EDITING TOOLS

BLOCKS	Copy a range of disk blocks. Similar to CMOVE.
THRU	Load an inclusive range of blocks.
WIPE	Clear the currently edited screen. Put a zero in the first two bytes, indicating to the poly-FORTH printing routines that this block is empty.

Redefine all the poly-FORTH editing instructions in their respective lower cases letters. This will ease the use of FORTH editor to edit normal texts in blocks like this one.

THE SHORTEST CASE

This was the shortest and the most elegant CASE definition I came across. It uses the regular FORTH colon compiler to compile the cases.

It appeared in an article in Computer Design, April 21, 1983, by Whitney and Conrad. The earliest reference to this routine I could dig out was in 'An N-level File System' by Bill Bilobran, Proceedings 1980 FORML Conference, p. 210.

I was totally confused with all the case statements published in FORTH Dimensions sometimes ago, and I didn't feel the urge or the need to implement any of them. But this one I do like and I will recommend it to anybody who has to do N-way branchings.

ASCII DUMP

Poly-FORTH had a DUMP instruction to dump memory in integers. It is sometimes necessary to see the Ascii character printouts.

ASCII	Given the starting address, print the next 32 bytes in Ascii format. Leave a blank for control codes.
BYTES	Given the starting address, print the next 32 bytes in the 3 U.R format.
DUMP	Dump a range of memory in both the Ascii and byte formats.

150 LIST

```
( BLOCK COPY, 27-AUG-83, CHT)
: BLOCKS ( SOURCE DEST COUNT --- )
      EMPTY-BUFFERS
      OVER + SWAP DO    DUP I COPY UPDATE    1+    LOOP
      FLUSH DROP    ;
: THRU    ( BEGIN END --- )
      1+ SWAP DO    I LOAD    LOOP    ;
: t T ;    : p P ;    : l L ;    : u U ;    : d D ;    : n N ;
: f F ;    : x X ;    : m M ;    : list LIST ;
: copy COPY ;    : flush FLUSH ;
: i EDITOR I FORTH ;    : n N ;    : b B ;    : k K ;
: WIPE    SCR @ BLOCK DUP 1024 BLANK    0 SWAP !
      UPDATE    ;
: wipe WIPE ;
```

151 LIST

```
( CASE, AL WHITNEY & MARVIN C. CONRAD, COMPUTER DESIGN, )
( APRIL 21, 1983, P. 81.       CHT, 11-MAY-83             )
: :CASE    CREATE ]
           DOES>    SWAP 2* + @    2 +    EXECUTE    ;
: LEFT    ." turn left. "    ;
: RIGHT   ." turn right. "    ;
: STRAIGHT    ." go straight. " ;
:CASE    DIRECTION    LEFT RIGHT STRAIGHT    ;
0 DIRECTION
1 DIRECTION
2 DIRECTION
```

152 LIST

```
( DUAL DUMP , CHT, 28-APR-83)
: DELAY 20000 0 DO LOOP ;
: ASCII ( ADDRESS --- )    CR DUP 5 U.R    DELAY DELAY
        3 SPACES
        32 OVER + SWAP DO    I C@  DUP 32 < OVER 126 >
        OR IF DROP 46 THEN EMIT LOOP 3 SPACES ;
: BYTES ( ADDRESS --- )
        32 OVER + SWAP DO    I @ 5 U.R    2 +LOOP ;
: DUMP    ( ADDRESS # --- )
        OVER + SWAP DO  I ASCII    I BYTES    32 +LOOP ;
```

Poly-FORTH VLIST

Poly-FORTH does not have VLIST due to the hashed vocabulary structure. However, it is not difficult to implement it for those fig-FORTH addicts who cannot live without it.

ARM, ?TERMINAL ?TERMINAL has to be invented, too.
LINKS An array to store the link addresses of 8 threads.
INIT-LINKS Move the 8 vocabulary link address into LINKS.
NAMES Print the 8 names pointed to by the link addresses in LINKS.
REPLACE Change the link addresses in LINKS to the addresses contained in the link fields of the instructions pointed to by the link addresses in LINKS. 0 in the link field indicates end of chain.
?END Stop when ?TERMINAL is true or all links are zero.
VLIST Print instructions in 8 columns.

LISTING CP/M DISK

A number of years ago, I bought a fig-FORTH CP/M system on an 8" single density floppy disk. I didn't have a CP/M system to use it. (I still do not have one.) However, I was very interested in whatever was on that disk. It didn't take long to find out that my poly-FORTH could read the disk sector by sector. It was just a big puzzle to piece the sectors together in order to make some senses of the information. From the large assembly code file, I found the sector skewing sequence and built a transform table to convert the logical sector to physical sector. This way, one can print out a range of sectors in proper sequence.

COPYING LSI-11 BLOCKS

LSI-11 fig-FORTH puts blocks into a RT-11 file. After writing some 200 blocks of codes in this system, I wanted to move these blocks back to the regular poly-FORTH disk.

PHYSICAL Translate the logical sector number to the physical sector number in the RT-11 file.
MOVE-BLOCK Copy 8 sectors in a block of the fig-FORTH disk to the corresponding block in the poly-FORTH disk.
MOVE-DISK Copy a range of blocks from fig-FORTH disk to poly-FORTH disk.

153 LIST

```
( POLY-FORTH VLIST, CHT, 28-APR-83)
OCTAL
: ARM  1 177560 ! ;
: ?TERMINAL   177560 @ 200 AND ;
DECIMAL
VARIABLE LINKS 20 ALLOT
: INIT-LINKS CONTEXT LINKS 20 MOVE   ARM   ;
: NAMES   18 2 DO I LINKS + @    ?DUP IF   DUP 2 - C@   5 U.R
          SPACE 1- 3 TYPE   ELSE 9 SPACES   THEN   2 +LOOP ;
: REPLACE 18 2 DO I LINKS + @   ?DUP IF
          2+ @   I LINKS + ! THEN   2 +LOOP ;
: ?END    0   18 2 DO   I LINKS + @   OR   2 +LOOP  0=
          ?TERMINAL   OR   ;
: VLIST   INIT-LINKS   BEGIN   CR NAMES
          REPLACE   ?END END   ;
```

154 LIST

```
( LIST Z80-FORTH, 10/17/80)
( #SECTOR1 #SECTOR2 SECTORS      : PRINT FROM SEC1 TO SECT2 )
27 EMIT 51 EMIT 8 EMIT 16 EMIT 24 EMIT 32 EMIT 40 EMIT 48 EMIT
 0 EMIT
: DELAY  150 0 DO LOOP  ;
 : SEC 250 0 DO DELAY LOOP ;
: EMIT DUP 32 < OVER 126 > OR IF DROP 46 THEN EMIT ;
: SECT ( #BLOCK #SECT--> #BLOCK, 1 SECT = 128 BYTES) SEC
 128 * OVER BLOCK + 128 0 DO DUP I + C@ EMIT DELAY LOOP DROP ;
0 CONSTANT SEQU 6 , 12 , 18 , 24 , 4 , 10 , 16 , 22 , 2 , 8 , 14
 , 20 , 1 , 7 , 13 , 19 , 25 , 5 , 11 , 17 , 23 , 3 , 9 , 15 ,
 21 ,  ' SEQU CONSTANT SEQU
: SECTOR 2130 - 26 /MOD SWAP 2* SEQU + @ SWAP 26 * + 2130 + ;
: #BLOCK 8 /MOD SWAP ;
: SECTORS SWAP DO  CR I 4 U.R I SECTOR #BLOCK SECT LOOP ;
```

155 LIST

```
( RT-11 FIG-FORTH DISK COPY, CHT, 27-NOV-82)
: PHYSICAL ( LOGICAL-SECTOR --- PHYSICAL-SECTOR )
      26 /MOD SWAP    ( TRACK, SECTOR)
      2* DUP 25 >    IF 25 - THEN    ( OFFSET ODD SECTOR)
      OVER 1- 6 *    26 /MOD DROP + ( TRACK-OFFSET OF 6 SECTORS)
      26 /MOD DROP    ( TRACK, PHYSICAL SECTOR )
      SWAP 26 * +     ;
: MOVE-BLOCK    ( BLOCK# --- )
      8 0 DO    DUP 8 * I + ( LOGICAL )    2+ ( FIG-FORTH OFFSET)
                PHYSICAL
                8 /MOD 250 + BLOCK    ( DRIVE 1)
                SWAP 128 * +          ( SOURCE ADDR )
                OVER BLOCK    I 128 * +    ( DESTINATION)
                128 MOVE    UPDATE LOOP    DROP    ;
: MOVE-DISK ( LIM START --- )
      DO    I MOVE-BLOCK    LOOP    FLUSH    ;
```