    HL
        CALL    L059F           ; routine EX-D-STR
        CALL    L18A8           ; routine CL-CHAN
        CALL    L059F           ; routine EX-D-STR
        POP     DE
        LD      HL,($5C4F)      ; sv CHANS
        OR      A
        SBC     HL,DE
        LD      DE,($5CDA)      ; sv N_STR1
        ADD     HL,DE
        LD      ($5CDA),HL      ; sv N_STR1
        CALL    L18A8           ; routine CL-CHAN
        CALL    L17B7           ; routine RCL-T-CH

        RET                     ; RETURN


; -----------------------------------------
; THE 'USE STREAM OR TEMPORARY CHANNEL' ROUTINE
; -----------------------------------------
;

;; OP-STRM
L1859:  LD      A,($5CD8)       ; sv D_STR1
        INC     A
        JR      Z,L186A         ; forward to OP-CHAN

        DEC     A
        RST     10H             ; CALBAS
        DEFW    $1601           ; main CHAN-OPEN
        LD      HL,($5C51)      ; sv CURCHL
        LD      ($5CDA),HL      ; sv N_STR1
        RET


;; OP-CHAN
L186A:  LD      A,($5CD9)       ; sv L_STR1 device letter.

        CP      $4D             ; is character "M" ?
        JR      NZ,L1883        ; forward to CHECK-N

        CALL    L1B05           ; routine OP-TEMP-M creates a temporary
                                ; microdrive channel, starts motor, and
                                ; fetches record zero of named file.
        XOR     A
        CALL    L1532           ; routine SEL-DRIVE
        LD      ($5CDA),IX      ; sv N_STR1
        BIT     2,(IX+$43)      ; RECFLG
        RET     Z

        RST     20H             ; Shadow Error Restart
        DEFB    $16             ; Wrong file type


;; CHECK-N
L1883:  CP      $4E             ; is character "N" ?
        JR      NZ,L188F        ; forward to CHECK-R
```

```
        CALL    L0F46           ; routine OP-TEMP-N
        LD      ($5CDA),IX      ; sv N_STR1
        RET

; ---

;   Finally, check for the RS232 channel before producing an error.

;; CHECK-R
L188F:  CP      $54             ; is character "T" ?
        JR      Z,L1899         ; forward to USE-R

        CP      $42             ; is character "B" ?
        JR      Z,L1899         ; forward to USE-R

        RST     20H             ; Shadow Error Restart
        DEFB    $00             ; Nonsense in BASIC

; ---

;; USE-R
L1899:  CALL    L0B17           ; routine OP-RS-CH
        LD      ($5CDA),DE      ; sv N_STR1
        PUSH    DE              ;
        POP     IX              ;
        SET     7,(IX+$04)      ; channel letter
        RET                     ; return.


; ---------------------------------
; THE 'CLOSE 'MOVE' CHANNEL' ROUTINE
; ---------------------------------
;

;; CL-CHAN
L18A8:  LD      A,($5CD8)       ; sv D_STR1
        INC     A
        RET     NZ

        LD      A,($5CD9)       ; sv L_STR1 device letter.
        CP      $4D             ; is character "M" ?
        JR      NZ,L18BC        ; forward to CL-CHK-N

        LD      IX,($5CDA)      ; sv N_STR1
        CALL    L138E           ; routine CLOSE-M2
        RET                     ;


;; CL-CHK-N
L18BC:  CP      $4E             ; is character "N" ?
        RET     NZ              ;

        LD      IX,($5CDA)      ; sv N_STR1
        LD      ($5C51),IX      ; sv CURCHL
        CALL    L0FAE           ; routine SEND-NEOF
        RET


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

```
; THE 'SAVE DATA BLOCK INTO MICRODRIVE' ROUTINE
; ---------------------------------------------
;

;; SA-DRIVE
L18CB:  LD      A,($5CD6)       ; fetch drive number from D_STR1
        CALL    L1532           ; routine SEL-DRIVE starts motor.

        IN      A,($EF)         ; read microdrive port.
        AND     $01             ; isolate 'write protect' bit.
        JR      NZ,L18D9        ; forward, if not low, to STAR-SA

        RST     20H             ; Shadow Error Restart
        DEFB    $0E             ; 'Drive 'write' protected'

; ---

;; STAR-SA
L18D9:  LD      HL,($5CE9)      ; sv HD_0D
        LD      ($5CE4),HL      ; sv D_STR2

        CALL    L1B05           ; routine OP-TEMP-M creates a temporary
                                ; microdrive channel, starts motor, and
                                ; attempts to fetch record zero of named file.

        BIT     0,(IX+$18)      ; test CHFLAG
        JR      NZ,L18ED        ; forward, with no existing file, to NEW-NAME

        CALL    L138E           ; routine CLOSE-M2 closes temporary channel
                                ; and stops the motor.

        RST     20H             ; Shadow Error Restart
        DEFB    $0C             ; Writing to a 'read' file

; ---

;; NEW-NAME
L18ED:  SET     2,(IX+$43)      ; update RECFLG signal not a PRINT type file.

;   Note. the microdrive motor has been left running by OP-TEMP-M so the next
;   two lines are not necessary.  Redundant code elsewhere suggests that
;   OP-TEMP-M once stopped the drive.

        LD      A,(IX+$19)      ; fetch drive from CHDRIV.
        CALL    L1532           ; routine SEL-DRIVE stops and then restarts the
                                ; motor.

        PUSH    IX              ; transfer the channel base address
        POP     HL              ; to the HL register pair.

        LD      DE,$0052        ; prepare offset to data buffer.
        ADD     HL,DE           ; add to address start of data.
        EX      DE,HL           ; transfer this destination to DE.

        LD      HL,$5CE6        ; set source to the nine byte header at HD_00
        LD      BC,$0009        ; nine bytes to copy.
        LD      (IX+$0B),C      ; update CHBYTE_lo with length saved so far.

        LDIR                    ; block move the header info into the buffer.
```

```
        PUSH    DE              ; save destination.

;   Now calculate the number of sectors required using a similar method to
;   the one used for calculating the number of records to load.
;   Note. there is an error in the calculation as one byte should be subtracted
;   from the total bytes to ensure that there is at least one byte in the EOF
;   record. The next instruction should be to load HL with eight.

L190B:  LD      HL,$0009        ; start with the nine header bytes. ??
        LD      BC,($5CE7)      ; fetch data length from HD_0B.
        ADD     HL,BC           ; add to give total size of block.

        SRL     H               ; halve MSB to convert to 512 byte chunks.
        INC     H               ; increment to include EOF block. Wrong.

;   Note.
;   511  bytes = 502 bytes + 9 header = $01FF, SRL=$00, INC=$01 sectors OK.
;   512  bytes = 503 bytes + 9 header = $0200, SRL=$01, INC=$02 sectors WRONG!!
;   513  bytes = 504 bytes + 9 header = $0201, SRL=$01, INC=$02 sectors OK.


        PUSH    HL              ; preserve register H the sector counter.

        CALL    L1D43           ; routine FREESECT calculates free sectors on
                                ; cartridge.

        POP     HL              ; bring back the sector estimate in H.
        LD      A,E             ; load accumulator with actual sectors.
        CP      H               ; compare with estimate
        JR      NC,L1921        ; forward, if equal or greater, to SA-DRI-2

        RST     20H             ; Shadow Error Restart
        DEFB    $0F             ; 'Microdrive full'

; ---

;; SA-DRI-2
L1921:  POP     DE              ; bring back destination.
        LD      HL,($5CE4)      ; fetch start from D_STR2
        LD      BC,($5CE7)      ; fetch data length from HD_0B

;; SA-DRI-3
L1929:  LD      A,B             ; test for
        OR      C               ; zero bytes.
        JR      Z,L194F         ; forward, if all chunks saved, to SA-DRI-4

        LD      A,(IX+$0C)      ; fetch high byte of byte counter from CHBYTE_hi
        CP      $02             ; compare to 2 which would indicate 512 bytes.
        JR      NZ,L1943        ; forward, if less, to SA-DRI-WR

; a sector is written to microdrive.

        PUSH    HL              ; preserve start of data.
        PUSH    BC              ; preserve length.

        CALL    L12EE           ; routine WRITE-PRC.

        POP     BC              ; restore length.
```

```
        PUSH    IX              ; transfer the channel base address
        POP     HL              ; to the HL register pair.

        LD      DE,$0052        ; add offset to
        ADD     HL,DE           ; point to data buffer.
        EX      DE,HL           ; transfer this destination to DE.

        POP     HL              ; restore the start of data.

;; SA-DRI-WR
L1943:  LDI                     ; transfer one byte at a time decrementing BC
                                ; the total byte counter.

;   now increment the channel byte counter which started at zero and has a
;   limit of 512 bytes.

        INC     (IX+$0B)        ; increment CHBYTE_lo
        JR      NZ,L1929        ; back, if not 256, to SA-DRI-3

        INC     (IX+$0C)        ; increment CHBYTE_hi
        JR      L1929           ; back to SA-DRI-3 to check high byte.

; ---

;; SA-DRI-4
L194F:  SET     1,(IX+$43)      ; update RECFLG mark this as EOF record.

        CALL    L12EE           ; routine WRITE-PRC writes last record in set.

        LD      A,($5CEF)       ; fetch user-alterable system variable COPIES
        DEC     A               ; decrement
        JR      Z,L196A         ; forward, if zero, to END-SA-DR

        LD      ($5CEF),A       ; place decremented value back in COPIES

        RES     1,(IX+$43)      ; update RECFLG - signal not the EOF record.
        LD      A,$00           ; prepare to start saving at record zero again.
        LD      (IX+$0D),A      ; update the channel record counter CHREC.

        JR      L18ED           ; back to NEW-NAME

; ---

;; END-SA-DR
L196A:  XOR     A               ; set accumulator to zero.
        CALL    L1532           ; routine SEL-DRIVE stops the motor.

        JP      L119F           ; jump to DEL-M-BUF

; -----------------------------------------------------
; THE 'GET HEADER INFORMATION FROM MICRODRIVE' ROUTINE
; -----------------------------------------------------
; this routine extracts the nine bytes of global header information that
; is prepended to the data saved on microdrive. This relates to the type -
; Basic, Code and length etc. and is the equivalent of a tape header without
; the name which, in contrast, does have to be saved to every record.
; It is obtained therefore from the start of data at record zero.
;
```

```
; Note. the destination for this data, (program area or variable location),
; has already been calculated and since opening a channel will move this
; destination up in memory, the "Start of data" is transferred to the D_STR2
; location, otherwise used for the second filename during moves, so that its
; value is adjusted by REST-N-AD during OP-TEMP-M.

;; F-M-HEAD
L1971:  LD      HL,($5CE1)      ; copy start of data from D_STR2(+3)
        LD      ($5CE4),HL      ; to dynamic location D_STR2(+6)

        CALL    L1B05           ; routine OP-TEMP-M creates a temporary
                                ; microdrive channel, starts motor, and
                                ; fetches record zero of named file.

        BIT     0,(IX+$18)      ; test CHFLAG for valid first record.
        JR      Z,L1982         ; forward, if OK, to F-HD-2

        RST     20H             ; Shadow Error Restart
        DEFB    $11             ; 'File not found'

; ---

;; F-HD-2
L1982:  BIT     2,(IX+$43)      ; test RECFLG is it a print file
        JR      NZ,L198A        ; forward, if not, to F-HD-3

        RST     20H             ; Shadow Error Restart
        DEFB    $16             ; 'Wrong file type'

; ---

;; F-HD-3
L198A:  PUSH    IX              ; transfer the channel base address
        POP     HL              ; to the HL register pair.

        LD      DE,$0052        ; offset to CHDATA
        ADD     HL,DE           ; add to address start of data.
        LD      DE,$5CE6        ; set destination to nine system variables
                                ; starting at location HD_00.
        LD      BC,$0009        ; nine bytes to copy.

        LDIR                    ; block move to HD_00 - HD_11.

        RET                     ; return.

; -------------------------------------------------
; THE 'LOAD OR VERIFY BLOCK FROM MICRODRIVE' ROUTINE
; -------------------------------------------------
;   This subroutine is called once only from LV-ANY to load a block of code,
;   previously SAVED to a number of sectors, from microdrive.
;   At this stage a temporary channel has already been created and it holds
;   the first 512 byte record containing at the start the nine header bytes.
;   There will be an accurate microdrive map for the drive which has its
;   motor running.
;   The block could be a program, code bytes or an array and the first
;   receiving location is in HL and the length in DE.

;; LV-MCH
L199A:  LD      ($5CE9),HL      ; save start in system variable HD_0D
```

```
;   now directly read the header values at the start of the data buffer.

        LD      E,(IX+$53)      ; directly read the saved length
        LD      D,(IX+$54)      ; from the data buffer into DE.

;   now calculate how many 512 byte microdrive records need to be read in
;   by taking the total minus one to ensure an EOF record.
;   e.g.
;   1023 bytes = 1014 bytes + 9 header - 1 = $03FE, SRL=$01, INC=$02 sectors
;   1024 bytes = 1015 bytes + 9 header - 1 = $03FF, SRL=$01, INC=$02 sectors
;   1025 bytes = 1016 bytes + 9 header - 1 = $0400, SRL=$02, INC=$03 sectors

        LD      HL,$0008        ; add eight in effect +9 for header -1.
        ADD     HL,DE           ; add the program/code length.

;   the MSB is the number of 256 chunks.

        SRL     H               ; shift right to halve and give 512 byte
                                ; chunks.
        INC     H               ; increment to include the extra sector.

        LD      A,H             ; use accumulator to store record count
        LD      ($5CE7),A       ; in the temporary system variable HD_0B

;   the microdrive map is now saved on the machine stack, for later recall,
;   and at the same time the current map locations are all set to zero.
;   The new map is to be used for records rather than sectors.

        CALL    L1A04           ; routine SA-MAP saves the thirty two bytes
                                ; of the map on the machine stack safely
                                ; dipping into the 80 bytes of spare memory.

;   now, since this is record zero, subtract the nine header bytes from the
;   current record length and put back.

        LD      DE,$0009        ;

        LD      L,(IX+$45)      ; RECLEN_lo
        LD      H,(IX+$46)      ; RECLEN_hi

        OR      A               ; clear carry

        SBC     HL,DE           ;

        LD      (IX+$45),L      ; RECLEN_lo
        LD      (IX+$46),H      ; RECLEN_hi

;

        PUSH    IX              ; transfer the channel base address
        POP     HL              ; to the HL register pair.

        LD      DE,$005B        ; prepare offset $0052 to data and then an
        ADD     HL,DE           ; extra nine bytes. Add to skip the header.

        LD      DE,($5CE9)      ; set destination from HD_0D

        JR      L19EA           ; forward to LOOK-MAP to enter the record
```

```
                                ; loading loop at the mid-point as record
                                ; zero is already in the channel.

; ---

;   The record loading loop loads records in random order.  Consider that
;   multiple copies of a filename may have been saved so there may be several
;   sectors with the same record number.

;; USE-REC
L19D0:  CALL    L1A5D           ; routine F-REC2 fetches only a header and
                                ; record that matches the name specified
                                ; in CHNAME and only if the map bit is reset
                                ; indicating no sector with this record number
                                ; has already been loaded.

        LD      A,(IX+$44)      ; re-fetch record number from RECNUM.

; Note. the next test is a nonsense as a record zero has already been marked
; so no sector with record zero could be reloaded.

        OR      A               ; test for a record zero.
        JR      Z,L19D0         ; back, if so, to USE-REC.

; now calculate the destination if this 512 byte sector.

        RLA                     ; double recnum to give 512 byte chunks
        DEC     A               ; decrement to adjust for nine bytes of header.
        LD      D,A             ; place in MSB of offset

        LD      E,$F7           ; set LSB of offset to $00 - $09 for header.
        LD      HL,($5CE9)      ; fetch start of data from HD_0D

        ADD     HL,DE           ; add to calculate destination for this sector.
        EX      DE,HL           ; transfer destination to DE.

        PUSH    IX              ; transfer the channel base address
        POP     HL              ; to the HL register pair.

        LD      BC,$0052        ; prepare offset to start of 512 byte buffer
        ADD     HL,BC           ; add so that HL addresses start of data.

; -> The mid loop entry point.

;; LOOK-MAP
L19EA:  EXX                     ; preserve HL and DE by using alternate
                                ; registers.

        CALL    L13BF           ; routine CHK-MAP-2 sets HL to the map byte
                                ; and B to the mask.

; Note. the routine also resets the zero flag if this record has previously
; been loaded but this is not possible.

        JR      NZ,L19D0        ; back, if already loaded, to USE-REC.

;   since this is the first time for this record mark so that not loaded again.

        LD      A,(HL)          ; mark the record bit
```

```
        OR      B               ; by setting it so that it is not
        LD      (HL),A          ; considered for loading again.

        EXX                     ; restore HL (source) and DE (destination).

        CALL    L1A39           ; routine LD-VE-M loads or verifies a
                                ; data record.

;   now decrement the record count which is beyond reach of IY register.

        LD      A,($5CE7)       ; fetch count of records to be loaded HD_0B
        DEC     A               ; decrement
        LD      ($5CE7),A       ; and place back in system variable HD_0B

        JR      NZ,L19D0        ; back, if not finished to USE-REC

;   the block is loaded

        CALL    L1A1E           ; routine RE-MAP restores the true microdrive
                                ; map from the stack.

        RET                     ; return.


; -----------------------------------------
; THE 'SAVE MICRODRIVE MAP CONTENTS' ROUTINE
; -----------------------------------------
;   This routine saves the sector-mapped microdrive map on the machine stack
;   at the same time setting each of the 32 vacated locations to zero.

;; SA-MAP
L1A04:  POP     HL              ; drop the return address into HL
        LD      ($5CC9),HL      ; and save in unused system variable SECTOR

        LD      L,(IX+$1A)      ; fetch address of microdrive map from CHMAP
        LD      H,(IX+$1B)      ; fetch address of microdrive map from CHMAP
        LD      BC,$1000        ; set word counter B to sixteen and C to zero.

;   now enter a loop stacking two bytes at a time.

;; SA-MAP-LP
L1A11:  LD      E,(HL)          ; fetch first byte to E.
        LD      (HL),C          ; set location to zero.
        INC     HL              ; bump address.
        LD      D,(HL)          ; fetch second byte to D.
        LD      (HL),C          ; set location to zero.
        INC     HL              ; bump address.

        PUSH    DE              ; save DE on machine stack.

        DJNZ    L1A11           ; back, for 16 pairs, to SA-MAP-LP

        LD      HL,($5CC9)      ; restore return address from SECTOR
        JP      (HL)            ; and jump to location.

; -------------------------------------------
; THE 'RESTORE MICRODRIVE MAP CONTENTS' ROUTINE
; -------------------------------------------
;   This routine is the opposite of the above and restores the sector-mapped
```

```
;   microdrive map from the machine stack back to its original location
;   overwriting the now redundant record-indicating map.

;; RE-MAP
L1A1E:  POP     HL              ; drop the subroutine return address.
        LD      ($5CC9),HL      ; store in the multi-purpose variable SECTOR.

        LD      L,(IX+$1A)      ; fetch address of microdrive map from CHMAP.
        LD      H,(IX+$1B)      ; fetch address of microdrive map from CHMAP.
        LD      DE,$001F        ; thirty one locations are added.
        ADD     HL,DE           ; to address the last location.
        LD      B,$10           ; set the pop counter to sixteen.

;; RE-MAP-LP
L1A2E:  POP     DE              ; pop two bytes of the map from the stack.

        LD      (HL),D          ; insert a map byte.
        DEC     HL              ; decrement the address.
        LD      (HL),E          ; insert second map byte.
        DEC     HL              ; decrement the address again.

        DJNZ    L1A2E           ; back, sixteen times, to RE-MAP-LP.

        LD      HL,($5CC9)      ; restore the return address from SECTOR.
        JP      (HL)            ; and jump to address.

; ---------------------
; THE 'LD-VE-M' ROUTINE
; ---------------------
;   The Load or Verify from Microdrive routine.
;   This routine loads or verifies up to 512 bytes of data currently in the
;   microdrive channel data buffer.

;; LD-VE-M
L1A39:  LD      C,(IX+$45)      ; RECLEN_lo
        LD      B,(IX+$46)      ; RECLEN_hi

; now test if a VERIFY operation by performing the equivalent of bit 7,(iy+$7c)

        LD      A,($5CB6)       ; load system variable FLAGS_3 to accumulator.
        BIT     7,A             ; test FLAGS_3 value - performing VERIFY ?

        JR      NZ,L1A49        ; forward, if so, to VE-M-E

;   the operation is a LOAD.

        LDIR                    ; block copy the bytes.
        RET                     ; return.

; ---

;   the operation is a VERIFY.

;; VE-M-E
L1A49:  LD      A,(DE)          ; fetch a byte from the destination.
        CP      (HL)            ; compare to that of source
        JR      NZ,L1A55        ; forward, with mismatch, to VE-FAIL

        INC     HL              ; increment source address.
```

```
        INC     DE              ; increment destination address.
        DEC     BC              ; decrement byte count.
        LD      A,B             ; test for
        OR      C               ; zero.
        JR      NZ,L1A49        ; back, if not, to VE-M-E

        RET                     ; return.

; ---

;; VE-FAIL
L1A55:  RST     20H             ; Shadow Error Restart
        DEFB    $15             ; 'Verification has failed'

; ----------------------------------------
; THE 'FETCH RECORD FROM MICRODRIVE' ROUTINE
; ----------------------------------------
;   Entered at F-REC2,
;   Note. the first entry point f-rec1 is unused.

;; f-rec1
L1A57:  LD      A,(IX+$19)      ; fetch drive number.
        CALL    L1532           ; routine SEL-DRIVE starts motor.

; -->

;; F-REC2
L1A5D:  LD      BC,$04FB        ; Set sector counter to 5 * 255 = 1275
        LD      ($5CC9),BC      ; Update System Variable SECTOR

;; UNTILFIVE
L1A64:  CALL    L1280           ; routine G-HD-RC fetches the next header and
                                ; matching record to pass tape head.

        JR      C,L1A7B         ; forward, with name mismatch, to F-ERROR

        JR      Z,L1A7B         ; forward, with unused record, to F-ERROR

        CALL    L13BF           ; routine CHK-MAP-2  checks RECORD.

        JR      NZ,L1A7B        ; forward, if already loaded, to F-ERROR

        PUSH    IX              ; transfer the channel base address
        POP     HL              ; to the HL register pair.

        LD      DE,$0052        ;
        ADD     HL,DE           ;
        CALL    L142B           ; routine CHKS-BUFF
        RET     Z               ;


;; F-ERROR
L1A7B:  CALL    L13F7           ; routine DEC-SECT
        JR      NZ,L1A64        ; back to UNTILFIVE

        RST     20H             ; Shadow Error Restart
        DEFB    $11             ; File not found
```

```
; --------------------------------------
; THE 'RESTORE ADDRESS OF FILENAME' ROUTINE
; --------------------------------------
;   This subroutine performs a similar function to the Main ROM POINTERS routine
;   by adjusting the extra system variables that point to filenames within
;   the sliding, dynamic areas.
;   On entry HL points to the start of the New Room and BC holds the number of
;   bytes created.

;; REST-N-AD
L1A82:  PUSH    HL              ; Preserve HL throughout.

        PUSH    HL              ; Preserve HL for second call.

        LD      DE,($5CE4)      ; Fetch D_STR2 - start of 2nd filename.
        CALL    L1A9D           ; routine TST-PLACE may adjust fetched value.
        LD      ($5CE4),DE      ; Store in System Variable  D_STR2

        POP     HL              ; Restore HL for second call.

        LD      DE,($5CDC)      ; Fetch D_STR1 - start of 1st filename.
        CALL    L1A9D           ; routine TST-PLACE
        LD      ($5CDC),DE      ; Store in System Variable D_STR1

        POP     HL              ; Restore original HL value.

        RET                     ; return.

; --------------------------
; THE 'TEST PLACE' SUBROUTINE
; --------------------------
;   This subroutine is used twice from above to test if the filename address
;   is within the Spectrum's dynamic RAM area.
;   HL = location before new room.
;   DE = address of filename.
;   BC = amount of room just created.

;; TST-PLACE
L1A9D:  SCF                     ; adjust for one before.
        SBC     HL,DE           ; subtract filename address from start of room
        RET     NC              ; and if before new room then return.

        LD      HL,($5C65)      ; fetch STKEND and if the filename is above
        SBC     HL,DE           ; then it is not in dynamic memory.
        RET     C               ;

        EX      DE,HL           ; add the number of bytes created
        ADD     HL,BC           ; to the filename address
        EX      DE,HL           ; to bring it into line.

        RET                     ; return.


; ----------------------------------
; THE 'CALLS TO THE COMMANDS' ROUTINE
; ----------------------------------
;

;; ERASE-RUN
```

```
L1AAB:  CALL    L1D79           ; routine ERASE
        JR      L1AC9           ; forward to ENDC

; ---

;; MOVE-RUN
L1AB0:  CALL    L17F5           ; routine MOVE
        JR      L1AC9           ; forward to ENDC

; ---

;; CAT-RUN
L1AB5:  CALL    L1C52           ; routine CAT
        JR      L1AC9           ; forward to ENDC

; ---

;; FOR-RUN
L1ABA:  CALL    L1B5D           ; routine FORMAT
        JR      L1AC9           ; forward to ENDC

; ---

;; OP-RUN
L1ABF:  CALL    L1ACC           ; routine OP-M-STRM
        JR      L1AC9           ; forward to ENDC

; ---

;; SAVE-RUN
L1AC4:  CALL    L18CB           ; routine SA-DRIVE
        JR      L1AC9           ; forward to ENDC

; ---

;; ENDC
L1AC9:  JP      L05C1           ; jump to END1

; ----------------------------------------
; THE 'OPEN A PERMANENT "M" CHANNEL' ROUTINE
; ----------------------------------------
;

;; OP-M-STRM
L1ACC:  LD      A,($5CD8)       ; sv D_STR1
        ADD     A,A             ;
        LD      HL,$5C16        ; sv STRMS_00
        LD      E,A             ;
        LD      D,$00           ;
        ADD     HL,DE           ;
        PUSH    HL              ;
        CALL    L1B05           ; routine OP-TEMP-M creates a temporary
                                ; microdrive channel, starts motor, and
                                ; fetches record zero of named file.
        BIT     0,(IX+$18)      ; CHFLAG
        JR      Z,L1AE9         ; forward to MAKE-PERM

        IN      A,($EF)         ;
        AND     $01             ; isolate write prot.
```

```
        JR      NZ,L1AE9        ; forward to MAKE-PERM

        RST     20H             ; Shadow Error Restart
        DEFB    $0E             ; Drive 'write' protected

; ---

;; MAKE-PERM
L1AE9:  RES     7,(IX+$04)      ; channel letter
        XOR     A               ;
        CALL    L1532           ; routine SEL-DRIVE
        BIT     0,(IX+$18)      ; CHFLAG
        JR      NZ,L1AFF        ; forward to STORE-DSP

        BIT     2,(IX+$43)      ; RECFLG
        JR      Z,L1AFF         ; forward to STORE-DSP

        RST     20H             ; Shadow Error Restart
        DEFB    $16             ; Wrong file type

; ---

;; STORE-DSP
L1AFF:  EX      DE,HL           ;
        POP     HL              ;
        LD      (HL),E          ;
        INC     HL              ;
        LD      (HL),D          ;
        RET                     ;


; -----------------------------------------
; THE 'OPEN A TEMPORARY "M" CHANNEL' ROUTINE
; -----------------------------------------
; (Hook Code: $22)
;

;; OP-TEMP-M
L1B05:  CALL    L10A5           ; routine SET-T-MCH creates a temporary channel
                                ; using either an existing microdrive map from
                                ; a channel also using this drive or allocating
                                ; a new one initialized to $FF bytes.
                                ; fields CHREC etc. are set to zero.

        PUSH    HL              ; preserve the offset to this channel from CHANS

        LD      A,(IX+$19)      ; fetch drive number 1 - 8 from CHDRIV
        CALL    L1532           ; routine SEL-DRIVE starts motor and disables
                                ; interrupts.

        LD      BC,$0032        ; now set temporary unused
        LD      ($5CC9),BC      ; system variable SECTOR_lo to fifty
                                ; and set SECTOR_hi to zero.

; now enter a loop

;; OP-F-L
L1B16:  CALL    L1280           ; routine G-HD-RC fetches any header and
                                ; matching record
```

```
        PUSH    AF              ; preserve return status flags.

; maintain the 'maximum sectors to visit' so only one rotation of tape occurs.

        LD      A,(IX+$29)      ; fetch sector from HDNUMB
        ADD     A,$03           ; add 3
        LD      HL,$5CC9        ; address current (max+3) in SECTOR_lo
        CP      (HL)            ; compare
        JR      C,L1B26         ; forward, if less, to OP-F-X

        LD      (HL),A          ; update with new max sectors to visit.

;; OP-F-X
L1B26:  POP     AF              ; restore status flags.

        JR      C,L1B49         ; forward, with no name match, to OP-F-4

        JR      Z,L1B46         ; forward, if unused, to OP-F-3
                                ; to reset map bit.

; the fetched record is one from the file named in CHNAME

        RES     0,(IX+$18)      ; update CHFLAG to indicate success.

        LD      A,(IX+$44)      ; fetch the record number within file RECNUM
        OR      A               ; test for zero - first record.
        JR      NZ,L1B41        ; forward, if not, to OP-F-2

        PUSH    IX              ; transfer the channel base address
        POP     HL              ; to the HL register pair.

        LD      DE,$0052        ; prepare offset to data and
        ADD     HL,DE           ; add to address start of the 512 byte buffer

        CALL    L142B           ; routine CHKS-BUFF checks that checksum agrees.
        JR      Z,L1B5B         ; forward, if OK, to DP-F-5

;; OP-F-2
L1B41:  CALL    L1258           ; routine GET-R-2 repeatedly calls the
                                ; subroutine G-HD-RC (as at start of loop)
                                ; until the validated record matching CHREC
                                ; (zero) is loaded.

        JR      L1B5B           ; forward, with success, to DP-F-5.

; ---

;; OP-F-3
L1B46:  CALL    L13E3           ; routine RES-B-MAP resets bit for unused
                                ; sectors.

; the branch was here

;; OP-F-4
L1B49:  LD      HL,$5CCA        ; address visited sector count SECTOR_hi

        LD      A,(HL)          ; fetch sector counter.
        INC     A               ; increment
```

```
        LD      (HL),A          ; and put back in SECTOR_hi.
        DEC     HL              ; address the max sector value.
        CP      (HL)            ; compare.
        JR      C,L1B16         ; back, if less than one revolution, to OP-F-L

; else a full revolution occurred without finding the record.

        RES     1,(IX+$43)      ; RECFLG
        RES     2,(IX+$43)      ; RECFLG

; the branch was here with record zero of named file.

;; DP-F-5
L1B5B:  POP     HL              ; restore the offset from CHANS.

        RET                     ; return.

; --------------------------------
; THE 'FORMAT "M" COMMAND' ROUTINE
; --------------------------------
; e.g. FORMAT "m";1;"demos"


;; FORMAT
L1B5D:  CALL    L10A5           ; routine SET-T-MCH creates a temporary
                                ; microdrive channel with name of cartridge.

        LD      A,(IX+$19)      ; fetch drive number from CHDRIV
        CALL    L1565           ; routine SW-MOTOR starts the motor.

        LD      BC,$32C8        ; decimal 1300
        CALL    L1652           ; routine DELAY-BC

        DI                      ; Disable Interrupts.

        IN      A,($EF)         ; read microdrive port.
        AND     $01             ; isolate write prot. bit.
        JR      NZ,L1B75        ; forward, if not low, to FORMAT-1

        RST     20H             ; Shadow Error Restart
        DEFB    $0E             ; Drive 'write' protected

; ---

;; FORMAT-1
L1B75:  LD      A,$E6           ; enable writing.
        OUT     ($EF),A         ; update microdrive port.

        LD      BC,$00FF        ; assume 255 sectors will fit on a tape.
        LD      ($5CC9),BC      ; set system variable SECTOR.

        PUSH    IX              ; transfer the channel base address
        POP     HL              ; to the HL register pair.

        LD      DE,$002C        ; offset to HDNAME
        ADD     HL,DE           ;
        EX      DE,HL           ; make destination HDNAME
        LD      HL,$FFE2        ;
        ADD     HL,DE           ; make source CHNAME
```

```
        LD      BC,$000A        ; ten bytes to copy.
        LDIR                    ; copy - C is now zero.

; now prepare an 'unusable' record.

        XOR     A               ; make accumulator zero.

        LD      (IX+$47),A      ; set first character of RECNAM to zero.
        SET     0,(IX+$28)      ; mark HDFLAG indicate a header.
        RES     0,(IX+$43)      ; mark RECFLG indicate a record.
        SET     1,(IX+$43)      ; mark RECFLG indicate an EOF record.

        PUSH    IX              ; transfer the channel base address
        POP     DE              ; to the DE register pair for a change.

        LD      HL,$0043        ; offset to RECFLG - start of record descriptor.
        ADD     HL,DE           ; add offset to start of record descriptor.

        CALL    L1426           ; routine CHKS-HD-R inserts 14 byte checksum.

;   Now enter a loop to write the blocks to the cartridge

;; WR-F-TEST
L1BAB:  CALL    L13F7           ; routine DEC-SECT decrements sector originally
                                ; set to $FF
        JR      Z,L1BDF         ; forward, if BC is zero, to TEST-SCT  ->

        LD      (IX+$29),C      ; insert reduced sector number in HDNUMB

        PUSH    IX              ; transfer the base channel address
        POP     HL              ; to the HL register pair.

        LD      DE,L0028        ; offset to the header
        ADD     HL,DE           ; add to address HDFLAG.

        CALL    L1426           ; routine CHKS-HD-R inserts 14 byte checksum
                                ; preserving the HL value.

        LD      DE,$FFF4        ; subtract twelve
        ADD     HL,DE           ; to address the header PREAMBLE.

        CALL    L15AD           ; routine OUT-M-HD writes the header to tape.

        LD      BC,$01B2        ; set timer for gap - 434 decimal.
        CALL    L1652           ; routine DELAY-BC

        PUSH    IX              ; transfer start of channel
        POP     HL              ; to HL register pair.

        LD      DE,$0037        ; adjust HL to point to PREAMBLE at
        ADD     HL,DE           ; start of record descriptor.
        CALL    L16AF           ; routine WR-BLK writes record to tape.

        LD      BC,$0100        ; a short delay.
        CALL    L1652           ; routine DELAY-BC

        CALL    L163E           ; routine TEST-BRK
```

```
        JR      L1BAB           ; loop back to WR-F-TEST for sectors 254 - 1.

; ---

; -> the branch was to here when all sectors from 254 down to 1 have been
;    written.

;; TEST-SCT
L1BDF:  LD      BC,$0087        ; use value 35 decimal.
        CALL    L1652           ; routine DELAY-BC

        LD      A,$EE           ; signal disable writing.
        OUT     ($EF),A         ; output to microdrive port.

        LD      A,(IX+$19)      ; select drive number from CHDRIV.
        CALL    L1532           ; routine SEL-DRIVE.

        LD      BC,$0032        ; set max sector to fifty, read sectors to zero.
        LD      ($5CC9),BC      ; insert both values in SECTOR

;; CHK-SCT
L1BF6:  CALL    L13A9           ; routine GET-M-HD2 reads the next valid header
                                ; to pass the tape head.
        LD      A,(IX+$29)      ; fetch the unique sector number from HDNUMB
        ADD     A,$03           ; add three to value.
        LD      HL,$5CC9        ; address system variable SECTOR
        CP      (HL)            ; and compare to total of sectors to visit.
        JR      C,L1C05         ; forward if less to CHK-SCT2

        LD      (HL),A          ; else insert new value for sectors to visit.

;; CHK-SCT2
L1C05:  CALL    L13C4           ; routine CHECK-MAP checks if sector is free
                                ; on the microdrive map.
        JR      Z,L1C1E         ; forward, if so, to CHK-NSECT

        PUSH    IX              ; transfer channel base address
        POP     HL              ; to the HL register pair.

        LD      DE,$0043        ; offset to the start of record descriptor.
        ADD     HL,DE           ; add to address RECFLG.

        CALL    L165A           ; routine READ-BLK reads in a block.

        JR      NZ,L1C1E        ; forward, with bad read, to CHK-NSECT
                                ; leaving map bit set.

        CALL    L1426           ; routine CHKS-HD-R check the header checksum
        JR      NZ,L1C1E        ; forward, with error, to CHK-NSECT

        CALL    L13E3           ; routine RES-B-MAP resets the map bit marking
                                ; the sector as usable.

;; CHK-NSECT
L1C1E:  LD      HL,$5CCA        ; address SECTOR_hi the visited sector counter.
        LD      A,(HL)          ; fetch the value.
        INC     A               ; increment
        LD      (HL),A          ; and place back.
```

```
        DEC     HL              ; decrement to address max sectors to visit.
        CP      (HL)            ; compare counter to limit.
        JR      C,L1BF6         ; back, if counter is less, to CHK-SCT

        LD      L,(IX+$1A)      ; load L from CHMAP_lo
        LD      H,(IX+$1B)      ; load H from CHMAP_hi

;   Register HL now addresses the microdrive maps which at this stage have
;   sectors 0 and 255 marked as unusable. If as is usual, the lower numbered
;   sectors have overwritten the higher numbered sectors then typically
;   the top seventy sectors, or so, will be marked as unusable though not on an
;   emulated machine which at this stage will only have 0 and 255 marked
;   unusable. On a real machine the splice will show up as an unusable sector
;   and there may be some other sectors unusable due to dirt on the recording
;   film.
;   What happens next is unique to this ROM and is no doubt due to extensive
;   testing and analysis of the microdrives by Sinclair Research.
;   Microdrive sectors are encountered in descending order, as they are
;   written, and the following routine marks any sector following a bad sector
;   as bad also.  One can conclude that Sinclair Research's test programme
;   revealed that the first sectors to fail were those adjacent to contaminated
;   or damaged sectors.
;   This perhaps explains why my use of the microdrives with ROM 2 has been
;   more reliable than early reviews, no doubt with ROM 1, suggested.

        LD      DE,$001F        ; add thirty one to start at the end of the map
        ADD     HL,DE           ; - the byte that refers to sector 255.
        LD      B,$20           ; count the thirty two bytes of a map.

        SCF                     ; set carry flag to ensure that sector 255
                                ; is unusable - but it is already marked so ??

;; PREP-MARK
L1C35:  LD      A,(HL)          ; fetch a byte representing eight sectors.

        LD      C,A             ; and store it in C - Note. unnecessary.

        RRA                     ; rotate right accumulator  C->76543210->C

        OR      C               ; combine with original value. Why not OR (HL) ?

        LD      (HL),A          ; store the modified byte back in the map.

        DEC     HL              ; point to the next byte for lower-numbered
                                ; sectors.

L1C3B:  DJNZ    L1C35           ; loop back to PREP-MARK for all 32 map bytes.

;   Note. the above routine is untidy.  There is no need to set the carry flag
;   and no need to store the original value in C.  While it achieves it's aims,
;   if sector one is bad it has no effect on the next sector to be encountered.
;   That would be hard to implement but the first sector that is marked bad,
;   the highest numbered sector, is marked so solely because it is adjacent to
;   the overwritten section.

;   Note. from details of addresses Andrew Pennell gave in the magazine "Your
;   Sinclair" it can be deduced that the unpublished ROM 3 had two extra
;   instruction bytes at this point and together with a cleanup, this may have
;   addressed the above issue.
```

```
;   Now prepare to overwrite the unusable sectors (which are mapped as usable)
;   with record descriptors which are usable.

        CALL    L1E49           ; routine IN-CHK marks the channel record
                                ; descriptor fields as usable by blanking
                                ; both RECFLG and RECLEN and then inserting
                                ; the descriptor checksum.

;   A loop is now entered to write usable datablocks to every sector indicated
;   as usable in the microdrive map.

;; MARK-FREE
L1C40:  CALL    L1349           ; routine CHK-FULL checks if there is still a
                                ; usable sector on the cartridge.
        JR      NZ,L1C4D        ; forward, if so, to MK-BLK.

;   The FORMAT operation is now complete.

        XOR     A               ; select no motor
        CALL    L1532           ; routine SEL-DRIVE stops the microdrive motor.

        CALL    L119F           ; routine DEL-M-BUF deletes the microdrive
                                ; buffer and the microdrive map.

        RET                     ; return.                       >>>>>>>

; ---

;; MK-BLK
L1C4D:  CALL    L135A           ; routine SEND-BLK writes block to microdrive
                                ; cartridge as indicated by the microdrive map
                                ; which is then updated by the routine.

        JR      L1C40           ; loop back to MARK-FREE


; -------------------------
; THE 'CAT COMMAND' ROUTINE
; -------------------------
;

;; CAT
L1C52:  LD      A,($5CD8)       ; fetch output stream from S_STR1

        RST     10H             ; CALBAS
        DEFW    $1601           ; main CHAN-OPEN

        CALL    L10A5           ; routine SET-T-MCH sets a temporary channel.

        LD      A,(IX+$19)      ; fetch drive number from CHDRIV.
        CALL    L1532           ; routine SEL-DRIVE starts the motor.

        LD      BC,$0032        ; set maximum sector to 50 and initialize
                                ; value of sectors read to zero.
        LD      ($5CC9),BC      ; update system variable SECTOR

;   On the original Interface 1 ROM operations like CAT and ERASE were quite
;   slow as the routines assumed the theoretical maximum number of sectors was
```

```
;   256.  In reality, the maximum number of sectors on a microdrive is
;   approximately 180, so the original routines spent the last 3 seconds
;   reading about 75 sectors for the second time.  The improved algorithm above
;   is to keep a record of the maximum sector + 3 and when the number of
;   visited sectors is equal to this number then a complete revolution of the
;   tape has been made and the operation can cease. The overhead of three is
;   to ensure that bad sectors or the tape splice do not cause the operation to
;   end prematurely.
;   Happily, this algorithm also works with emulators which usually provide the
;   full 256 sectors.

;; CAT-LP
L1C68:  CALL    L13A9           ; routine GET-M-HD2 reads in 14 byte header.

        LD      A,(IX+$29)      ; fetch value of sector from HDNUMB

        ADD     A,$03           ; add 3 to value.

        LD      HL,$5CC9        ; address system variable SECTOR_lo
        CP      (HL)            ; compare to contents
        JR      C,L1C77         ; forward if A is less to CAT-LP-E

        LD      (HL),A          ; else update SECTOR_lo with higher value.

;; CAT-LP-E
L1C77:  CALL    L1E5E           ; routine G-RDES loads only a
                                ; 14 byte record descriptor.
        JR      NZ,L1C68        ; back, with error or mismatch, to CAT-LP

; a record can be considered in use if either the RECLEN is maximum $0200 or
; the RECFLG indicates that it is the seldom full EOF record.

        LD      A,(IX+$43)      ; RECFLG
        OR      (IX+$46)        ; RECLEN_hi
        AND     $02             ;
        JR      NZ,L1C8B        ; forward, if used, to IN-NAME

; else mark sector free in microdrive map and find next sector.

        CALL    L13E3           ; routine RES-B-MAP
        JR      L1CF4           ; forward to F-N-SCT

; a name is to be inserted in the 512 byte data buffer workspace, if it is not
; there already. Secret files are not listed.

;; IN-NAME
L1C8B:  LD      A,(IX+$47)      ; take first character of RECNAM
        OR      A               ; test for zero.
        JR      Z,L1CF4         ; forward, if CHR$ 0, to F-N-SCT

        PUSH    IX              ; transfer base address
        POP     HL              ; to HL register.

        LD      DE,$0052        ; offset to start of data buffer.
        ADD     HL,DE           ; add to address names.
        LD      DE,$000A        ; set DE to ten, the length of a name.
        LD      B,$00           ; set high byte to zero.
        LD      C,(IX+$0D)      ; fetch name total from CHREC initially zero.
```

```
;; SE-NAME
L1CA0:  LD      A,C             ; test name count for zero
        OR      A               ;
        JR      Z,L1CDA         ; forward, with first name, to INS-NAME

        PUSH    HL              ; save buffer address.
        PUSH    IX              ; save channel base address.
        PUSH    BC              ; save name total.

        LD      B,$0A           ; set character counter to ten.

;; T-NA-1
L1CAA:  LD      A,(HL)          ; take letter of buffered name.
        CP      (IX+$47)        ; compare to that in RECNAM
        JR      NZ,L1CB5        ; forward, with mismatch, to T-NA-2

        INC     HL              ; increment
        INC     IX              ; both pointers.
        DJNZ    L1CAA           ; back, for all ten, to T-NA-1


;; T-NA-2
L1CB5:  POP     BC              ; restore
        POP     IX              ; all
        POP     HL              ; pointers.

; if all ten characters match then find next sector.

        JR      Z,L1CF4         ; forward to F-N-SCT

; if buffered name is higher than new name then re-order to create a slot.

        JR      NC,L1CC1        ; forward to ORD-NAM

; else add ten to buffer address and compare with following name performing
; a simple insert if the end of the list is reached.

        ADD     HL,DE           ; add ten to address.
        DEC     C               ; decrement name counter.
        JR      L1CA0           ; back to SE-NAME

; ---


;; ORD-NAM
L1CC1:  PUSH    HL              ; save pointer to start of name slot.
        PUSH    DE              ; save the value ten.
        PUSH    BC              ; save the buffered name counter.

        PUSH    HL              ; save address of name slot again.
        SLA     C               ; double name count.
        LD      H,B             ; set H to zero.
        LD      L,C             ; HL = 2 * count
        ADD     HL,BC           ; HL = 4 * count
        ADD     HL,BC           ; HL = 6 * count        Note. add hl,hl doubles.
        ADD     HL,BC           ; HL = 8 * count        c.f. Main ROM
        ADD     HL,BC           ; HL = 10 * count

        LD      B,H             ; transfer number of bytes
```

```
        LD      C,L             ; to be moved to BC register.

        POP     HL              ; restore address of insertion point.
        DEC     HL              ; decrement and then add
        ADD     HL,BC           ; bytes to be moved to point to end of block.

        EX      DE,HL           ; now make DE
        ADD     HL,DE           ; the destination
        EX      DE,HL           ; ten bytes higher.

        LDDR                    ; slide the block of higher names upwards.

        POP     BC              ; restore name count.
        POP     DE              ; restore ten value.
        POP     HL              ; restore insertion point.

;; INS-NAME
L1CDA:  PUSH    IX              ; save channel base address.
        LD      B,$0A           ; set character count to ten.

;; MOVE-NA
L1CDE:  LD      A,(IX+$47)      ; fetch a character from new name at RECNAM
        LD      (HL),A          ; insert into buffer.
        INC     IX              ; increment both
        INC     HL              ; pointers.
        DJNZ    L1CDE           ; loop back to MOVE-NA

        POP     IX              ; restore channel base address.

        LD      A,(IX+$0D)      ; fetch count of names from CHREC
        INC     A               ; increment
        LD      (IX+$0D),A      ; and store back in CHREC

        CP      $32             ; compare to maximum of 50.
        JR      Z,L1CFF         ; forward, if buffer filled, to BF-FILLED


;; F-N-SCT
L1CF4:  LD      HL,$5CCA        ; sv SECTOR_hi
        LD      A,(HL)          ; fetch actual count of used sectors.
        INC     A               ; and increment.
        LD      (HL),A          ; update SECTOR_hi
        DEC     HL              ; address system variable SECTOR_lo
        CP      (HL)            ; compare
        JP      C,L1C68         ; jump to CAT-LP

;; BF-FILLED
L1CFF:  PUSH    IX              ;

        XOR     A               ; clear accumulator
        CALL    L1532           ; routine SEL-DRIVE stops the motor.

        PUSH    IX              ; transfer the channel base address
        POP     HL              ; to the HL register pair.

        LD      DE,$002C        ; offset to cartridge name HDNAME.
        ADD     HL,DE           ; add the offset to address the name.

        CALL    L1D5B           ; routine PRNAME prints name and a carriage
```

```
                                ; return.

        LD      A,$0D           ; prepare an extra carriage return.
        CALL    L1D71           ; routine PRCHAR outputs it.

        PUSH    IX              ;
        POP     HL              ;

        LD      DE,$0052        ; offset to CHDATA - the 512 byte data buffer.
        ADD     HL,DE           ; add to address list of up to fifty names.

        LD      B,(IX+$0D)      ; load B with count of names from CHREC
        LD      A,B             ; test for
        OR      A               ; zero.
        JR      Z,L1D27         ; forward, if so, to NONAMES


;; OT-NAMS
L1D22:  CALL    L1D5B           ; routine PRNAME
        DJNZ    L1D22           ; loop back to OT-NAMS


;; NONAMES
L1D27:  CALL    L1D43           ; routine FREESECT

        LD      A,E             ;
        SRL     A               ;

        RST     10H             ; CALBAS
        DEFW    $2D28           ; main STACK-A

        LD      A,$0D           ;
        CALL    L1D71           ; routine PRCHAR

        RST     10H             ; CALBAS
        DEFW    $2DE3           ; main PRINT-FP

        LD      A,$0D           ;
        CALL    L1D71           ; routine PRCHAR

        POP     IX              ;

        CALL    L119F           ; routine DEL-M-BUF

        RET                     ; return.


; ----------------------
; THE 'FREESECT' ROUTINE
; ----------------------
;   This routine is called from SAVE and CAT to calculate the number of free
;   sectors that are present on a microdrive from the map information.
;   The count of free sectors is returned in the E register.

;; FREESECT
L1D43:  LD      L,(IX+$1A)      ; address of microdrive map.
        LD      H,(IX+$1B)      ; for channel transferred to HL.

        LD      E,$00           ; initialize sector count to zero.
```

```
        LD      C,$20           ; there are thirty two bytes to examine.

;; FR-SC-LP
L1D4D:  LD      A,(HL)          ; fetch a byte from the map.
        INC     HL              ; address next map location.

        LD      B,$08           ; count eight bits.

;; FR-S-LPB
L1D51:  RRA                     ; rotate right.
        JR      C,L1D55         ; forward, with carry, to FR-S-RES.

        INC     E               ; increment the free sector count.

;; FR-S-RES
L1D55:  DJNZ    L1D51           ; loop back for all eight bits to FR-S-LPB.

        DEC     C               ; decrement byte count.
        JR      NZ,L1D4D        ; loop back for thirty two bytes to FR-SC-LP.

        RET                     ; return.

; --------------------
; THE 'PRNAME' ROUTINE
; --------------------
;   This routine outputs a ten character name, followed by a carriage return,
;   and is used by the CAT command to first print the cartridge name and then
;   the filenames on the cartridge.
;   Note. For a routine that can output to any stream, it seems straightforward
;   until one notices the call to TEMPS at the end.  This applies the permanent
;   colour screen attributes to the temporary set and has been placed within
;   the routine as a security measure to ensure that if the cartridge name
;   or filename contains a string of colour control codes that render filenames
;   invisible then their effect does not last beyond the current name.
;
;   On the other hand, colour control codes can be used in the cartridge name
;   without affecting the cartridge contents display.

;; PRNAME
L1D5B:  PUSH    BC              ; preserve name count.

        LD      B,$0A           ; ten characters per name.

;; PRNM-LP
L1D5E:  LD      A,(HL)          ; fetch a character.

        CALL    L1D71           ; routine PRCHAR

        INC     HL              ; point to next character.
        DJNZ    L1D5E           ; loop back for all ten to PRNM-LP

        LD      A,$0D           ; prepare a carriage return.
        CALL    L1D71           ; routine PRCHAR

        PUSH    HL              ; preserve character address.

        RST     10H             ; CALBAS
        DEFW    $0D4D           ; main TEMPS restores temporary colours from
                                ; the permanent colours after each name.
```

```
        POP     HL              ; restore character address.
        POP     BC              ; restore name count.

        RET                     ; return.


; --------------------
; THE 'PRCHAR' ROUTINE
; --------------------
;   The PRINT CHARACTER routine utilizes the output restart in the main ROM
;   which outputs to any stream and so a stream such as the "T" channel
;   could be sent output. The IX register has therefore to be preserved.

;; PRCHAR
L1D71:  PUSH    IX              ; preserve this ad hoc channel address.

        RST     10H             ; CALBAS
        DEFW    $0010           ; main PRINT-A

        POP     IX              ; restore this channel address.

        RET                     ; return.


; ---------------------------
; THE 'ERASE COMMAND' ROUTINE
; ---------------------------
; (Hook Code: $24)
;   The ERASE command is in two stages and uses the first 32 bytes of the
;   otherwise unused data buffer to map out the sectors to be marked clear.
;   The first stage performs this mapping and in one revolution of the tape
;   it should find all sectors that have the specified name.  It should also
;   find the EOF record, which all files have, and which contains in the
;   RECNUM field the maximum record number. For example with four records the
;   numbers will be 0, 1, 2, 3.
;   Once the number of marked records equals the max record plus one then the
;   second stage can begin which is to mark free all the records.
;
;   There are two circumstances under which the procedure might not be so
;   straightforward.
;   The first is if the user has pressed BREAK during a previous ERASE
;   operation after a few records were marked free.
;   The second is if the file has been saved with the System Variable COPIES
;   holding a value larger than 1. For example with a value of 5, there will
;   be five EOF records and five records with RECNUM equal to zero etc.
;
;   For the first case the command will make five revolutions of the tape
;   before marking all found sectors free.
;   This may happen in the second case also if more multi records were found
;   before the first EOF record was encountered.
;   It is more likely that the ERASE command will have to be invoked several
;   times to erase the file.  It is simpler to issue the command within a
;   loop.  Multiple copy files are usually saved as part of a well-considered
;   scheme and are seldom subsequently erased.

;; ERASE
L1D79:  CALL    L10A5           ; routine SET-T-MCH creates a temporary channel
                                ; using either an existing microdrive map from
```

```
                                ; a channel also using this drive or allocating
                                ; a new one initialized to $FF bytes.

        LD      A,(IX+$19)      ; fetch drive number from CHDRIV.
        CALL    L1532           ; routine SEL-DRIVE starts motor.

        IN      A,($EF)         ; read microdrive port.
        AND     $01             ; isolate 'write prot.' bit.
        JR      NZ,L1D8A        ; forward, if not zero, to ERASE-1

        RST     20H             ; Shadow Error Restart
        DEFB    $0E             ; Drive 'write' protected


;; ERASE-1
L1D8A:  PUSH    IX              ; transfer address of start of channel.
        POP     HL              ; to the HL register.

        LD      DE,$0052        ; prepare offset to data buffer.
        ADD     HL,DE           ; add to address start.

;   A pseudo microdrive map will also be created within the buffer conserving
;   memory.  This is initialized to $00 bytes.

        PUSH    HL              ; transfer buffer address
        POP     DE              ; from HL to DE register
        INC     DE              ; and increment address.

        LD      BC,$001F        ; set counter to 31 and B to zero.
        XOR     A               ; set A to zero.
        LD      (HL),A          ; insert zero in first location.
        LDIR                    ; copy to other 31 addresses

        LD      A,$FF           ; prepare, as a default, to examine every sector.
        LD      (IX+$0D),A      ; update CHREC with max record number.

        LD      BC,$04FB        ; prepare decimal 1275 (5+ revolutions)
        LD      ($5CC9),BC      ; update system variable SECTOR

;   Note. if the EOF record is not found, or if the number of found sectors
;   doesn't equal the maximum record then 5+ revolutions of the tape will
;   occur after which all mapped sectors will be erased. Normally with a
;   simple file it's all over in less than two revolutions.

;; ERASE-LP
L1DA7:  CALL    L13F7           ; routine DEC-SECT decrements the 1275 counter.
        JR      Z,L1E03         ; forward, if zero, to ERASE-MK

        CALL    L13A9           ; routine GET-M-HD2 reads the next 14-byte
                                ; header to pass the tape heads.

        CALL    L1E5E           ; routine G-RDES reads the corresponding
                                ; 14-byte record descriptor for this sector.

        JR      NZ,L1DE5        ; forward, with read error, to TST-NUM

;   now check if sector is in use.  Considered it so if next position is
;   at $0200 or if it is the EOF record.
```

```
        LD      A,(IX+$43)      ; RECFLG
        OR      (IX+$46)        ; RECLEN_hi
        AND     $02
        JR      NZ,L1DC3        ; forward, if in use, to ERASE-2
                                ; to consider for erasure.

; the sector is not used so reset the REAL microdrive map bit.

        CALL    L13E3           ; routine RES-B-MAP resets sector bit on
                                ; the REAL microdrive map.

        JR      L1DE5           ; forward to TST-NUM

; ---

;   consider for erasure if filename matches.

;; ERASE-2
L1DC3:  PUSH    IX              ; transfer channel base address
        POP     HL              ; to the HL register.

        LD      DE,$0047        ; offset to 10 characters of filename.
        ADD     HL,DE           ; add so HL addresses the start of RECNAM.
        LD      BC,$000A        ; ten bytes to compare against required CHNAME.

        CALL    L1403           ; routine CHK-NAME

        JR      NZ,L1DE5        ; forward, with no match, to TST-NUM

; the name matches so sector is marked free.

        CALL    L13EB           ; routine TEST-PMAP obtains address of sector
                                ; bit in HL and bit mask in B.

        LD      A,B             ; transfer mask to B
        OR      (HL)            ; combine with addressed byte
        LD      (HL),A          ; and update setting the sector bit.

        BIT     1,(IX+$43)      ; test RECFLG is this an EOF record.
        JR      Z,L1DE5         ; forward, if not, to TST-NUM

; All files should have an EOF record and, if this is it, then the endpoint
; can be reduced from $FF to record number plus one as range starts at 1.

        LD      A,(IX+$44)      ; fetch record number from RECNUM
        INC     A               ; increment as CHREC value starts at one not
                                ; zero.
        LD      (IX+$0D),A      ; update the endpoint CHREC

;; TST-NUM
L1DE5:  PUSH    IX              ; transfer the channel base address
        POP     HL              ; to the HL register.

        LD      DE,$0052        ; add offset to data
        ADD     HL,DE           ; to address the pseudomap.

        LD      E,$00           ; initialize E to zero.
        LD      C,$20           ; and C counter to thirty two.
```

```
;; LP-P-MAP
L1DF0:  LD      A,(HL)          ; fetch a byte from pseudomap
        INC     HL              ; and increment the address.

        LD      B,$08           ; set bit counter to eight.

;; LP-B-MAP
L1DF4:  RRA                     ; rotate end bit to carry.
        JR      NC,L1DF8        ; forward, with no carry, to NOINC-C

        INC     E               ; increment recno

;; NOINC-C
L1DF8:  DJNZ    L1DF4           ; back to LP-B-MAP for all eight bits.

        DEC     C               ; decrement byte counter.
        JR      NZ,L1DF0        ; back to LP-P-MAP for all 32 bytes.

; now E holds the number of records marked for erasure in range 1 to NR.

        LD      A,(IX+$0D)      ; fetch records to be erased from CHREC
        CP      E               ; compare to records marked for erasure.
        JR      NZ,L1DA7        ; back, if not exact match, to ERASE-LP

;   Now the second stage begins. Since the pseudomap has a representation of
;   all the records to be erased we can load the headers one by one, and
;   rewrite the corresponding records with a clear one in the channel.
;   The same record is written after all the appropriate headers. Fields
;   like RECNUM only have relevance when the record is in use.
;
;   First prepare a clear record descriptor.  The actual data buffer does not
;   have to be clear and in fact contains the pseudomap. Note also that the
;   checksum for the data need not be calculated but the checksum for the
;   record descriptor is required to be accurate.

;; ERASE-MK
L1E03:  CALL    L1E49           ; routine IN-CHK marks the channel record
                                ; descriptor fields as usable by blanking
                                ; both RECFLG and RECLEN and then inserting
                                ; the descriptor checksum.

; now enter a loop for all marked records.

;; ERASE-MK2
L1E06:  CALL    L13A9           ; routine GET-M-HD2 reads the next header
                                ; to pass the tape heads.
        CALL    L13EB           ; routine TEST-PMAP checks if the sector,
                                ; (in HDNUMB) is marked to be erased in the
                                ; pseudomap.
        JR      Z,L1E31         ; forward, if not, to T-OTHER

; this record is marked for erasure.

        PUSH    HL              ; save pseudomap sector bit address.
        PUSH    BC              ; save pseudomap bit mask which has one set bit.

        LD      A,$E6           ; enable writing.
        OUT     ($EF),A         ; output to microdrive port.
```

```
        LD      BC,$0168        ; set counter to 360 decimal.
        CALL    L1652           ; routine DELAY-BC pauses briefly as the
                                ; record now approaches the tape heads.

        PUSH    IX              ; transfer channel base address
        POP     HL              ; to the HL register pair.

        LD      DE,$0037        ; offset to record PREAMBLE.
        ADD     HL,DE           ; add to form start of save address.

        CALL    L15B3           ; routine OUT-M-BUF rewrites descriptor and
                                ; data buffer. The descriptor is checksummed,
                                ; the data is not.

        LD      A,$EE           ; disable writing
        OUT     ($EF),A         ; output to microdrive port

; now update bit the real microdrive map and the pseudomap.

        CALL    L13E3           ; routine RES-B-MAP resets appropriate bit
                                ; for the now free sector in the REAL
                                ; microdrive map.

        POP     BC              ; restore the pseudomap bit mask.
        POP     HL              ; restore the pseudomap sector bit address.

        LD      A,B             ; transfer bitmask to B.
        CPL                     ; the set bit is now reset and the other seven
                                ; bits are set.
        AND     (HL)            ; reset the bit in the pseudomap
        LD      (HL),A          ; and update.

; now check if there are any more sectors to do.

;; T-OTHER
L1E31:  PUSH    IX              ; transfer channel base address
        POP     HL              ; to the HL register.

        LD      DE,$0052        ; prepare offset to the pseudomap
        ADD     HL,DE           ; and add to address start of map.

        LD      B,$20           ; set byte count to thirty two.

;; CHK-W-MAP
L1E3A:  LD      A,(HL)          ; fetch a byte representing eight sectors.
        OR      A               ; test for zero.
        JR      NZ,L1E06        ; back, if a byte is not zero, to ERASE-MK2

        INC     HL              ; increment the map address
        DJNZ    L1E3A           ; loop back to CHK-W-MAP for all 32 bytes.

; at this point all records have been erased and it only remains to clear up.

        XOR     A               ; select no motor
        CALL    L1532           ; routine SEL-DRIVE stops the motor.

        CALL    L119F           ; routine DEL-M-BUF deletes the adhoc buffer.

        RET                     ; return.
```

```
; ---------------------------------
; THE 'PREPARE 'FREE SECTOR'' ROUTINE
; ---------------------------------
;   The two indicators within the current channel are marked clear and the
;   RECORD DESCRIPTOR is checksumed in preparation for writing to each sector
;   to be marked free.

;; IN-CHK
L1E49:  XOR     A               ; clear accumulator A.
        LD      (IX+$43),A      ; blank RECFLG.
        LD      (IX+$45),A      ; blank RECLEN_lo.
        LD      (IX+$46),A      ; blank RECLEN_hi.

        PUSH    IX              ; transfer the start of channel
        POP     HL              ; to the HL register pair.

        LD      DE,$0043        ; prepare the offset to RECFLG.
        ADD     HL,DE           ; add to form start of record descriptor.

        CALL    L1426           ; routine CHKS-HD-R inserts 14-byte checksum.

        RET                     ; return.

; -------------------------------------
; THE 'OBTAIN RECORD DESCRIPTOR' ROUTINE
; -------------------------------------
; This routine is used by CAT, ERASE and the GET-DESC Hook Code $33.
; It loads and verifies the 14 byte record descriptor from RECFLG to RECNAM.
; This is normally loaded with the following data block
; or with the header block.
; The Zero Flag is set upon successful completion.

;; G-RDES
L1E5E:  PUSH    IX              ; transfer channel address
        POP     HL              ; to HL register.

        LD      DE,$0043        ; offset to RECFLG
        ADD     HL,DE           ; add to form first receiving location.
        CALL    L15E2           ; routine GET-M-HD reads in 15 bytes.
        CALL    L1426           ; routine CHKS-HD-R checksums the first 14 bytes
        RET     NZ              ; return with checksum error.

        BIT     0,(IX+$43)      ; test bit 0 of RECFLAG - should be zero
        RET                     ; return.

; ----------------------
; THE 'HOOK-CODE' ROUTINE
; ----------------------
; This accesses the twenty six hook codes now reduced to the range $00 - $19.

;; HOOK-CODE
L1E71:  CP      $1A             ; compare to upper limit.
        JR      C,L1E77         ; forward, if valid, to CLR-ERR.

        RST     20H             ; Shadow Error Restart.
        DEFB    $12             ; Hook code error.
```

```
; ---

;; CLR-ERR
L1E77:  LD      (IY+$00),$FF    ; set ERR_NR to one less than zero - no error.

        SET     2,(IY+$01)      ; update FLAGS signal 'L' mode.

        INC     HL              ; step past the hook code location in RAM.
        EX      (SP),HL         ; make this the return address.
        PUSH    HL              ; push back what was at stack pointer - the
                                ; preserved value of AF on entry.

        ADD     A,A             ; double the code.
        LD      D,$00           ; set D to zero for indexing.
        LD      E,A             ; transfer the code to E.

        LD      HL,L1E99        ; address: HOOK-TAB the base of the Hook Codes.
        ADD     HL,DE           ; index into this table.

        LD      E,(HL)          ; low byte to E.
        INC     HL              ; increment pointer.
        LD      D,(HL)          ; high byte to D.

        POP     AF              ; restore AF from machine stack.

        LD      HL,L0700        ; push the address UNPAGE
        PUSH    HL              ; on the machine stack.

        EX      DE,HL           ; transfer address to HL.
        JP      (HL)            ; jump to Hook Code routine.


; ---------------------------
; THE 'HOOK CODE +32' ROUTINE
; ---------------------------
; (Hook Code: $32)
;   This allows the user to call any address in the shadow ROM.

;; HOOK-32
L1E94:  LD      HL,($5CED)      ; sv HD_11
        JP      (HL)            ; jump to routine.


; ---------------------------
; THE 'HOOK CODE +31' ROUTINE
; ---------------------------
; (Hook Code: $31)
; This Hook Code ensures that the extra System Variables are created. Since
; this has already occurred, as is the case with all Hook Codes, then all that
; remains to do is to return to the address on the stack - the UNPAGE location.

;; HOOK-31
L1E98:  RET                     ; return.


; ------------------------------
; THE 'HOOK CODE ADDRESSES' TABLE
; ------------------------------
;   The addresses of the Hook Codes.  The last two are new to this ROM.
```

```
;; HOOK-TAB
L1E99:  DEFW    L1ECD           ; $1B - CONS-IN
        DEFW    L1EE0           ; $1C - CONS-OUT
        DEFW    L0B88           ; $1D - BCHAN-IN
        DEFW    L0D07           ; $1E - BCHAN-OUT
        DEFW    L1EF0           ; $1F - PRT-OUT
        DEFW    L1EF5           ; $20 - KBD-TEST
        DEFW    L1532           ; $21 - SEL-DRIVE
        DEFW    L1B05           ; $22 - OP-TEMP-M
        DEFW    L138E           ; $23 - CLOSE-M2
        DEFW    L1D79           ; $24 - ERASE
        DEFW    L1EFD           ; $25 - READ-SEQ
        DEFW    L12DA           ; $26 - WR-RECD
        DEFW    L1F0B           ; $27 - RD-RANDOM
        DEFW    L1F3F           ; $28 - RD-SECTOR
        DEFW    L1F7A           ; $29 - RD-NEXT
        DEFW    L1F85           ; $2A - WR-SECTOR
        DEFW    L10A5           ; $2B - SET-T-MCH
        DEFW    L119F           ; $2C - DEL-M-BUF
        DEFW    L0F46           ; $2D - OP-TEMP-N
        DEFW    L1F18           ; $2E - CLOSE-NET
        DEFW    L1F25           ; $2F - GET-PACK
        DEFW    L0E4F           ; $30 - SEND-PACK
        DEFW    L1E98           ; $31 - HOOK-31
        DEFW    L1E94           ; $32 - HOOK-32

        DEFW    L1FE4           ; $33 - GET-DESC
        DEFW    L1FF6           ; $34 - OP-B-CHAN


; ---------------------------
; THE 'CONSOLE INPUT' ROUTINE
; ---------------------------
; (Hook Code: $1B)

;; CONS-IN
L1ECD:  EI                      ; enable interrupts.
        RES     5,(IY+$01)      ; update FLAGS signal no new key pressed.

;; WTKEY
L1ED2:  HALT                    ; wait for an interrupt.

        RST     10H             ; CALBAS
        DEFW    $02BF           ; main KEYBOARD

        BIT     5,(IY+$01)      ; test FLAGS - new key ?
        JR      Z,L1ED2         ; loop back, if not, to WTKEY

        LD      A,($5C08)       ; place decoded key in system variable LASTK
        RET                     ; return.


; ----------------------------
; THE 'CONSOLE OUTPUT' ROUTINE
; ----------------------------
; (Hook Code: $1C)
;   outputs a character to the unalterable system stream for the console.
```

```
;; CONS-OUT
L1EE0:  PUSH    AF              ; save character to be output.
        LD      A,$FE           ; use system stream $FE - upper screen.

; ->

;; OUT-CODE
L1EE3:  LD      HL,$5C8C        ; address system variable SCR_CT.
        LD      (HL),$FF        ; load with a high number to suppress scroll
                                ; prompt.

        RST     10H             ; CALBAS
        DEFW    $1601           ; main CHAN-OPEN opens selected stream.

        POP     AF              ; fetch the preserved print character.

        RST     10H             ; CALBAS
        DEFW    $0010           ; main PRINT-A prints character in accumulator.


        RET                     ; return.


; ----------------------------
; THE 'PRINTER OUTPUT' ROUTINE
; ----------------------------
; (Hook code: $1D)
;   outputs a character to stream 3.

;; PRT-OUT
L1EF0:  PUSH    AF              ; preserve character to be printed
        LD      A,$03           ; select stream 3
        JR      L1EE3           ; back to OUT-CODE


; ---------------------------
; THE 'KEYBOARD TEST' ROUTINE
; ---------------------------
; ( Hook Code: $20 )
; Normally a single reset bit in A determines which half row is read but by
; resetting all bits the entire keyboard is read.  A pressed key will cause
; a bit to be reset.  Routine returns with zero flag set if no keys pressed,
; NZ otherwise.

;; KBD-TEST
L1EF5:  XOR     A               ; reset all eight high-order bits.
        IN      A,($FE)         ; read the entire keyboard.
        AND     $1F             ; retain any unpressed keys - will be $1F if
                                ; no key.
        SUB     $1F             ; subtract to give zero if no keys.
        RET                     ; return.


; ------------------------------
; THE 'READ SEQUENTIAL' HOOK CODE
; ------------------------------
; (Hook Code: $25)
;
```

```
;; READ-SEQ
L1EFD:  BIT     1,(IX+$43)      ; RECFLG
        JR      Z,L1F08         ; forward to INCREC

        LD      (IY+$00),$07    ; set ERR_NR to '8 End of file'
        RST     28H             ; Error Main ROM

; ---

;; INCREC
L1F08:  INC     (IX+$0D)        ; increment the required record in CHREC
                                ; and continue into next routine...

; --------------------------
; THE 'READ RANDOM' HOOK CODE
; --------------------------
; (Hook Code: $27)
;   reads a PRINT record randomly.

;; RD-RANDOM
L1F0B:  CALL    L1252           ; routine GET-RECD gets the record specified
                                ; by CHREQ matching filename CHNAME from the
                                ; cartridge in the drive CHDRIV which is
                                ; started.

        BIT     2,(IX+$43)      ; test RECFLG - is it a PRINT type file.
        RET     Z               ; return if so.

        CALL    L119F           ; routine DEL-M-BUF reclaims the permanent
                                ; channel thus losing the buffer contents.

        RST     20H             ; Shadow Error Restart
        DEFB    $16             ; 'Wrong file type'

; -----------------------------------
; THE 'CLOSE NETWORK CHANNEL' HOOK CODE
; -----------------------------------
; (Hook Code: $2E)
; Hook Code Only

;; CLOSE-NET
L1F18:  CALL    L0FAE           ; routine SEND-NEOF

        PUSH    IX              ; pick up start address
        POP     HL              ; of the channel.

        LD      BC,$0114        ; bytes to reclaim.

        RST     10H             ; CALBAS.
        DEFW    $19E8           ; main RECLAIM-2.

        RET                     ; return.


; -----------------------------------
; THE 'GET PACKET FROM NETWORK' ROUTINE
; -----------------------------------
; (Hook Code: $2F)
```

```
;; GET-PACK
L1F25:  LD      A,($5CC6)       ; sv IOBORD
        OUT     ($FE),A         ;
        DI
        CALL    L0FD3           ; routine WT-SC-E
        JR      NC,L1F3A        ; forward to GP-ERROR

        CALL    L0EB5           ; routine GET-NBLK
        JR      NZ,L1F3A        ; forward to GP-ERROR

        EI                      ;
        AND     A               ;
        JP      L0D4D           ; jump to BORD-REST

; ---

;; GP-ERROR
L1F3A:  SCF                     ;
        EI                      ;
        JP      L0D4D           ; jump to BORD-REST


; ---------------------------
; THE 'READ SECTOR' HOOK CODE
; ---------------------------
; (Hook Code: $28)
; fetches header from sector specified by CHREC.
; If the sector is from a PRINT type file then it returns with success.
; Otherwise if a program or code file the data area is 'cleared'.

;; RD-SECTOR
L1F3F:  LD      HL,$00FF        ; ensure every sector is tried.
                                ; Note. was $F0 (240) in original ROM which
                                ; would not be compatible with emulators.
        LD      ($5CC9),HL      ; update temporary variable SECTOR

;; NO-GOOD
L1F45:  CALL    L13A9           ; routine GET-M-HD2 reads the next header
                                ; to pass the tape heads.

        LD      A,(IX+$29)      ; fetch sector number from HDNUMB
        CP      (IX+$0D)        ; compare with required sector in CHREC
        JR      Z,L1F57         ; forward, with match, to USE-C-RC

        CALL    L13F7           ; routine DEC-SECT decrements the counter.
        JR      NZ,L1F45        ; loop back, if not zero, to NO-GOOD

        RST     20H             ; Shadow Error Restart
        DEFB    $11             ; 'File not found'

; ---

;; USE-C-RC
L1F57:  PUSH    IX              ; transfer channel base address
        POP     HL              ; to the HL register.

        LD      DE,$0043        ; offset to RECFLG
        ADD     HL,DE           ; add to address start of record descriptor.
```

```
        CALL    L15EB           ; routine GET-M-BUF reads in the record
                                ; descriptor and the 512 bytes of data.

        CALL    L1426           ; routine CHKS-HD-R checksums the descriptor.

        JR      NZ,L1F75        ; forward, with error, to DEL-B-CT

        LD      DE,$000F        ; additional offset to data.
        ADD     HL,DE           ; add to address data.

        CALL    L142B           ; routine CHKS-BUFF checksums the data buffer.
        JR      NZ,L1F75        ; forward, with error, to DEL-B-CT

        OR      A               ; clear carry
        BIT     2,(IX+$43)      ; test RECFLG - is this a PRINT file ?
        RET     Z               ; return if so.

;; DEL-B-CT
L1F75:  CALL    L1FD4           ; routine CLR-BUFF sets descriptor and data
                                ; contents to same values.
        SCF                     ; signal error.
        RET                     ; return from hook-code.


; --------------------------------
; THE 'READ NEXT SECTOR' HOOK CODE
; --------------------------------
; (Hook Code: $29)
; This hook code just reads the next header to pass the tape head and then,
; without further qualification, reads the corresponding data using the
; routine above. If not a PRINT file then the data is cleared.
; It needlessly sets up a sector counter in the System Variable SECTOR.

;; RD-NEXT
L1F7A:  LD      HL,$00FF        ; set count to 255. Note. not used.
        LD      ($5CC9),HL      ; insert in system variable SECTOR.

        CALL    L13A9           ; routine GET-M-HD2 reads the next header
                                ; to pass the tape heads.

        JR      L1F57           ; back to USE-C-RC to read and validate the
                                ; corresponding descriptor and data.

; ----------------------------
; THE 'WRITE SECTOR' HOOK CODE
; ----------------------------
; (Hook Code: $2A)
; writes to microdrive the sector in CHREC.

;; WR-SECTOR
L1F85:  LD      HL,$00FF        ; set counter to ensure at least one revolution
        LD      ($5CC9),HL      ; of the tape and update SECTOR

        PUSH    IX              ; transfer base address
        POP     HL              ; of channel to HL.

        LD      DE,$0037        ; offset to header preamble
        ADD     HL,DE           ; add and
        PUSH    HL              ; preserve location on machine stack.
```

```
        LD      DE,$000C        ; offset past preamble to  RECFLG
        ADD     HL,DE           ; the start of the record descriptor.

        CALL    L1426           ; routine CHKS-HD-R insert checksum byte.

        LD      DE,$000F        ; 15 byte offset to start  of data.
        ADD     HL,DE           ; add to address first of 512 bytes.
        CALL    L142B           ; routine CHKS-BUFF inserts buffer checksum.

;; WR-S-1
L1FA1:  CALL    L13A9           ; routine GET-M-HD2 reads any header.
        LD      A,(IX+$29)      ; fetch sector from HDNUMB
        CP      (IX+$0D)        ; compare to required sector in CHREC
        JR      Z,L1FB3         ; forward, with match, to WR-S-2

        CALL    L13F7           ; routine DEC-SECT decrements the counter
        JR      NZ,L1FA1        ; back, if not zero, to WR-S-1

; else the header was not found after a complete tape revolution.

        RST     20H             ; Shadow Error Restart
        DEFB    $11             ; File not found

; ---

;; WR-S-2
L1FB3:  IN      A,($EF)         ; read microdrive port.
        AND     $01             ; isolate 'write prot.' bit.
        JR      NZ,L1FBB        ; forward, if not, to WR-S-3

        RST     20H             ; Shadow Error Restart
        DEFB    $0E             ; Drive 'write' protected

; ---

;; WR-S-3
L1FBB:  LD      A,$E6           ; enable writing
        OUT     ($EF),A         ; output to port.

        LD      BC,$0168        ; set delay to 360
        CALL    L1652           ; routine DELAY-BC pauses briefly as the
                                ; record now approaches the tape heads.

        POP     HL              ; restore pointer to RECFLG
        CALL    L15B3           ; routine OUT-M-BUF writes descriptor and
                                ; data buffer.

        LD      A,$EE           ; disable writing
        OUT     ($EF),A         ; output to port.

        CALL    L13C4           ; routine CHECK-MAP fetches bit mask for map
                                ; location addressed by HL into B register.

        LD      A,B             ; transfer mask to accumulator
        OR      (HL)            ; combine with any set bits already there.
        LD      (HL),A          ; update map marking sector used.

        RET                     ; return.
```

```
; ----------------------------------
; THE 'CLEAR BUFFER CONTENTS' ROUTINE
; ----------------------------------
; This routine sets the contents of the 14 byte record descriptor and
; the 512 byte data buffer to the same value so that they are unreadable.
; This is invoked when the possibility that a secret file, whose name begins
; with CHR$ 0 has been read.

;; CLR-BUFF
L1FD4:  PUSH    IX              ; transfer the channel base
        POP     HL              ; address to HL.

        LD      DE,$0028        ; offset to HDFLAG.
        ADD     HL,DE           ; add to base address.

        LD      D,H             ; transfer same
        LD      E,L             ; address to DE and
        INC     DE              ; make one higher.

        LD      BC,$0229        ; set counter to 553 bytes.
        LDIR                    ; fill with HDFLAG contents.

        RET                     ; return.

; --------------------------------------
; THE 'FETCH RECORD DESCRIPTOR' HOOK CODE
; --------------------------------------
; (Hook Code: $33)
; Note. new in this ROM.
; This Hook Code reads the next header and corresponding record descriptor
; returning with carry flag set with header mismatch or if the name starts
; with CHR$ 0 and should therefore be secret.

;; GET-DESC
L1FE4:  CALL    L13A9           ; routine GET-M-HD2 reads the next 14-byte
                                ; header to pass the tape heads.
        CALL    L1E5E           ; routine G-RDES reads the corresponding
                                ; 14-byte record descriptor for this sector.
        JR      NZ,L1FF1        ; forward, with checksum error, to NOT-RECV

; a valid header and matching descriptor has been read.

        LD      A,(IX+$47)      ; fetch first character of RECNAM.
        OR      A               ; test for CHR$ 0.
        RET     NZ              ; return if not a secret file.

; but if a secret file then ensure that the 14 descriptor bytes (read) and
; the 512 buffer bytes (not read) are cleared to the same value.

;; NOT-RECV
L1FF1:  CALL    L1FD4           ; routine CLR-BUFF (above).
        SCF                     ; signal error.
        RET                     ; return from hook code.

; --------------------------------
; THE 'OPEN "B" CHANNEL' HOOK CODE
; --------------------------------
; (Hook Code: $34)
```

```
; New in this ROM.

;; OP-B-CHAN
L1FF6:  LD      A,$42           ; letter "B"
        LD      ($5CD9),A       ; place in system variable L_STR1
        CALL    L0B17           ; routine OP-RS-CH opens an RS232 channel.
        RET                     ; return.

; ---

        DEFB    $FF             ; spare

; ---

.end

; -----------------------------
; THE 'SHADOW' SYSTEM VARIABLES
; -----------------------------
;
; X1  23734 $5CB6  FLAGS3                 ; IY+$7C - Flags
; X2  23735 $5CB7  VECTOR                 ; Address used to extend BASIC.
; X10 23737 $5CB9  SBRT                   ; 10 bytes of Z80 code to Page ROM.
; 2   23747 $5CC3  BAUD                   ; BAUD=(3500000/(26*baud rate)) -2
; 1   23749 $5CC5  NTSTAT                 ; Own network station number.
; 1   23750 $5CC6  IOBORD                 ; Border colour during I/O
; N2  23751 $5CC7  SER_FL                 ; 2 byte workspace used by RS232
; N2  23753 $5CC9  SECTOR                 ; 2 byte workspace used by Microdrive.
; N2  23755 $5CCB  CHADD_                 ; Temporary store for CH_ADD
; 1   23757 $5CCC  NTRESP                 ; Store for network response code.
; --  ----- -----  ------                 ; ---------------------------------
; 1   23758 $5CCD  NTDEST                 ;  Destination station number 0 - 64.
; 1   23759 $5CCE  NTSRCE                 ;  Source station number.
; X2  23760 $5CD0  NTNUMB                 ;  Network block number 0 - 65535
; N1  23762 $5CD2  NTTYPE                 ;  Header type block.
; X1  23763 $5CD3  NTLEN                  ;  Data block length 0 - 255.
; N1  23764 $5CD4  NTDCS                  ;  Data block checksum.
; N1  23765 $5CD5  NTHDS                  ;  Header block checksum.
; --  ----- -----  ------                 ; ---------------------------------
; N2  23766 $5CD6  D_STR1                 ; 2 byte drive number 1 - 8.
; N1  23768 $5CD8  S_STR1                 ; Stream number 1 - 15. [ also 0 ]
; N1  23769 $5CD9  L_STR1                 ; Device type "M", "N", "T" or "B"
; N2  23770 $5CDA  N-STR1                 ; Length of filename.
; N2  23772 $5CDC             (dynamic)   ; Address of filename.
; --  ----- -----  ------                 ; ---------------------------------
; N1  23774 $5CDE  D_STR2                 ; 2 byte drive   ; File type.
; N1  23775 $5CDF                         ; number.        ; Length of
; N1  23776 $5CE0  S_STR2                 ; Stream number. ; Data.
; N1  23777 $5CE1  L_STR2                 ; Device type.   ; Start of
; N1  23778 $5CE2  N-STR2                 ; Length of      ; data.   \
; N1  23779 $5CE3                         ; filename.      ; Program  \
; N1  23780 $5CE4             (dynamic)   ; Address of     ; length. ; Start of
; N1  23781 $5CE5             (dynamic)   ; filename       ;         ; data.
; --  ----- -----  ------                 ; ---------------------------------
; N1  23782 $5CE6  HD_00                  ; File type .      _
; N2  23783 $5CE7  HD_0B                  ; Length of data.   /\
; N2  23785 $5CE9  HD_0D                  ; Start of data.   /
; N2  23787 $5CEB  HD_0F                  ; Program length. /
; N2  23789 $5CED  HD_11                  ; Line number.
```

```
; --  ----- -----  ------               ; ---------------------------------
; 1   23791 $5CEF  COPIES               ; Number of copies made by SAVE.
; --  ----- -----  ------               ; ---------------------------------

; Note. the System Variables HD_00 to HD_11 take their names from their
; position in the standard audio tape header. The ten bytes HD_01 to HD_0A
; would be the tape filename and are not held within the above area.
; The area D_STR2 is multipurpose and sometimes the HD_?? variables are
; copied to this region and sometimes the D_STR1 variables are copied there.

; ---------------------------
; THE 'MICRODRIVE MAPS' FORMAT
; ---------------------------
;
; The creation of the extra system variables moves the start of CHANS up to
; address 23792.  It is at this location that the first of a possible eight
; Microdrive Maps will be created.  Each map is 32 bytes in size containing
; 256 bits for each possible sector and as each map is created, CHANS moves
; up by another 32 bytes.

; 10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
; 00000000 00000000 00000000 00000000 00010101 01010100 00000000 00000000
; 00000000 01100000 00000000 00000000 00000000 00000000 00000111 11111111
; 11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111

; Note. The continuous loop tape is formatted in such a way that sector $FE
; is written first and sector $01 is the last to be written. Sectors $00 and
; $FF are therefore always unavailable.  As there is only room for about 180
; sectors on a 100 foot long tape, the higher numbered sectors are later
; overwritten by the lower numbered sectors.
; Where the tape is spliced together one or two bad sectors will appear.
; When saving bytes there isn't enough time to copy the next 512 bytes from
; the program/code area to the buffer between sectors so a program or
; code/data block is written to alternating sectors as with the 3K example
; above.  As the tape cartridge fills up it becomes more difficult to find
; usable sectors and LOAD/SAVE operations take longer.
; A growing number of Spectrum emulators feature the microdrives and they
; usually make available all 254 sectors so a typical cartridge will hold 126
; Kilobytes compared to say 92 K on real hardware.
;
; During a LOAD operation the entire sector map is pushed on the machine stack
; and the microdrive map is used to map loaded records after which the previous
; map is 'popped' of the stack and reverts to mapping sectors again.
;
; ------------------------------
; THE 'STANDARD CHANNELS' FORMAT
; ------------------------------
; The twenty bytes of the standard channels as set up my Main ROM.
;
; CHANS                 $09F4           ; PRINT-OUT
;                       $10A8           ; KEY-INPUT
;                       $4B             ; 'K'
;
;                       $09F4           ; PRINT-OUT
;                       $15C4           ; REPORT-J
;                       $53             ; 'S'
;
;                       $0F81           ; ADD-CHAR
;                       $15C4           ; REPORT-J
```

```
;                       $52             ; 'R'
;
;                       $09F4           ; PRINT-OUT
;                       $15C4           ; REPORT-J
;                       $50             ; 'P'
;
;                       $80             ; End Marker

; ------------------------------
; THE 'MICRODRIVE CHANNEL' FORMAT
; ------------------------------
;
;   2 IX+$00              $0008           ; main ERROR-1
;   2 IX+$02              $0008           ; main ERROR-1
;   1 IX+$04              $CD             ; inverted or regular "M" character
;   2 IX+$05              $12B3           ; MCHAN-OUT
;   2 IX+$07              $11FD           ; M-INPUT
;   2 IX+$09              $0253           ; length of channel.
;   2 IX+$0B  CHBYTE      $0000           ; position of next byte rec'd/stored
;   1 IX+$0D  CHREC       $00             ; record number, also temporary sector
;  10 IX+$0E  CHNAME      "          "    ; filename with trailing spaces.
;   1 IX+$18  CHFLAG      %0000000x       ; bit 0 used
;   1 IX+$19  CHDRIV                      ; drive number 0 - 7.
;   2 IX+$1A  CHMAP                       ; address of MAP for this microdrive.
; ------------------------------------------------------------------------------
;  12 IX+$1C                              ; 12 bytes of header preamble
; ------------------------------------------------------------------------------
;   1 IX+$28  HDFLAG                      ; Flag byte.
;                                         ; bit 0 set indicates a header.
;   1 IX+$29  HDNUMB                      ; Sector number. [1-254]
;   2 IX+$2A                              ; Two unused bytes.
;  10 IX+$2C  HDNAME                      ; Cartridge name with trailing spaces.
;   1 IX+$36  HDCHK                       ; Header checksum.
; ------------------------------------------------------------------------------
;  12 IX+$37                              ; 12 bytes of data block preamble.
; ------------------------------------------------------------------------------
;   1 IX+$43  RECFLG                      ; Flag byte.
;                                         ; bit 0 reset indicates a record.
;                                         ; bit 1 reset no EOF, set EOF
;                                         ; bit 2 reset indicates a PRINT FILE
;   1 IX+$44  RECNUM                      ; Record number in the range 0-255
;   2 IX+$45  RECLEN                      ; Number of databytes in record 0-512.
;  10 IX+$47  RECNAM                      ; Filename with trailing spaces.
;   1 IX+$51  DESCHK                      ; Checksum of the preceding 14 bytes
; ------------------------------------------------------------------------------
; 512 IX+$52  CHDATA                      ; the 512 bytes of data.
;   1 +$0252  DCHK                        ; Checksum of preceding 512 bytes.

; ---------------------------
; THE 'NETWORK CHANNEL' FORMAT
; ---------------------------
;
;   2 IX+$00              $0008           ; main ERROR-1
;   2 IX+$02              $0008           ; main ERROR-1
;   1 IX+$04              $4E             ; "N" character
;   2 IX+$05              $0E09           ; NCHAN-OUT
;   2 IX+$07              $0DA9           ; N-INPUT
;   2 IX+$09              $0114           ; Length of channel 276 bytes.
;   1 IX+$0B NCIRIS                       ; The destination station number.
```

```
;   1 IX+$0C NCSELF                     ; This Spectrum's station number.
;   2 IX+$0D NCNUMB                     ; The block number.
;   1 IX+$0F NCTYPE                     ; The packet type code . 0 data, 1 EOF
;   1 IX+$10 NCOBL                      ; Number of bytes in data block.
;   1 IX+$11 NCDCS                      ; The data checksum.
;   1 IX+$12 NCHCS                      ; The header checksum.
;   1 IX+$13 NCCUR                      ; The position of last buffer char taken
;   1 IX+$14 NCIBL                      ; Number of bytes in the input buffer.
;   1 IX+$15 NCB                        ; A 255 byte data buffer.

; -----------------------------
; THE 'RS232 "T" CHANNEL' FORMAT
; -----------------------------
;
;   2 IX+$00              $0008         ; main ERROR-1
;   2 IX+$02              $0008         ; main ERROR-1
;   1 IX+$04              $54           ; "T" character
;   2 IX+$05              $0C3A         ; TCHAN-OUT
;   2 IX+$07              $0B76         ; T-INPUT
;   2 IX+$09              $000B         ; length of channel.

; -----------------------------
; THE 'RS232 "B" CHANNEL' FORMAT
; -----------------------------
; created by overwriting a "T" channel
;
;   2 IX+$00              $0008         ; main ERROR-1
;   2 IX+$02              $0008         ; main ERROR-1
;   1 IX+$04              $42           ; "B" character
;   2 IX+$05              $0D07         ; BCHAN-OUT
;   2 IX+$07              $0B7C         ; B-INPUT
;   2 IX+$09              $000B         ; length of channel.


; Acknowledgements
; ----------------
; Dr Ian Logan          for main ROM labels ( and half on Interface 1)
; Dr Frank O'Hara       for main ROM labels.
; Gianluca Carri        for Interface 1 v1.2 labels

; Credits
; -------
; Jonathan Needle       for requesting said labels on comp.sys.sinclair
;                       thereby kick-starting this project.
;                       Also for the Interface1-aware Spectaculator emulator
;                       and help with PORTS.
; Paul Dunn             for help with PORTS and the SPIN emulator.
; Chris Born            for documentation.
```