FigForth 1.1#

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  0
00 ******  fig-FORTH  MODEL  ******
01
02      Through the courtesy of
03
04
05       FORTH INTEREST GROUP
06          P. O. BOX 1105
07       SAN CARLOS, CA. 94070
08
09        Implemented on the
0A          ATARI  800/400
0B                by
0C           Steve Calfee
0D              1/26/81
0E
0F          Copywrite 1981
10
11             RELEASE 1
12       WITH COMPILER SECURITY
13                AND
14       VARIABLE LENGTH NAMES
15
16
17
18
19      Further distribution must
1A      include the above notice.
1B
1C
1D
1E
1F


****** fig-FORTH MODEL ****** Through the courtesy of FORTH
INTEREST GROUP P. O. BOX 1105 SAN CARLOS,
CA. 94070 Implemented on the ATARI 800/400 by Steve Calfee
1/26/81 Copywrite 1981 RELEASE 1 WITH COMPILER SECURITY AND
VARIABLE LENGTH
NAMES Further distribution must include the above notice.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  1
00 ******  fig-FORTH  MODEL  ******
01
02      Through the courtesy of
03
04
05       FORTH INTEREST GROUP
06          P. O. BOX 1105
07       SAN CARLOS, CA. 94070
08
09        Implemented on the
```

```
0A          ATARI  800/400
0B                by
0C           Steve Calfee
0D              1/26/81
0E
0F          Copywrite 1981
10
11              RELEASE 1
12       WITH COMPILER SECURITY
13                AND
14       VARIABLE LENGTH NAMES
15
16
17
18
19      Further distribution must
1A      include the above notice.
1B
1C
1D
1E
1F
```

****** fig-FORTH MODEL ****** Through the courtesy of FORTH INTEREST GROUP P. O. BOX 1105 SAN CARLOS, CA. 94070 Implemented on the ATARI 800/400 by Steve Calfee 1/26/81 Copywrite 1981 RELEASE 1 WITH COMPILER SECURITY AND VARIABLE LENGTH NAMES Further distribution must include the above notice.

* *

SCR # 2

```
00 BREAK Abort.
01
02 IOCB already open.
03
04 Non-existant device.
05
06 IOCB is write-only.
07
08 Invalid command (for this device
09 )
0A Device or file not open.
0B
0C Bad IOCB #
0D
0E IOCB is read-only
0F
10 End Of File
11
12 Truncated Record
13
14 Device Timeout
15
16 Device NAK (Negative AcKnowledge
17 )
18 Serial Bus input framing error
```

19
1A Cursor out of range
1B
1C Serial Bus data-frame overrun
1D
1E Serial Bus data-frame checksum e
1F rror.

BREAK Abort. IOCB already open. Non-existant device. IOCB is write-only. Invalid command (for this device) Device or file not open. Bad IOCB # IOCB is read-only End Of File Truncated Record Device Timeout Device NAK (Negative AcKnowledge) Serial Bus input framing error Cursor out of range Serial Bus data-frame overrun Serial Bus data-frame checksum error.

* *

SCR # 3
00 Device-done error
01
02 Read-after-write compare error
03
04 Function not implemented in hand
05 ler
06 Insufficient RAM
07
08
09
0A
0B
0C
0D
0E
0F
10
11
12
13
14
15
16
17
18
19
1A
1B
1C
1D
1E
1F

Device-done error Read-after-write compare error Function not implemented in handler Insufficient RAM

* *

SCR # 4

```
00 (  ERROR MESSAGES  )   135  159
01  9  8  7  10  ;S
02 empty stack
03
04 dictionary full
05
06 has incorrect address mode
07
08 isn't unique
09
0A
0B
0C disc range ??
0D
0E full stack !
0F
10 disc error !
11
12
13
14
15
16
17
18
19
1A
1B
1C
1D
1E
1F


(  ERROR MESSAGES  )
135 159 9 8 7 10 ;S empty stack dictionary full has incorrect
address mode isn't unique disc range ?? full stack ! disc error
!

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  5
00 (  ERROR MESSAGES  )
01
02 compilation only, use in definit
03 ion
04 execution only
05
06 conditionals not paired
07
08 definition not finished
09
0A in protected dictionary
0B
0C use only when loading
0D
0E off current editing screen
0F
10 declare vocabulary
```

```
11
12 outside allocated file space
13
14 writing off current line
15
16
17
18
19
1A string stack empty !!
1B
1C
1D
1E
1F


(  ERROR MESSAGES  )
compilation only,
use in definition execution only conditionals not paired
definition not finished in protected dictionary use only when
loading off current editing screen declare vocabulary outside
allocated file space writing off current line string stack
empty !!

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  6
00 (  TARGET COMPILER ERROR MESSAGE
01 S                 WFR-79JUN02 )
02
03
04 below lower bound of virtual mem
05 ory
06 disc compiler assembly error in
07 mode of
08 can't find in TARGET
09
0A target redef.
0B
0C T: error, is it paired with T;
0D ?
0E above virtual memory bounds
0F
10
11
12
13
14
15
16
17
18
19
1A
1B
1C
1D
1E
```

1F

(TARGET COMPILER ERROR MESSAGES   WFR-79JUN02)
below lower bound of virtual memory disc compiler assembly
error in mode of can't find in TARGET target redef. T: error,
is it paired with T; ? above virtual memory bounds

* *

SCR # 7
00 (<UNUSED>) ;S
01
02
03
04
05
06
07
08
09
0A
0B
0C
0D
0E
0F
10
11
12
13
14
15
16
17
18
19
1A
1B
1C
1D
1E
1F

(<UNUSED>)
;S

* *

SCR # 8
00 (<UNUSED>) ;S
01
02
03
04
05
06
07

```
08
09
0A
0B
0C
0D
0E
0F
10
11
12
13
14
15
16
17
18
19
1A
1B
1C
1D
1E
1F


( <UNUSED> )
;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  9
00 (  compile assembler
01    and editor  SRC 7/6/80   )
02 BASE  @  ( PRESERVE THE RADIX )
03
04 DECIMAL  31  WIDTH  !
05
06 13 LOAD ( ASSEMBLER )
07
08 HEX 1E LOAD ( DECUS FORTH ADDS)
09
0A HEX 15 LOAD   ( EDITOR )
0B
0C FORTH  DEFINITIONS
0D
0E 25 CONSTANT LPWORDS
0F
10 27 CONSTANT FORMY
11 : SAVENFAs ( MOVE FORTH NFAS TO
12 ORIGIN AREA ) #LINKS 0 DO
13 ' FORTH 4 + I 4 * + @
14 22 I 2* + +ORIGIN ! LOOP ;
15 DECIMAL
16 HERE  28 +ORIGIN ! ( FENCE )
17
18 HERE    30  +ORIGIN  !  ( DP )
19
1A HERE  FENCE  !
```

```
1B 1 WARNING ! ( DISK WARNINGS )
1C SAVENFAs : TASK  ;
1D BASE !
1E ;S
1F


(  compile assembler                and editor  SRC 7/6/80   )
BASE @ ( PRESERVE THE RADIX )
DECIMAL 31 WIDTH ! 13 LOAD
( ASSEMBLER )
HEX 1E LOAD
( DECUS FORTH ADDS)
HEX 15 LOAD
( EDITOR )
FORTH DEFINITIONS 25 CONSTANT LPWORDS
27 CONSTANT FORMY
: SAVENFAs
        ( MOVE FORTH NFAS TO ORIGIN AREA )
        #LINKS 0
        DO
          ' FORTH 4 + I 4 * + @ 22 I 2* + +ORIGIN !
        LOOP
        ;
DECIMAL HERE 28 +ORIGIN ! ( FENCE )
HERE 30 +ORIGIN ! ( DP )
HERE FENCE ! 1 WARNING ! ( DISK WARNINGS )
SAVENFAs
: TASK
        ;
BASE ! ;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  A
00 ( <UNUSED> ) ;S
01
02
03
04
05
06
07
08
09
0A
0B
0C
0D
0E
0F
10
11
12
13
14
15
16
17
```

```
18
19
1A
1B
1C
1D
1E
1F


( <UNUSED> )
;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  B
00 ( CLONING WORDS  7/21/80-SRC )
01  HEX FORTH DEFINITIONS
02 : COPYDISK DR0  4E 0 DO I I 0800
03   + EDITOR COPY FORTH LOOP ;
04 : CLONE DR0 0 PHYSOFF !
05   4E B + 0 DO I I 0800 +
06   EDITOR COPY FORTH LOOP DR0 ;
07 : 1.2TO1.3 DR0 8 OFFSET ! 4E 0
08   DO I  I 0803 + EDITOR COPY
09   FORTH LOOP DR0 ;
0A : 1.3TO1.4 DR0 B PHYSOFF ! 4E 0
0B   DO I I 0801 + COPY LOOP DR0 ;
0C
0D : OBJ  DR0 0 PHYSOFF !
0E   C 0 DO I I 0800 + EDITOR
0F   COPY FORTH LOOP DR0 ;
10   CR
11  ." INSERT SRCE DISK IN DRIVE 1
12  " CR
13  ." INSERT DEST DISK IN DRIVE 2"
14  CR
15  ." TYPE CLONE TO COPY ALL OF IT
16 " CR ." INCLUDING BOOT PROGRAM"
17   CR ." TYPE COPYDISK TO COPY"
18  CR ." SCREENS 0 TO 4E"
19   CR ." TYPE OBJ  TO COPY JUST"
1A   CR ." THE BOOTSTRAP B BLOCKS"
1B CR ." TYPE 1.2TO1.3 TO COPY "
1C CR ." OR TYPE 1.3TO1.4 TO COPY"
1D CR ." YOUR OLD DISK SOURCES "
1E CR ." TO THE NEW VERSION "
1F CR  ;S


( CLONING WORDS  7/21/80-SRC )
HEX FORTH DEFINITIONS
: COPYDISK
        DR0 4E 0
        DO
          I I 0800 + EDITOR COPY FORTH
        LOOP
        ;
: CLONE
```

```
        DR0 0 PHYSOFF ! 4E B + 0
        DO
          I I 0800 + EDITOR COPY FORTH
        LOOP
        DR0
        ;
: 1.2TO1.3
        DR0 8 OFFSET ! 4E 0
        DO
          I I 0803 + EDITOR COPY FORTH
        LOOP
        DR0
        ;
: 1.3TO1.4
        DR0 B PHYSOFF ! 4E 0
        DO
          I I 0801 + COPY
        LOOP
        DR0
        ;
: OBJ
        DR0 0 PHYSOFF ! C 0
        DO
          I I 0800 + EDITOR COPY FORTH
        LOOP
        DR0
        ;
CR ." INSERT SRCE DISK IN DRIVE 1 " CR ." INSERT DEST DISK IN
DRIVE 2" CR ." TYPE CLONE TO COPY ALL OF IT" CR ." INCLUDING
BOOT PROGRAM" CR ." TYPE COPYDISK TO COPY" CR ." SCREENS 0 TO
4E" CR ." TYPE OBJ TO COPY JUST" CR ." THE BOOTSTRAP B BLOCKS"
CR ." TYPE 1.2TO1.3 TO COPY " CR ." OR TYPE 1.3TO1.4 TO COPY"
CR ." YOUR OLD DISK SOURCES " CR ." TO THE NEW VERSION " CR ;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  C
00 ( <UNUSED> ) ;S
01
02
03
04
05
06
07
08
09
0A
0B
0C
0D
0E
0F
10
11
12
13
14
15
```

```
16
17
18
19
1A
1B
1C
1D
1E
1F


( <UNUSED> )
;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  D
00 (  FORTH-65 ASSEMBLER
01                   WFR-79JUN03 )
02 HEX
03
04 VOCABULARY  ASSEMBLER  IMMEDIATE
05      ASSEMBLER  DEFINITIONS
06
07
08 (  LOCATE EXISTING REGISTERS )
09
0A FF CONSTANT  XSAVE     0FB  CONS
0B TANT  W    0FD  CONSTANT  UP
0C F8  CONSTANT  IP          F0  CO
0D NSTANT  N
0E
0F
10 (  LOCATE EXISTING CODE PROCEEDU
11 RES )
12 '  (DO)  0E  +    CONSTANT  POP
13    ( FROM COMPUTATION STACK *)
14 '  (DO)  0C  +    CONSTANT  POPT
15 WO
16 '  LIT  13  +  CONSTANT  PUT
17
18 '  LIT  11  +  CONSTANT  PUSH
19
1A '  LIT  18  +  CONSTANT  NEXT
1B
1C '  EXECUTE  NFA  11  -  CONSTANT
1D   SETUP
1E     -->
1F


(  FORTH-65 ASSEMBLER                              WFR-79JUN03 )
HEX VOCABULARY ASSEMBLER IMMEDIATE ASSEMBLER DEFINITIONS
(  LOCATE EXISTING REGISTERS )
FF CONSTANT XSAVE
0FB CONSTANT W
0FD CONSTANT UP
F8 CONSTANT IP
```

```
F0 CONSTANT N
(  LOCATE EXISTING CODE PROCEEDURES )
' (DO) 0E + CONSTANT POP
( FROM COMPUTATION STACK *)
' (DO) 0C + CONSTANT POPTWO
' LIT 13 + CONSTANT PUT
' LIT 11 + CONSTANT PUSH
' LIT 18 + CONSTANT NEXT
' EXECUTE NFA 11 - CONSTANT SETUP
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  E
00 (     ASSEMBLER, CONT.
01                    WFR-780CT03 )
02 0   VARIABLE  INDEX       -2  AL
03 LOT
04 0909 , 1505 , 0115 , 8011 , 8009
05  , 1D0D , 8019 , 8080 ,
06 0080 , 1404 , 8014 , 8080 , 8080
07  , 1C0C , 801C , 2C80 ,
08
09
0A 2  VARIABLE  MODE
0B
0C : .A   0 MODE ! ;    : #    1 MO
0D DE ! ;    : MEM  2 MODE ! ;
0E : ,X   3 MODE ! ;    : ,Y   4 MO
0F DE ! ;    : X)   5 MODE ! ;
10 : )Y   6 MODE ! ;    : )    F MO
11 DE ! ;
12
13
14 : BOT       ,X    0  ;     ( ADD
15 RESS THE BOTTOM OF THE STACK *)
16 : SEC       ,X    2  ;        (
17 ADDRESS SECOND ITEM ON STACK *)
18 : RP)       ,X  101  ;      ( AD
19 DRESS BOTTOM OF RETURN STACK *)
1A -->
1B
1C
1D
1E
1F


(     ASSEMBLER, CONT.                          WFR-780CT03 )
0 VARIABLE INDEX
-2 ALLOT 0909 , 1505 , 0115 , 8011 , 8009 , 1D0D , 8019 , 8080
, 0080 , 1404 , 8014 , 8080 , 8080 , 1C0C , 801C , 2C80 , 2
VARIABLE MODE
: .A
        0 MODE !
        ;
: #
        1 MODE !
        ;
```

```
: MEM
        2 MODE !
        ;
: ,X
        3 MODE !
        ;
: ,Y
        4 MODE !
        ;
: X)
        5 MODE !
        ;
: )Y
        6 MODE !
        ;
: )
        F MODE !
        ;
: BOT
        ,X 0
        ;
( ADDRESS THE BOTTOM OF THE STACK *)
: SEC
        ,X 2
        ;
( ADDRESS SECOND ITEM ON STACK *)
: RP)
        ,X 101
        ;
( ADDRESS BOTTOM OF RETURN STACK *)
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  F
00 (  UPMODE,  CPU
01                    WFR-78OCT23 )
02
03
04 : UPMODE   IF  MODE  C@  8  AND
05  0=  IF  8  MODE +! ENDIF ENDIF
06      1 MODE C@  0F AND  -DUP  IF
07   0  DO  DUP + LOOP  ENDIF
08      OVER  1+  @  AND  0=  ;
09
0A
0B
0C : CPU   <BUILDS  C,  DOES>  C@
0D  C,  MEM  ;
0E      00 CPU BRK,   18 CPU CLC,
0F  D8 CPU CLD,  58 CPU CLI,
10      B8 CPU CLV,   CA CPU DEX,
11  88 CPU DEY,  E8 CPU INX,
12      C8 CPU INY,   EA CPU NOP,
13  48 CPU PHA,  08 CPU PHP,
14      68 CPU PLA,   28 CPU PLP,
15  40 CPU RTI,  60 CPU RTS,
16      38 CPU SEC,   F8 CPU SED,
17  78 CPU SEI,  AA CPU TAX,
```

```
18      A8 CPU TAY,   BA CPU TSX,
19  8A CPU TXA,  9A CPU TXS,
1A      98 CPU TYA,
1B
1C -->
1D
1E
1F


(  UPMODE,  CPU                                      WFR-78OCT23 )
: UPMODE
        IF
          MODE C@ 8 AND 0=
          IF
            8 MODE +!
          ENDIF
        ENDIF
        1 MODE C@ 0F AND -DUP
        IF
          0
          DO
            DUP +
          LOOP
        ENDIF
        OVER 1+ @ AND 0=
        ;
: CPU
        <BUILDS C, DOES> C@ C, MEM
        ;
00 CPU BRK,
18 CPU CLC,
D8 CPU CLD,
58 CPU CLI,
B8 CPU CLV,
CA CPU DEX,
88 CPU DEY,
E8 CPU INX,
C8 CPU INY,
EA CPU NOP,
48 CPU PHA,
08 CPU PHP,
68 CPU PLA,
28 CPU PLP,
40 CPU RTI,
60 CPU RTS,
38 CPU SEC,
F8 CPU SED,
78 CPU SEI,
AA CPU TAX,
A8 CPU TAY,
BA CPU TSX,
8A CPU TXA,
9A CPU TXS,
98 CPU TYA,
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
SCR #  10
00 (  M/CPU,   MULTI-MODE OP-CODES
01                  WFR-79MAR26 )
02 : M/CPU   <BUILDS  C,  ,  DOES>
03
04        DUP  1+  C@  80  AND  IF
05  10  MODE  +!  ENDIF  OVER
06        FF00  AND  UPMODE  UPMODE
07   IF  MEM  CR  LATEST  ID.
08        3  ERROR  ENDIF  C@  MODE
09   C@
0A        INDEX  +  C@  +  C,  MODE
0B   C@  7  AND  IF  MODE  C@
0C        0F  AND  7  <  IF  C,  EL
0D SE  ,  ENDIF  ENDIF  MEM  ;
0E
0F
10    1C6E 60 M/CPU ADC,  1C6E 20 M
11 /CPU AND,  1C6E C0 M/CPU CMP,
12    1C6E 40 M/CPU EOR,  1C6E A0 M
13 /CPU LDA,  1C6E 00 M/CPU ORA,
14    1C6E E0 M/CPU SBC,  1C6C 80 M
15 /CPU STA,  0D0D 01 M/CPU ASL,
16    0C0C C1 M/CPU DEC,  0C0C E1 M
17 /CPU INC,  0D0D 41 M/CPU LSR,
18    0D0D 21 M/CPU ROL,  0D0D 61 M
19 /CPU ROR,  0414 81 M/CPU STX,
1A    0486 E0 M/CPU CPX,  0486 C0 M
1B /CPU CPY,  1496 A2 M/CPU LDX,
1C    0C8E A0 M/CPU LDY,  048C 80 M
1D /CPU STY,  0480 14 M/CPU JSR,
1E    8480 40 M/CPU JMP,  0484 20 M
1F /CPU BIT,       -->


(  M/CPU,   MULTI-MODE OP-CODES                    WFR-79MAR26 )
: M/CPU
        <BUILDS C, , DOES> DUP 1+ C@ 80 AND
        IF
          10 MODE +!
        ENDIF
        OVER FF00 AND UPMODE UPMODE
        IF
          MEM CR LATEST ID. 3 ERROR
        ENDIF
        C@ MODE C@ INDEX + C@ + C, MODE C@ 7 AND
        IF
          MODE C@ 0F AND 7 <
          IF
            C,
          ELSE
            ,
          ENDIF
        ENDIF
        MEM
        ;
1C6E 60 M/CPU ADC,
1C6E 20 M/CPU AND,
1C6E C0 M/CPU CMP,
```

```
1C6E 40 M/CPU EOR,
1C6E A0 M/CPU LDA,
1C6E 00 M/CPU ORA,
1C6E E0 M/CPU SBC,
1C6C 80 M/CPU STA,
0D0D 01 M/CPU ASL,
0C0C C1 M/CPU DEC,
0C0C E1 M/CPU INC,
0D0D 41 M/CPU LSR,
0D0D 21 M/CPU ROL,
0D0D 61 M/CPU ROR,
0414 81 M/CPU STX,
0486 E0 M/CPU CPX,
0486 C0 M/CPU CPY,
1496 A2 M/CPU LDX,
0C8E A0 M/CPU LDY,
048C 80 M/CPU STY,
0480 14 M/CPU JSR,
8480 40 M/CPU JMP,
0484 20 M/CPU BIT,
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  11
00 (  ASSEMBLER CONDITIONALS
01                    WFR-79MAR26 )
02 : BEGIN, HERE 1 ;    IMMEDIATE
03 : UNTIL,    ?EXEC >R 1 ?PAIRS R>
04  C, HERE 1+ - C, ; IMMEDIATE
05 : IF, C, HERE 0 C, 2 ; IMMEDIATE
06  : ENDIF, ?EXEC 2 ?PAIRS HERE
07 OVER  C@
08  IF  SWAP  !  ELSE  OVER  1+  -
09  SWAP  C!  ENDIF  ; IMMEDIATE
0A : ELSE, 2 ?PAIRS HERE 1+ 1 JMP,
0B SWAP HERE OVER 1+ - SWAP C! 2 ;
0C  IMMEDIATE
0D : THEN, [COMPILE] ENDIF, ;
0E IMMEDIATE      : END, [COMPILE]
0F UNTIL, ; IMMEDIATE
10 : NOT    20  +  ;
11      ( REVERSE ASSEMBLY TEST )
12 90  CONSTANT  CS    ( ASSEMBLER
13 TEST FOR CARRY SET )
14 D0  CONSTANT  0=    ( ASSEMBLER
15 TEST FOR EQUAL ZERO )
16 10  CONSTANT  0<    ( ASSEMBLER
17 TEST FOR LESS THAN ZERO )
18 90  CONSTANT  >=    ( ASSEMBLER
19 TEST FOR GREATER OR EQUAL ZERO )
1A  ( >= IS ONLY CORRECT AFTER SUB,
1B  OR CMP,  )
1C 50 CONSTANT VS      ( ASSEMBLER
1D TEST FOR OVERFLOW BIT SET )
1E -->
1F
```

```
(  ASSEMBLER CONDITIONALS                          WFR-79MAR26 )
: BEGIN,
        HERE 1
        ;
IMMEDIATE
: UNTIL,
        ?EXEC >R 1 ?PAIRS R> C, HERE 1+ - C,
        ;
IMMEDIATE
: IF,
        C, HERE 0 C, 2
        ;
IMMEDIATE
: ENDIF,
        ?EXEC 2 ?PAIRS HERE OVER C@
        IF
          SWAP !
        ELSE
          OVER 1+ - SWAP C!
        ENDIF
        ;
IMMEDIATE
: ELSE,
        2 ?PAIRS HERE 1+ 1 JMP,
        SWAP HERE OVER 1+ - SWAP C! 2
        ;
IMMEDIATE
: THEN,
        [COMPILE]
      ENDIF,
      ;
IMMEDIATE
: END,
        [COMPILE]
      UNTIL,
      ;
IMMEDIATE
: NOT
        20 +
        ;
( REVERSE ASSEMBLY TEST )
90 CONSTANT CS
( ASSEMBLER TEST FOR CARRY SET )
D0 CONSTANT 0=
( ASSEMBLER TEST FOR EQUAL ZERO )
10 CONSTANT 0<
( ASSEMBLER TEST FOR LESS THAN ZERO )
90 CONSTANT >=
( ASSEMBLER TEST FOR GREATER OR EQUAL ZERO )
( >= IS ONLY CORRECT AFTER SUB, OR CMP,  )
50 CONSTANT VS
( ASSEMBLER TEST FOR OVERFLOW BIT SET )
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  12
00 (  USE OF ASSEMBLER
01                  WFR-79APR28 )
```

```
02 : C;
03     ( END OF CODE DEFINITION *)
04    CURRENT  @  CONTEXT  !  ?EXEC
05   ?CSP  SMUDGE  ; IMMEDIATE
06
07
08 FORTH  DEFINITIONS
09
0A : CODE      ( CREATE WORD AT ASS
0B EMBLY CODE LEVEL *)
0C       ?EXEC  CREATE  [COMPILE]
0D ASSEMBLER
0E       ASSEMBLER  MEM  !CSP  ;
0F    IMMEDIATE
10 DECIMAL
11 '  ASSEMBLER  CFA    '  ;CODE  8
12   +  !  ( OVER-WRITE SMUDGE )
13
14 -->
15
16
17
18
19
1A
1B
1C
1D
1E
1F


(  USE OF ASSEMBLER                                  WFR-79APR28 )
: C;
        ( END OF CODE DEFINITION *)
        CURRENT @ CONTEXT ! ?EXEC ?CSP SMUDGE
        ;
IMMEDIATE FORTH DEFINITIONS
: CODE
        ( CREATE WORD AT ASSEMBLY CODE LEVEL *)
        ?EXEC CREATE [COMPILE] ASSEMBLER ASSEMBLER MEM !CSP
        ;
IMMEDIATE DECIMAL ' ASSEMBLER CFA '
        ;CODE
        8 + ! ( OVER-WRITE SMUDGE )
        -->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  13
00 ( EXEC, routines ) BASE @ HEX
01 ASSEMBLER DEFINITIONS
02 CODE xec  IP LDA,  W STA, IP 1+
03 LDA, W 1+ STA, ( save IP )
04  PLA, CLC, 1 # ADC, IP STA,
05  PLA,
06  0 # ADC, IP 1+ STA, ( get new
07  IP)
08 W 1+ LDA,  PHA,  W LDA,  PHA,
```

```
09 ( save last IP ) NEXT JMP, C;
0A CODE xec2 IP LDA, W STA, IP 1+
0B LDA, W 1+ STA,     ( save IP to
0C continue in the code routine )
0D PLA, IP STA, PLA, IP 1+ STA,
0E ( Restore old IP )
0F     W ) JMP, C;
10
11 : EXEC,     ( addr -- ^ EXECUTE
12 COLON WORD IN A CODE DEF )
13 ( addr = PFA OF COLON WORD )
14 ' xec JSR,
15 CFA  , ' xec2 CFA , ;
16
17 FORTH  DEFINITIONS  DECIMAL
18
19 HERE   28  +ORIGIN  !  ( FENCE )
1A
1B HERE    30  +ORIGIN  !  ( DP )
1C
1D '  ASSEMBLER  2  +
1E 32  +ORIGIN  !  ( VOC-LINK )
1F HERE  FENCE  !   BASE !  ;S


( EXEC, routines )
BASE @ HEX ASSEMBLER DEFINITIONS
CODE xec
        IP LDA,
        W STA,
        IP 1+ LDA,
        W 1+ STA,
        ( save IP )
        PLA,
        CLC,
        1 # ADC,
        IP STA,
        PLA,
        0 # ADC,
        IP 1+ STA,
        ( get new   IP)
        W 1+ LDA,
        PHA,
        W LDA,
        PHA,
        ( save last IP )
        NEXT JMP,
        C;
CODE xec2
        IP LDA,
        W STA,
        IP 1+ LDA,
        W 1+ STA,
        ( save IP to continue in the code routine )
        PLA,
        IP STA,
        PLA,
        IP 1+ STA,
        ( Restore old IP )
```

```
        W ) JMP,
        C;
: EXEC,
        ( addr -- ^ EXECUTE COLON WORD IN A CODE DEF )
        ( addr = PFA OF COLON WORD )
        ' xec JSR,
        CFA , ' xec2 CFA ,
        ;
FORTH DEFINITIONS DECIMAL HERE 28 +ORIGIN ! ( FENCE )
HERE 30 +ORIGIN ! ( DP )
' ASSEMBLER 2 + 32 +ORIGIN ! ( VOC-LINK )
HERE FENCE ! BASE ! ;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  14
00 ( LPWORDS FOR JOYSTICK CONTROLLE
01 R JACKS )
02 CODE STROBE BOT LDA, D301 STA,
03  80 # ORA, D301 STA,
04  POP JMP, C; : PRT D303 C@ FB
05  AND D303 C! FF D301 C! D303
06 C@ 4 OR D303 C! BEGIN D013 C@  1
07  AND 0= UNTIL 7F AND STROBE ;
08 : LYP1 DUP IF 0 DO DUP I + C@ PR
09 T LOOP DROP
0A ELSE DROP DROP THEN ;
0B     : LYPE LYP1 20 PRT ;
0C : CRLP 0D PRT 0A PRT ; : FFLP 0C
0D  PRT CRLP ;
0E : .LP S->D SWAP OVER DABS <# #S
0F   SIGN #>
10     LYPE ;
11
12 : LISTLP DUP SCR !  CRLP
13 0E PRT ( [ SCREEN ] LYPE ) .LP
14 0F PRT 10 0 DO CRLP I  DUP .LP
15  LINE C/L -TRAILING LYPE LOOP
16 CRLP  ; : SHOWLP 1+ SWAP
17  DO I LISTLP 3 0 DO CRLP
18 LOOP LOOP ;
19
1A ;S
1B
1C
1D
1E
1F


( LPWORDS FOR JOYSTICK CONTROLLER JACKS )
CODE STROBE
        BOT LDA,
        D301 STA,
        80 # ORA,
        D301 STA,
        POP JMP,
        C;
: PRT
```

```
        D303 C@ FB AND D303 C! FF D301 C! D303 C@ 4 OR D303 C!
        BEGIN
          D013 C@ 1 AND 0=
        UNTIL
        7F AND STROBE
        ;
: LYP1
        DUP
        IF
          0
          DO
            DUP I + C@ PRT
          LOOP
          DROP
        ELSE
          DROP DROP
        THEN
        ;
: LYPE
        LYP1 20 PRT
        ;
: CRLP
        0D PRT 0A PRT
        ;
: FFLP
        0C PRT CRLP
        ;
: .LP
        S->D SWAP OVER DABS <# #S SIGN #> LYPE
        ;
: LISTLP
        DUP SCR ! CRLP 0E PRT ( [ SCREEN ] LYPE )
        .LP 0F PRT 10 0
        DO
          CRLP I DUP .LP LINE C/L -TRAILING LYPE
        LOOP
        CRLP
        ;
: SHOWLP
        1+ SWAP
        DO
          I LISTLP 3 0
          DO
            CRLP
          LOOP
        LOOP
        ;
;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  15
00 HEX VOCABULARY EDITOR IMMEDIATE
01
02 1A LOAD 1B LOAD ( GRAPHICS )
03
04 : EDIT SCR ! [COMPILE] EDITOR ;
05
06 EDITOR DEFINITIONS
```

```
07 0 VARIABLE TOPFLAG
08 : ULL DUP TOPFLAG ! 0 GR.   2203
09   LMARGN ! 3 0 POS. ( 32 CHAR )
0A   1 2FE C! ( PRINT ALL CHARS )
0B SCR @ BLOCK + 200 TYPE ( PRINT )
0C   0 2FE C! ( CURSOR CNTRLS )
0D CR ." DOIT" CR 0AAAA 2B2 ! ;
0E : UL 0 ULL ;     ( SHOW UPPER 16
0F  LINES )
10 : LL 200 ULL ; ( SHOW  LOWER  16
11   LINES )
12 : DOIT 10 0 DO -1 2B2 !
13
14 3 I POS. ( POINT CURSOR )
15
16 SCR @ BLOCK I 20 * + TOPFLAG @ +
17
18 ICBAL ! 20 ICBLL ! GET DROP
19
1A LOOP UPDATE 0 GR. TOPFLAG @ 0=
1B   IF UL ELSE LL ENDIF ;
1C : FLUSH 2602 LMARGN !
1D   [COMPILE] FORTH FLUSH ;
1E -->
1F


HEX VOCABULARY EDITOR IMMEDIATE 1A LOAD
1B LOAD
( GRAPHICS )
: EDIT
        SCR ! [COMPILE] EDITOR
        ;
EDITOR DEFINITIONS 0 VARIABLE TOPFLAG
: ULL
        DUP TOPFLAG ! 0 GR. 2203 LMARGN ! 3 0 POS. ( 32 CHAR )
        1 2FE C! ( PRINT ALL CHARS )
        SCR @ BLOCK + 200 TYPE ( PRINT )
        0 2FE C! ( CURSOR CNTRLS )
        CR ." DOIT" CR 0AAAA 2B2 !
        ;
: UL
        0 ULL
        ;
( SHOW UPPER 16 LINES )
: LL
        200 ULL
        ;
( SHOW  LOWER  16  LINES )
: DOIT
        10 0
        DO
          -1 2B2 ! 3 I POS. ( POINT CURSOR )
          SCR @ BLOCK I 20 * + TOPFLAG @ + ICBAL ! 20 ICBLL !
          GET DROP
        LOOP
        UPDATE 0 GR. TOPFLAG @ 0=
        IF
          UL
```

```
        ELSE
          LL
        ENDIF
        ;
: FLUSH
        2602 LMARGN ! [COMPILE] FORTH FLUSH
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  16
00 (  TEXT,  LINE, WHERE USED IN
01  EDITOR    7/7/80-SRC )
02 FORTH  DEFINITIONS   HEX
03
04
05
06 : TEXT                  ( ACCEPT
07  FOLLOWING TEXT TO PAD *)
08     HERE  C/L  1+   BLANKS  WORD
09   HERE  PAD  C/L  1+  CMOVE  ;
0A : #OFLINES B/BUF B/SCR * C/L / ;
0B
0C : LINE             ( RELATIVE TO
0D  SCR, LEAVE ADDRESS OF LINE *)
0E  DUP  #OFLINES            MINUS
0F AND IF ." NOT ON SCREEN" ABORT
10 ENDIF  ( KEEP ON THIS SCREEN )
11       SCR  @  (LINE)  DROP  ;
12
13  -->
14
15
16
17
18
19
1A
1B
1C
1D
1E
1F


(  TEXT,  LINE, WHERE USED IN    EDITOR    7/7/80-SRC )
FORTH DEFINITIONS HEX
: TEXT
        ( ACCEPT FOLLOWING TEXT TO PAD *)
        HERE C/L 1+ BLANKS WORD HERE PAD C/L 1+ CMOVE
        ;
: #OFLINES
        B/BUF B/SCR * C/L /
        ;
: LINE
        ( RELATIVE TO SCR, LEAVE ADDRESS OF LINE *)
        DUP #OFLINES MINUS AND
        IF
```

```
          ." NOT ON SCREEN" ABORT
        ENDIF
        ( KEEP ON THIS SCREEN )
        SCR @ (LINE) DROP
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  17
00 (  LINE EDITING COMMANDS )
01      EDITOR DEFINITIONS
02 : -MOVE   ( MOVE IN BLOCK BUFFER
03  ADDR FROM-2,  LINE TO-1 *)
04      LINE  C/L  CMOVE  UPDATE  ;
05
06 : HL                       ( HOLD
07  NUMBERED LINE AT PAD *)
08     LINE  PAD  1+  C/L  DUP  PAD
09   C!  CMOVE  ;
0A : BL                      ( ERASE
0B  LINE-1 WITH BLANKS *)
0C LINE  C/L  BLANKS  UPDATE ;
0D
0E : SL                     ( SPREAD
0F  MAKING LINE # BLANK *)
10       DUP  1  -  ( LIMIT )
11 #OFLINES 2 -  ( FIRST TO MOVE )
12       DO  I  LINE  I  1+  -MOVE
13   -1  +LOOP  BL ;
14 : DL            ( DELETE LINE-1,
15  BUT HOLD IN PAD *)
16    DUP  HL  #OFLINES   1   -
17    DUP  ROT
18       DO  I  1+  LINE  I  -MOVE
19  LOOP  BL ;
1A : CL  ( COPY LINE-2 OF SCREEN-1
1B  TO PAD )
1C SCR @ >R SCR ! HL R> SCR ! ;
1D
1E -->
1F



(  LINE EDITING COMMANDS )
EDITOR DEFINITIONS
: -MOVE
        ( MOVE IN BLOCK BUFFER ADDR FROM-2,  LINE TO-1 *)
        LINE C/L CMOVE UPDATE
        ;
: HL
        ( HOLD NUMBERED LINE AT PAD *)
        LINE PAD 1+ C/L DUP PAD C! CMOVE
        ;
: BL
        ( ERASE LINE-1 WITH BLANKS *)
        LINE C/L BLANKS UPDATE
        ;
```

```
: SL
        ( SPREAD MAKING LINE # BLANK *)
        DUP 1 - ( LIMIT )
        #OFLINES 2 - ( FIRST TO MOVE )
        DO
          I LINE I 1+ -MOVE -1
        +LOOP
        BL
        ;
: DL
        ( DELETE LINE-1, BUT HOLD IN PAD *)
        DUP HL #OFLINES 1 - DUP ROT
        DO
          I 1+ LINE I -MOVE
        LOOP
        BL
        ;
: CL
        ( COPY LINE-2 OF SCREEN-1  TO PAD )
        SCR @ >R SCR ! HL R> SCR !
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  18
00 (  LINE EDITING COMMANDS
01                    WFR-790105 )
02 : RL
03  ( REPLACE ON LINE-1, FROM PAD )
04       PAD  1+  SWAP  -MOVE  ;
05
06
07
08 : $                          ( PUT
09  FOLLOWING TEXT ON LINE-1 )
0A       1  TEXT  RL  QUIT  ;
0B
0C
0D
0E : %                 ( INSERT TEXT
0F  FOLLOWING AFTER LINE-1 *)
10     1   TEXT  1+  DUP  SL  RL  ;
11
12
13
14 : IL       (  INSERT  PAD  AFTER
15  LINE-1 ) 1+  DUP  SL  RL  ;
16
17
18 : TL   ( TYPE LINE BY #-1,  SAVE
19  ALSO IN PAD *)
1A     DUP . ." $ "
1B     DUP C/L *  R#  !       HL
1C      PAD  1+  C/L  TYPE  CR ;
1D
1E -->
1F
```

```
(  LINE EDITING COMMANDS                              WFR-790105 )
: RL
        ( REPLACE ON LINE-1, FROM PAD )
        PAD 1+ SWAP -MOVE
        ;
: $
        ( PUT FOLLOWING TEXT ON LINE-1 )
        1 TEXT RL QUIT
        ;
: %
        ( INSERT TEXT FOLLOWING AFTER LINE-1 *)
        1 TEXT 1+ DUP SL RL
        ;
: IL
        (  INSERT  PAD  AFTER LINE-1 )
        1+ DUP SL RL
        ;
: TL
        ( TYPE LINE BY #-1,  SAVE ALSO IN PAD *)
        DUP . ." $ " DUP C/L * R# ! HL PAD 1+ C/L TYPE CR
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  19
00 (  SCREEN EDITING COMMANDS )
01 FORTH DEFINITIONS
02
03
04 : COPY  (  DUPLICATE  SCREEN-2,
05  ONTO SCREEN-1 *)
06   SWAP BLOCK DROP PREV @ !
07   UPDATE FLUSH  ;
08
09
0A : LIST 2602 LMARGN ! LIST ;
0B
0C
0D : SHOW 1+ SWAP DO I LIST LOOP ;
0E
0F
10 : L SCR @ LIST ( RE-LIST SCR ) ;
11
12 : N SCR @ 1+ LIST ;  ( LIST NEXT
13  SCR)
14
15 : WHERE ( OFFSET BLK --- ) DUP
16 SCR ! ." SCR # " . CR C/L /MOD
17 EDITOR TL FORTH 2 + SPACES
18 5E EMIT [COMPILE] EDITOR QUIT ;
19
1A BASE   @   DECIMAL
1B     ' EDITOR 2 + 32 +ORIGIN !
1C  ( VOC-LINK ) BASE !
1D      ;S
1E
1F
```

```
(  SCREEN EDITING COMMANDS )
FORTH DEFINITIONS
: COPY
        (  DUPLICATE  SCREEN-2,  ONTO SCREEN-1 *)
        SWAP BLOCK DROP PREV @ ! UPDATE FLUSH
        ;
: LIST
        2602 LMARGN ! LIST
        ;
: SHOW
        1+ SWAP
        DO
          I LIST
        LOOP
        ;
: L
        SCR @ LIST ( RE-LIST SCR )
        ;
: N
        SCR @ 1+ LIST
        ;
( LIST NEXT SCR)
: WHERE
        ( OFFSET BLK --- )
        DUP SCR ! ." SCR # " . CR C/L /MOD EDITOR TL FORTH 2 +
        SPACES 5E EMIT [COMPILE] EDITOR QUIT
        ;
BASE @ DECIMAL ' EDITOR 2 + 32 +ORIGIN ! ( VOC-LINK )
BASE ! ;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  1A
00 ( OS & HDW CONSTANTS ) : CN CONS
01 TANT ;
02 D200 CN F1AUD D201 CN C1AUD
03
04 D202 CN F2AUD D203 CN C2AUD
05
06 D204 CN F3AUD D205 CN C3AUD
07
08 D206 CN F4AUD D207 CN C4AUD
09
0A D20F CN SKCTL D208 CN AUDCTL
0B
0C  230 CN DLST   22F CN DMCT
0D
0E   14 CN RTCLK  2F0 CN CRSINH
0F
10  2F4 CN CHBAS  2C4 CN COL0
11
12  2C5 CN COL1   2C6 CN COL2
13
14  2C7 CN COL3   2C8 CN COL4
15
16 D01F CN CONSOL 2FC CN CH
17
```

```
18  2BF CN BOTSC   52 CN LMARGN
19
1A 2FB CN ATACHR
1B
1C  ;S
1D
1E
1F


( OS & HDW CONSTANTS )
: CN
        CONSTANT ;
D200 CN F1AUD
D201 CN C1AUD
D202 CN F2AUD
D203 CN C2AUD
D204 CN F3AUD
D205 CN C3AUD
D206 CN F4AUD
D207 CN C4AUD
D20F CN SKCTL
D208 CN AUDCTL
230 CN DLST
22F CN DMCT
14 CN RTCLK
2F0 CN CRSINH
2F4 CN CHBAS
2C4 CN COL0
2C5 CN COL1
2C6 CN COL2
2C7 CN COL3
2C8 CN COL4
D01F CN CONSOL
2FC CN CH
2BF CN BOTSC
52 CN LMARGN
2FB CN ATACHR
;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


SCR #  1B
00 ( CIO CALL ROUTINES )
01
02 340 VARIABLE IOC 0 VARIABLE IOB
03
04 : IOCB 7 MIN 0 MAX 10 * DUP IOB
05 ! 340 + IOC ! ;
06 : .IOC <BUILDS , DOES> @ IOC @ +
07  ;
08 1 .IOC ICDNO 2 .IOC ICCOM 3 .IOC
09  ICSTA
0A 4 .IOC ICBAL 6 .IOC ICPTL
0B
0C 8 .IOC ICBLL A .IOC I1CAX B .IOC
0D  I2CAX
0E CODE CIO TXA, PHA, IOB LDX, E456
0F  JSR, PLA, TAX, NEXT JMP, C;
```

```
10 CODE Get XSAVE STX, IOB LDX, E45
11 6 JSR,
12 XSAVE LDX, PHA, 0 # LDA, PUSH JM
13 P, C;
14 : GET 7 ICCOM C! Get ;
15
16 : CLOSE 0C ICCOM C! CIO ;
17
18 : OPEN 3 ICCOM C! ICBAL ! I1CAX
19 C! I2CAX C! CIO ;
1A CODE ACIO XSAVE STX, BOT LDA, IO
1B B LDX, E456 JSR,
1C XSAVE LDX, POP JMP, C;
1D
1E -->
1F


( CIO CALL ROUTINES )
340 VARIABLE IOC
0 VARIABLE IOB
: IOCB
        7 MIN 0 MAX 10 * DUP IOB ! 340 + IOC !
        ;
: .IOC
        <BUILDS , DOES> @ IOC @ +
        ;
1 .IOC ICDNO 2 .IOC ICCOM 3 .IOC ICSTA 4 .IOC ICBAL 6 .IOC
ICPTL 8 .IOC ICBLL A .IOC I1CAX B .IOC I2CAX
CODE CIO
        TXA,
        PHA,
        IOB LDX,
        E456 JSR,
        PLA,
        TAX,
        NEXT JMP,
        C;
CODE Get
        XSAVE STX,
        IOB LDX,
        E456 JSR,
        XSAVE LDX,
        PHA,
        0 # LDA,
        PUSH JMP,
        C;
: GET
        7 ICCOM C! Get
        ;
: CLOSE
        0C ICCOM C! CIO
        ;
: OPEN
        3 ICCOM C! ICBAL ! I1CAX C! I2CAX C! CIO
        ;
CODE ACIO
        XSAVE STX,
        BOT LDA,
```

```
        IOB LDX,
        E456 JSR,
        XSAVE LDX,
        POP JMP,
        C;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  1C
00 ( COLLEEN GRAPHICS )
01
02 3A53 VARIABLE S: 2FD CONSTANT
03        FILDAT 0 VARIABLE Qbase
04 : PBASE Qbase @ ; : SPB HIMEM @
05 1+      F800 AND 800 - DUP
06   Qbase ! 17F + HIMEM ! ; SPB
07       : POS. 54 C! 55 ! ;
08 : GR. 1 IOCB CLOSE 0 ICBLL ! DUP
09  F      AND SWAP 30 AND 10
0A   XOR 0C + S: OPEN SPB ;
0B       : GRAPHICS GR. ;
0C : LOC. POS. GET ; 1 VARIABLE Col
0D or
0E : C. DUP Color C! FILDAT C! ;
0F       : PUT 0B ICCOM C! ACIO ;
10 : PL. POS. ICBLL 0SET Color C@ P
11 UT ;
12 : SE. SWAP 10 * + SWAP 2C4 + C!
13 ;
14 : DR. POS. 11 ICCOM C! Color C@
15 DUP 2FB C! FILDAT C! CIO ;
16 : PLOT PL. ;      : LOCATE LOC.
17 ;
18 : SETCOLOR SE. ;  : COLOR C. ;
19
1A : POSITION POS. ; : DRAWTO DR. ;
1B
1C : CLEAR   0 0 POS. 7D PUT ;
1D
1E : XIO18 ( FILL ) DUP 2FD C! 2FB
1F C!      12 ICCOM C! CIO ; -->


( COLLEEN GRAPHICS )
3A53 VARIABLE S:
2FD CONSTANT FILDAT
0 VARIABLE Qbase
: PBASE
        Qbase @
        ;
: SPB
        HIMEM @ 1+ F800 AND 800 - DUP Qbase ! 17F + HIMEM !
        ;
SPB
: POS.
        54 C! 55 !
        ;
: GR.
```

```
        1 IOCB CLOSE 0 ICBLL ! DUP F AND SWAP 30 AND 10 XOR 0C
        + S: OPEN SPB
        ;
: GRAPHICS
        GR.
        ;
: LOC.
        POS. GET
        ;
1 VARIABLE Color
: C.
        DUP Color C! FILDAT C!
        ;
: PUT
        0B ICCOM C! ACIO
        ;
: PL.
        POS. ICBLL 0SET Color C@ PUT
        ;
: SE.
        SWAP 10 * + SWAP 2C4 + C!
        ;
: DR.
        POS. 11 ICCOM C! Color C@ DUP 2FB C! FILDAT C! CIO
        ;
: PLOT
        PL.
        ;
: LOCATE
        LOC.
        ;
: SETCOLOR
        SE.
        ;
: COLOR
        C.
        ;
: POSITION
        POS.
        ;
: DRAWTO
        DR.
        ;
: CLEAR
        0 0 POS. 7D PUT
        ;
: XIO18
        ( FILL )
        DUP 2FD C! 2FB C! 12 ICCOM C! CIO
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  1D
00 ( SOUND CONTROLLERS RND PLAYER/M
01 ISSILES )
02 : SOUND 3 D20F C! 0 D208 C! SWAP
03
```

```
04 10 * + 100 * + SWAP 2 * D200 + !
05  ;
06 : PADDLE 270 + C@ ;
07        : PTRIG 27C + C@ ;
08 : STICK 278 + C@ ;
09        : STRIG 284 + C@ ;
0A : RND D20A C@ ;
0B         22F CONSTANT DMACTL
0C D01D CONSTANT GRACTL
0D        D407 CONSTANT PMBASE
0E D01B CONSTANT PRIOR
0F        D016 CONSTANT VDELAY
10  2C0 CONSTANT COLPM
11         26F CONSTANT GPRIOR
12 PBASE 1 - HIMEM !
13
14 : PLAYER Qbase 1+ C@ PMBASE C! 3
15          GRACTL C! 2 - IF 1C
16   ELSE 0C ENDIF DMACTL @ E3 AND
17 OR       DMACTL C! ;
18 : HPOS! D000 + C! ;
19          ( H-posn plyr# -> )
1A : SIZE! D008 + C! ;
1B          ( size-code plyr# -> )
1C : COLPM! COLPM + C! ;
1D          ( color plyr# -> )
1E : NOPLY GRACTL 0SET D000 11 0 FI
1F LL       ; ;S


( SOUND CONTROLLERS RND PLAYER/MISSILES )
: SOUND
        3 D20F C! 0 D208 C! SWAP 10 * + 100 * + SWAP 2 * D200 +
        !
        ;
: PADDLE
        270 + C@
        ;
: PTRIG
        27C + C@
        ;
: STICK
        278 + C@
        ;
: STRIG
        284 + C@
        ;
: RND
        D20A C@
        ;
22F CONSTANT DMACTL
D01D CONSTANT GRACTL
D407 CONSTANT PMBASE
D01B CONSTANT PRIOR
D016 CONSTANT VDELAY
2C0 CONSTANT COLPM
26F CONSTANT GPRIOR
PBASE 1 - HIMEM !
: PLAYER
```

```
        Qbase 1+ C@ PMBASE C! 3 GRACTL C! 2 -
        IF
          1C
        ELSE
          0C
        ENDIF
        DMACTL @ E3 AND OR DMACTL C!
        ;
: HPOS!
        D000 + C!
        ;
( H-posn plyr# -> )
: SIZE!
        D008 + C!
        ;
( size-code plyr# -> )
: COLPM!
        COLPM + C!
        ;
( color plyr# -> )
: NOPLY
        GRACTL 0SET D000 11 0 FILL
        ;
;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  1E
00 ( DECUS-FORTH ADDITIONS )
01
02 : 1+! 1 SWAP +! ; : 1- 1 - ;
03
04 : 0SET 0 SWAP ! ; : 2* DUP + ;
05
06 : HD DUP 0A < IF 30 ELSE 37
07  ENDIF + EMIT ;
08 : CHH DUP 0F0 AND 10 / HD 0F AND
09   HD ;
0A : CH? C@ CHH ;
0B
0C : HH DUP 0FF00 AND 100 / 0FF AND
0D   CHH  CHH  ;
0E : H? @ HH ;
0F
10 : BDUMP 1+ SWAP DO I HH SPACE  I
11
12   8 0 DO DUP I + CH? SPACE  LOOP
13   DROP  ." \"  CR  8  +LOOP ;
14
15 : \ 10 0 DO SP@ 0E + I - @ SP@
16   12 + @ I 2 / + C!
17    2 +LOOP       DROP DROP DROP
18 DROP DROP DROP  DROP DROP DROP
19 QUIT ;
1A : TBL <BUILDS DOES> ;
1B
1C : ALLOC DUP + ALLOT ; ( FOR RAM
1D BASED SYSTEMS,)
1E : ARRAY <BUILDS ALLOC DOES> ;
```

```
1F ;S


( DECUS-FORTH ADDITIONS )
: 1+!
        1 SWAP +!
        ;
: 1-
        1 -
        ;
: 0SET
        0 SWAP !
        ;
: 2*
        DUP +
        ;
: HD
        DUP 0A <
        IF
          30
        ELSE
          37
        ENDIF
        + EMIT
        ;
: CHH
        DUP 0F0 AND 10 / HD 0F AND HD
        ;
: CH?
        C@ CHH
        ;
: HH
        DUP 0FF00 AND 100 / 0FF AND CHH CHH
        ;
: H?
        @ HH
        ;
: BDUMP
        1+ SWAP
        DO
          I HH SPACE I 8 0
          DO
            DUP I + CH? SPACE
          LOOP
          DROP ." \" CR 8
        +LOOP
        ;
: \
        10 0
        DO
          SP@ 0E + I - @ SP@ 12 + @ I 2 / + C! 2
        +LOOP
        DROP DROP DROP DROP DROP DROP DROP DROP DROP QUIT
        ;
: TBL
        <BUILDS DOES>
        ;
: ALLOC
        DUP + ALLOT
```

```
        ;
( FOR RAM BASED SYSTEMS,)
: ARRAY
        <BUILDS ALLOC DOES>
        ;
;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  1F
00 ( DISPLAY LIST STUFF )
01
02 0 VARIABLE 3BYT 0 VARIABLE DLADR
03
04 : DINST DLADR @ C@ DUP 0F AND IF
05
06 DUP 0F AND 1 = IF 40 AND IF ." J
07 VB "
08 ELSE ." JMP " ENDIF DLADR 1+! DL
09 ADR  @
0A @ DUP DLADR ! HH 3BYT 0SET ELSE
0B DUP 0F AND
0C 8 OVER < IF ." MAP" ELSE ." CHR"
0D
0E ENDIF 7 AND . DUP 10 AND IF ." H
0F "
10 THEN DUP 20 AND IF ." V" THEN DU
11 P
12 80 AND IF ." I" ENDIF DUP 0B0
13
14 AND IF DUP 40 AND IF ." ," ENDIF
15
16 ENDIF 40 AND IF 3 DLADR @ 1+ H?
17 ELSE
18 1 ENDIF 3BYT ! ENDIF ELSE ." BLK
19 "
1A DUP 80 AND IF ." I," ENDIF 70
1B
1C AND 10 / . 1 3BYT ! ENDIF CR
1D
1E 3BYT @ DLADR +! ; ;S
1F


( DISPLAY LIST STUFF )
0 VARIABLE 3BYT
0 VARIABLE DLADR
: DINST
        DLADR @ C@ DUP 0F AND
        IF
          DUP 0F AND 1 =
          IF
            40 AND
            IF
              ." JVB "
            ELSE
              ." JMP "
            ENDIF
            DLADR 1+! DLADR @ @ DUP DLADR ! HH 3BYT 0SET
```

```
          ELSE
            DUP 0F AND 8 OVER <
            IF
              ." MAP"
            ELSE
              ." CHR"
            ENDIF
            7 AND . DUP 10 AND
            IF
              ." H"
            THEN
            DUP 20 AND
            IF
              ." V"
            THEN
            DUP 80 AND
            IF
              ." I"
            ENDIF
            DUP 0B0 AND
            IF
              DUP 40 AND
              IF
                ." ,"
              ENDIF
            ENDIF
            40 AND
            IF
              3 DLADR @ 1+ H?
            ELSE
              1
            ENDIF
            3BYT !
          ENDIF
        ELSE
          ." BLK" DUP 80 AND
          IF
            ." I,"
          ENDIF
          70 AND 10 / . 1 3BYT !
        ENDIF
        CR 3BYT @ DLADR +!
        ;
;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  20
00 ( WRITE BOOTABLE OBJECT 1 OF 2 )
01
02 BASE @ FORTH DEFINITIONS HEX
03 SAVENFAs ( PRESERV ALL NFAS )
04 (  LATEST  0C  +ORIGIN ! )
05 ( TOP NFA )
06 HERE    1C  +ORIGIN ! ( FENCE )
07
08 HERE    1E  +ORIGIN ! ( DP )
09
0A HERE DUP FENCE ! 0 +ORIGIN - 80
```

```
0B /     1+ CONSTANT #SECT
0C CODE CALLDK XSAVE STX, E453 JSR,
0D         TYA, PHA, ( STATUS )
0E   XSAVE LDX, PUSH JMP, C;
0F
10 : DKIO 301 ! ( CMD, DRIVE # )
11         30A ! ( SECT. # )
12   304 ! ( RAM BUFFER ADDR )
13         CALLDK ( JSR DKHND)
14   DUP 0< IF ." ERROR " 0FF AND
15         BASE @ SWAP DECIMAL
16   . BASE ! QUIT ENDIF DROP ;
17
18 : WTSEC 5701 DKIO ;
19       : RDSEC 5201 DKIO ;
1A : FORMAT ." FORMAT DRIVE " DUP .
1B
1C   ." -ARE YOU SURE?" 0 PAD ! PAD
1D  1      EXPECT PAD C@ 59 ( Y) =
1E   IF 2100 OR PAD 0 ROT DKIO ELSE
1F         DROP THEN ; -->


( WRITE BOOTABLE OBJECT 1 OF 2 )
BASE @ FORTH DEFINITIONS HEX SAVENFAs ( PRESERV ALL NFAS )
(  LATEST  0C  +ORIGIN ! )
( TOP NFA )
HERE 1C +ORIGIN ! ( FENCE )
HERE 1E +ORIGIN ! ( DP )
HERE DUP FENCE ! 0 +ORIGIN - 80 / 1+ CONSTANT #SECT
CODE CALLDK
        XSAVE STX,
        E453 JSR,
        TYA,
        PHA,
        ( STATUS )
        XSAVE LDX,
        PUSH JMP,
        C;
: DKIO
        301 ! ( CMD, DRIVE # )
        30A ! ( SECT. # )
        304 ! ( RAM BUFFER ADDR )
        CALLDK ( JSR DKHND)
        DUP 0<
        IF
          ." ERROR " 0FF AND BASE @ SWAP DECIMAL . BASE ! QUIT
        ENDIF
        DROP
        ;
: WTSEC
        5701 DKIO
        ;
: RDSEC
        5201 DKIO
        ;
: FORMAT
        ." FORMAT DRIVE " DUP . ." -ARE YOU SURE?" 0 PAD ! PAD
        1 EXPECT PAD C@ 59 ( Y)
```

```
        =
        IF
          2100 OR PAD 0 ROT DKIO
        ELSE
          DROP
        THEN
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  21
00 ( WRITE BOOTABLE OBJECT 2 OF 2 )
01
02  0 VARIABLE BOOT ( ->CODE)
03 : WTOBJ FLUSH EMPTY-BUFFERS
04  ." INSERT NEW DISK, TYPE Y" CR
05 0      PAD ! ( DEFAULT )
06  PAD 3 EXPECT PAD C@ 59 = IF BOO
07 T      @ 1 WTSEC #SECT 0 DO I
08  80 * +ORIGIN I 2 + WTSEC I 2 +
09 .      LOOP ." DONE" CR THEN ;
0A ( FOLLOWING IS BOOT SECTOR CODE
0B )     HERE BOOT ! ( PT TO US )
0C ASSEMBLER 1FF , 480 , E4C0 , #SE
0D CT     # LDA, 0= IF, 0 +ORIGIN ,
0E  1 , ENDIF, N STA, 52 # LDA, 302
0F        STA, 48C LDA, 30A STA,
10  48D LDA, 30B STA, ( FIRST SECTO
11 R)     1 # LDA, ( DRV) 301 STA,
12  48A LDA, 304 STA, 48B LDA,
13        305 STA, ( ORIGIN)
14  BEGIN, 30A INC, 0= IF, 30B INC,
15        ENDIF, E453 JSR, 303 LDA,
16  .A ASL, CS IF, RTS, ( FRETURN )
17        ENDIF, 304 LDA, 80 # EOR,
18  304 STA, 0< NOT IF, 305 INC,
19        ENDIF, ( BUMP PTR.)
1A  N DEC, 0= UNTIL, 48A LDA, 0A ST
1B A,     48B LDA, 0B STA, CLC,
1C  RTS, FORTH BASE ! ." n FORMAT"
1D CR ." to Format Disk Drive n" CR
1E  ." WTOBJ to write boot version
1F of        current object" CR ;S


( WRITE BOOTABLE OBJECT 2 OF 2 )
0 VARIABLE BOOT
( ->CODE)
: WTOBJ
        FLUSH EMPTY-BUFFERS ." INSERT NEW DISK,
        TYPE Y" CR 0 PAD ! ( DEFAULT )
        PAD 3 EXPECT PAD C@ 59 =
        IF
          BOOT @ 1 WTSEC #SECT 0
          DO
            I 80 * +ORIGIN I 2 + WTSEC I 2 + .
          LOOP
          ." DONE" CR
```

```
        THEN
        ;
( FOLLOWING IS BOOT SECTOR CODE )
HERE BOOT ! ( PT TO US )
ASSEMBLER 1FF , 480 , E4C0 , #SECT # LDA,
0=
IF,
  0 +ORIGIN , 1 ,
ENDIF,
N STA,
52 # LDA,
302 STA,
48C LDA,
30A STA,
48D LDA,
30B STA,
( FIRST SECTOR)
1 # LDA,
( DRV)
301 STA,
48A LDA,
304 STA,
48B LDA,
305 STA,
( ORIGIN)
BEGIN,
  30A INC,
  0=
  IF,
    30B INC,
  ENDIF,
  E453 JSR,
  303 LDA,
  .A ASL,
  CS
  IF,
    RTS,
    ( FRETURN )
  ENDIF,
  304 LDA,
  80 # EOR,
  304 STA,
  0< NOT
  IF,
    305 INC,
  ENDIF,
  ( BUMP PTR.)
  N DEC,
  0=
UNTIL,
48A LDA,
0A STA,
48B LDA,
0B STA,
CLC,
RTS,
FORTH BASE ! ." n FORMAT" CR ." to Format Disk Drive n" CR ."
WTOBJ to write boot version of current object" CR ;S
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  22
00 ( <UNUSED> ) ;S
01
02
03
04
05
06
07
08
09
0A
0B
0C
0D
0E
0F
10
11
12
13
14
15
16
17
18
19
1A
1B
1C
1D
1E
1F


( <UNUSED> )
;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  23
00 ( COLLEEN TO DEVSYS COMMUNICATIO
01 NS WORDS --  SEND )
02 D303 CONSTANT PIADIR D301 CONSTA
03 NT PIA
04 : NIBSEND ( SEND LOW NIBBLE OF
05   TOS )
06 80 OR PIA C! ( SEND DATA )
07  BEGIN PIA C@ 40 AND ( WAIT FOR
08  ACK ) END 0 PIA C! ( ACK-ACK)
09  BEGIN PIA C@ 40 AND 0= END
0A  ( WAIT FOR ACK-ACK-ACK )   ;
0B : DBSND ( TOS=BLOCK PTR )
0C  400 0 DO I OVER + ( DATA PTR )
0D  C@ DUP NIBSEND
0E  ( LOW NIBBLE ) 10 / NIBSEND  (
0F HIGH NIBBLE ) LOOP DROP ;
```

```
10 : BSND ( SET UP PIA AND SEND A
11  BLOCK -- TOS = BLOCK NUMBER )
12  PIADIR C@ FB AND PIADIR C!
13  8F PIA C! ( SET DATA DIRECTION)
14  PIADIR C@ 4 OR PIADIR C!
15  0 PIA C!
16  BEGIN PIA C@ 40 AND 0= END
17  BLOCK DBSND ( SEND THE BLOCK )
18  ;
19
1A ( FRST LAST SMOV MOVE BLOCKS )
1B : SMOV 1+ SWAP DO I .     I BSND
1C  LOOP ;
1D
1E  -->
1F


( COLLEEN TO DEVSYS COMMUNICATIONS WORDS --  SEND )
D303 CONSTANT PIADIR
D301 CONSTANT PIA
: NIBSEND
        ( SEND LOW NIBBLE OF    TOS )
        80 OR PIA C! ( SEND DATA )
        BEGIN
          PIA C@ 40 AND ( WAIT FOR  ACK )
        END
        0 PIA C! ( ACK-ACK)
        BEGIN
          PIA C@ 40 AND 0=
        END
        ( WAIT FOR ACK-ACK-ACK )
        ;
: DBSND
        ( TOS=BLOCK PTR )
        400 0
        DO
          I OVER + ( DATA PTR )
          C@ DUP NIBSEND ( LOW NIBBLE )
          10 / NIBSEND ( HIGH NIBBLE )
        LOOP
        DROP
        ;
: BSND
        ( SET UP PIA AND SEND A   BLOCK -- TOS = BLOCK NUMBER )
        PIADIR C@ FB AND PIADIR C! 8F PIA C!
        ( SET DATA DIRECTION)
        PIADIR C@ 4 OR PIADIR C! 0 PIA C!
        BEGIN
          PIA C@ 40 AND 0=
        END
        BLOCK DBSND ( SEND THE BLOCK )
        ;
( FRST LAST SMOV MOVE BLOCKS )
: SMOV
        1+ SWAP
        DO
          I . I BSND
        LOOP
```

```
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  24
00 ( COLLEEN TO DEVSYS COMMUNICATIO
01 NS WORDS -- RECEIVE )
02 CODE NIBRECV 0 # LDA, PIA   STA,
03  BEGIN, PIA  BIT, 0< UNTIL,
04 PIA  LDA, 0F #  AND, 40 #  LDY,
05  PIA STY,
06 BEGIN, PIA  BIT, 0< NOT UNTIL,
07  PHA, 0 #  LDA, PIA  STA,
08 PUSH JMP, C;
09  ( READY TO RECEIVE )
0A  : DBREC ( TOS - BLOCK POINTER )
0B  400 0 DO NIBRECV NIBRECV 10 * +
0C  ( GET A BYTE ) OVER I +
0D  C! ( AND STORE IT IN BUFFER )
0E  LOOP DROP ;
0F
10 : BREC ( SET UP PIA AND RECV A
11  BLOCK -- TOS = BLOCK NUMBER )
12  PIADIR C@ FB AND PIADIR C!
13  40 PIA C! ( ONLY SEND ACK BIT )
14  PIADIR C@ 4 OR PIADIR C!
15  BUFFER DBREC ( RECV THE BLOCK )
16  UPDATE FLUSH ( WRITE TO DISK )
17  ;
18 ( FRST LAST RMOV -- MOVE A SET
19  OF BLOCKS )
1A : RMOV 1+ SWAP DO I . I BREC
1B  LOOP ;  ;S
1C
1D
1E
1F


( COLLEEN TO DEVSYS COMMUNICATIONS WORDS -- RECEIVE )
CODE NIBRECV
        0 # LDA,
        PIA STA,
        BEGIN,
          PIA BIT,
          0<
        UNTIL,
        PIA LDA,
        0F # AND,
        40 # LDY,
        PIA STY,
        BEGIN,
          PIA BIT,
          0< NOT
        UNTIL,
        PHA,
        0 # LDA,
        PIA STA,
```

```
        PUSH JMP,
        C;
( READY TO RECEIVE )
: DBREC
        ( TOS - BLOCK POINTER )
        400 0
        DO
          NIBRECV NIBRECV 10 * + ( GET A BYTE )
          OVER I + C! ( AND STORE IT IN BUFFER )
        LOOP
        DROP
        ;
: BREC
        ( SET UP PIA AND RECV A   BLOCK -- TOS = BLOCK NUMBER )
        PIADIR C@ FB AND PIADIR C! 40 PIA C!
        ( ONLY SEND ACK BIT )
        PIADIR C@ 4 OR PIADIR C! BUFFER DBREC
        ( RECV THE BLOCK )
        UPDATE FLUSH ( WRITE TO DISK )
        ;
( FRST LAST RMOV -- MOVE A SET   OF BLOCKS )
: RMOV
        1+ SWAP
        DO
          I . I BREC
        LOOP
        ;
;S
```

* *

```
SCR #  25
00 ( LINE PRINTER WORDS 1/27/81
01  SRC )      3A50 VARIABLE P:
02  CODE PCIO XSAVE STX, 70 # LDX,
03 E456 JSR, XSAVE LDX,  TYA, PHA,
04 PUSH JMP, C; 0 VARIABLE LPCNT
05 : PERR? DUP 0< IF FF AND
06 ." P: ERROR "   ERROR THEN
07  DROP ;
08 : LPOPEN 3 3B2 C! P: 3B4 ! 2 3B8
09  ! 8 3BA ! PCIO PERR? ;
0A : LYP1 3B8 ! 3B4 ! 0B 3B2 C! PCI
0B O PERR? ; : LPEMIT SP@ 1 LYP1 DR
0C OP ; : LPCR 9B LPEMIT 1 LPCNT +!
0D  ; : LYPE DUP IF DUP 50 > IF
0E  1 LPCNT +! THEN LYP1 ELSE DROP
0F DROP THEN 20 SP@ 1 LYP1 DROP ;
10 : CRLP LPCR LPCNT @ 3D > IF
11 LPCR LPCR LPCR LPCR 0 LPCNT !
12 THEN ;
13 : FFLP CRLP BEGIN LPCNT @ WHILE
14 CRLP REPEAT ;
15 : SHRINK 1B LPEMIT 14 LPEMIT
16  CRLP ; : EXPAND 1B LPEMIT 13
17  LPEMIT CRLP ;
18 : .CLP 0 <# # # #> LYPE ;
19 : .LP 0 <# #S #>  LYPE ;
1A : LINELP DUP .CLP SCR @ (LINE)
```

```
1B  -TRAILING 1 MAX LYPE CRLP   ;
1C 4353 VARIABLE SCR# 2052 , 2023 ,
1D  : LISTLP DUP SCR ! SCR# 6 LYPE
1E .LP LPCR B/SCR B/BUF * C/L /
1F 0 DO I LINELP LOOP ; -->


( LINE PRINTER WORDS 1/27/81     SRC )
3A50 VARIABLE P:
CODE PCIO
        XSAVE STX,
        70 # LDX,
        E456 JSR,
        XSAVE LDX,
        TYA,
        PHA,
        PUSH JMP,
        C;
0 VARIABLE LPCNT
: PERR?
        DUP 0<
        IF
          FF AND ." P: ERROR " ERROR
        THEN
        DROP
        ;
: LPOPEN
        3 3B2 C! P: 3B4 ! 2 3B8 ! 8 3BA ! PCIO PERR?
        ;
: LYP1
        3B8 ! 3B4 ! 0B 3B2 C! PCIO PERR?
        ;
: LPEMIT
        SP@ 1 LYP1 DROP
        ;
: LPCR
        9B LPEMIT 1 LPCNT +!
        ;
: LYPE
        DUP
        IF
          DUP 50 >
          IF
            1 LPCNT +!
          THEN
          LYP1
        ELSE
          DROP DROP
        THEN
        20 SP@ 1 LYP1 DROP
        ;
: CRLP
        LPCR LPCNT @ 3D >
        IF
          LPCR LPCR LPCR LPCR 0 LPCNT !
        THEN
        ;
: FFLP
        CRLP
```

```
        BEGIN
          LPCNT @
        WHILE
          CRLP
        REPEAT
        ;
: SHRINK
        1B LPEMIT 14 LPEMIT CRLP
        ;
: EXPAND
        1B LPEMIT 13 LPEMIT CRLP
        ;
: .CLP
        0 <# # # #> LYPE
        ;
: .LP
        0 <# #S #> LYPE
        ;
: LINELP
        DUP .CLP SCR @ (LINE) -TRAILING 1 MAX LYPE CRLP
        ;
4353 VARIABLE SCR#
2052 , 2023 ,
: LISTLP
        DUP SCR ! SCR# 6 LYPE .LP LPCR B/SCR B/BUF * C/L / 0
        DO
          I LINELP
        LOOP
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  26
00 ( MORE LINE PRINTER WORDS
01  1/27/81  SRC              )
02 : LPSPC 0 DO 20 LPEMIT LOOP ;
03 : SHOWLP 1+ SWAP C/L 20 = IF
04   DO CRLP
05   SCR# 6 LYPE I .LP
06   1F LPSPC SCR# 6 LYPE I 1+
07   .LP CRLP
08   I 20 0 DO DUP SCR ! I .CLP
09   I SCR @ (LINE) LYPE
0A   5 LPSPC
0B   DUP 1+ SCR ! I LINELP LOOP
0C   DROP 2 +LOOP
0D   ELSE DO CRLP    I LISTLP LOOP
0E   ENDIF FFLP ;
0F
10 : LPINDEX 1+ SWAP DO I .LP
11 0 I (LINE) -TRAILING LYPE LPCR
12 LOOP ;
13 LPOPEN
14 ;S
15
16
17
18
```

```
19
1A
1B
1C
1D
1E
1F


( MORE LINE PRINTER WORDS        1/27/81  SRC             )
: LPSPC
        0
        DO
          20 LPEMIT
        LOOP
        ;
: SHOWLP
        1+ SWAP C/L 20 =
        IF
          DO
            CRLP SCR# 6 LYPE I .LP 1F LPSPC SCR# 6 LYPE I 1+
            .LP CRLP I 20 0
            DO
              DUP SCR ! I .CLP I SCR @ (LINE) LYPE 5 LPSPC DUP
              1+ SCR ! I LINELP
            LOOP
            DROP 2
          +LOOP
        ELSE
          DO
            CRLP I LISTLP
          LOOP
        ENDIF
        FFLP
        ;
: LPINDEX
        1+ SWAP
        DO
          I .LP 0 I (LINE) -TRAILING LYPE LPCR
        LOOP
        ;
LPOPEN ;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  27
00 ( FORMATTED LIST PROGRAM) : THAT
01  ; VOCABULARY FORMX IMMEDIATE
02 FORMX DEFINITIONS : CN CONSTANT
03 ;  : OCTAL 8 BASE ! ;
04   BASE @ OCTAL 40 CN SPACBYT 54
05 CN COMCHR : IARRAY 0 VARIABLE -2
06  ALLOT ; : 0> DUP 0= IF DROP  0
07 ELSE  0< 0= THEN ;
08 0 VARIABLE INDENT 106 CN FCONS
09 111 CN ICONS 0 VARIABLE TLFLG
0A 0 VARIABLE KERKNT 100 CN MAXLIN
0B : NXSPACE >R 1+ >R 0   R> R> DO
0C SPACBYT I C@ = IF DROP I LEAVE
```

```
0D THEN LOOP ; : NXNSPACE >R 1+ >R
0E 0 R> R> DO SPACBYT I C@ = 0= IF
0F
10  DROP I LEAVE THEN LOOP ; : GTNX
11 WD DUP IF + OVER SWAP NXSPACE
12 ELSE DROP THEN DUP IF OVER SWAP
13 NXNSPACE DUP IF OVER OVER
14 NXSPACE DUP IF OVER - ELSE DROP
15 OVER OVER - 1+ THEN ELSE DUP
16 THEN ELSE DUP THEN ; : TORLCR TL
17 FLG @ IF CRLP ELSE CR THEN KERKN
18 T 0SET ; : TORLY DUP 1+ KERKNT +
19 ! TLFLG @ IF LYPE ELSE TYPE SPAC
1A E THEN ; : DOIND INDENT @ 0> IF
1B INDENT @ 0 DO 0 0 TORLY LOOP THE
1C N ; : PRWORD DUP 1+ KERKNT @ + M
1D AXLIN > IF TORLCR THEN KERKNT @
1E 0= IF DOIND THEN OVER OVER TORLY
1F  ; : 1SET 1 SWAP ! ;  -->


( FORMATTED LIST PROGRAM)
: THAT
        ;
VOCABULARY FORMX IMMEDIATE FORMX DEFINITIONS
: CN
        CONSTANT ;
: OCTAL
        8 BASE !
        ;
BASE @ OCTAL 40 CN SPACBYT
54 CN COMCHR
: IARRAY
        0 VARIABLE -2
ALLOT
;
: 0>
        DUP 0=
        IF
          DROP 0
        ELSE
          0< 0=
        THEN
        ;
0 VARIABLE INDENT
106 CN FCONS
111 CN ICONS
0 VARIABLE TLFLG
0 VARIABLE KERKNT
100 CN MAXLIN
: NXSPACE
        >R 1+ >R 0 R> R>
        DO
          SPACBYT I C@ =
          IF
            DROP I LEAVE
          THEN
        LOOP
        ;
```

```
: NXNSPACE
        >R 1+ >R 0 R> R>
        DO
          SPACBYT I C@ = 0=
          IF
            DROP I LEAVE
          THEN
        LOOP
        ;
: GTNXWD
        DUP
        IF
          + OVER SWAP NXSPACE
        ELSE
          DROP
        THEN
        DUP
        IF
          OVER SWAP NXNSPACE DUP
          IF
            OVER OVER NXSPACE DUP
            IF
              OVER -
            ELSE
              DROP OVER OVER - 1+
            THEN
          ELSE
            DUP
          THEN
        ELSE
          DUP
        THEN
        ;
: TORLCR
        TLFLG @
        IF
          CRLP
        ELSE
          CR
        THEN
        KERKNT 0SET
        ;
: TORLY
        DUP 1+ KERKNT +! TLFLG @
        IF
          LYPE
        ELSE
          TYPE SPACE
        THEN
        ;
: DOIND
        INDENT @ 0>
        IF
          INDENT @ 0
          DO
            0 0 TORLY
          LOOP
        THEN
        ;
```

```
: PRWORD
        DUP 1+ KERKNT @ + MAXLIN >
        IF
          TORLCR
        THEN
        KERKNT @ 0=
        IF
          DOIND
        THEN
        OVER OVER TORLY
        ;
: 1SET
        1 SWAP !
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  28
00 : ( 51 WORD 6 ALLOT ;
01
02 : IA IARRAY ; IA L1G 10  , ( :)
03 ( CODE) ( ,CODE) ( SUBROUTINE)
04  ( IA) ( IARRAY) ( LABEL) ( TBL)
05  IA L2G 2 , ( ;) ( C;)
06  IA L3G 2 , ( NXT,) ( NEXT,) IA
07 L4G 6 , ( IF) ( DO) ( IF,)
08   ( CASE) ( BEGIN) ( BEGIN,) IA
09 L5G 3 , ( ELSE,) ( ELSE)
0A   ( WHILE) IA L6G 16 , ( THEN,)
0B ( THEN) ( END,) ( END) ( SOB,)
0C ( BACK) ( UNTIL) ( AGAIN) ( REPE
0D AT) ( ENDIF,)
0E   ( UNTIL,) ( LOOP) ( +LOOP) ( E
0F NDIF) IA L7G 7 , ( CONSTANT)
10 ( IR)   ( VARIABLE)       ( CN)
11 ( ARRAY) ( INTEGER) ( ORCON)
12    IA L8G 1 , ( () IA L9G 3 , (
13 LD,) ( ST,) ( LOAD)
14 IA LAG 1 , ( ;CODE)
15
16 : CMPWORD DUP >R C@ OVER = R>
17 SWAP IF >R OVER
18   R> SWAP OVER DUP C@ DUP 4 > IF
19  DROP 4 THEN 0
1A   DO I OVER + 1+ C@ >R OVER R>
1B SWAP I + C@
1C   = 0= IF 0 LEAVE THEN LOOP
1D
1E   0= IF DROP DROP 0 THEN ELSE 0
1F THEN ;   -->


: (
        51 WORD 6 ALLOT
        ;
: IA
IARRAY ;
IA L1G
```

```
        10 , ( :)
        ( CODE)
        ( ,CODE)
        ( SUBROUTINE)
        ( IA)
        ( IARRAY)
        ( LABEL)
        ( TBL)
IA L2G
        2 , ( ;)
        ( C;)
IA L3G
        2 , ( NXT,)
        ( NEXT,)
IA L4G
        6 , ( IF)
        ( DO)
        ( IF,)
        ( CASE)
        ( BEGIN)
        ( BEGIN,)
IA L5G
        3 , ( ELSE,)
        ( ELSE)
        ( WHILE)
IA L6G
        16 , ( THEN,)
        ( THEN)
        ( END,)
        ( END)
        ( SOB,)
        ( BACK)
        ( UNTIL)
        ( AGAIN)
        ( REPEAT)
        ( ENDIF,)
        ( UNTIL,)
        ( LOOP)
        ( +LOOP)
        ( ENDIF)
IA L7G
        7 , ( CONSTANT)
        ( IR)
        ( VARIABLE)
        ( CN)
        ( ARRAY)
        ( INTEGER)
        ( ORCON)
IA L8G
        1 , ( ()
IA L9G
        3 , ( LD,)
        ( ST,)
        ( LOAD)
IA LAG
        1 , ( ;CODE)
: CMPWORD
        DUP >R C@ OVER = R> SWAP
        IF
```

```
          >R OVER R> SWAP OVER DUP C@ DUP 4 >
          IF
            DROP 4
          THEN
          0
          DO
            I OVER + 1+ C@ >R OVER R> SWAP I + C@ = 0=
            IF
              0 LEAVE
            THEN
          LOOP
          0=
          IF
            DROP DROP 0
          THEN
        ELSE
          0
        THEN
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  29
00 : GSCAN DUP @ SWAP 2+ SWAP 0 DO
01 CMPWORD IF LEAVE
02   0 ELSE 6 + THEN LOOP IF 0 ELSE
03  DROP 1 THEN ;
04 : NEWCR KERKNT @ IF TORLCR THEN
05 ;
06 : DUPBC OVER >R >R OVER R> SWAP
07 R> ;
08 : FINDCHAR SWAP >R SWAP 1+ R>
09 DO DUP I C@ =
0A   IF DROP I LEAVE 0 THEN LOOP IF
0B  0 THEN ;
0C : PRNEWL PRWORD TORLCR ;
0D
0E : >= OVER OVER = IF DROP DROP
0F 1 ELSE > THEN ;   -->
10
11
12
13
14
15
16
17
18
19
1A
1B
1C
1D
1E
1F


: GSCAN
```

```
        DUP @ SWAP 2+ SWAP 0
        DO
          CMPWORD
          IF
            LEAVE 0
          ELSE
            6 +
          THEN
        LOOP
        IF
          0
        ELSE
          DROP 1
        THEN
        ;
: NEWCR
        KERKNT @
        IF
          TORLCR
        THEN
        ;
: DUPBC
        OVER >R >R OVER R> SWAP R>
        ;
: FINDCHAR
        SWAP >R SWAP 1+ R>
        DO
          DUP I C@ =
          IF
            DROP I LEAVE 0
          THEN
        LOOP
        IF
          0
        THEN
        ;
: PRNEWL
        PRWORD TORLCR
        ;
: >=
        OVER OVER =
        IF
          DROP DROP 1
        ELSE
          >
        THEN
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  2A
00 : EL1G NEWCR INDENT 0SET PRWORD
01 GTNXWD PRNEWL
02   10 INDENT ! ;
03
04 : EL2G NEWCR PRNEWL INDENT 0SET
05 ;
06 : EL3G NEWCR PRNEWL ;
```

```
07
08 : EL4G NEWCR PRNEWL 2 INDENT +!
09 ;
0A : EL5G NEWCR -2 INDENT +! PRNEWL
0B  2 INDENT +! ;
0C : EL6G NEWCR -2 INDENT +! PRNEWL
0D  ;
0E : EL7G PRWORD GTNXWD PRNEWL INDE
0F NT 0SET ;
10 : EL8G DUPBC 51 FINDCHAR DUP
11
12   IF SWAP DROP OVER - 1+ PRNEWL
13 ELSE DROP PRWORD THEN ;
14 : EL9G PRNEWL ;
15
16 : ELAG NEWCR 10 INDENT ! PRNEWL
17 ;
18 : ASSWRD DUP 4 >= IF OVER OVER +
19  1- C@ COMCHR = IF
1A    OVER DUP C@ ICONS =  SWAP 1+
1B C@ FCONS = AND
1C   IF 2 ELSE 1 THEN ELSE 0 THEN E
1D LSE 0 THEN ;
1E -->
1F
```

```
: EL1G
        NEWCR INDENT 0SET PRWORD GTNXWD PRNEWL 10 INDENT !
        ;
: EL2G
        NEWCR PRNEWL INDENT 0SET
        ;
: EL3G
        NEWCR PRNEWL
        ;
: EL4G
        NEWCR PRNEWL 2 INDENT +!
        ;
: EL5G
        NEWCR -2 INDENT +! PRNEWL 2 INDENT +!
        ;
: EL6G
        NEWCR -2 INDENT +! PRNEWL
        ;
: EL7G
        PRWORD GTNXWD PRNEWL INDENT 0SET
        ;
: EL8G
        DUPBC 51 FINDCHAR DUP
        IF
          SWAP DROP OVER - 1+ PRNEWL
        ELSE
          DROP PRWORD
        THEN
        ;
: EL9G
        PRNEWL
        ;
```

```
: ELAG
        NEWCR 10 INDENT ! PRNEWL
        ;
: ASSWRD
        DUP 4 >=
        IF
          OVER OVER + 1- C@ COMCHR =
          IF
            OVER DUP C@ ICONS = SWAP 1+ C@ FCONS = AND
            IF
              2
            ELSE
              1
            THEN
          ELSE
            0
          THEN
        ELSE
          0
        THEN
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  2B
00 : PRCWRD L1G GSCAN IF EL1G ELSE
01 L2G GSCAN IF EL2G ELSE
02   L3G GSCAN IF EL3G ELSE L4G GSC
03 AN IF EL4G ELSE L5G GSCAN
04   IF EL5G ELSE L6G GSCAN IF EL6G
05  ELSE L7G GSCAN IF EL7G
06   ELSE L8G GSCAN IF EL8G ELSE L9
07 G GSCAN IF EL9G ELSE
08   LAG GSCAN IF ELAG ELSE ASSWRD
09 IF ASSWRD 2 =
0A   IF EL4G ELSE PRNEWL THEN ELSE
0B PRWORD
0C   THEN THEN THEN THEN THEN THEN
0D THEN THEN THEN THEN THEN ;
0E : FORLST TORLCR DUP TLFLG @ IF L
0F ISTLP ELSE
10  TORLCR LIST THEN TORLCR TORLCR
11 DUP BLK !
12   BLOCK DUP 1777 + SWAP KERKNT 0
13 SET INDENT 0SET 0 BEGIN GTNXWD
14   DUP IF PRCWRD THEN DUP 0= END
15 DROP DROP DROP BLK 0SET ;
16 : ASTER TORLCR 40 0 DO 52 SP@ 1
17 TORLY DROP LOOP TORLCR ;
18 : FORSHW 1+ OVER DO ASTER I FORL
19 ST TORLCR LOOP DROP ;
1A FORTH DEFINITIONS : FLST FORMX T
1B LFLG 0SET FORLST ; : FLSTLP FORM
1C X TLFLG 1SET FORLST FFLP ; : FSH
1D W FORMX TLFLG 0SET FORSHW ; : FS
1E HWLP FORMX  TLFLG 1SET FORSHW
1F FFLP  ; BASE !  ;S
```

```
: PRCWRD
        L1G GSCAN
        IF
          EL1G
        ELSE
          L2G GSCAN
          IF
            EL2G
          ELSE
            L3G GSCAN
            IF
              EL3G
            ELSE
              L4G GSCAN
              IF
                EL4G
              ELSE
                L5G GSCAN
                IF
                  EL5G
                ELSE
                  L6G GSCAN
                  IF
                    EL6G
                  ELSE
                    L7G GSCAN
                    IF
                      EL7G
                    ELSE
                      L8G GSCAN
                      IF
                        EL8G
                      ELSE
                        L9G GSCAN
                        IF
                          EL9G
                        ELSE
                          LAG GSCAN
                          IF
                            ELAG
                          ELSE
                            ASSWRD
                            IF
                              ASSWRD 2 =
                              IF
                                EL4G
                              ELSE
                                PRNEWL
                              THEN
                            ELSE
                              PRWORD
                            THEN
                          THEN
                        THEN
                      THEN
                    THEN
                  THEN
                THEN
```

```
              THEN
            THEN
          THEN
        THEN
        ;
: FORLST
        TORLCR DUP TLFLG @
        IF
          LISTLP
        ELSE
          TORLCR LIST
        THEN
        TORLCR TORLCR DUP BLK ! BLOCK DUP 1777 + SWAP KERKNT
        0SET INDENT 0SET 0
        BEGIN
          GTNXWD DUP
          IF
            PRCWRD
          THEN
          DUP 0=
        END
        DROP DROP DROP BLK 0SET
        ;
: ASTER
        TORLCR 40 0
        DO
          52 SP@ 1 TORLY DROP
        LOOP
        TORLCR
        ;
: FORSHW
        1+ OVER
        DO
          ASTER I FORLST TORLCR
        LOOP
        DROP
        ;
FORTH DEFINITIONS
: FLST
        FORMX TLFLG 0SET FORLST
        ;
: FLSTLP
        FORMX TLFLG 1SET FORLST FFLP
        ;
: FSHW
        FORMX TLFLG 0SET FORSHW
        ;
: FSHWLP
        FORMX TLFLG 1SET FORSHW FFLP
        ;
BASE ! ;S

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  2C
00 ( RS232 SUPPORT )
01
02 CODE SIO XSAVE STX, BOT LDA, E45
03 9      JSR, ( SIOV) XSAVE LDX,
```

```
04  BOT STA, BOT 1+ STY, NEXT JMP,
05 C;
06 : SERR DUP 0< IF 0 100 U/ BASE @
07         DECIMAL ." SIO ERROR "
08   . BASE ! QUIT ELSE DROP THEN ;
09
0A CODE DORL XSAVE STX, 506 JSR,
0B
0C  HERE 8 + JSR, XSAVE LDX, NEXT
0D        JMP, 0C ) JMP, C;
0E : GETR: HERE 2E7 ! ( SET MEMLO )
0F        FLUSH EMPTY-BUFFERS
10  150  300 !  ( DDEVIC,DUNIT)
11
12  403F 302 !  ( ? CMD,EXPECT DATA
13 )      5    306 C! ( TIMEOUT)
14  500  304 !  ( BUFFER ADDR)
15
16  0C   308 !  ( LENGTH )
17
18  0    30A !  ( AUXES )
19        0 SIO SERR ( ERRORS?)
1A  500 300 0C CMOVE 0 SIO SERR DOR
1B L
1C  ( RUN RELOCATOR ) 2E7 @ HERE -
1D
1E  ALLOT HERE FENCE ! ; -->
1F


( RS232 SUPPORT )
CODE SIO
        XSAVE STX,
        BOT LDA,
        E459 JSR,
        ( SIOV)
        XSAVE LDX,
        BOT STA,
        BOT 1+ STY,
        NEXT JMP,
        C;
: SERR
        DUP 0<
        IF
          0 100 U/ BASE @ DECIMAL ." SIO ERROR " . BASE ! QUIT
        ELSE
          DROP
        THEN
        ;
CODE DORL
        XSAVE STX,
        506 JSR,
        HERE 8 + JSR,
        XSAVE LDX,
        NEXT JMP,
        0C ) JMP,
        C;
: GETR:
        HERE 2E7 ! ( SET MEMLO )
```

```
        FLUSH EMPTY-BUFFERS 150 300 ! ( DDEVIC,DUNIT)
        403F 302 ! ( ? CMD,EXPECT DATA)
        5 306 C! ( TIMEOUT)
        500 304 ! ( BUFFER ADDR)
        0C 308 ! ( LENGTH )
        0 30A ! ( AUXES )
        0 SIO SERR ( ERRORS?)
        500 300 0C CMOVE 0 SIO SERR DORL ( RUN RELOCATOR )
        2E7 @ HERE - ALLOT HERE FENCE !
        ;
-->

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SCR #  2D
00 ( RS232 )
01
02 : R1: " R1: " DROP ;
03
04 : R1OPEN 0 8 R1: OPEN ICSTA CH?
05 ;
06 : RYPE -DUP IF 1 IOCB 0B ICCOM C
07 !       ICBLL ! ICBAL ! CIO
08   20 ICCOM C! 0 I1CAX ! CIO ELSE
09         DROP THEN ;
0A : CRR 0A9B SP@ 2 RYPE DROP ;
0B       : REMIT SP@ 1 RYPE DROP ;
0C : SET9600 1 IOCB 0E I1CAX ! 24
0D         ICCOM C! R1: ICBAL !
0E   CIO ICSTA CH? ;
0F
10 : LINER SCR @ (LINE) -TRAILING
11         RYPE ;
12 100 VARIABLE LSPD
13
14 : LISTR DUP SCR ! CRR " SCR#" RY
15 PE      0 <# #S #> RYPE CRR 10 0
16   DO I 0 <# # # #> RYPE I LINER
17         CRR LOOP ;
18  ;S
19
1A
1B
1C
1D
1E
1F


( RS232 )
: R1:
        " R1: " DROP
        ;
: R1OPEN
        0 8 R1: OPEN ICSTA CH?
        ;
: RYPE
        -DUP
        IF
```

```
          1 IOCB 0B ICCOM C! ICBLL ! ICBAL ! CIO 20 ICCOM C! 0
          I1CAX ! CIO
        ELSE
          DROP
        THEN
        ;
: CRR
        0A9B SP@ 2 RYPE DROP
        ;
: REMIT
        SP@ 1 RYPE DROP
        ;
: SET9600
        1 IOCB 0E I1CAX ! 24 ICCOM C! R1: ICBAL ! CIO ICSTA CH?
        ;
: LINER
        SCR @ (LINE) -TRAILING RYPE
        ;
100 VARIABLE LSPD
: LISTR
        DUP SCR ! CRR " SCR#" RYPE 0 <# #S #> RYPE CRR 10 0
        DO
          I 0 <# # # #> RYPE I LINER CRR
        LOOP
        ;
;S
```

'Screens'

```
SCR # 14
  0 ( ERROR MESSAGES )
  1 Stack empty
  2 Dictionary full
  3 Wrong address mode
  4 Isn't unique
  5 Value error
  6 Disk address error
  7 Stack full
  8 Disk Error!
  9
 10
 11
 12
 13
 14
 15

SCR # 15
  0 ( ERROR MESSAGES )
  1 Use only in Definitions
  2 Execution only
  3 Conditionals not paired
  4 Definition not finished
  5 In protected dictionary
  6 Use only when loading
  7 Off current screen
  8 Declare VOCABULARY
  9
 10
```

```
 11
 12
 13
 14
 15

SCR # 16
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15

SCR # 17
  0 ( CASSETTE LOAD )
  1
  2
  3
  4 ( LOAD DEBUG )
  5    21 LOAD
  6
  7 ( LOAD ASSEMBLER )
  8    39 LOAD
  9
 10
 11
 12
 13 ;S
 14
 15

SCR # 18
  0 ( FULL LOAD )
  1
  2
  3
  4 ( LOAD DEBUG )
  5    21 LOAD
  6
  7 ( LOAD EDITOR )
  8   27 LOAD
  9
 10 ( LOAD ASSEMBLER )
 11    39 LOAD
 12
 13 ;S
 14
 15
```

```
SCR # 19
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15

SCR # 20
  0 ( ATARI FORTH DEFS )
  1     BASE @ HEX
  2
  3 : PON    1 PFLAG ! ; ( PRT ON )
  4 : POFF   0 PFLAG ! ; ( PRT OFF )
  5
  6 : BEEP  0C0 0 DO
  7    08 0D01F C! 6 0 DO LOOP
  8    00 0D01F C! 6 0 DO LOOP
  9    LOOP ;
 10
 11 : ASCII BL WORD HERE 1+ C@
 12  STATE @ IF COMPILE CLIT C,
 13  THEN ;  IMMEDIATE
 14
 15 BASE ! ;S

SCR # 21
  0 ( DEBUGGER AIDS -- DUMP , CDUMP )
  1
  2 BASE @ HEX
  3
  4
  5
  6
  7
  8 : H. BASE @ HEX OVER U. BASE ! ;
  9
 10 : B?   BASE @ DUP DECIMAL . BASE ! ;
 11 : FREE   2E5 @ HERE -  U. ." bytes" CR ;
 12
 13
 14 -->
 15

SCR # 22
  0 ( DEBUGGER AIDS -- DUMP , CDUMP )
  1     DECIMAL
  2 : ?EXIT  ?TERMINAL
```

```
  3          IF LEAVE ENDIF ;
  4 : U.R    0 SWAP D.R ;
  5 : LDMP  DUP 8 + SWAP DO I C@ 4 .R
  6          LOOP ;
  7 : DUMP   OVER + SWAP DO CR I 5 U.R I
  8          LDMP ?EXIT 8 +LOOP CR ;
  9 : CDMP   DUP 16 + SWAP DO
 10          I C@ EMIT LOOP ;
 11  HEX
 12 : CDUMP   OVER + SWAP DO CR I 5 U.R I
 13       SPACE 1 2FE C! CDMP 0 2FE C!
 14        ?EXIT 10 +LOOP CR ;
 15  DECIMAL -->

SCR # 23
  0 ( STACK PRINTER )
  1
  2 HEX
  3
  4 : DEPTH SP@ 12 +ORIGIN @ SWAP - 2 / ;
  5 : S.  ( PRINTS THE STACK )
  6    DEPTH -DUP IF
  7       0 DO  CR  ." TOP+" I .
  8       SP@ I 2 * + @ U. LOOP
  9    ELSE ." Stack Empty" THEN CR ;
 10
 11
 12
 13 BASE !
 14
 15 -->

SCR # 24
  0 ( DEFINITION TRACER )
  1      BASE @ HEX
  2 0 VARIABLE .WORD
  3 ' CLIT CFA CONSTANT .CLIT
  4 ' 0BRANCH CFA CONSTANT ZBRAN
  5 ' BRANCH CFA CONSTANT BRAN
  6 ' ;S CFA CONSTANT SEMIS
  7 ' (LOOP) CFA CONSTANT PLOOP
  8 ' (+LOOP) CFA CONSTANT PPLOOP
  9 ' (.") CFA CONSTANT PDOTQ
 10 : PWORD 2+ NFA ID.  ;
 11 : 1BYTE  PWORD .WORD @ C@ . 1 .WORD +! ;
 12 : 1WORD PWORD .WORD @ @  . 2 .WORD +! ;
 13 : NP DUP SEMIS = IF PWORD CR CR
 14     PROMPT QUIT THEN ?TERMINAL IF
 15      PROMPT QUIT THEN ;    -->

SCR # 25
  0 ( DEFINITION TRACER )
  1
  2 : BRNCH PWORD ." to " .WORD @ .WORD @ @ + . 2 .WORD +! ;
  3
  4 : STG  PWORD 22 EMIT .WORD @   DUP COUNT TYPE 22 EMIT
  5    C@ .WORD @ + 1+ .WORD ! ;
  6
  7 ' LIT CFA CONSTANT .LIT
```

```
  8
  9 : CKIT DUP ZBRAN = OVER BRAN =
 10  OR OVER PLOOP = OR OVER PPLOOP =
 11 OR IF BRNCH ELSE DUP .LIT =
 12 IF 1WORD ELSE DUP .CLIT =
 13  IF 1BYTE ELSE DUP PDOTQ = IF STG
 14  ELSE PWORD THEN THEN THEN THEN ;
 15 -->

SCR # 26
  0 ( DEFINITION TRACER )
  1     ' : 12 + CONSTANT DOCOL
  2
  3 : T?PR CR CR ." Primitive" CR CR ;
  4 : ?DOCOL DUP 2 - @ DOCOL -  IF
  5   T?PR PROMPT QUIT THEN ;
  6
  7 : .SETUP ~[COMPILE] ' ?DOCOL .WORD ! ;
  8
  9 : NXT1  .WORD @ U. 2 SPACES .WORD
 10     @ @ 2 .WORD +! ;
 11
 12 : DECOMP .SETUP CR CR BEGIN NXT1 NP
 13       CKIT CR AGAIN ;
 14
 15 BASE !     ;S

SCR # 27
  0 ( ** EDITOR ** )
  1
  2  BASE @  HEX
  3
  4 ( THIS EDITOR IS PATTERNED AFTER
  5 ( THE EXAMPLE EDITOR IN THE fig
  6 ( "INSTALLATION MANUAL" 8/80 WFR
  7
  8 : TEXT HERE C/L 1+ BLANKS WORD
  9        HERE PAD C/L 1+ CMOVE ;
 10
 11 : LINE DUP FFF0 AND 17 ?ERROR SCR
 12        @ (LINE) DROP ;
 13
 14 : MARK  10 0 DO I LINE UPDATE
 15     DROP LOOP ;        -->

SCR # 28
  0 ( EDITOR )
  1 VOCABULARY EDITOR IMMEDIATE
  2 : WHERE DUP B/SCR / DUP SCR ! ." SCR # " DECIMAL .
  3 SWAP C/L /MOD C/L * ROT BLOCK + CR C/L -TRAILING TYPE CR HERE
  4 C@ - SPACES 1 2FE C! 1C EMIT 0 2FE C! ~[COMPILE] EDITOR QUIT ;
  5
  6 EDITOR DEFINITIONS
  7
  8 : #LOCATE R# @ C/L /MOD ;
  9 : #LEAD #LOCATE LINE SWAP ;
 10 : #LAG  #LEAD DUP >R + C/L R> - ;
 11
 12
```

```
 13 : -MOVE LINE C/L CMOVE UPDATE ;
 14
 15 -->

SCR # 29
  0 ( EDITOR )
  1 : H LINE PAD 1+ C/L DUP PAD C!
  2      CMOVE ;
  3 : E LINE C/L BLANKS UPDATE ;
  4 : S DUP 1 - 0E DO I LINE I 1+
  5     -MOVE -1 +LOOP E ;
  6 : D DUP H 0F DUP ROT
  7     DO I 1+ LINE I -MOVE LOOP E ;
  8
  9
 10    -->
 11
 12
 13
 14
 15

SCR # 30
  0 ( EDITOR )
  1
  2 : M   R# +! CR SPACE #LEAD TYPE
  3       17 EMIT #LAG TYPE #LOCATE
  4       . DROP ;
  5 : T   DUP C/L * R# ! DUP H 0 M ;
  6 : L   SCR @ LIST 0 M ;
  7 : R   PAD 1+ SWAP -MOVE ;
  8 : P   1 TEXT R ;
  9 : I   DUP S R ;
 10 : TOP   0 R# ! ;
 11
 12
 13   -->
 14
 15

SCR # 31
  0 ( EDITOR )
  1
  2
  3 : CLEAR  SCR ! 10 0 DO FORTH I
  4          EDITOR E LOOP ;
  5
  6
  7
  8
  9
 10 : COPY   B/SCR * OFFSET @ + SWAP
 11          B/SCR * B/SCR OVER +
 12          SWAP DO DUP FORTH I
 13          BLOCK 2 - ! 1+ UPDATE
 14          LOOP DROP FLUSH ;
 15   -->

SCR # 32
```

```
 0 ( EDITOR )
 1
 2 : 1LINE   #LAG PAD COUNT MATCH R#
 3           +! ;
 4
 5
 6 : FIND   BEGIN 3FF R# @ < IF TOP
 7          PAD HERE C/L 1+ CMOVE 0
 8          ERROR ENDIF 1LINE UNTIL
 9          ;
10
11 : DELETE   >R #LAG + FORTH R -
12         #LAG R MINUS R# +! #LEAD
13         + SWAP CMOVE R> BLANKS
14         UPDATE ;
15 -->

SCR # 33
 0 ( EDITOR )
 1
 2 : N   FIND 0 M ;
 3
 4 : F   1 TEXT N ;
 5
 6 : B   PAD C@ MINUS M ;
 7
 8 : X   1 TEXT FIND PAD C@ DELETE
 9       0 M ;
10
11 : TILL   #LEAD + 1 TEXT 1LINE 0=
12          0 ?ERROR #LEAD + SWAP -
13          DELETE 0 M ;
14
15 -->

SCR # 34
 0 ( END OF EDITOR )
 1
 2 : C   1 TEXT PAD COUNT #LAG ROT
 3       OVER MIN >R FORTH R R# +!
 4       R - >R DUP HERE R CMOVE
 5       HERE #LEAD + R> CMOVE R>
 6       CMOVE UPDATE 0 M ;
 7
 8
 9 FORTH DEFINITIONS DECIMAL
10
11 LATEST 12 +ORIGIN !
12 HERE 28 +ORIGIN !
13 HERE 30 +ORIGIN !
14 ' EDITOR 6 + 32 +ORIGIN !
15 HERE FENCE !    BASE !    ;S

SCR # 35
 0
 1
 2
 3
 4
```

```
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15

SCR # 36
  0 ( DISK COPY ROUTINE  32K RAM )
  1
  2    BASE @ DECIMAL
  3 16384 CONSTANT BUFHEAD
  4 0 VARIABLE BLK#  0 VARIABLE ADRS
  5 : GET   ADRS @ BLK# @ ;
  6 : RD    GET DUP 718 = IF LEAVE THEN 1 R/W ;
  7 : WRT   GET DUP 718  = IF LEAVE THEN  0 R/W ;
  8 : +BLK  1 BLK# +!  128 ADRS +! ;
  9 : DSETUP   BLK# ! BUFHEAD ADRS ! ;
 10 : GKEY ." HIT ANY KEY "  KEY CR DROP ;
 11 : RDIN  CR ." Insert SOURCE disk  "  GKEY DSETUP
 12 90 0 DO RD +BLK  LOOP ;
 13 : WRTO  CR ." Insert DESTINATION disk  "  GKEY DSETUP
 14 90 0 DO WRT +BLK  LOOP ;
 15 -->

SCR # 37
  0 ( DISK COPY ROUTINE )
  1
  2 ( INSERT SOURCE DISK IN DRIVE #1
  3
  4 ( SIMPLY TYPE  "DISKCOPY" !
  5
  6 : MS1 CR CR
  7 ." SINGLE-DRIVE DISK COPY" CR CR ;
  8
  9
 10 : %COPY      0 DO I 90 *
 11              DUP DUP RDIN WRTO
 12              90 +  . LOOP ;
 13 : DISKCOPY   CR MS1 CR 8 %COPY ;
 14
 15 BASE !   ;S

SCR # 38
  0
  1
  2
  3
  4
  5
  6
  7
  8
  9
```

```
 10
 11
 12
 13
 14
 15

SCR # 39
  0 ( ** ASSEMBLER **   IN FORTH )
  1
  2 (  ASSEMBLER COMFORMS TO THE
  3 ( fig "INSTALLATION GUIDE" WITH
  4 ( THE FOLLOWING EXCEPTIONS:
  5
  6 (  SHIFTS ARE: "XXX.A" FOR A-REG.
  7 ( SHIFTS.
  8 (  CONDITIONAL BRANCHES ARE
  9 (  PATTERNED AFTER THE BRANCH OP-
 10 (  CODES:   "IFEQ," IS USED IN-
 11 ( STEAD OF "0= IF," FOR BETTER
 12 ( CLARITY.  SEE SCREEN 43.
 13
 14
 15 -->

SCR # 40
  0 ( ASSEMBLER )
  1
  2 VOCABULARY ASSEMBLER IMMEDIATE
  3
  4 BASE @  HEX
  5
  6 : CODE  ~[COMPILE] ASSEMBLER
  7         CREATE SMUDGE ;
  8
  9 ASSEMBLER DEFINITIONS
 10
 11 : SB  <BUILDS C, DOES> @ C, ;
 12       ( SINGLE BYTE OPERATORS)
 13
 14
 15 -->

SCR # 41
  0 ( ASSEMBLER )
  1
  2 00 SB BRK, 18 SB CLC, D8 SB CLD,
  3 58 SB CLI, B8 SB CLV, CA SB DEX,
  4 88 SB DEY, E8 SB INX, C8 SB INY,
  5 EA SB NOP, 48 SB PHA, 08 SB PHP,
  6 68 SB PLA, 28 SB PLP, 40 SB RTI,
  7 60 SB RTS, 38 SB SEC, F8 SB SED,
  8 78 SB SEI, A8 SB TAX, BA SB TSX,
  9 8A SB TXA, 9A SB TXS, 98 SB TYA,
 10
 11 0A SB ASL.A,   2A SB ROL.A,
 12 4A SB LSR.A,   6A SB ROR.A,
 13
 14 : NOT  0= ;  ( REVERSE LOGICAL )
```

```
 15 : 0=  1 ; ( PUSH A TRUE )  -->

SCR # 42
  0 ( ASSEMBLER )
  1
  2 : 3BY  <BUILDS C, DOES> @ C, , ;
  3
  4 4C 3BY JMP,   6C 3BY JMP(),
  5 20 3BY JSR,
  6
  7 : ?ER5    5 ?ERROR ;
  8
  9 : IF.  <BUILDS C, DOES> C@ C, 0
 10         C, HERE ;
 11 : THEN,   DUP HERE SWAP - DUP
 12         7F > ?ER5 DUP -80 < ?ER5
 13   SWAP -1 + C! ;  IMMEDIATE
 14 : ENDIF, ~[COMPILE] THEN, ; IMMEDIATE
 15 -->

SCR # 43
  0 ( ASSEMBLER )
  1
  2 30 IF. IFPL,  ( BPL )
  3 10 IF. IFMI,  ( BMI )
  4 70 IF. IFVC,  ( BVC )
  5 50 IF. IFVS,  ( BVS )
  6 B0 IF. IFCC,  ( BCC )
  7 90 IF. IFCS,  ( BCS )
  8 F0 IF. IFNE,  ( BNE )
  9 D0 IF. IFEQ,  ( BEQ )
 10
 11 : BEGIN,   HERE ; IMMEDIATE
 12 : END,   IF D0 ELSE F0 THEN  C,
 13          HERE 1+ - DUP
 14         -80 < ?ER5 C, ; IMMEDIATE
 15 : UNTIL,  ~[COMPILE]  END,  ; IMMEDIATE        -->

SCR # 44
  0 ( ASSEMBLER )
  1
  2 0D VARIABLE MODE ( ABS. MODE )
  3
  4 : MODE=   MODE @ = ; ( CK MODE )
  5 : 256<   DUP 100 ( HEX) U< ;
  6 : MODEFIX  256< IF -08 MODE +!
  7            THEN ;
  8      ( MODE=MODE-8 IF ADR<256 )
  9 : CKMODE   MODE= IF MODEFIX
 10                  THEN ;
 11 : M0 <BUILDS C, DOES> SWAP
 12      0D CKMODE  1D CKMODE SWAP
 13      C@ MODE @ OR C,    256< IF
 14      C, ELSE , THEN 0D MODE ! ;
 15  DECIMAL  46 LOAD    ;S

SCR # 45
  0 B jDISKNAMEDAT
  1
```

```
  2 APX-20029fig-FORTH 1.1 Rev. 2.0Patrick L. Mullarky01/15/821
  3                                                               J
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15

SCR # 46
  0 ( ASSEMBLER )
  1     HEX
  2 : X)  01 MODE ! ;   ( ~[ADDR,X]  )
  3 : #   09 MODE ! ;   ( IMMEDIATE )
  4 : )Y  11 MODE ! ;   ( ~[ADDR],Y  )
  5 : ,X  1D MODE ! ;   ( ADDR,X    )
  6 : ,Y  19 MODE ! ;   ( ADDR,Y    )
  7
  8
  9 00 M0 ORA, 20 M0 AND, 40 M0 EOR,
 10 60 M0 ADC, 80 M0 STA, A0 M0 LDA,
 11 C0 M0 CMP, E0 M0 SBC,
 12
 13 : BIT,   256< IF 24 C, C, ELSE
 14          2C C, , THEN ;
 15 -->

SCR # 47
  0 ( ASSEMBLER )
  1
  2 : STOREADD  C, 256<  IF C, ELSE ,
  3             THEN 0D MODE ! ;
  4
  5 : ZPAGE      OVER 100 < IF F7 AND
  6              THEN  ;
  7 : XYMODE  MODE @ 19 = MODE @ 1D
  8           = OR ;
  9 : M1  <BUILDS C, DOES> C@ MODE @
 10       1D = IF 10 ELSE 0 THEN OR
 11       ZPAGE STOREADD ;
 12
 13 0E M1 ASL, 2E M1 ROL, 4E M1 LSR,
 14 6E M1 ROR, CE M1 DEC, EE M1 INC,
 15 -->

SCR # 48
  0 ( ASSEMBLER )
  1
  2 : OPCODE  C@ ZPAGE XYMODE IF 10
  3           OR THEN ;
  4 : M2  <BUILDS C, DOES> OPCODE
  5       MODE @ 9 = IF 4 - THEN
  6       STOREADD ;
```

```
  7
  8 AC M2 LDY,   AE M2 LDX,
  9 CC M2 CPY,   EC M2 CPX,
 10
 11 : M3    <BUILDS C, DOES> OPCODE
 12         STOREADD ;
 13
 14 8C M3 STY,    8E M3 STX,
 15 -->

SCR # 49
  0 ( END OF ASSEMBLER )
  1
  2 FORTH DEFINITIONS
  3
  4
  5 LATEST 0C +ORIGIN !  ( NTOP )
  6
  7 HERE   1C +ORIGIN !  ( FENCE )
  8
  9 HERE   1E +ORIGIN !  ( DP )
 10
 11
 12
 13
 14
 15 BASE !      ;S

SCR # 50
  0 ( COLOR COMMANDS )
  1 BASE @ HEX
  2 : SETCOLOR  2 * SWAP 10 * OR SWAP
  3           02C4 ( COLPF0 ) + C! ;
  4 : SE.  SETCOLOR ;  ( ALIAS )
  5
  6 (  REGISTER#-3, COLOR-2, LUM-1
  7
  8 (    0-3          0-F     0-7
  9
 10 -->
 11
 12
 13
 14
 15

SCR # 51
  0 ( GRAPHICS COMMANDS )
  1 E456 CONSTANT CIO
  2   1C VARIABLE MASK
  3  340 CONSTANT IOCX
  4   53 VARIABLE SNAME
  5
  6 CODE GR.  1 # LDA,  GFLAG STA,
  7          XSAVE STX,    0 ,X LDA,
  8  # 30 LDX,     IOCX 0B + ,X STA,
  9 # 3 LDA,  IOCX 2 + ,X STA,
 10 SNAME FF AND # LDA,  IOCX 4 + ,X
 11  STA,   SNAME 100 / # LDA,
```

```
 12 IOCX 5 + ,X STA,   MASK LDA,
 13 IOCX 0A + ,X STA,    CIO JSR,
 14 XSAVE LDX,   0 # LDY,  POP JMP,
 15 -->

SCR # 52
  0 ( GRAPHICS COMMANDS )
  1
  2 CODE &GR     XSAVE STX, # 30 LDX,
  3              # C LDA,  IOCX 2 +
  4              ,X STA,   CIO JSR,
  5              XSAVE LDX, 0 # LDA,
  6              GFLAG STA, NEXT JMP,
  7
  8 : XGR  &GR  0 GR.  &GR ;
  9   ( EXIT GRAPHICS MODE )
 10
 11 -->
 12
 13
 14
 15

SCR # 53
  0 ( GRAPHICS I/O )
  1
  2 CODE CPUT   0 ,X LDA,  PHA,
  3   XSAVE STX,  # 30 LDX,
  4   # B LDA,  IOCX 2 + ,X STA, TYA,
  5   IOCX 8 + ,X STA, IOCX 9 + ,X
  6   STA, PLA,  CIO JSR,  XSAVE LDX,
  7   POP JMP,
  8
  9 54 CONSTANT ROWCRS
 10 55 CONSTANT COLCRS
 11
 12 : POS  ROWCRS C! COLCRS ! ;
 13 : PLOT   POS CPUT ;
 14
 15 -->

SCR # 54
  0 ( GRAPHICS I/O )
  1
  2 : GTYPE  -DUP IF OVER + SWAP
  3         DO I C@ CPUT LOOP ELSE
  4         DROP ENDIF ;
  5
  6 : (G")  R COUNT DUP 1+ R> + >R
  7         GTYPE ;
  8
  9 : G"  22 STATE @ IF COMPILE (G")
 10       WORD HERE C@ 1+ ALLOT
 11      ELSE WORD HERE COUNT GTYPE
 12      ENDIF  ; IMMEDIATE
 13
 14
 15 -->
```

```
SCR # 55
  0 ( DRAW, FIL )
  1
  2 2FB CONSTANT ATACHR
  3 2FD CONSTANT FILDAT
  4
  5 CODE GCOM    XSAVE STX, 0 ,X LDA,
  6    # 30 LDX,  IOCX 2 + ,X STA,
  7    CIO JSR,  XSAVE LDX, POP JMP,
  8
  9 : DRAW   POS ATACHR C! 11 GCOM ;
 10
 11 : FIL  FILDAT C!  12 GCOM ;
 12
 13
 14 BASE ! ;S
 15

SCR # 56
  0 ( SOUND COMMANDS )
  1     BASE @ HEX
  2
  3 D208 CONSTANT AUDCTL
  4 D200 CONSTANT AUDBASE
  5
  6 : SOUND  ( CH# FREQ DIST VOL --- )
  7   3 DUP 0D20F C! 232 C!
  8   SWAP 16 * + ROT DUP + AUDBASE +
  9     ROT OVER C! 1+ C! ;
 10
 11 : FILTER!  AUDCTL C! ;
 12    ( N --- )
 13
 14
 15 BASE ! ;S

SCR # 57
  0 ( GRAPHICS TESTS )
  1
  2 : BOX  0 10 10 PLOT  1 50 10 DRAW
  3        1 50 25 DRAW  1 10 25 DRAW
  4        1 10 10 DRAW ;
  5
  6 : FBOX  XGR   5 GR.   BOX
  7       10 25 POS  2 FIL ;
  8
  9
 10
 11
 12
 13
 14
 15

SCR # 58
  0 ( DOS OBJECT READER )
  1
  2 BASE @ HEX
  3
```

```
  4 0 VARIABLE BLOCK#  0 VARIABLE BYTES  0 VARIABLE BYTPTR
  5 0 VARIABLE ADDRSS  0 VARIABLE #BYTES
  6 : GETCOUNT  7F + C@ 7F AND  BYTES ! 0 BYTPTR ! ;
  7 : FNEXTBLK  7D + DUP C@ 100 * SWAP 1+ C@ + 3FF AND 1 - ;
  8 : LINKBLOCK  FNEXTBLK
  9   DUP BLOCK# ! DUP 0 > IF BLOCK THEN ;
 10 : BLK-CK  BYTES @ 0= IF BLOCK# @ BLOCK LINKBLOCK
 11   GETCOUNT THEN ;
 12 : NEXTBYTE  BLK-CK -1 BYTES +! BYTPTR @ 1 BYTPTR +!
 13 BLOCK# @ BLOCK + C@ ;
 14 : NEXTWORD  NEXTBYTE NEXTBYTE 100 * + ;
 15 -->

SCR # 59
  0 ( DOS OBJECT READER )
  1
  2 : ADRCALC  NEXTWORD DUP ADDRSS ! NEXTWORD SWAP - 1+  #BYTES ! ;
  3
  4 : BLOCKSET  DUP BLOCK# ! BLOCK GETCOUNT ;
  5
  6 : LOADOBJ  BLOCKSET  NEXTWORD 1+ IF CR ." Not an Object file"
  7  CR QUIT THEN
  8     BEGIN
  9          ADRCALC
 10         #BYTES @ 0 DO NEXTBYTE ADDRSS @ C! 1 ADDRSS +! LOOP
 11         BLOCK# @ BLOCK FNEXTBLK
 12    1+ 0= BYTES @ 0= AND   END ;
 13
 14
 15 BASE ! ;S

SCR # 60
  0 ( FLOATING POINT WORDS )
  1  BASE @  HEX
  2 : FDROP   DROP DROP DROP ;
  3 : FDUP    >R >R DUP R> DUP ROT
  4           SWAP R ROT ROT R> ;
  5 CODE FSWAP
  6   XSAVE STX, # 6 LDY,
  7 BEGIN,  0 ,X LDA, PHA, INX, DEY,
  8 0= END, XSAVE LDX,  # 6 LDY,
  9 BEGIN, 6 ,X LDA,  0 ,X STA, INX,
 10 DEY, 0= END, XSAVE LDX,  # 6 LDY,
 11  BEGIN, PLA, 0B ,X STA, DEX, DEY,
 12 0= END,   XSAVE LDX,  NEXT JMP,
 13
 14 XSAVE 100 * 86 + CONSTANT XSAV
 15 : XS,   XSAV ,  ;       -->

SCR # 61
  0 ( FLOATING POINT WORDS )
  1 CODE FOVER     DEX, DEX, DEX,
  2  DEX, DEX, DEX, XSAVE STX,
  3  # 6 LDY,  BEGIN,  0C ,X LDA,
  4  0 ,X STA, INX, DEY, 0= END,
  5  XSAVE LDX,  NEXT JMP,
  6
  7 XSAVE 100 * A6 + CONSTANT XLD
  8 : XL,   XLD  ,  ;
```

```
  9
 10 CODE AFP   XS, D800 JSR, XL, NEXT JMP,
 11 CODE FASC  XS, D8E6 JSR, XL, NEXT JMP,
 12 CODE IFP   XS, D9AA JSR, XL, NEXT JMP,    -->
 13
 14
 15

SCR # 62
  0 ( FLOATING POINT WORDS )
  1
  2 CODE FPI   XS, D9D2 JSR, XL, NEXT JMP,
  3 CODE FADD  XS, DA66 JSR, XL, NEXT JMP,
  4 CODE FSUB  XS, DA60 JSR, XL, NEXT JMP,
  5 CODE FMUL  XS, DADB JSR, XL, NEXT JMP,
  6 CODE FDIV  XS, DB28 JSR, XL, NEXT JMP,
  7 CODE FLG  XS, DECD JSR, XL, NEXT JMP,
  8 CODE FLG10 XS, DED1 JSR, XL, NEXT JMP,
  9 CODE FEX  XS, DDC0 JSR, XL, NEXT JMP,
 10 CODE FEX10 XS, DDCC JSR, XL, NEXT JMP,
 11 CODE FPOLY XS, DD40 JSR, XL, NEXT JMP,
 12   -->
 13
 14
 15

SCR # 63
  0 ( FLOATING POINT WORDS )
  1
  2 D4 CONSTANT FR0
  3 E0 CONSTANT FR1
  4 FC CONSTANT FLPTR
  5 F3 CONSTANT INBUF
  6 F2 CONSTANT CIX
  7
  8 -->
  9
 10
 11
 12
 13
 14
 15
```