Ohio Scientific FORTH Listing

This is a printout of the screens from the OSI version of fig-FORTH from Software Consultants. Sorry about the poor quality of the scan but this printout is from 1982/83ish and has faded a bit.

JAL 2015/07

```
 0 ****************** fig-FORTH  under OS-65U ******************
 1 (  ERROR MESSAGES  )
 2    2 DIT MESSAGES  )
 3 (  AUTO-LOAD SCREEN USING OBJ GET )
 4 (  AUTO-LOAD SCREEN USING LOAD )
 5 (  RESET AUTO-LOAD OBJ SCREENS  )
 6 (  TERMINAL  &  PRINTER  TOOLS  )
 7 (  TERMINAL  &  PRINTER  TOOLS  )
 8 (  TERMINAL  &  PRINTER  TOOLS  )
 9 (  LISTING WORDS SUPPORT  )
10 (  LIST, INDEX.  REQUIRES TERM & PRN TOOLS  )
11 (  TRIAD, VLIST.  REQUIRES TERM & PRINTER TOOLS )
12 (  ASSEMBLER  --  1 OF 6  )
13 (  ASSEMBLER  --  2 OF 6  )
14 (  ASSEMBLER  --  3 OF 6  )
15 (  ASSEMBLER  OPCODES  --  4 OF 6  )
16 (  ASSEMBLER  OPCODES  --  5 OF 6  )
17 (  ASSEMBLER  OPCODES  --  6 OF 6  )
18 (  DOUBLE PRECISION SUPPORT  )
19 (  DISK I/O SUPPORT.  REQUIRES DP SUPPORT  )
20 (  CASE STATEMENT  )
21 (  OS-65U DISK DIRECTORY.  REQUIRES DISK I/O SUPPORT  )
22 (  OS-65U DISK DIRECTORY  )
23 (  OS-65U DISK DIRECTORY  )
24 (  VIDEO EDITOR.  REQUIRES TERM & PRN TOOLS, CASE STMT  )
25 (  VIDEO EDITOR  )
26 (  VIDEO EDITOR  )
27 (  VIDEO EDITOR  )
28 (  VIDEO EDITOR  )
29 (  VIDEO EDITOR  )
30 (  VIDEO EDITOR  )
31 (  VIDEO EDITOR  )
32 (  VIDEO EDITOR  )
33 (  RANDOM NUMBER GENERATOR  )
34 (  #H, #D, .4H, S?   PRINT STACK CONTENTS  )
35 (  LOAD/UNLOAD ASSEMBLER  )
36 (  GET/PUT COMPILED CODE  )
37 (  GET/PUT COMPILED CODE  )
38 (  GET/PUT COMPILED CODE  )
39
40
41
42
43
44
45 (  MINIMAL EDITOR  )
46 (  MINIMAL EDITOR  )
47
48 (  FIND FILE IN DIRECTORY  )
49
50
51
52
53
```

```
 54
 55
 56
 57
 58
 59
 60     TOWERS OF HANOI,  REQUIRES TERM & PRN TOOLS   )
 61 (   TOWERS OF HANOI   )
 62 (   TOWERS OF HANOI   )
 63 (   TOWERS OF HANOI   )
 64 (   TOWERS OF HANOI   )
 65 (   TOWERS OF HANOI   )
 66 OBJ CODE, STARTS @ 7191 ; ENDS @ 7518 ; # SCRS 1
 67 OBJ CODE, STARTS @ 7518 ; ENDS @ 7989 ; # SCRS 1
 68 OBJ CODE, STARTS @ 24576 ; ENDS @ 25956 ; # SCRS 2
 69 OBJ CODE, STARTS @ 24576 ; ENDS @ 25956 ; # SCRS 2
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
100
```

```
SCR # 0
  0 ******************  fig-FORTH  under OS-65U ********************
  1 *                          REVISION 1.0                               *
  2 *                                                                     *
  3 *    The software contained on this disk is proprietary to:           *
  4 *                                                                     *
  5 *                       SOFTWARE CONSULTANTS                          *
  6 *                         7053 ROSE TRAIL                             *
  7 *                        MEMPHIS, TN  38134                           *
  8 *                          (901) 377-3503                             *
  9 *                                                                     *
 10 *                         fig-FORTH MODEL                             *
 11 *                      Through the courtesy of:                       *
 12 *                       FORTH INTEREST GROUP                          *
 13 *                          P. O. BOX 1105                             *
 14 *                       SAN CARLOS, CA  94070                         *
 15 ****************************************************************

SCR # 1
  0 (  ERROR MESSAGES  )
  1 EMPTY STACK
  2 DICTIONARY FULL
  3 HAS INCORRECT ADDRESS MODE
  4 ISN'T UNIQUE
  5 CANNOT LOAD OBJECT SCREEN
  6 DISC RANGE ?
  7 FULL STACK
  8 DISC ERROR !
  9
 10 NOT VALID FOR PRINTER!
 11 NOT ENOUGH ROOM BEFORE OS!
 12
 13
 14
 15

SCR # 2
  0    2 BIT MESSAGES  )
  1 OOOPILATION ONLY, USE IN DEFINITION
  2 EXECUTION ONLY
  3 CONDITIONALS NOT PAIRED
  4 DEFINITION NOT FINISHED
  5 IN PROTECTED DICTIONARY
  6 USE ONLY WHEN LOADING
  7 CANNOT EDIT OBJ SCREEN
  8 DECLARE VOCABULARY
  9
 10
 11
 12
 13
 14
 15
```

```
SCR # 3
( AUTO-LOAD SCREEN USING OBJ GET )
( RESET OBJ GET SCREENS WITH SCREEN 5 )

CR ." LOADING...." CR CR

  36 LOAD          ( COMPILED CODE GET/PUT )
  66 GET           ( TERMINAL & PRINTER TOOLS )
  67 GET           ( LIST, INDEX, TRIAD, & VLIST )
HEX 6000 HERE - ALLOT DECIMAL ( SKIP OVER OS )
  68 GET           ( CASE STATEMENT & VIDEO EDITOR )

: TASK ;

TOF 0 LIST CR CR       ;S



SCR # 4
( AUTO-LOAD SCREEN USING LOAD )

CR ." LOADING...." CR CR

  36 LOAD          ( COMPILED CODE GET/PUT )
   6 LOAD          ( TERMINAL & PRINTER TOOLS )
   9 LOAD          ( LIST, INDEX, TRIAD, & VLIST )
HEX 4000 HERE - ALLOT DECIMAL ( SKIP OVER OS )
  20 LOAD          ( CASE STATEMENT & VIDEO EDITOR )
  24 LOAD

: TASK ;

TOF 0 LIST CR CR       ;S



SCR # 5
( RESET AUTO-LOAD OBJ SCREENS )

FORTH DEFINITIONS DECIMAL
FORGET OP.LEN
36 LOAD
6 LOAD  66 PUT OUTDIST
9 LOAD  67 PUT OBJ?

HERE 8191 > 11 TERROR  ( CHECK IF INTO OS AREA )

HEX 6000 HERE - ALLOT DECIMAL

20 LOAD 24 LOAD  68 PUT DO-CASES

;S

```

```
SCR # 6
  0 (  TERMINAL  &  PRINTER  TOOLS  )
  1 FORTH   DEFINITIONS   HEX
  2
  3 2DA6  CONSTANT  OUTDIST        ( 65U OUTPUT DISTRIBUTOR )
  4   04  CONSTANT  PRNVAL         ( VALUE FOR PRINTER DEVICE )
  5   01  CONSTANT  CRTVAL         ( VALUE FOR TERMINAL )
  6   05  CONSTANT  OFS            ( DISPLAY OFFSET, LIST,INDEX )
  7
  8   00  VARIABLE  LINES      ( # OF LINES PRINTED THIS PAGE )
  9   37  CONSTANT  MAXLINES   ( MAX LINES PER PAGE IN HEX )
 10
 11 : PRN?         ( IS PRINTER ACTIVE?  LEAVES BOOLEAN )
 12      OUTDIST C@  PRNVAL  AND  ;
 13
 14 -->
 15

SCR # 7
  0 (  TERMINAL  &  PRINTER  TOOLS  )
  1
  2 : TOF        ( PRINTER TOP OF FORM, TERMINAL CLEAR SCREEN )
  3      PRN?  IF ( PRINTER )  OC EMIT  0 LINES !
  4      ELSE  ( CRT ) 0C EMIT  THEN  ;
  5                                                   1C 7E EMIT
  6 : NEWLINE              ( CR & HANDLE PAGING )
  7      CR  PRN?  IF ( PRINTER )  LINES @ 1+  DUP  MAXLINES  =
  8      IF ( EOP )  TOF  DROP  0  THEN  LINES !  THEN  ;
  9
 10 : CRT                      ( SET OUTPUT TO TERMINAL )
 11      CRTVAL  OUTDIST  C!  ;
 12
 13 : PRN                      ( SET OUTPUT TO PRINTER )
 14      PRNVAL  OUTDIST  C!  ;
 15 -->

SCR # 8
  0 (  TERMINAL  &  PRINTER  TOOLS  )
  1
  2 : ?PRN                   ( ERROR IF PRINTER ACTIVE )
  3      PRN?  IF ( ACTIVE )  CRT  CR  0A MESSAGE  QUIT  THEN  ;
  4
  5 : CURSOR                   ( POSITION CURSOR, COL-2, ROW-1 )
  6      ?PRN  ( NOT FOR PRN )  1A EMIT  EMIT  EMIT  ;
  7                             SWAP 11  7E  EMIT
  8 : FIRSTPRN      ( DO 1ST WHEN PRINTING, EITHER CRT OR PRN )
  9      PRN? IF ( PRINTER ) 0 LINES ! ELSE ( CRT ) CR CR THEN ;
 10
 11 : LASTPRN      ( DO LAST WHEN PRINTING, EITHER CRT OR PRN )
 12      PRN?  IF ( PRINTER )  TOF  THEN  CRT  ;
 13
 14 DECIMAL  ;S
 15
```

```
SCR # 9
  0 (  LISTING WORDS SUPPORT  )
  1
  2 FORTH DEFINITIONS DECIMAL
  3
  4 : OBJT      ( IS SCR # ON TOS OBJECT CODE )
  5      DUP  SCR  !  BLOCK  C@  32  <  ;
  6
  7 : OBJ.INFO      ( PRINT INFO ABOUT OBJECT SCREEN )
  8      ." OBJ CODE, STARTS @ "  SCR @  BLOCK  DUP  2+ @  .
  9      ." ; ENDS @ "  DUP  4 + @  .  ." ; # SCRS "  12 + @  .  ;
 10
 11 : OBJ.1ST.WORD      ( DISPLAY STARTING WORD OF SCR ON TOS )
 12      OBJ? IF  PREV @ 16 +  ID.  ELSE  ." NOT OBJECT CODE"
 13      THEN  CR  ;
 14 -->
 15

SCR # 10
  0 (  LIST, INDEX.  REQUIRES TERM & PRN TOOLS  )
  1
  2 : LIST      ( LIST SCREEN # ON STACK )
  3      BASE C@ SWAP  DECIMAL  CR  OFS SPACES  DUP  ." SCR # "
  4      .  OBJT  IF ( OBJ CODE )  OBJ.INFO  ELSE  16 0  DO
  5      CR  I  OFS 5 +  .R  SPACE  I  SCR @  .LINE  LOOP
  6      THEN  CR  BASE  C!  ;
  7
  8 : INDEX      ( PRINT FIRST LINE OF SCREENS FROM-2, TO-1 )
  9      FIRSTPRN  1+  SWAP  DO  NEWLINE  I  OFS 5 +  .R  SPACE
 10      I  OBJT  IF ( OBJ CODE)  OBJ.INFO  ELSE ( GOOD SCR )
 11      0 I .LINE  ?TERMINAL  IF ( BREAK )  LEAVE  THEN
 12      THEN  LOOP  CR  LASTPRN  CR  ;
 13 -->
 14
 15

SCR # 11
  0 (  TRIAD, VLIST, REQUIRES TERM & PRINTER TOOLS )
  1
  2 : TRIAD      ( PRINT 3 SCREENS ON PAGE, CONTAINING # ON STACK )
  3      FIRSTPRN  3 / 3 *  3 OVER +  SWAP  DO  I LIST  CR  LOOP
  4      LASTPRN  CR  ;
  5
  6 : VLIST   ( STANDARD VLIST )
  7      128 OUT !  CONTEXT @ @   BEGIN  OUT @ C/L >
  8      IF CR 0 OUT ! THEN  DUP ID. 2 SPACES  PFA LFA @
  9      DUP 0=  ?TERMINAL  OR  UNTIL DROP ;
 10
 11 ;S
 12
 13
 14
 15
```

```
SCR # 12
  0 (  ASSEMBLER  -  1 OF 6  )
  1 VOCABULARY  ASSEMBLER  HEX
  2 : CODE  [COMPILE] ASSEMBLER
  3           <BUILDS  -4 ALLOT  HERE  2+  ,  ;
  4 ' ASSEMBLFR CFA  ' ;CODE 0A + !   (  ;CODE SETS ASSEMBLER  )
  5 ASSEMBLER  DEFINITIONS   HEX
  6
  7 : ?RANGE   DUP   007F  >   OVER  FF80  <  OR  19 ?ERROR  ;
  8
  9 : BEGIN,   HERE  1  ;
 10 : END,    C,   1  ?PAIRS   HERE  1+  -  ?RANGE  C,   ;
 11
 12 : IF,    C,  HERE  1   1 ALLOT  ;
 13 : THEN,   1    ?PAIRS   HERE OVER  - 1 -  ?RANGE  SWAP  C!  ;
 14
 15 : NOT    20   XOR  ;   -->

SCR # 13
  0 (  ASSEMBLER  -  2 OF 6  )
  1   00  VARIABLE  MODE
  2   02  VARIABLE  BYTFS
  3   90  CONSTANT  CS
  4   B0  CONSTANT  CC
  5   D0  CONSTANT  0=
  6   10  CONSTANT  0<
  7   50  CONSTANT  VS
  8   70  CONSTANT  VC
  9 03A1  CONSTANT  PUT
 10 039F  CONSTANT  PUSH
 11 06B7  CONSTANT  PUSH0A
 12 04C7  CONSTANT  POP
 13 04C5  CONSTANT  POPTWO
 14 03D9  CONSTANT  SETUP
 15 03A6  CONSTANT  NEXT     00F8  CONSTANT N     -->

SCR # 14
  0 (  ASSEMBLER  -  3 OF 6  )
  1 : MODE+,      MOOE @  DUP  5 = IF  ROT  DUP
  2           >R ROT ROT R>  FF00   AND  IF  08 +  2 BYTES !  THEN
  3           THEN SWAP  C@   +  C,  ,  BYTES @  2 -  ALLOT
  4           05  MODE !   1 BYTES !  ;
  5
  6 : ADJUSTED   MODE @   9 =  IF  1 MODE !  THEN
  7          MODE @ 19 = OVER C@ A1 = AND  IF 10 MODE !  THEN  ;
  8
  9 : LABEL  HERE [COMPILE] ' !  ;
 10
 11 : 1OP0  <BUILDS  C,  DOES>            MODE+,  ;
 12 : 1OP1  <BUILDS  C,  DOES>  ADJUSTED  MODE+,  ;
 13 : 1OP2  <BUILDS  C,  DOES>            C@  C,  ;    -->
 14
 15
```

```
SCR # 15
 (  ASSEMBLER  OPCODES  -  4 OF 6  )
 60 :OPO ADC,      20 :OPO AND,      01 :OPO ASL,

 1F :OPO BIT,      00 :OP2 BRK,

 18 :OP2 CLC,      D8 :OP2 CLD,      58 :OP2 CLI,
 B8 :OP2 CLV,
 C0 :OPO CMP,      DF :OP1 CPX,      BF :OP1 CPY,

 C1 :OPO DEC,      CA :OP2 DEX,      88 :OP2 DEY,

 40 :OPO EOR,

 E1 :OPO INC,      E8 :OP2 INX,      C8 :OP2 INY,
 -->


SCR # 16
 (  ASSEMBLER  OPCODES  -  5 OF 6  )
 3F :OPO JMP,      5F :OPO JMP(,     13 :OPO JSR,

 A0 :OPO LDA,      A1 :OP1 LDX,      9F :OP1 LDY,
 41 :OPO LSR,
 EA :OP2 NOP,      00 :OPO ORA,

 48 :OP2 PHA,      08 :OP2 PHP,
 68 :OP2 PLA,      28 :OP2 PLP,

 21 :OPO ROL,      61 :OPO ROR,
 40 :OP2 RTI,      60 :OP2 RTS,

 E0 :OPO SBC,      38 :OP2 SEC,      F8 :OP2 SED,
 78 :OP2 SEI,      80 :OPO STA,
 81 :OPO STX,      7F :OPO STY,      -->

SCR # 17
 (  ASSEMBLER  OPCODES  -  6 OF 6  )
 AA :OP2 TAX,      A8 :OP2 TAY,      98 :OP2 TYA,
 BA :OP2 TSX,
 8A :OP2 TXA,      9A :OP2 TXS,

 : ,A  09 MODE !  0 BYTES !  0  ;
 : #   09 MODE !  1 BYTES !     ;
 : ,X  1D MODE !  2 BYTES !     ;
 : ,Y  19 MODE !  2 BYTES !     ;
 : X)  01 MODF !  1 BYTES !     ;
 : )Y  11 MODE !  1 BYTES !     ;
 : BOT 15 MODE !  1 BYTES !  0  ;
 : SEC     BOT              2+  ;
 : R       ,X     101 +         ;

 FORTH DEFINITIONS DECIMAL ;S
```

```
SCR # 18
( DOUBLE PRECISION SUPPORT  )

: D@           ( DP FETCH  1:ADDRESS, RETURNS DP CONTENTS )
    DUP  @  SWAP  2+  @  ;
: D!           ( DP STORE  2:DP #, 1:ADDRESS )
    SWAP  OVER  2+  !  !  ;
: D-           ( DP SUBTRACT  2:DP #, 1:DP #, RETURNS DP DIFF )
    DMINUS  D+  ;
: D=           ( DP EQUAL TEST  2:DP #, 1:DP #, RETURNS BOOLEAN )
    D-  OR  0=  ;
: D+!          ( DP PLUS STORE  2:DP #, 1:ADDR OF DP # )
    DUP  >R  D@  D+  R>  D!  ;

;S


SCR # 19
(  DISK I/O SUPPORT.  REQUIRES DP SUPPORT  )

FORTH  DEFINITIONS  HEX

26A1  CONSTANT  TCB      ( TRANSFER CONTROL BLOCK )

: DISK     ( DISK READ/WRITE.  4:R/W FLAG, 0 = WRITE, 1 = READ )
    TCB  1+  D!                       ( 3:LENGTH  2:MEMORY ADDR )
    TCB  7  +  !   TCB  5  +  !             ( 1:DP DISK ADDRESS )
    -DISC  0  ?ERROR  ;

DECIMAL  ;S




SCR # 20
(  CASE STATEMENT  )
: DO-CASES           ( START CASE STATEMENTS )
    ?COMP  CSP  @  !CSP  4  ;   IMMEDIATE

: CASE               ( TEST A CASE )
    4 ?PAIRS  COMPILE OVER  COMPILE =  COMPILE 0BRANCH
    HERE  0 ,  COMPILE DROP  5  ;   IMMEDIATE

: ESAC               ( END A CASE )
    5 ?PAIRS  COMPILE BRANCH  HERE  0 ,  SWAP  2
    [COMPILE] ENDIF  4  ;   IMMEDIATE

: CASES-DONE        ( END CASE STATEMENTS )
    4 ?PAIRS  COMPILE DROP  BEGIN  SP@  CSP @  =  0=
    WHILE  2  [COMPILE] ENDIF  REPEAT  CSP !  ;   IMMEDIATE
;S
```

```
SCR # 21
  0 (  OS-65U DISK DIRECTORY.  REQUIRES DISK I/O SUPPORT  )
  1 FORTH  DEFINITIONS  HEX
  2
  3 00 VARIABLE DIRADDR  2 ALLOT      ( NEXT DSK ADDR TO BE READ )
  4 00 VARIABLE DIRLIMIT 2 ALLOT      ( END OF DIRECTORY ADDR )
  5
  6 : DIRINIT        ( SET UP & READ FIRST DIR SECTOR )
  7       1 100 PAD 6200. DISK     ( GET 1ST SECTOR )
  8       6300.  DIRADDR  D!   PAD 1C + C@  100  *
  9       PAD 1D + @  6200.  D+  DIRLIMIT  D!  ;
 10
 11 : DIRNEXT       ( READ NEXT DIR SECTOR, RETURN 0 IF EOF )
 12       DIRADDR D@  DIRLIMIT D@  D=  IF ( EOF )  0 ( ERR )
 13       ELSE ( DO READ )  1  100  PAD  DIRADDR D@  DISK
 14       100.  DIRADDR  D+! ( BUMP ADDR )  1 ( NO ERR )  THEN  ;
 15 -->

SCR # 22
  0 (  OS-65U DISK DIRECTORY  )
  1
  2 : DIR.NAME,TYPE    ( PRINT NAME & TYPE FROM DIR  1:ADDR OF NAME )
  3       DUP  C@  1  =  IF ( DELETED )  ." *AVAIL"  12 SPACES
  4       ELSE ( FILE NAME )  6 0  DO  DUP  I +  C@  EMIT  LOOP
  5       4 SPACES  8 +  C@  DUP  0C AND  DO-CASES ( FILE TYPE )
  6       0  CASE  ." DATA "  ESAC   4  CASE  ." BASIC"  ESAC
  7       8  CASE  ." OTHER"  ESAC   CASES-DONE   4 SPACES
  8       3 AND  DO-CASES ( ACCESS RIGHTS )  0  CASE  ." NONE "  ESAC
  9       1  CASE  ." READ "  ESAC   2  CASE  ." WRITE"  ESAC
 10       3  CASE  ." R/W  "  ESAC   CASES-DONE   THEN  ;
 11 : DIR.ENTRY       ( PRINT ONE DIRECTORY ENTRY  1:OFFSET INTO PAD )
 12       PAD +  DUP  DIR.NAME,TYPE  8 +  DUP DUP  0 SWAP C!  D@
 13       0C  D.R ( DISK ADDR )  3 +  DUP  0 SWAP C!  D@
 14       0C  D.R ( LENGTH )  NEWLINE  ?TERMINAL  IF  QUIT  THEN  ;
 15 -->

SCR # 23
  0 (  OS-65U DISK DIRECTORY  )
  1
  2 : PRN.DIR.SECT      ( PRINT THIS DIRECTORY SECTOR  1:OFFSET )
  3       1 ( NOT LAST USED SECT FLAG )  SWAP  100 SWAP  DO
  4       I PAD +  C@  IF  I  DIR.ENTRY  ELSE ( DIR END )
  5       DROP  0 ( END OF DIR FLAG )  LEAVE  THEN  10  +LOOP ;
  6
  7 : DIR            ( PRINT OS-65U DISK DIRECTORY )
  8       FIRSTPRN  0D SPACES  ." OS-65U FILE DIRECTORY"  NEWLINE
  9       NEWLINE  ." NAME        TYPE     ACCESS      ADDRESS     "
 10       ." LENGTH"  NEWLINE  30 0  DO  ." -"  LOOP  NEWLINE
 11       DIRINIT  1  10  BEGIN  PRN.DIR.SECT  AND  WHILE
 12       DIRNEXT  0  REPEAT  LASTPRN  CR  ;
 13
 14 DECIMAL  ;S
 15
```

```
SCR # 24
( VIDEO EDITOR.  REQUIRES TERM & PRN TOOLS, CASE STMT  )

VOCABULARY EDITOR IMMEDIATE  EDITOR DEFINITIONS  DECIMAL

  00  VARIABLE  T#           ( TEMP STORAGE FOR R# )
  08  VARIABLE  TABAMT       ( # OF SPACES TO TAB )

: .CUR               ( PRINT CURSOR AT TOS )
     64 /MOD 2+  SWAP  OFS + 6 +  SWAP  CURSOR  ;

: .R#            ( PRINT CURSOR AT CURRENT POS )
     R#  @  .CUR  ;

: !R#                ( STORE CURSOR POSITION, STAY ON SCREEN )
     1024  +  1024  MOD  R#  !  ;
-->

SCR # 25
( VIDEO EDITOR  )

: +R#                ( ADD TOS TO PRESENT CURSOR POSITION )
     R#  @  +  !R#  ;

: +.R#              ( ADD TOS TO PCP, STORE IT, AND PRINT )
     +R#  .R#  ;

: R#->T#          ( MOVE R# TO T# )
     R#  @  T#  !  ;

: T#->R#         ( MOVE T# TO R# )
     T#  @  R#  !  ;

-->


SCR # 26
( VIDEO EDITOR  )

: SOL                 ( COMPUTE START OF LINE )
     R#  @  64  /  64  *  ;

: EOL                 ( COMPUTE END OF LINE )
     SOL  63  +  ;

: +LIN                 ( ADVANCE TO START OF NEXT LINE )
     EOL  1+  !R#  .R#  ;

: CURADR            ( GET MEMORY ADDRESS OF CURSOR POS )
     SCR  @  BLOCK  R#  @  +  ;

: !BLK                 ( STORE BYTE ON TOS @ CURSOR POSITION )
     CURADR  C!  UPDATE  1  +.R# ;   -->
```

```
SCR # 27
( VIDEO EDITOR )

: .LIN              ( REPRINT CURRENT LINE & PUT CURSOR BACK )
     SOL  DUP  .CUR  SCR @  BLOCK  +  64  TYPE  .R#  ;

: LINE             ( RETURNS ADDR OF START OF CURRENT LINE )
     SOL  SCR  @  BLOCK  +  ;

: ERASELINE         ( ERASE CURRENT LINE & DISPLAY )
     LINE  64  BLANKS  UPDATE  .LIN  ;

: MOVELINE          ( MOVE CURR LINE UP OR DOWN.  1:+64 OR -64 )
     LINE  SWAP  +R#  LINE  64  CMOVE  .LIN  ;

-->

SCR # 28
( VIDEO EDITOR )

: (EDIT)            ( EDIT SCREEN  2:START POS  1:SCREEN )
     DUP  OBJ?  23  ?ERROR  ( CHECK IF OBJ SCREEN )
     TOF  LIST  !R#  .R#  BEGIN ( PERMANENT LOOP )
     KEY  DO-CASES   13 CASE ( CR )  +LIN  ESAC
12 — 26 CASE ( UP )  -64  +.R#  ESAC
     11 CASE ( DOWN )  64  +.R#  ESAC
     08 CASE ( LEFT )  -1  +.R#  ESAC
16 — 24 CASE ( RIGHT )  1  +.R#  ESAC
     09 CASE ( TAB )  TABAMT @  +.R#  ESAC
20 — 18 CASE ( SHIFT TAB )  TABAMT @  MINUS  +.R#  ESAC
     05 CASE ( CTRL E )  ERASELINE  ESAC
     27 CASE ( ESC )  0 18 CURSOR  QUIT  ESAC

-->

SCR # 29
( VIDEO EDITOR )

     19 CASE ( CTRL S )  R#->T#  896  !R#  BEGIN  T# @  64  -
          R# @  <  WHILE  64 MOVELINE  -128 R# +!  REPEAT
          T#->R#  ERASELINE  ESAC
     04 CASE ( CTRL D )  R#->T#  64 R# +!  BEGIN  R# @
          1024  <  WHILE  -64 MOVELINE  128 R# +!  REPEAT
          960  !R#  ERASELINE  T#->R#  .R#  ESAC
     01 CASE ( CTRL A )  CURADR 1 -  R#->T#  EOL  1  -
          !R#  CURADR  DO  I C@  I 1+ C!  -1 +LOOP
          T#->R#  32  !BLK  -1 +R#  .LIN  ESAC
    127 CASE ( DELETE )  EOL  DUP  T# !  R# @  -  CURADR
           DUP  1+  SWAP  ROT  CMOVE SCR @ BLOCK  T# @  +
           32  SWAP  C!  UPDATE  .LIN  ESAC

-->
```

[illegible]

VC414H.

```
SCR # 28
  0 (  VIDEO EDITOR  )
  1
  2 : (EDIT)          ( EDIT SCREEN  2:START POS  1:SCREEN )
       DUP  OBJ?  23  ?ERROR  ( CHECK IF OBJ SCREEN )
       TOF  LIST  !R#  .R#  BEGIN ( PERMANENT LOOP )
       KEY DUP 126 = IF KEY THEN       DROP
       DO-CASES   13 CASE ( CR )  +LIN  ESAC
       12 CASE ( UP )  -64  +.R#  ESAC
       11 CASE ( DOWN )  64  +.R#  ESAC
       08 CASE ( LEFT )  -1  +.R#  ESAC
 10    16 CASE ( RIGHT )  1  +.R#  ESAC
 11    09 CASE ( TAB )  TABAMT @  +.R#  ESAC
 12    20 CASE ( SHIFT TAB )  TABAMT @  MINUS  +.R#  ESAC
 13    05 CASE ( CTRL E )  ERASELINE  ESAC
 14    27 CASE ( ESC )  0 18 CURSOR  QUIT  ESAC
 15 -->
```

MODIFIED FOR VOLKER CRAIG TERMINAL

JOHN LOCKETT.

```
SCR # 30
( VIDEO EDITOR )

      14            N
      ~~18~~ CASE ( CTRL ~~P~~ ) LINE PAD 64 CMOVE ESAC
      15 CASE ( CTRL O ) PAD LINE 64 CMOVE
         .LIN UPDATE ESAC
      02
      ~~20~~ CASE ( CTRL ~~T~~B ) 0 !R# .R# ESAC
      ( DEFAULT ) DUP 32 < OVER 126 > OR IF ( INVALID )
      7 EMIT ELSE ( VALID ) DUP EMIT !BLK 0 THEN
      CASES-DONE AGAIN ;

: N                ( EDIT NEXT SCREEN )
      0 SCR @ 1+ (EDIT) ;

: S                ( EDIT SAME SCREEN AGAIN )
      0 SCR @ (EDIT) ;
-->

SCR # 31
( VIDEO EDITOR )

: P                ( EDIT PREVIOUS SCREEN )
      0 SCR @ 1 - (EDIT) ;

: CLEAR            ( CLEAR SCREEN ON TOS )
      BLOCK 1024 BLANKS UPDATE ;

: COPY             ( SCREEN COPY 2:FROM 1:TO )
      SWAP BLOCK SWAP BLOCK 1024 CMOVE UPDATE ;

-->





SCR # 32
( VIDEO EDITOR )

FORTH DEFINITIONS

: EDIT          ( EDIT SCREEN ON TOS )
      0 SWAP [COMPILE] EDITOR EDITOR (EDIT) ;

: WHERE            ( GO TO EDIT MODE, DISPLAY COMPILE ERROR )
      0 ROT HERE C@ - ROT [COMPILE] EDITOR
      EDITOR (EDIT) ;

FORTH DEFINITIONS

;S


```

```
SCR # 33
  0 (  RANDOM NUMBER GENERATOR  )
  1
  2 BASE  C@ ( SAVE PRESENT BASE )  DECIMAL
  3
  4   99  VARIABLE  LASTRND
  5
  6 : RANDOM      ( GET RANDOM NUMBER,  2:LOW LIMIT  1:HI LIMIT )
  7      OVER  -  1+  LASTRND  @  99  *  ABS  DUP
  8      LASTRND  !  8  /  SWAP  MOD  +  ;
  9
 10 BASE  C! ( SET BACK TO ORIGINAL BASE )
 11
 12 ;S
 13
 14
 15

SCR # 34
  0 (  #H, #D, .4H, ST   PRINT STACK CONTENTS  )
  1 FORTH DEFINITIONS HEX
  2 E0  CONSTANT  S0
  3 : #H       ( MASK PRINT HEX DIGIT )
  4      10 M/MOD ROT 9 OVER < IF 7 + THEN 30 + HOLD ;
  5 : #D       ( MASK PRINT DECIMAL DIGIT )
  6      0A M/MOD ROT 9 OVER < IF 7 + THEN 30 + HOLD ;
  7 : .4H      ( PRINT 4 HEX DIGITS )
  8      0 <# #H #H #H #H #> TYPE SPACE ;
  9
 10 : ST       ( PRINT STACK CONTENTS )
 11      CR S0 SP@ - 2 - 2 / -DUP CR IF ." STACK   HEX   DEC" 0 DO
 12      CR SP@ I 2 * + @ ." S" I 1+ 0 <# #D #D #> TYPE
 13      ." =  " DUP .4H BASE @ DECIMAL SWAP 0 5 D.R BASE ! LOOP
 14      ELSE ." NOTHING ON STACK." THEN CR CR ;
 15 DECIMAL ;S

SCR # 35
  0 (  LOAD/UNLOAD ASSEMBLER  )
  1 FORTH  DEFINITIONS  HEX
  2 1  QUIET  !       HERE ( SAVE FOR LINE 4 )
  3 A800  HERE  -  ALLOT  ( PUT ASSEM IN HIGH MEMORY )
  4 ( HERE )  VARIABLE  SV.HERE
  5 VOC-LINK @  VARIABLE SV.VOC-LINK
  6
  7 DECIMAL  12  LOAD  ( ASSEMBLER )
  8
  9 : KILL.ASSM      ( REMOVE ASSEMBLER FROM DICTIONARY )
 10      ' SV.HERE  LFA  @  SV.HERE @  !  TRAVERSE  1+  !
 11      SV.VOC-LINK @  VOC-LINK  !  [COMPILE] FORTH  ;
 12
 13 SV.HERE @  DP  !  ( PUT DICTIONARY POINTER BACK )
 14
 15 0  QUIET  !   DECIMAL  ;S
```

```
SCR # 36
( GFT/PUT COMPILED CODE )
FORTH DEFINITIONS DECIMAL

1010  CONSTANT  GP.LEN  ( AMT OF SCREEN USED FOR CODE )
   0  CONSTANT  GP.LOC  ( PSUEDO-CONSTANT POINTER TO BUFFER )

: GP.BLK      ( GET A BLOCK, SAVE BUFFER ADDRESS )
     BLOCK  ' GP.LOC  !  ;

: GP.R#+      ( UPDATE MEMORY ADDRESS )
     GP.LEN  R*  +!  ;

: GP.#SCR     ( COMPUTE # OF SCREENS REQUIRED )
     PAD 4 + @  PAD 2+ @  -  ( PAD+2 = START, PAD+4 = END )
     GP.LEN  /MOD  SWAP  IF  1+  THEN  ;
-->

SCR # 37
( GFT/PUT COMPILED CODE )
( HEADER INFO CONTAINED AT PAD.  PUT ON ALL SCREENS )
( PAD    = SCREEN TYPE FLAG, 0 FOR COMPILED CODE )
( PAD+2  = STARTING ADDRESS    PAD+4  = ENDING ADDRESS )
( PAD+6  = CURRENT             PAD+8  = CONTEXT )
( PAD+10 = VOC-LINK            PAD+12 = # OF SCREENS )

: MPUT      ( PUT MEMORY TO DISK )
     DUP  PAD 2+ @  R*  !  GP.#SCR  DUP  PAD 12 +  !  +  SWAP
     DO  I  BUFFER  PAD  OVER  14  CMOVE  14  +  R* @  SWAP
     GP.LEN  CMOVE  GP.R*+  UPDATE  LOOP  ;

: MGET      ( GET MEMORY FROM DISK )
     DUP  DUP  GP.BLK  GP.LOC 2+ @  R*  !  GP.LOC 12 + @  +
     SWAP  DO  I  GP.BLK  GP.LOC 14 +  R* @  GP.LEN  CMOVE
     GP.R*+  LOOP  ;      -->

SCR # 38
( GET/PUT COMPILED CODE )
: GP.SETUP      ( SET PARAMETERS IN PAD, FOLLOWED BY WORD NAME )
     0  PAD  !  [COMPILE] '  NFA  PAD 2+  !  HERE  PAD 4 +  !
     LATEST  PAD 6 + !  CONTEXT @ @  PAD 8 +  !
     VOC-LINK @  PAD 10 +  !  ;

: SCRST         ( DISPLAY # OF SCREENS REQUIRED,  FBWN )
     GP.SETUP  GP.#SCR  CR  .  ." SCREENS REQUIRED'  CR  ;

: PUT           ( PUT COMPILED CODE TO DISK,  1:SCR#, FBWN )
     GP.SETUP  MPUT  FLUSH  ;

: GET           ( GET COMPILED CODE FROM DISK,  1:SCR # )
     MGET  GP.LOC 6 + @  CURRENT @  !  GP.LOC 10 + @  VOC-LINK
     !  GP.LOC 4 + @  DP  !  GP.LOC 8 + @  CONTEXT @  !  ;
;S
```

```
SCR # 45
  0 (  MINIMAL EDITOR  )
  1 FORTH  DEFINITIONS   HEX
  2 : TEXT   HERE  C/L 1+  BLANKS  WORD  HERE PAD C/L 1+ CMOVE ;
  3 : LINE   DUP FFF0 AND 17 ?ERROR SCR @ (LINE) DROP ;
  4 VOCABULARY  EDITOR  IMMEDIATE   HEX
  5 : WHERE   DUP B/SCR / DUP SCR ! ." SCR # " DECIMAL . SWAP C/L
  6    /MOD C/L * ROT BLOCK + CR C/L TYPE CR HERE C@ - SPACES
  7    .! "' [COMPILE] EDITOR QUIT ;
  8 EDITOR  DEFINITIONS
  9 : -MOVE   LINE C/L CMOVE UPDATE ;
 10 : H   LINE PAD 1+ C/L DUP PAD C! CMOVE ;
 11 : E   LINE C/L BLANKS UPDATE ;
 12 : S   DUP 1 - 0F DO I LINE I 1+ -MOVE -1 +LOOP E ;
 13 : D   DUP H 0F DUP ROT DO I 1+ LINE 1 -MOVE LOOP E ;
 14 : L   SCR @ LIST ;
 15 : R   PAD 1+ SWAP -MOVE ;      -->

SCR # 46
  0 (  MINIMAL EDITOR  )
  1 : P   1 TEXT R ;
  2 : I   DUP S R ;
  3 : CLEAR   SCR ! 10 0 DO FORTH I EDITOR E LOOP ;
  4 : COPY   B/SCR * SWAP B/SCR * B/SCR OVER + SWAP DO DUP FORTH I
  5      BLOCK 2 - ! 1+ UPDATE LOOP DROP FLUSH ;
  6
  7 FORTH  DEFINITIONS  DECIMAL  ;S
  8
  9
 10
 11
 12
 13
 14
 15
```

```
SCR # 48
 ( FIND FILE IN DIRECTORY )
 HEX
 : FINDFL      ( RETURNS 3:DP ADDRESS, 2:DP LENGTH, 1:BOOLEAN )
      DIRINIT  1  10  BEGIN   ( FILE NAME @ HERE )
      1 SWAP 100 SWAP DO I PAD + C@ IF  I PAD + >R R @ HERE 1+ @
      = R 2+ @ HERE 3 + @ = R> 4 + @ HERE 5 + @ = AND AND IF DROP
      DROP  I PAD + 8 + DUP DUP 0 SWAP C! D@ ( DK ADDR )
      ROT 3 + DUP 0 SWAP C! D@ ( LENGTH ) 1 ( BOOLEAN )  0
      2 LEAVE  THEN ELSE DROP 0 SWAP 0 LEAVE THEN 10 +LOOP AND
      WHILE DIRNEXT 0 REPEAT ;

 : TEST CR ." FILE NAME: " QUERY 0 WORD FINDFL ;
 DECIMAL ;S



```

```
SCR # 60
( TOWERS OF HANOI.  REQUIRES TERM & PRN TOOLS  )

FORTH  DEFINITIONS  DECIMAL

: MYSELF  ( IN DEFINITION, A RECURSIVE USE OF A NEW WORD )
     LATEST PFA CFA , ;  IMMEDIATE
  00  CONSTANT  LI
: 2DROP  DROP DROP ;
: PICK  2 * SP@ + @ ;
: 4DUP  4 PICK  4 PICK  4 PICK  4 PICK  ;
  12  CONSTANT  NMAX  ( MAX NUMBER OF RINGS )
NMAX  VARIABLE  (N)     : N  (N) @ ;  ( FORMERLY A CONSTANT )
  43  CONSTANT  COLOR  ( + )
  00  VARIABLE  RING  N 2 - ALLOT  ( ARRAY 1..N )
: 2DUP  OVER OVER  ;
-->

SCR # 61
(  TOWERS OF HANOI  )
: DELAY  ( IDEALLY, CENTISECONDS DELAY )
     0 DO 12 0 DO 127 127 * DROP LOOP LOOP ;
: POS     ( LOCATION POS -> CO-ORDINATE )
     2 N * 1+ * N + ;
: HALFDISPLAY  ( COLOR SIZE HALFDISPLAY )
     0 DO  DUP EMIT  LOOP  DROP ;
: <DISPLAY>     ( LINE COLOR SIZE <DISPLAY> )
     2DUP HALFDISPLAY  ROT 3 <  IF BL ELSE 124 ( V LINE )
     THEN EMIT  HALFDISPLAY  ;
: DISPLAY        ( SIZE POS LINE COLOR DISPLAY )
     SWAP >R ROT ROT OVER - R ( COLOR SIZE POS-SIZE LINE )
     CURSOR  R> ( COLOR SIZE LINE ) ROT ROT <DISPLAY> ;
-->



SCR # 62
(  TOWERS OF HANOI  )
: PRESENCE         ( TOWER RING PRESENCE -> BOOLEAN )
     RING + C@ = ;
: LINE             ( TOWER LINE -> DISPLAY LINE OF TOP )
     4 SWAP N 0 DO DUP I PRESENCE 0= ROT + SWAP LOOP DROP ;
: 1-  1 - ;

: RAISE            ( SIZE TOWER RAISE )
     DUP  POS SWAP LINE 1 SWAP DO
     2DUP I BL DISPLAY  2DUP I 1-  COLOR DISPLAY
     -1  +LOOP  2DROP  ;
: LOWER            ( SIZE TOWER LOWER )
     DUP POS SWAP LINE 1+ 2 DO
     2DUP  I 1-  BL DISPLAY  2DUP I COLOR DISPLAY
     LOOP  2DROP  ;
-->
```

```
SCR # 63
( TOWERS OF HANOI )
: MOVELEFT          ( SIZE SOURCE-TOWER DESTINY-TOWER MOVELEFT )
     POS  1- SWAP POS 1- DO DUP R 1+ I BL DISPLAY
     DUP R 1 COLOR DISPLAY -1  +LOOP DROP ;
: MOVERIGHT         ( SIZE SOURCE-TOWER DESTINY-TOWER MOVERIGHT )
     POS 1+  SWAP POS 1+ DO DUP R 1- 1 BL DISPLAY
     DUP R 1 COLOR DISPLAY  LOOP DROP ;
: TRAVERSE          ( SIZE SOURCE-TOWER DESTINY-TOWER TRAVERSE )
     2DUP >  IF MOVELEFT ELSE MOVERIGHT THEN ;
: MOVE              ( SIZE SOURCE-TOWER DESTINY-TOWER MOVE 0 )
     ?TERMINAL  IF 0 N 4 + CURSOR QUIT THEN
     ROT ROT 2DUP RAISE  >R 2DUP R> ROT TRAVERSE
     2DUP RING + 1- C!  SWAP LOWER ;
-->


SCR # 64
( TOWERS OF HANOI )
: MULTIMOV           ( SIZE SOURCE DESTINY SPARE MULTIMOV )
     4 PICK 1 = IF DROP MOVE ELSE
     >R >R SWAP 1- SWAP R> R>  4DUP SWAP MYSELF
     4DUP DROP  ROT 1+ ROT ROT  MOVE
     ROT ROT SWAP  MYSELF  THEN  ;

: MAKETOWER          ( TOWER MAKETOWER )
     POS 4  N + 3 DO DUP I CURSOR  124 EMIT ( | )  LOOP DROP ;
: MAKEBASE           ( NO ARGUEMENTS )
     0 N 4 + CURSOR N 6 * 3 + 0 DO 45 EMIT ( - ) LOOP ;
: MAKERING           ( TOWER SIZE MAKERING )
     2DUP RING + 1- C!  SWAP LOWER ;
-->


SCR # 65
( TOWERS OF HANOI )
: SETUP   ( NO ARGUEMENTS )
     TOF N 1+ 0 DO 1 RING I + C!  LOOP  3 0 DO I MAKETOWER LOOP
     MAKEBASE 0 N DO 0 I MAKERING -1 +LOOP ;
: TOWERS          ( QUANTITY TOWERS )
     ABS 1 MAX NMAX MIN (N) !
     SETUP  N 2 0 1
     OVER POS  N 4 + CURSOR  N 0 DO  7 EMIT  50 DELAY  LOOP
     ROT MULTIMOV
     0 22 CURSOR ;

;S
```

```
10 REM *** BASIC EXECUTIVE ***
20 REM %NET 6/79
30 IF PEEK(13316)=1 THEN POKE 13404,44: REM OMIT CD-23 FAULT CLEAR
40 REM
50 REM SETUP CONSOLE INPUT DEVICE
60 I% = PEEK (11664)
70 REM
80 REM  LOCKUP SYSTEM
90 POKE2888,0:FLAG 21: REM  INPUT ESCAPE
100 POKE 2073,96: POKE 14639,0: REM  CTRL C,O
110 POKE 11668, 2^(I%-1)
120 REM
130 REM SETUP CONSOLE OUTPUT DEVICE
140 J% = PEEK (11665)
150 POKE 11686, 2^(J%-1)
160 REM
170 PRINT: PRINT
179 FR=INT(10*PEEK(11946)/56+.5)/10
180 PRINT "09-65U V1.2 -";FR;"MHz"
185 PRINT "10/24/79"
190 REM
200 REM SETUP FOR MASTER HARD DISK SYSTEM
210 IF PEEK(9832) < 8 THEN POKE 61448,0: POKE 61439,0
220 REM
230 REM  GET USER FUNCTION
240 REM
242 REM  NETWORK SUPPORT PARTITION USER # %NET
243 NS = 1
245 IF (PEEK(14948)=76)AND(PEEK(55381)=NS) GOTO 300:REM %NET
290 REM
300 REM  UNLOCK SYSTEM
310 REM
330 POKE 14639,255: POKE 2073,76: REM  CTRL C,O
335 PRINT: PRINT "SYSTEM OPEN"
340 REM
350 PRINT: PRINT PEEK(132)+PEEK(133)*256-24576; "BYTES FREE": PRINT
355 IF (PEEK(14948)=76) AND (PEEK(55381)=NS) THEN RUN"NETWRK": REM %NE
400 RUN"FORTH"
```

```
20 REM            FORTH 1 FIG-FORTH UNDER OS-65U
40 REM               WRITTEN BY LARRY HINSLEY
60 REM
80 REM          This program is proprietary to and
100 REM            considered a trade secret of:
110 REM
120 REM                  Software Consultants
140 REM                    7053 Rose Trail
160 REM                  Memphis, TN 38134
180 REM                    (901) 377-3503
200 REM
240 REM     7936 BYTES OF MACHINE LANGUAGE BEFORE THIS PROGRAM
260 REM
280 REM ..... roll basic out to disk .....
300 POKE8778,192:POKE8779,36:POKE9432,243:POKE9433,40
320 POKE9435,232:POKE9436,40:CB=9889
340 CLOSE:OPEN"BASIC","PASS",1
360 POKECB+1,0:FORI=1TO3:T(I)=PEEK(9906+I)
370 POKECB+I+1,T(I):NEXTI:REM disk address (save for later)
380 POKECB+5,0:POKECB+6,2:REM save basic pages 0 & 1
400 POKECB+7,0:POKECB+8,71:REM swap buffer at $4700
420 T=USR(1):IFT<>0GOTO2000
440 T=PEEK(CB+2)+2:IFT<256GOTO540:REM bump disk addr 2 pages
460 T=PEEK(CB+3)+1:IFT<256GOTO500
480 T=PEEK(CB+4)+1:POKECB+4,T:T=0
500 POKECB+3,T
520 T=PEEK(CB+2)-254
540 POKECB+2,T
560 POKECB+5,0:POKECB+6,32:REM save rest of basic
580 POKECB+7,0:POKECB+8,2:REM start at $0200
600 T=USR(1):IFT<>0GOTO2000
1000 REM ..... set up & run forth .....
1020 CLOSE:OPEN"SCREEN",1:CLOSE
1040 POKE0,0:FORI=1TO3:T=PEEK(9906+I):POKEI,T:NEXTI
1050 FORI=0TO4:POKEI+4,T(I):NEXTI
1060 T=255:POKE14633,T:REM KILL CTRL O
1080 POKE14684,T:POKE14646,T:POKE14725,T:REM KILL INDIRECT FILE
1100 POKE14677,T:POKE14689,T:POKE14721,T:REM  "          "          "
1120 POKE8778,0:POKE8779,126:T=USR(0)
```