```
SCR # 6
( INPUT-OUTPUT, TIM                                WFR-780519 )
CODE EMIT   XSAVE STX,  BOT 1+ LDA,  7F # AND,
            72C6 JSR,  XSAVE LDX,  POP JMP,
CODE KEY   XSAVE STX,  BEGIN,  BEGIN,  8 # LDX,
      BEGIN,  6E02 LDA,  .A LSR,  CS END,  7320 JSR,
      BEGIN,  731D JSR,  0 X) CMP,  0 X) CMP,  0 X) CMP,
      0 X) CMP,  0 X) CMP,  6E02 LDA,  .A LSR,  PHP,  TYA,
      .A LSR,  PLP,  CS IF,  80 # ORA,  THEN,  TAY,  DEX,
      0= END,  731D JSR,  FF # EOR,  7F # AND,  0= NOT END,
      7F # CMP,  0= NOT END,  XSAVE LDX,  PUSHOA JMP,
CODE CR  XSAVE STX,  728A JSR,  XSAVE LDX, NEXT JMP,

CODE ?TERMINAL   1 # LDA,  6E02 BIT,  0= NOT IF,
     BEGIN,  731D JSR,  6E02 BIT,  0= END,  INY,  THEN,
     TYA,  PUSHOA  JMP,
DECIMAL       ;S


SCR # 7
( INPUT-OUTPUT, APPLE                              WFR-780730 )
CODE HOME   FC58 JSR,  NEXT JMP,
CODE SCROLL   FC70 JSR,  NEXT JMP,

HERE  '  KEY  2  -  !    ( POINT KEY TO HERE )
   FDOC JSR,  7F # AND,  PUSHOA JMP,
HERE  ' EMIT  2  -  !   (  POINT EMIT TO HERE  )
   BOT 1+ LDA,  80 # ORA,  FDED JSR,  POP JMP,
HERE  '  CR  2  -  !      ( POINT CR TO HERE )
    FD8E JSR,  NEXT JMP,
HERE  '  ?TERMINAL  2  -  !   ( POINT ?TERM TO HERE )
   C000 BIT,  0<
      IF,  BEGIN,  C010 BIT,  C000 BIT,  0< NOT END,  INY,
        THEN,  TYA,  PUSHOA JMP,

DECIMAL    ;S


SCR # 8
( INPUT-OUTPUT, SYM-1                              WFR-781015 )
HEX
CODE KEY    8A58 JSR,  7F # AND,  PUSHOA JMP,

CODE EMIT   BOT 1+ LDA,    8A47 JSR,  POP JMP,

CODE CR    834D JSR,  NEXT JMP,

CODE ?TERMINAL   ( BREAK TEST FOR ANY KEY )
    8B3C JSR,  CS
    IF,  BEGIN,  8B3C JSR,  CS NOT  END,  INY,  THEN,
           TYA,  PUSHOA  JMP,




DECIMAL    ;S
```

```
SCR # 12
( COLD AND WARM ENTRY,  USER PARAMETERS                 WFR-79APR29 )
ASSEMBLER  OBJECT  MEM  HEX
NOP,    HERE JMP,    ( WORD ALIGNED VECTOR TO COLD )
NOP,    HERE JMP,    ( WORD ALIGNED VECTOR TO WARM )
0000 ,   0001 ,   ( CPU, AND REVISION PARAMETERS )
0000    ,        ( TOPMOST WORD IN FORTH VOCABULARY )
  7F    ,        ( BACKSPACE CHARACTER )
3BA0    ,        ( INITIAL USER AREA )
009E    ,        ( INITIAL TOP OF STACK )
01FF    ,        ( INITIAL TOP OF RETURN STACK )
0100    ,        ( TERMINAL INPUT BUFFER )
001F    ,        ( INITIAL NAME FIELD WIDTH )
0001    ,        ( INITIAL WARNING = 1 )
0200    ,        ( INITIAL FENCE )
0000    ,        ( COLD START VALUE FOR DP )
0000    ,        ( COLD START VALUE FOR VOC-LINK )  -->


SCR # 13
( START OF NUCLEUS,  LIT, PUSH, PUT, NEXT           WFR-78DEC26 )
CODE LIT                        ( PUSH FOLLOWING LITERAL TO STACK *)
     IP )Y LDA,  PHA,  IP INC,  0= IF,  IP 1+ INC,  THEN,
     IP )Y LDA,        IP INC,  0= IF,  IP 1+ INC,  THEN,
LABEL PUSH      ( PUSH ACCUM AS HI-BYTE, ML STACK AS LO-BYTE *)
     DEX,  DEX,
LABEL PUT         ( REPLACE BOTTOM WITH ACCUM. AND ML STACK *)
     BOT 1+ STA,  PLA,  BOT STA,
LABEL NEXT              ( EXECUTE NEXT FORTH ADDRESS, MOVING IP *)
     1 # LDY,  IP )Y LDA,  W 1+ STA,      ( FETCH CODE ADDRESS )
         DEY,  IP )Y LDA,  W    STA,
     CLC,  IP LDA,  2 # ADC,  IP STA,          ( MOVE IP AHEAD )
     CS IF,  IP 1+ INC,  THEN,
     W 1 - JMP,   ( JUMP INDIR. VIA W THRU CODE FIELD TO CODE )

-->


SCR # 14
( SETUP                                                WFR-790225 )
                    HERE  2+  ,            ( MAKE SILENT WORD *)
     IP )Y LDA,  PHA,  TYA,  'T LIT OB +  0=  NOT  END,

LABEL SETUP  ( MOVE # ITEMS FROM STACK TO 'N' AREA OF Z-PAGE *)
   .A ASL,  N 1 - STA,
   BEGIN,  BOT LDA,  N ,Y STA,  INX,  INY,
        N 1 - CPY,  0= END,  0 # LDY,  RTS,

CODE EXECUTE                   ( EXECUTE A WORD BY ITS CODE FIELD *)
                                          ( ADDRESS ON THE STACK *)
   BOT LDA,  W STA,  BOT 1+ LDA,  W 1+ STA,
   INX,  INX,  W 1 - JMP,



-->
```

```
SCR # 15
( BRANCH, 0BRANCH     W/16-BIT OFFSET              WFR-79APR01 )
CODE BRANCH             ( ADJUST IP BY IN-LINE 16 BIT LITERAL *)
   CLC,  IP )Y LDA,  IP    ADC,        PHA,
   INY,  IP )Y LDA,  IP 1+ ADC,  IP 1+ STA,
                           PLA,  IP    STA,  NEXT 2+ JMP,

CODE 0BRANCH            ( IF BOT IS ZERO, BRANCH FROM LITERAL *)
   INX,  INX,  FE ,X LDA,  FF ,X ORA,
   '  BRANCH  0= NOT END,  ( USE 'BRANCH' FOR FALSE )
 LABEL BUMP:                     ( TRUE JUST MOVES IP 2 BYTES *)
   CLC,  IP LDA,  2 # ADC,  IP STA,
   CS IF,  IP 1+ INC, THEN,  NEXT JMP,

-->



SCR # 16
(  LOOP CONTROL                                    WFR-79MAR20 )
CODE (LOOP)       ( INCREMENT LOOP INDEX, LOOP UNTIL => LIMIT *)
    XSAVE STX,  TSX,  R INC,  0= IF,  R 1+ INC,  THEN,
  LABEL L1:  CLC,  R 2+ LDA,  R SBC,  R 3 + LDA,  R 1+ SBC,
  LABEL L2:    XSAVE LDX,      ( LIMIT-INDEX-1 )
    .A ASL, '  BRANCH  CS END, ( BRANCH UNTIL D7 SIGN=1 )
    PLA,  PLA,  PLA,  PLA,  BUMP: JMP,  ( ELSE EXIT LOOP )

CODE (+LOOP)            ( INCREMENT INDEX BY STACK VALUE +/-   *)
    INX,  INX,  XSAVE STX,  ( POP INCREMENT )
    FF ,X LDA,  PHA,  PHA,  FE ,X LDA,  TSX,  INX,  INX,
    CLC,  R ADC,  R STA,  PLA,  R 1 + ADC,  R 1 + STA,
    PLA,  L1:  0< END,      ( AS FOR POSITIVE INCREMENT )
    CLC,  R    LDA,  R 2+  SBC, ( INDEX-LIMIT-1 )
          R 1+ LDA,  R 3 + SBC,    L2: JMP,
-->

SCR # 17
(  (DO-                                            WFR-79MAR30 )

CODE (DO)               ( MOVE TWO STACK ITEMS TO RETURN STACK *)
   SEC 1+ LDA,  PHA,  SEC LDA,  PHA,
   BOT 1+ LDA,  PHA,  BOT LDA,  PHA,

LABEL POPTWO    INX,  INX,
LABEL POP       INX,  INX,  NEXT JMP,

CODE I                      ( COPY CURRENT LOOP INDEX TO STACK *)
                            ( THIS WILL LATER BE POINTED TO 'R' )

-->



```

```
SCR # 18
( DIGIT                                             WFR-781202 )
CODE DIGIT     ( CONVERT ASCII CHAR-SECOND, WITH BASE-BOTTOM *)
                   ( IF OK RETURN DIGIT-SECOND, TRUE-BOTTOM; *)
                                    ( OTHERWISE FALSE-BOTTOM. *)
  SEC,  SEC LDA,   30 # SBC,
  0< NOT IF,  0A # CMP, ( ADJUST FOR ASCII LETTER )
            0< NOT IF,  SEC,  07 # SBC,  0A # CMP,
                       0< NOT IF,
 SWAP ( AT COMPILE TIME )  THEN,  BOT CMP, ( TO BASE )
                      0< IF,  SEC STA,  1 # LDA,
                      PHA,  TYA,  PUT JMP,
                      ( STORE RESULT SECOND AND RETURN TRUE )
   THEN,  THEN,  THEN,   ( CONVERSION FAILED )
   TYA,  PHA,  INX,  INX,  PUT JMP,  ( LEAVE BOOLEAN FALSE )

-->
```

```
SCR # 19
(  FIND FOR VARIABLE LENGTH NAMES                    WFR-790225 )
CODE (FIND)  ( HERE, NFA ... PFA, LEN BYTE, TRUE; ELSE FALSE *)
     2 # LDA,  SETUP JSR,  XSAVE STX,
BEGIN,  0 # LDY,  N )Y LDA,  N 2+ )Y EOR,  3F # AND,  0=
   IF, ( GOOD ) BEGIN, INY, N )Y LDA,  N 2+ )Y EOR,  .A ASL,  0=
          IF, ( STILL GOOD ) SWAP  CS ( LOOP TILL D7 SET )
      END,    XSAVE LDX,  DEX,  DEX,  DEX,  DEX,  CLC,
              TYA,  5 # ADC,  N ADC,  SEC STA,  0 # LDY,
        TYA,   N 1+ ADC,  SEC 1+ STA,  BOT 1+ STY,
     N )Y LDA,  BOT STA,  1 # LDA,  PHA,  PUSH JMP, ( FALSE )
     THEN,  CS  NOT ( AT LAST CHAR? )  IF,  SWAP  THEN,
    BEGIN,  INY,  N )Y LDA,  0< END, ( TO LAST CHAR )
  THEN,  INY,  ( TO LINK )  N )Y LDA,  TAX,  INY,
         N )Y LDA,  N 1+ STA,  N STX,  N ORA, ( 0 LINK ? )
    0= END, ( LOOP FOR ANOTHER NAME )
    XSAVE LDX,  0 # LDA, PHA,  PUSH JMP, ( FALSE )   -->
```

```
SCR # 20
(  ENCLOSE                                          WFR-780926 )
CODE ENCLOSE   ( ENTER WITH ADDRESS-2, DELIM-1.  RETURN WITH *)
   ( ADDR-4, AND OFFSET TO FIRST CH-3, END WORD-2, NEXT CH-1 *)
  2 # LDA,  SETUP JSR,  TXA,  SEC,  8 # SBC,  TAX,
  SEC 1+ STY,  BOT 1+ STY,  ( CLEAR HI BYTES )   DEY,
  BEGIN,  INY,  N 2+ )Y LDA,  ( FETCH CHAR )
     N CMP,  0=  NOT END,  ( STEP OVER LEADING DELIMITERS )
  BOT 4 + STY,  ( SAVE OFFSET TO FIRST CHAR )
  BEGIN,  N 2+ )Y LDA,  0=
      IF, ( NULL )  SEC STY, ( IN EW )  BOT STY,  ( IN NC )
            TYA,  BOT 4 + CMP,  0=
          IF, ( Y=FC )  SEC INC, ( BUMP EW )  THEN,  NEXT JMP,
       THEN,  SEC STY,  ( IN EW )  INY,  N CMP, ( DELIM ? )
     0= END,  ( IS DELIM )  BOT STY, ( IN NC )  NEXT JMP,

-->
```

```
SCR # 21
( TERMINAL VECTORS                                     WFR-79MAR30 )
( THESE WORDS ARE CREATED WITH NO EXECUTION CODE, YET.             )
( THEIR CODE FIELDS WILL BE FILLED WITH THE ADDRESS OF THEIR       )
( INSTALLATION SPECIFIC CODE.                                      )

CODE EMIT              ( PRINT ASCII VALUE ON BOTTOM OF STACK *)

CODE KEY         ( ACCEPT ONE TERMINAL CHARACTER TO THE STACK *)

CODE ?TERMINAL       ( 'BREAK' LEAVES 1 ON STACK; OTHERWISE 0 *)

CODE CR          ( EXECUTE CAR. RETURN, LINE FEED ON TERMINAL *)

-->


```

```
SCR # 22
( CMOVE,                                               WFR-79MAR20 )
CODE CMOVE   ( WITHIN MEMORY; ENTER W/  FROM-3, TO-2, QUAN-1 *)
  3 # LDA,  SETUP JSR,         ( MOVE 3 ITEMS TO 'N' AREA )
  BEGIN,  BEGIN,  N CPY,  0=  ( DECREMENT BYTE COUNTER AT 'N' )
             IF,  N 1+ DEC,  0<          ( EXIT WHEN DONE )
                  IF,  NEXT JMP,  THEN,  THEN,
         N 4 + )Y LDA,  N 2+ )Y STA,  INY,  0=
      END,           ( LOOP TILL Y WRAPS, 22 CYCLES/BYTE  )
      N 5 + INC,  N 3 + INC,      ( BUMP HI BYTES OF POINTERS )
   JMP,  ( BACK TO FIRST 'BEGIN' )

-->




```

```
SCR # 23
( U*,  UNSIGNED MULTIPLY FOR 16 BITS          RS-WFR-80AUG16 )
CODE U*         ( 16 BIT MULTIPLICAND-2,  16 BIT MULTIPLIER-1 *)
               ( 32 BIT UNSIGNED PRODUCT: LO WORD-2, HI WORD-1 *)
   SEC    LDA,  N    STA,  SEC    STY,
   SEC 1+ LDA,  N 1+ STA,  SEC 1+ STY,  ( multiplicand to n )
   10 # LDY,
   BEGIN,  BOT 2+ ASL,  BOT 3 + ROL,  BOT ROL,  BOT 1+ ROL,
            ( double product while sampling D15 of multiplier )
        CS IF, ( set ) CLC,
            ( add multiplicand to partial product 32 bits )
               N    LDA,  BOT 2 + ADC,  BOT 2 + STA,
               N 1+ LDA,  BOT 3 + ADC,  BOT 3 + STA,
             CS IF,  BOT INC,  0= IF,  BOT 1+ INC, ENDIF, ENDIF,
          ENDIF,  DEY,  0=    ( corrected for carry bug )
      UNTIL,      NEXT JMP,   C;
-->
```

```
SCR # 24
( U/, UNSIGNED DIVIDE FOR 31 BITS                      WFR-79APR29 )
CODE  U/          ( 31 BIT DIVIDEND-2, -3,  16 BIT DIVISOR-1   *)
                  ( 16 BIT REMAINDER-2,  16 BIT QUOTIENT-1     *)
  SEC 2 + LDA,  SEC     LDY,  SEC 2 + STY,  .A ASL,  SEC    STA,
  SEC 3 + LDA,  SEC 1+  LDY,  SEC 3 + STY,  .A ROL,  SEC 1+ STA,
  10 # LDA,  N STA,
  BEGIN,  SEC 2 + ROL,  SEC 3 + ROL,  SEC,
          SEC 2 + LDA,  BOT     SBC,  TAY,
          SEC 3 + LDA,  BOT 1+  SBC,
          CS IF,  SEC 2+ STY,  SEC 3 + STA,  THEN,
          SEC ROL,  SEC 1+ ROL,
          N DEC,  0=
      END,    POP  JMP,
-->
```

```
SCR # 25
(  LOGICALS                                           WFR-79APR20 )

CODE AND             ( LOGICAL BITWISE AND OF BOTTOM TWO ITEMS *)
   BOT    LDA,  SEC    AND,  PHA,
   BOT 1+ LDA,  SEC 1+ AND,  INX,  INX,  PUT JMP,

CODE OR              ( LOGICAL BITWISE 'OR' OF BOTTOM TWO ITEMS *)
   BOT    LDA,  SEC     ORA,  PHA,
   BOT 1+ LDA,  SEC 1 + ORA,  INX,  INX,  PUT JMP,

CODE XOR           ( LOGICAL 'EXCLUSIVE-OR' OF BOTTOM TWO ITEMS *)
   BOT    LDA,  SEC    EOR,  PHA,
   BOT 1+ LDA,  SEC 1+ EOR,  INX,  INX,  PUT JMP,

-->
```

```
SCR # 26
(  STACK INITIALIZATION                               WFR-79MAR30 )
CODE SP@                          ( FETCH STACK POINTER TO STACK *)
               TXA,
LABEL PUSHOA   PHA,  0 # LDA,  PUSH JMP,

CODE SP!                                     ( LOAD SP FROM 'SO' *)
   06 # LDY,  UP )Y LDA,  TAX,  NEXT JMP,

CODE RP!                                       ( LOAD RP FROM RO *)
   XSAVE STX,  08 # LDY,  UP )Y LDA,  TAX,  TXS,
             XSAVE LDX,  NEXT JMP,

CODE ;S              ( RESTORE IP REGISTER FROM RETURN STACK *)
   PLA,  IP STA,  PLA,  IP 1+ STA,  NEXT JMP,

-->
```

```
SCR # 27
( RETURN STACK WORDS                                  WFR-79MAR29 )
CODE LEAVE          ( FORCE EXIT OF DO-LOOP BY SETTING LIMIT *)
   XSAVE STX,  TSX,  R LDA,  R 2+ STA,               ( TO INDEX *)
   R 1+ LDA,  R 3 + STA,  XSAVE LDX,  NEXT JMP,

CODE >R              ( MOVE FROM COMP. STACK TO RETURN STACK *)
   BOT 1+ LDA,  PHA,  BOT LDA,  PHA,  INX,  INX,  NEXT JMP,

CODE R>              ( MOVE FROM RETURN STACK TO COMP. STACK *)
   DEX,  DEX,  PLA,  BOT STA,  PLA,  BOT 1+ STA,  NEXT JMP,

CODE R       ( COPY THE BOTTOM OF RETURN STACK TO COMP. STACK *)
   XSAVE STX,  TSX,  R LDA,  PHA,  R 1+ LDA,
   XSAVE LDX,  PUSH JMP,
'   R    -2  BYTE.IN  I  !
-->


SCR # 28
( TESTS AND LOGICALS                                  WFR-79MAR19 )

CODE 0=             ( REVERSE LOGICAL STATE OF BOTTOM OF STACK *)
   BOT LDA,  BOT 1+ ORA,  BOT 1+ STY,
   0= IF,  INY,  THEN,  BOT STY,  NEXT JMP,

CODE 0<              ( LEAVE TRUE IF NEGATIVE; OTHERWISE FALSE *)
   BOT 1+ ASL,  TYA,  .A ROL,  BOT 1+ STY,  BOT STA,  NEXT JMP,


-->






SCR # 29
( MATH                                                WFR-79MAR19 )
CODE +            ( LEAVE THE SUM OF THE BOTTOM TWO STACK ITEMS *)
   CLC,  BOT LDA,  SEC ADC,  SEC STA,  BOT 1+ LDA,  SEC 1+ ADC,
         SEC 1+ STA,  INX,  INX,  NEXT JMP,
CODE D+              ( ADD TWO DOUBLE INTEGERS, LEAVING DOUBLE *)
   CLC,  BOT 2 + LDA,  BOT 6 + ADC,  BOT 6 + STA,
         BOT 3 + LDA,  BOT 7 + ADC,  BOT 7 + STA,
         BOT     LDA,  BOT 4 + ADC,  BOT 4 + STA,
         BOT 1 + LDA,  BOT 5 + ADC,  BOT 5 + STA,  POPTWO JMP,
CODE MINUS           ( TWOS COMPLEMENT OF BOTTOM SINGLE NUMBER *)
   SEC,  TYA,  BOT    SBC,  BOT    STA,
         TYA,  BOT 1+ SBC,  BOT 1+ STA,  NEXT JMP,
CODE DMINUS          ( TWOS COMPLEMENT OF BOTTOM DOUBLE NUMBER *)
   SEC,  TYA,  BOT 2 + SBC,  BOT 2 + STA,
         TYA,  BOT 3 + SBC,  BOT 3 + STA,
      1  BYTE.IN  MINUS  JMP,                 -->
```

```
SCR # 30
( STACK MANIPULATION                                   WFR-79MAR29 )
CODE OVER              ( DUPLICATE SECOND ITEM AS NEW BOTTOM *)
   SEC LDA,  PHA,  SEC 1+ LDA,  PUSH JMP,

CODE DROP                             ( DROP BOTTOM STACK ITEM *)
   POP  -2  BYTE.IN  DROP  ! ( C.F. VECTORS DIRECTLY TO 'POP' )

CODE SWAP         ( EXCHANGE BOTTOM AND SECOND ITEMS ON STACK *)
   SEC LDA,  PHA,  BOT LDA,  SEC STA,
   SEC 1+ LDA,  BOT 1+ LDY,  SEC 1+ STY,  PUT JMP,

CODE DUP                         ( DUPLICATE BOTTOM ITEM ON STACK *)
   BOT LDA,  PHA,  BOT 1+ LDA,  PUSH JMP,

-->


SCR # 31
( MEMORY INCREMENT,                                    WFR-79MAR30 )

CODE +!   ( ADD SECOND TO MEMORY 16 BITS ADDRESSED BY BOTTOM *)
   CLC,  BOT X) LDA,  SEC ADC,  BOT X) STA,
   BOT INC,  0= IF,  BOT 1+ INC,  THEN,
   BOT X) LDA,  SEC 1+ ADC,  BOT X) STA,  POPTWO JMP,

CODE TOGGLE            ( BYTE AT ADDRESS-2, BIT PATTERN-1  ... *)
       SEC X) LDA,  BOT EOR,  SEC X) STA,  POPTWO JMP,

-->






SCR # 32
( MEMORY FETCH AND STORE                                WFR-781202 )
CODE @                      ( REPLACE STACK ADDRESS WITH 16 BIT *)
    BOT X) LDA,  PHA,                ( CONTENTS OF THAT ADDRESS *)
    BOT INC,  0= IF, BOT 1+ INC,  THEN,  BOT X) LDA,  PUT JMP,

CODE C@       ( REPLACE STACK ADDRESS WITH POINTED 8 BIT BYTE *)
    BOT X) LDA,  BOT STA,  BOT 1+ STY,  NEXT JMP,

CODE !         ( STORE SECOND AT 16 BITS ADDRESSED BY BOTTOM *)
    SEC LDA,  BOT X) STA,  BOT INC,  0= IF,  BOT 1+ INC,  THEN,
    SEC 1+ LDA,  BOT X) STA,  POPTWO JMP,

CODE C!           ( STORE SECOND AT BYTE ADDRESSED BY BOTTOM *)
    SEC LDA,  BOT X) STA,  POPTWO JMP,

DECIMAL     ;S
```

```
SCR # 33
  0 (  :,  ;,                                            WFR-79MAR30 )
  1
  2 : :                 ( CREATE NEW COLON-DEFINITION UNTIL ';' *)
  3                    ?EXEC !CSP CURRENT   @        CONTEXT    !
  4                CREATE  ]      ;CODE   IMMEDIATE
  5      IP 1+ LDA,  PHA,  IP LDA,  PHA,  CLC,  W LDA,  2 # ADC,
  6      IP STA,  TYA,  W 1+ ADC,  IP 1+ STA,  NEXT JMP,
  7
  8
  9 : ;                             ( TERMINATE COLON-DEFINITION *)
 10                     ?CSP  COMPILE     ;S
 11                   SMUDGE  [    ;   IMMEDIATE
 12
 13
 14
 15 -->


SCR # 34
  0 (  CONSTANT,  VARIABLE, USER                         WFR-79MAR30 )
  1 : CONSTANT              ( WORD WHICH LATER CREATES CONSTANTS *)
  2                       CREATE  SMUDGE  ,      ;CODE
  3       2 # LDY,  W )Y LDA,  PHA,  INY,  W )Y LDA,  PUSH JMP,
  4
  5 : VARIABLE              ( WORD WHICH LATER CREATES VARIABLES *)
  6      CONSTANT  ;CODE
  7        CLC,  W LDA,  2 # ADC,  PHA,  TYA,  W 1+ ADC,  PUSH JMP,
  8
  9
 10 : USER                                  ( CREATE USER VARIABLE *)
 11      CONSTANT  ;CODE
 12        2 # LDY,  CLC,  W )Y LDA,  UP ADC,  PHA,
 13        0 # LDA,  UP 1+ ADC,  PUSH JMP,
 14
 15 -->


SCR # 35
  0 (  DEFINED CONSTANTS                                 WFR-78MAR22 )
  1 HEX
  2 00  CONSTANT  0        01  CONSTANT  1
  3 02  CONSTANT  2        03  CONSTANT  3
  4 20  CONSTANT  BL                               ( ASCII BLANK *)
  5 40  CONSTANT  C/L                   ( TEXT CHARACTERS PER LINE *)
  6
  7 3BE0    CONSTANT   FIRST   ( FIRST BYTE RESERVED FOR BUFFERS *)
  8 4000    CONSTANT   LIMIT            ( JUST BEYOND TOP OF RAM *)
  9   80    CONSTANT   B/BUF            ( BYTES PER DISC BUFFER  *)
 10    8     CONSTANT  B/SCR  ( BLOCKS PER SCREEN = 1024 B/BUF / *)
 11
 12           00  +ORIGIN
 13 : +ORIGIN  LITERAL  +  ; ( LEAVES ADDRESS RELATIVE TO ORIGIN *)
 14 -->
 15
```

```
SCR # 36
( USER VARIABLES                                      WFR-78APR29 )
HEX                 ( 0 THRU 5 RESERVED,    REFERENCED TO $00A0 *)
( 06 USER  S0 )           ( TOP OF EMPTY COMPUTATION STACK *)
( 08 USER  R0 )                ( TOP OF EMPTY RETURN STACK *)
0A   USER  TIB                     ( TERMINAL INPUT BUFFER *)
0C   USER  WIDTH                ( MAXIMUM NAME FIELD WIDTH *)
0E   USER  WARNING                 ( CONTROL WARNING MODES *)
10   USER  FENCE                  ( BARRIER FOR FORGETTING *)
12   USER  DP                        ( DICTIONARY POINTER *)
14   USER  VOC-LINK                ( TO NEWEST VOCABULARY *)
16   USER  BLK                   ( INTERPRETATION BLOCK *)
18   USER  IN                 ( OFFSET INTO SOURCE TEXT *)
1A   USER  OUT                ( DISPLAY CURSOR POSITION *)
1C   USER  SCR                         ( EDITING SCREEN *)
-->


SCR # 37
( USER VARIABLES, CONT.                               WFR-79APR29 )
1E   USER  OFFSET              ( POSSIBLY TO OTHER DRIVES *)
20   USER  CONTEXT            ( VOCABULARY FIRST SEARCHED *)
22   USER  CURRENT        ( SEARCHED SECOND, COMPILED INTO *)
24   USER  STATE                    ( COMPILATION STATE *)
26   USER  BASE                ( FOR NUMERIC INPUT-OUTPUT *)
28   USER  DPL                  ( DECIMAL POINT LOCATION *)
2A   USER  FLD                      ( OUTPUT FIELD WIDTH *)
2C   USER  CSP                    ( CHECK STACK POSITION *)
2E   USER  R#                  (  EDITING CURSOR POSITION *)
30   USER  HLD      ( POINTS TO LAST CHARACTER HELD IN PAD *)
-->





SCR # 38
( HI-LEVEL MISC.                                      WFR-79APR29 )
: 1+       1   +  ;            ( INCREMENT STACK NUMBER BY ONE *)
: 2+       2   +  ;            ( INCREMENT STACK NUMBER BY TWO *)
: HERE     DP  @  ;         ( FETCH NEXT FREE ADDRESS IN DICT. *)
: ALLOT    DP  +! ;                ( MOVE DICT. POINTER AHEAD *)
: ,   HERE  !  2  ALLOT  ;        ( ENTER STACK NUMBER TO DICT. *)
: C,   HERE  C!  1   ALLOT  ;      ( ENTER STACK BYTE TO DICT. *)
: -   MINUS  +  ;                  ( LEAVE DIFF.  SEC - BOTTOM *)
: =   -   0=  ;                    ( LEAVE BOOLEAN OF EQUALITY *)
: <   -   0<  ;                   ( LEAVE BOOLEAN OF SEC < BOT *)
: >   SWAP  <  ;                  ( LEAVE BOOLEAN OF SEC > BOT *)
: ROT   >R  SWAP  R>  SWAP  ;        ( ROTATE THIRD TO BOTTOM *)
: SPACE     BL   EMIT  ;            ( PRINT BLANK ON TERMINAL *)
: -DUP     DUP  IF  DUP  ENDIF  ;       ( DUPLICATE NON-ZERO *)
-->

```

```
SCR # 39
( VARIABLE LENGTH NAME SUPPORT                          WFR-79MAR30 )
: TRAVERSE                                  ( MOVE ACROSS NAME FIELD *)
      ( ADDRESS-2, DIRECTION-1, I.E. -1=R TO L, +1=L TO R *)
        SWAP
        BEGIN  OVER  +  7F  OVER  C@  <  UNTIL  SWAP  DROP  ;

: LATEST        CURRENT  @  @  ;              ( NFA OF LATEST WORD *)


( FOLLOWING HAVE LITERALS DEPENDENT ON COMPUTER WORD SIZE )

: LFA     4  -  ;                      ( CONVERT A WORDS PFA TO LFA *)
: CFA     2  -  ;                      ( CONVERT A WORDS PFA TO CFA *)
: NFA     5  -  -1  TRAVERSE  ;        ( CONVERT A WORDS PFA TO NFA *)
: PFA      1  TRAVERSE  5  +  ;        ( CONVERT A WORDS NFA TO PFA *)
    -->


SCR # 40
( ERROR PROCEEDURES, PER SHIRA                          WFR-79MAR23 )
: !CSP      SP@  CSP  !  ;      ( SAVE STACK POSITION IN 'CSP' *)

: ?ERROR             ( BOOLEAN-2,  ERROR TYPE-1,  WARN FOR TRUE *)
          SWAP  IF         ERROR    ELSE  DROP  ENDIF  ;

: ?COMP   STATE @  0= 11 ?ERROR ;    ( ERROR IF NOT COMPILING *)

: ?EXEC   STATE  @  12  ?ERROR  ;    ( ERROR IF NOT EXECUTING *)

: ?PAIRS  -  13  ?ERROR  ;  ( VERIFY STACK VALUES ARE PAIRED *)

: ?CSP   SP@  CSP @ -  14  ?ERROR  ; ( VERIFY STACK POSITION *)

: ?LOADING                            ( VERIFY LOADING FROM DISC *)
          BLK  @  0=  16  ?ERROR  ;   -->


SCR # 41
( COMPILE,  SMUDGE,  HEX, DECIMAL                       WFR-79APR20 )

: COMPILE           ( COMPILE THE EXECUTION ADDRESS FOLLOWING *)
        ?COMP  R>  DUP  2+  >R  @  ,  ;

: [     0  STATE  !  ;  IMMEDIATE              ( STOP COMPILATION *)

: ]     C0  STATE  !  ;                  ( ENTER COMPILATION STATE *)

: SMUDGE    LATEST  20  TOGGLE  ;    ( ALTER LATEST WORD NAME *)

: HEX       10  BASE  !  ;               ( MAKE HEX THE IN-OUT BASE *)

: DECIMAL   0A  BASE  !  ;       ( MAKE DECIMAL THE IN-OUT BASE *)
-->

```

```
SCR # 42
( ;CODE                                              WFR-79APR20 )

: (;CODE)     ( WRITE CODE FIELD POINTING TO CALLING ADDRESS *)
        R>  LATEST  PFA  CFA  !  ;


: ;CODE                      ( TERMINATE A NEW DEFINING WORD *)
      ?CSP  COMPILE  (;CODE)
      [COMPILE]  [  SMUDGE  ;   IMMEDIATE
-->
```

```
SCR # 43
(  <BUILD,  DOES>                                    WFR-79MAR20 )

: <BUILDS   0  CONSTANT  ;  ( CREATE HEADER FOR 'DOES>' WORD *)

: DOES>            ( REWRITE PFA WITH CALLING HI-LEVEL ADDRESS *)
                             ( REWRITE CFA WITH 'DOES>' CODE *)
             R>  LATEST  PFA  !  ;CODE
        IP 1+ LDA,  PHA,  IP LDA,  PHA,  ( BEGIN FORTH NESTING )
        2 # LDY,  W )Y LDA,  IP STA,       ( FETCH FIRST PARAM )
        INY,  W )Y LDA,  IP 1+ STA,        ( AS NEXT INTERP. PTR )
        CLC,  W LDA,  4 # ADC,  PHA,  ( PUSH ADDRESS OF PARAMS )
        W 1+ LDA,  00 # ADC,  PUSH JMP,

-->
```

```
SCR # 44
(  TEXT OUTPUTS                                      WFR-79APR02 )
: COUNT    DUP 1+ SWAP C@  ;  ( LEAVE TEXT ADDR. CHAR. COUNT *)
: TYPE           ( TYPE STRING FROM ADDRESS-2, CHAR.COUNT-1 *)
        -DUP  IF OVER + SWAP
                 DO I C@ EMIT LOOP  ELSE DROP ENDIF ;
: -TRAILING   ( ADJUST CHAR. COUNT TO DROP TRAILING BLANKS *)
        DUP  0  DO  OVER  OVER  +  1  -  C@
        BL  -  IF  LEAVE  ELSE  1  -  ENDIF  LOOP  ;
: (.")               ( TYPE IN-LINE STRING, ADJUSTING RETURN *)
        R  COUNT  DUP  1+  R>  +  >R  TYPE  ;


: ."      22  STATE  @        ( COMPILE OR PRINT QUOTED STRING *)
    IF  COMPILE  (.")        WORD    HERE  C@  1+  ALLOT
        ELSE        WORD    HERE   COUNT  TYPE  ENDIF  ;
               IMMEDIATE     -->
```

```
SCR # 45
( TERMINAL INPUT                                        WFR-79APR29 )

: EXPECT             ( TERMINAL INPUT MEMORY-2,  CHAR LIMIT-1 *)
   OVER  +  OVER  DO  KEY  DUP  0E  +ORIGIN ( BS )  @  =
   IF  DROP  08  OVER  I  =  DUP  R>  2  -  +  >R  -
      ELSE ( NOT BS )  DUP  0D  =
          IF ( RET ) LEAVE  DROP  BL  0  ELSE  DUP  ENDIF
          I  C!  0  I  1+  !
      ENDIF EMIT  LOOP  DROP  ;
: QUERY     TIB  @  50  EXPECT  0  IN  !  ;
8081  HERE
: X   BLK @                         ( END-OF-TEXT IS NULL *)
     IF ( DISC ) 1 BLK +!  0 IN !  BLK @  7  AND  0=
        IF ( SCR END )  ?EXEC  R>  DROP  ENDIF ( disc dependent )
      ELSE  ( TERMINAL )   R>  DROP
        ENDIF  ;  !  IMMEDIATE     -->


SCR # 46
( FILL, ERASE, BLANKS, HOLD, PAD                        WFR-79APR02 )
: FILL                ( FILL MEMORY BEGIN-3,  QUAN-2,  BYTE-1 *)
        SWAP  >R  OVER  C!  DUP  1+  R>  1  -  CMOVE  ;

: ERASE             ( FILL MEMORY WITH ZEROS  BEGIN-2,  QUAN-1 *)
        0  FILL  ;

: BLANKS                    ( FILL WITH BLANKS BEGIN-2, QUAN-1 *)
        BL  FILL  ;

: HOLD                                      ( HOLD CHARACTER IN PAD *)
        -1  HLD  +!  HLD  @  C!  ;

: PAD          HERE  44  +  ;      ( PAD IS 68 BYTES ABOVE HERE *)
        ( DOWNWARD HAS NUMERIC OUTPUTS; UPWARD MAY HOLD TEXT *)
-->


SCR # 47
( WORD,                                                 WFR-79APR02 )
: WORD           ( ENTER WITH DELIMITER, MOVE STRING TO 'HERE' *)
   BLK  @  IF  BLK  @       BLOCK    ELSE  TIB  @  ENDIF
   IN  @  +  SWAP    ( ADDRESS-2, DELIMITER-1 )
   ENCLOSE         ( ADDRESS-4, START-3, END-2, TOTAL COUNT-1 )
   HERE  22  BLANKS       ( PREPARE FIELD OF 34 BLANKS )
   IN  +!          ( STEP OVER THIS STRING )
   OVER  -  >R     ( SAVE CHAR COUNT )
   R  HERE  C!     ( LENGTH STORED FIRST )
   +  HERE  1+
   R>  CMOVE  ;    ( MOVE STRING FROM BUFFER TO HERE+1 )




-->
```

```
SCR # 48
 ( (NUMBER-, NUMBER, -FIND,                          WFR-79APR29 )
 : (NUMBER)   ( CONVERT DOUBLE NUMBER, LEAVING UNCONV. ADDR. *)
    BEGIN  1+  DUP  >R  C@  BASE  @  DIGIT
       WHILE  SWAP  BASE  @  U*  DROP  ROT  BASE  @  U*  D+
       DPL  @  1+  IF  1  DPL  +!  ENDIF  R>  REPEAT  R>  ;

 : NUMBER   ( ENTER W/ STRING ADDR.  LEAVE DOUBLE NUMBER *)
       0  0  ROT  DUP  1+  C@  2D  =  DUP  >R  +  -1
    BEGIN  DPL  !  (NUMBER)  DUP  C@  BL  -
       WHILE  DUP  C@  2E  -  0  ?ERROR    0  REPEAT
       DROP  R>  IF  DMINUS  ENDIF  ;

 : -FIND       ( RETURN PFA-3, LEN BYTE-2, TRUE-1; ELSE FALSE *)
       BL  WORD          HERE  CONTEXT  @  @  (FIND)
       DUP  0=  IF  DROP  HERE  LATEST  (FIND)  ENDIF  ;
 -->
```

```
SCR # 49
 (  ERROR HANDLER                                     WFR-79APR20 )

 : (ABORT)          ABORT    ;  ( USER ALTERABLE ERROR ABORT *)

 : ERROR               ( WARNING:  -1=ABORT, 0=NO DISC, 1=DISC *)
     WARNING  @  0<          ( PRINT TEXT LINE REL TO SCR #4 *)
     IF  (ABORT)  ENDIF  HERE  COUNT  TYPE  ."   ? "
           MESSAGE    SP!  IN  @  BLK  @        QUIT    ;

 : ID.    ( PRINT NAME FIELD FROM ITS HEADER ADDRESS *)
      PAD  020  5F  FILL  DUP  PFA  LFA  OVER  -
      PAD  SWAP  CMOVE  PAD  COUNT  01F  AND  TYPE  SPACE  ;
 -->



```

```
SCR # 50
 (  CREATE                                            WFR-79APR28 )

 : CREATE                 ( A SMUDGED CODE HEADER TO PARAM FIELD *)
                         ( WARNING IF DUPLICATING A CURRENT NAME *)
       TIB  HERE  0A0  +  <  2  ?ERROR  ( 6502 only )
       -FIND    ( CHECK IF UNIQUE IN CURRENT AND CONTEXT )
       IF ( WARN USER )  DROP  NFA  ID.
                           4        MESSAGE    SPACE  ENDIF
       HERE  DUP  C@  WIDTH  @        MIN    1+  ALLOT
       DP  C@  0FD  =  ALLOT    ( 6502 only )
       DUP  A0  TOGGLE  HERE  1  -  80  TOGGLE ( DELIMIT BITS )
       LATEST  ,  CURRENT  @  !
       HERE  2+  ,  ;
 -->


```

```
SCR # 51
  0 (  LITERAL,  DLITERAL,  [COMPILE],  ?STACK        WFR-79APR29 )
  1
  2 : [COMPILE]         ( FORCE COMPILATION OF AN IMMEDIATE WORD *)
  3      -FIND  0=  0  ?ERROR  DROP  CFA  ,  ;  IMMEDIATE
  4
  5 : LITERAL                    ( IF COMPILING, CREATE LITERAL *)
  6      STATE  @  IF  COMPILE  LIT  ,  ENDIF  ;  IMMEDIATE
  7
  8 : DLITERAL            ( IF COMPILING, CREATE DOUBLE LITERAL *)
  9      STATE  @  IF  SWAP  [COMPILE]  LITERAL
 10                          [COMPILE]  LITERAL  ENDIF ; IMMEDIATE
 11
 12 (  FOLLOWING DEFINITION IS INSTALLATION DEPENDENT )
 13 : ?STACK    ( QUESTION UPON OVER OR UNDERFLOW OF STACK *)
 14      09E  SP@  <  1  ?ERROR  SP@  020  <  7  ?ERROR  ;
 15 -->


SCR # 52
  0 (  INTERPRET,                                     WFR-79APR18 )
  1
  2 : INTERPRET   ( INTERPRET OR COMPILE SOURCE TEXT INPUT WORDS *)
  3      BEGIN  -FIND
  4         IF  ( FOUND )  STATE  @  <
  5                IF  CFA  ,  ELSE  CFA  EXECUTE  ENDIF  ?STACK
  6            ELSE  HERE  NUMBER  DPL  @  1+
  7                IF  [COMPILE]  DLITERAL
  8                  ELSE  DROP  [COMPILE]  LITERAL  ENDIF  ?STACK
  9          ENDIF  AGAIN  ;
 10 -->
 11
 12
 13
 14
 15


SCR # 53
  0 (  IMMEDIATE,  VOCAB,  DEFIN,  FORTH,  (       DJK-WFR-79APR29 )
  1 : IMMEDIATE        ( TOGGLE PREC. BIT OF LATEST CURRENT WORD *)
  2         LATEST  40  TOGGLE  ;
  3
  4 : VOCABULARY  ( CREATE VOCAB WITH 'V-HEAD' AT VOC INTERSECT. *)
  5        <BUILDS  A081  ,  CURRENT  @  CFA  ,
  6        HERE  VOC-LINK  @  ,  VOC-LINK  !
  7        DOES>  2+  CONTEXT  !  ;
  8
  9 VOCABULARY  FORTH    IMMEDIATE       ( THE TRUNK VOCABULARY *)
 10
 11 : DEFINITIONS        ( SET THE CONTEXT ALSO AS CURRENT VOCAB *)
 12        CONTEXT  @  CURRENT  !  ;
 13
 14 : (              ( SKIP INPUT TEXT UNTIL RIGHT PARENTHESIS *)
 15        29  WORD  ;  IMMEDIATE   -->
```

```
SCR # 54
 ( QUIT, ABORT                                         WFR-79MAR30 )

 : QUIT                       ( RESTART, INTERPRET FROM TERMINAL *)
      0  BLK  !  [COMPILE]  [
      BEGIN  RP!  CR  QUERY  INTERPRET
             STATE  @  0=  IF  ." OK"  ENDIF  AGAIN  ;

 : ABORT                      ( WARM RESTART, INCLUDING REGISTERS *)
      SP!  DECIMAL                DRO
      CR  ." FORTH-65 V 4.0"
      [COMPILE]  FORTH  DEFINITIONS  QUIT  ;


 -->


```

```
SCR # 55
 ( COLD START                                           WFR-79APR29 )
 CODE COLD                   ( COLD START, INITIALIZING USER AREA *)
    HERE  02  +ORIGIN  !  ( POINT COLD ENTRY TO HERE )
          0C +ORIGIN LDA,  'T FORTH 4 + STA,  ( FORTH VOCAB. )
          0D +ORIGIN LDA,  'T FORTH 5 + STA,
          15 # LDY, ( INDEX TO VOC-LINK ) 0= IF, ( FORCED )
     HERE  06 +ORIGIN !   ( POINT RE-ENTRY TO HERE )
       0F # LDY,  ( INDEX TO WARNING )  THEN, ( FROM IF, )
       10 +ORIGIN LDA,   UP    STA,  ( LOAD UP )
       11 +ORIGIN LDA,   UP 1+ STA,
        BEGIN,  0C +ORIGIN ,Y LDA,  ( FROM LITERAL AREA )
                         UP )Y STA,  ( TO USER AREA )
             DEY,  0< END,
      'T ABORT  100  /MOD  # LDA,  IP 1+ STA,
                           # LDA,  IP    STA,
       6C # LDA,  W 1 - STA,    'T RP! JMP, ( RUN )  -->
```

```
SCR # 56
 ( MATH UTILITY                                     DJK-WFR-79APR29 )
 CODE S->D                     ( EXTEND SINGLE INTEGER TO DOUBLE *)
      BOT 1+ LDA,  0< IF, DEY, THEN,  TYA, PHA, PUSH JMP,

 : +-    0< IF MINUS ENDIF ;    ( APPLY SIGN TO NUMBER BENEATH *)

 : D+-                      ( APPLY SIGN TO DOUBLE NUMBER BENEATH *)
         0<  IF  DMINUS  ENDIF  ;

 : ABS     DUP  +-   ;                       ( LEAVE ABSOLUTE VALUE *)
 : DABS    DUP  D+-  ;          ( DOUBLE INTEGER ABSOLUTE VALUE *)

 : MIN                               ( LEAVE SMALLER OF TWO NUMBERS *)
         OVER  OVER  >  IF  SWAP  ENDIF  DROP  ;
 : MAX                                ( LEAVE LARGET OF TWO NUMBERS *)
         OVER  OVER  <  IF  SWAP  ENDIF  DROP  ; -->
```

```
SCR # 57
( MATH PACKAGE                                   DJK-WFR-79APR29 )
: M*      ( LEAVE SIGNED DOUBLE PRODUCT OF TWO SINGLE NUMBERS *)
        OVER  OVER  XOR  >R  ABS  SWAP  ABS  U*  R>  D+-  ;
: M/              ( FROM SIGNED DOUBLE-3-2, SIGNED DIVISOR-1 *)
               ( LEAVE SIGNED REMAINDER-2, SIGNED QUOTIENT-1 *)
        OVER  >R  >R  DABS  R  ABS  U/
        R>  R  XOR  +-  SWAP  R>  +-  SWAP  ;
: *       U*  DROP  ;                         ( SIGNED PRODUCT *)
: /MOD    >R  S->D  R>  M/  ;            ( LEAVE REM-2, QUOT-1 *)
: /       /MOD  SWAP  DROP  ;                 ( LEAVE QUOTIENT *)
: MOD     /MOD  DROP  ;                      ( LEAVE REMAINDER *)
: */MOD               ( TAKE RATION OF THREE NUMBERS, LEAVING *)
          >R  M*  R>  M/  ;                 ( REM-2, QUOTIENT-1 *)
: */      */MOD  SWAP  DROP  ;   ( LEAVE RATIO OF THREE NUMBS *)
: M/MOD    ( DOUBLE, SINGLE DIVISOR ...  REMAINDER, DOUBLE *)
           >R  0  R  U/  R>  SWAP  >R  U/  R>   ;   -->

SCR # 58
( DISC UTILITY, GENERAL USE                          WFR-79APR02 )
FIRST  VARIABLE  USE            ( NEXT BUFFER TO USE, STALEST *)
FIRST  VARIABLE  PREV      ( MOST RECENTLY REFERENCED BUFFER *)

: +BUF      ( ADVANCE ADDRESS-1 TO NEXT BUFFER. RETURNS FALSE *)
      84 ( I.E. B/BUF+4 )  +  DUP  LIMIT  =      ( IF AT PREV *)
      IF  DROP  FIRST  ENDIF  DUP  PREV  @  -  ;

: UPDATE     ( MARK THE BUFFER POINTED TO BY PREV AS ALTERED *)
      PREV  @  @  8000  OR  PREV  @  !  ;

: EMPTY-BUFFERS   ( CLEAR BLOCK BUFFERS; DON'T WRITE TO DISC *)
      FIRST  LIMIT  OVER  -  ERASE  ;

: DR0      0  OFFSET  !  ;                    ( SELECT DRIVE #0 *)
: DR1   07D0  OFFSET  !  ;   -->              ( SELECT DRIVE #1 *)

SCR # 59
( BUFFER                                             WFR-79APR02 )
: BUFFER                    ( CONVERT BLOCK# TO STORAGE ADDRESS *)
    USE  @  DUP  >R   ( BUFFER ADDRESS TO BE ASSIGNED )
    BEGIN  +BUF  UNTIL ( AVOID PREV )  USE  !  ( FOR NEXT TIME )
    R  @  0<  ( TEST FOR UPDATE IN THIS BUFFER )
    IF ( UPDATED, FLUSH TO DISC )
       R  2+ ( STORAGE LOC. )
       R  @  7FFF  AND  ( ITS BLOCK # )
       0          R/W     ( WRITE SECTOR TO DISC )
      ENDIF
    R  !  ( WRITE NEW BLOCK # INTO THIS BUFFER )
    R  PREV  !  ( ASSIGN THIS BUFFER AS 'PREV' )
    R>  2+  ( MOVE TO STORAGE LOCATION )  ;

-->

```

```
SCR # 60
( BLOCK                                             WFR-79APR02 )
: BLOCK          ( CONVERT BLOCK NUMBER TO ITS BUFFER ADDRESS *)
  OFFSET  @  +  >R   ( RETAIN BLOCK # ON RETURN STACK )
  PREV  @  DUP  @  R  -  DUP  +  ( BLOCK = PREV ? )
  IF ( NOT PREV )
     BEGIN  +BUF  0=  ( TRUE UPON REACHING 'PREV' )
        IF ( WRAPPED )  DROP  R  BUFFER
            DUP  R  1        R/W    ( READ SECTOR FROM DISC )
            2  - ( BACKUP )
          ENDIF
          DUP  @  R  -  DUP  +  0=
       UNTIL  ( WITH BUFFER ADDRESS )
     DUP  PREV  !
    ENDIF
    R>  DROP    2+  ;
-->

SCR # 61
( TEXT OUTPUT FORMATTING                            WFR-79MAY03 )

: (LINE)        ( LINE#, SCR#, ...  BUFFER ADDRESS, 64 COUNT *)
        >R  C/L  B/BUF  */MOD  R>  B/SCR  *  +
        BLOCK  +  C/L  ;

: .LINE   ( LINE#,  SCR#,  ...  PRINTED *)
        (LINE)  -TRAILING  TYPE  ;

: MESSAGE      ( PRINT LINE RELATIVE TO SCREEN #4 OF DRIVE 0 *)
    WARNING  @
    IF  ( DISC IS AVAILABLE )
        -DUP  IF  4  OFFSET  @  B/SCR  /  -  .LINE  ENDIF
        ELSE  ." MSG # "          .    ENDIF  ;
-->


SCR # 62
( LOAD,  -->                                        WFR-79APR02 )

: LOAD                             ( INTERPRET SCREENS FROM DISC *)
    BLK  @  >R  IN  @  >R  0  IN  !  B/SCR  *  BLK  !
    INTERPRET  R>  IN  !  R>  BLK  !  ;

: -->                  ( CONTINUE INTERPRETATION ON NEXT SCREEN *)
     ?LOADING  0  IN  !  B/SCR  BLK  @  OVER
     MOD  -  BLK  +!  ;    IMMEDIATE

-->





```

```
SCR # 63
( INSTALLATION DEPENDENT TERMINAL I-O, TIM       WFR-79APR26 )
( EMIT )          ASSEMBLER
  HERE  -2  BYTE.IN  EMIT  !        ( VECTOR EMITS' CF TO HERE )
  XSAVE STX,  BOT LDA,  7F # AND, 72C6 JSR,  XSAVE LDX,
  CLC,  1A # LDY,  UP )Y LDA,  01 # ADC,  UP )Y STA,
             INY,  UP )Y LDA,  00 # ADC,  UP )Y STA,  POP JMP,
                                    ( AND INCREMENT 'OUT' )
( KEY )
      HERE  -2  BYTE.IN  KEY  !   ( VECTOR KEYS' CF TO HERE )
      XSAVE STX,  BEGIN,  8 # LDX,
      BEGIN,  6E02 LDA,  .A LSR,  CS END,  7320 JSR,
      BEGIN,  731D JSR,  0 X) CMP,  0 X) CMP,  0 X) CMP,
      0 X) CMP,  0 X) CMP,  6E02 LDA,  .A LSR,  PHP,  TYA,
      .A LSR,  PLP,  CS IF,  80 # ORA,  THEN,  TAY,  DEX,
      0= END,  731D JSR,  FF # EOR,  7F # AND,  0= NOT END,
      XSAVE LDX,   PUSHOA JMP,      -->


SCR # 64
( INSTALLATION DEPENDENT TERMINAL I-O,  TIM      WFR-79APR02 )

( ?TERMINAL )
     HERE  -2  BYTE.IN  ?TERMINAL  !    ( VECTOR LIKEWISE )
     1 # LDA, 6E02 BIT, 0= NOT  IF,
     BEGIN,  731D JSR,  6E02 BIT,  0= END,  INY,  THEN,
      TYA,    PUSHOA JMP,

( CR )
    HERE  -2  BYTE.IN  CR  !  ( VECTOR CRS' CF TO HERE )
    XSAVE STX,  728A JSR,  XSAVE LDX, NEXT JMP,

-->




SCR # 65
( INSTALLATION DEPENDENT DISC                     WFR-79APR02 )
6900    CONSTANT  DATA                      ( CONTROLLER PORT *)
6901    CONSTANT  STATUS                    ( CONTROLLER PORT *)


: #HL            ( CONVERT DECIMAL DIGIT FOR DISC CONTROLLER *)
      0  0A  U/  SWAP  30  +  HOLD  ;

-->







```

```
SCR # 66
( D/CHAR, ?DISC,                                    WFR-79MAR23 )
CODE D/CHAR    ( TEST CHAR-1. EXIT TEST BOOL-2, NEW CHAR-1 *)
      DEX, DEX, BOT 1+ STY, C0 # LDA,
   BEGIN, STATUS BIT, 0= NOT END, ( TILL CONTROL READY )
      DATA LDA, BOT STA, ( SAVE CHAR )
      SEC CMP, 0= IF, INY, THEN, SEC STY, NEXT JMP,

: ?DISC         ( UPON NAK SHOW ERR MSG, QUIT. ABSORBS TILL *)
      1 D/CHAR >R 0=                  ( EOT, EXCEPT FOR SOH *)
    IF  ( NOT SOH ) R 15 =
         IF ( NAK ) CR
             BEGIN 4 D/CHAR EMIT
                UNTIL ( PRINT ERR MSG TIL EOT ) QUIT
           ENDIF ( FOR ENQ, ACK )
          BEGIN 4 D/CHAR DROP UNTIL ( AT EOT )
     ENDIF R> DROP ; -->
```

```
SCR # 67
( BLOCK-WRITE                                        WFR-790103 )
CODE BLOCK-WRITE    ( SEND TO DISC FROM ADDRESS-2, COUNT-1 *)
      2 # LDA, SETUP JSR,                  ( WITH EOT AT END *)
   BEGIN, 02 # LDA,
      BEGIN,   STATUS BIT, 0= END, ( TILL IDLE )
      N CPY, 0=
        IF, ( DONE ) 04 # LDA, STATUS STA, DATA STA,
             NEXT JMP,
        THEN,
      N 2+ )Y LDA, DATA STA, INY,
      0= END, ( FORCED TO BEGIN )

-->
```

```
SCR # 68
( BLOCK-READ,                                        WFR-790103 )

CODE BLOCK-READ   ( BUF.ADDR-1. EXIT AT 128 CHAR OR CONTROL *)
     1 # LDA, SETUP JSR,
   BEGIN, C0 # LDA,
      BEGIN,  STATUS BIT, 0= NOT END, ( TILL FLAG )
      50 ( BVC, D6=DATA )
     IF, DATA LDA, N )Y STA, INY,       SWAP
          0< END, ( LOOP TILL 128 BYTES )
       THEN, ( OR D6=0, SO D7=1, )
       NEXT JMP,

-->
```

```
SCR # 69
( R/W  FOR PERSCI 1070 CONTROLLER                    WFR-79MAY03 )
0A  ALLOT  HERE       ( WORKSPACE TO PREPARE DISC CONTROL TEXT )
        ( IN FORM:  C TT SS /D,  TT=TRACK, SS=SECTOR, D=DRIVE )
                                    ( C = I TO READ, 0 TO WRITE *)
: R/W                                   ( READ/WRITE DISC BLOCK *)
               ( BUFFER ADDRESS-3, BLOCK #-2, 1=READ 0=WRITE *)
   LITERAL  HLD  ! ( JUST AFTER WORKSPACE )   SWAP
   0  OVER  >  OVER  0F9F  >  OR  6  ?ERROR
   07D0  ( 2000 SECT/DR )  /MOD  #HL  DROP  2F  HOLD  BL  HOLD
   1A  /MOD  SWAP 1+ #HL  #HL  DROP  BL  HOLD  ( SECTOR 01-26 )
                     #HL  #HL  DROP  BL  HOLD  ( TRACK  00-76 )
   DUP
   IF  49 ( I=READ)  ELSE 4F ( O=WRITE )  ENDIF
   HOLD  HLD  @  0A  BLOCK-WRITE  ( SEND TEXT ) ?DISC
   IF  BLOCK-READ  ELSE  B/BUF  BLOCK-WRITE  ENDIF
   ?DISC  ;     -->


SCR # 70
(  FORWARD REFERENCES                                WFR-79MAR30 )
00  BYTE.IN  :          REPLACED.BY  ?EXEC
02  BYTE.IN  :          REPLACED.BY  !CSP
04  BYTE.IN  :          REPLACED.BY  CURRENT
08  BYTE.IN  :          REPLACED.BY  CONTEXT
0C  BYTE.IN  :          REPLACED.BY  CREATE
0E  BYTE.IN  :          REPLACED.BY  ]
10  BYTE.IN  :          REPLACED.BY  (;CODE)
00  BYTE.IN  ;          REPLACED.BY  ?CSP
02  BYTE.IN  ;          REPLACED.BY  COMPILE
06  BYTE.IN  ;          REPLACED.BY  SMUDGE
08  BYTE.IN  ;          REPLACED.BY  [
00  BYTE.IN  CONSTANT   REPLACED.BY  CREATE
02  BYTE.IN  CONSTANT   REPLACED.BY  SMUDGE
04  BYTE.IN  CONSTANT   REPLACED.BY  ,
06  BYTE.IN  CONSTANT   REPLACED.BY  (;CODE)     -->


SCR # 71
(  FORWARD REFERENCES                                WFR-79APR29 )
02  BYTE.IN  VARIABLE     REPLACED.BY  (;CODE)
02  BYTE.IN  USER         REPLACED.BY  (;CODE)
06  BYTE.IN  ?ERROR       REPLACED.BY  ERROR
0F  BYTE.IN  ."           REPLACED.BY  WORD
1D  BYTE.IN  ."           REPLACED.BY  WORD
00  BYTE.IN  (ABORT)      REPLACED.BY  ABORT
19  BYTE.IN  ERROR        REPLACED.BY  MESSAGE
25  BYTE.IN  ERROR        REPLACED.BY  QUIT
0C  BYTE.IN  WORD         REPLACED.BY  BLOCK
1E  BYTE.IN  CREATE       REPLACED.BY  MESSAGE
2C  BYTE.IN  CREATE       REPLACED.BY  MIN
04  BYTE.IN  ABORT        REPLACED.BY  DR0
2C  BYTE.IN  BUFFER       REPLACED.BY  R/W
30  BYTE.IN  BLOCK        REPLACED.BY  R/W       DECIMAL  ;S

```

```
SCR # 72
( ',  FORGET,                                  DJK-WFR-79DEC02 )
: '          ( FIND NEXT WORDS PFA; COMPILE IT, IF COMPILING *)
   -FIND  0=  0  ?ERROR  DROP  [COMPILE]  LITERAL  ;
                                          IMMEDIATE
HEX
: FORGET                       ( Dave Kilbridge's Smart Forget )
    [COMPILE]  '  NFA   DUP  FENCE  @  U<  15  ?ERROR
    >R  VOC-LINK  @  ( start with latest vocabulary )
  BEGIN  R  OVER  U<  WHILE  [COMPILE]  FORTH  DEFINITIONS
      @  DUP  VOC-LINK  !  REPEAT  ( unlink from voc list )
  BEGIN  DUP  4  -      ( start with phantom nfa )
      BEGIN  PFA  LFA  @  DUP  R  U<  UNTIL
      OVER  2  -  !  @  -DUP  0=  UNTIL ( end of list ? )
    R>  DP  !  ;    -->


SCR # 73
( CONDITIONAL COMPILER, PER SHIRA                    WFR-79APR01 )
: BACK      HERE  -  ,  ;             ( RESOLVE BACKWARD BRANCH *)

: BEGIN     ?COMP  HERE  1  ;                       IMMEDIATE

: ENDIF     ?COMP 2 ?PAIRS  HERE  OVER  -  SWAP  !  ;  IMMEDIATE

: THEN      [COMPILE]  ENDIF  ;     IMMEDIATE

: DO        COMPILE  (DO)  HERE  3  ;                 IMMEDIATE

: LOOP      3  ?PAIRS  COMPILE  (LOOP)  BACK  ;  IMMEDIATE

: +LOOP     3  ?PAIRS  COMPILE  (+LOOP)  BACK  ;     IMMEDIATE

: UNTIL     1  ?PAIRS  COMPILE  OBRANCH  BACK  ; IMMEDIATE  -->


SCR # 74
( CONDITIONAL COMPILER                               WFR-79APR01 )
: END       [COMPILE]  UNTIL  ;  IMMEDIATE

: AGAIN     1  ?PAIRS  COMPILE  BRANCH  BACK  ;   IMMEDIATE

: REPEAT    >R  >R  [COMPILE]  AGAIN
               R>  R>  2  -  [COMPILE]  ENDIF  ;  IMMEDIATE

: IF        COMPILE  OBRANCH   HERE  0  ,  2  ;  IMMEDIATE

: ELSE      2  ?PAIRS  COMPILE  BRANCH  HERE  0  ,
            SWAP  2  [COMPILE]  ENDIF  2  ;      IMMEDIATE

: WHILE    [COMPILE]  IF  2+  ;    IMMEDIATE

-->
```

```
SCR # 75
( NUMERIC PRIMITIVES                                  WFR-79APR01 )
: SPACES    0 MAX  -DUP  IF  0  DO  SPACE  LOOP  ENDIF  ;

: <#      PAD  HLD  !  ;

: #>      DROP  DROP  HLD  @  PAD  OVER  -  ;

: SIGN    ROT  0<  IF  2D  HOLD  ENDIF  ;

: #                      ( CONVERT ONE DIGIT, HOLDING IN PAD * )
          BASE @ M/MOD ROT 9 OVER < IF  7 + ENDIF 30  +  HOLD  ;

: #S      BEGIN  #  OVER  OVER  OR  0=  UNTIL  ;
-->


SCR # 76
( OUTPUT OPERATORS                                    WFR-79APR20 )
: D.R         ( DOUBLE INTEGER OUTPUT, RIGHT ALIGNED IN FIELD *)
       >R  SWAP  OVER  DABS  <#  #S  SIGN  #>
       R>  OVER  -  SPACES  TYPE  ;

: D.      0  D.R  SPACE  ;             ( DOUBLE INTEGER OUTPUT *)

: .R      >R  S->D  R>  D.R  ;        ( ALIGNED SINGLE INTEGER *)

: .       S->D  D.  ;                  ( SINGLE INTEGER OUTPUT *)

: ?       @  .  ;                   ( PRINT CONTENTS OF MEMORY *)

   .  CFA      MESSAGE  2A  +  !  ( PRINT MESSAGE NUMBER )
-->


SCR # 77
( PROGRAM DOCUMENTATION                               WFR-79APR20 )
HEX
: LIST                         ( LIST SCREEN BY NUMBER ON STACK *)
           DECIMAL   CR  DUP  SCR  !
          ." SCR # "   .  10  0  DO  CR  I  3  .R  SPACE
          I  SCR  @  .LINE  LOOP  CR  ;

: INDEX       ( PRINT FIRST LINE OF EACH SCREEN FROM-2,  TO-1 *)
          0C  EMIT ( FORM FEED )  CR  1+  SWAP
          DO  CR  I  3  .R  SPACE
              0  I  .LINE
              ?TERMINAL  IF  LEAVE  ENDIF  LOOP  ;
: TRIAD     ( PRINT 3 SCREENS ON PAGE, CONTAINING # ON STACK *)
          0C  EMIT ( FF )  3  /  3  *  3  OVER  +  SWAP
          DO  CR  I  LIST  LOOP  CR
          0F  MESSAGE  CR  ;     DECIMAL   -->
```

```
SCR # 78
  0 (  TOOLS                                                 WFR-79APR20 )
  1 HEX
  2 : VLIST                                      ( LIST CONTEXT VOCABULARY *)
  3                80  OUT  !    CONTEXT  @  @
  4      BEGIN  OUT  @  C/L  >  IF  CR  0  OUT  !  ENDIF
  5             DUP  ID.  SPACE  SPACE    PFA  LFA  @
  6             DUP  0=  ?TERMINAL  OR  UNTIL  DROP  ;
  7 -->
  8
  9
 10
 11
 12
 13
 14
 15


SCR # 79
  0 (  TOOLS                                                 WFR-79MAY03 )
  1 HEX
  2
  3 CREATE  MON          ( CALL MONITOR, SAVING RE-ENTRY TO FORTH *)
  4         0  C,    4C C,   ' LIT 18 + ,    SMUDGE
  5
  6
  7
  8
  9
 10 DECIMAL
 11 HERE                 FENCE  !
 12 HERE         28  +ORIGIN  !   ( COLD START FENCE )
 13 HERE         30  +ORIGIN  !   ( COLD START DP )
 14 LATEST       12  +ORIGIN  !   ( TOPMOST WORD )
 15 '  FORTH  6  +  32  +ORIGIN  !  ( COLD VOC-LINK )  ;S


SCR # 80
  0 -->
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
```