fig-FORTH GLOSSARY

This glossary contains all of the word definitions in Release 1 of fig-FORTH. The definitions are presented in the order of their ascii sort.

The first line of each entry shows a symbolic description of the action of the procedure on the parameter stack. The symbols indicate the order in which input parameters have been placed on the stack. Three dashes "---" indicate the execution point; any parameters left on the stack are listed. In this notation, the top of the stack is to the right.

The symbols include:

addr	memory address
b	8 bit byte (i.e. hi 8 bits zero)
c	7 bit ascii character (hi 9 bits zero)
d	32 bit signed double integer, most significant portion with sign on top of stack
f	boolean flag. 0=false, non-zero=true
ff	boolean false flag=0
n	16 bit signed integer number
u	16 bit unsigned integer
tf	boolean true flag=non-zero

The capital letters on the right show definition characteristics:

C	May only be used within a colon definition. A digit indicates number of memory addresses used, if other than one.
E	Intended for execution only.
L0	Level Zero definition of FORTH-78.
L1	Level One definition of FORTH-78.
P	Has precedence bit set. Will execute even when compiling.
U	A user variable.

Unless otherwise noted, all references to numbers are for 16 bit signed integers. On 8 bit data bus computers, the high byte of a number is on top of the stack, with the sign in the leftmost bit. For 32 bit signed double numbers, the most significant part (with the sign) is on top.

All arithmetic is implicitly 16 bit signed integer math, with error and under-flow indication unspecified.

```
!           n  addr  ---                      L0
       Store 16 bits of n at address.
       Pronounced “store”.

!CSP
       Save the stack position in CSP.  Used as
       part of the compiler security.

#           d1  ---  d2                       L0
       Generate from a double number d1, the
       next ascii character which is placed in
       an output string.  Result d2 is the
       quotient after division by BASE, and is
       maintained for further processing.  Used
       between <# and #>.  See #S.

#>          d  --- addr  count                L0
       Terminates numeric output conversion by
       dropping d, leaving the text address and
       character count suitable for TYPE.

#S          d1  ---  d2                       L0
       Generates ascii text in the text output
       buffer, by the use of #, until a zero
       double number n2 results.  Used between
       <# and #>.

'           ---  addr                       P,L0
       Used in the form:
                 '  nnnn
       Leaves the parameter field address of
       dictionary word nnnn.  As a complier
       directive,executes in a colon-definition
       to compile the address as a literal.  If
       the word is not found after a search of
       CONTEXT and CURRENT, an appropriate
       error message is given.  Pronounced
       "tick".

(                                           P,L0
       Used in the form:
                ( cccc)
       Ignore a comment that will be delimited
       by a right parenthesis on the same line.
       May occur during execution or in a
       colon-definition.  A blank after the
       leading parenthesis is required.

(.")                                          C+
       The run-time procedure, compiled by ."
       which transmits the following in-line
       text to the selected output device.
       See ."

(;CODE)                                        C
       The run-time procedure, compiled by
       ;CODE, that rewrites the code field of
       the most recently defined word to point
       to the following machine code sequence.
       See ;CODE.

(+LOOP)     n  ---                            C2
       The run-time procedure compiled by
       +LOOP, which increments the loop index
       by n and tests for loop completion.  See
       +LOOP.

(ABORT)
       Executes after an error when WARNING is
       -1.  This word normally executes ABORT,
       but may be altered (with care) to a
       user's alternative procedure.

(DO)                                           C
       The run-time procedure compiled by DO
       which moves the loop control parameters
       to the return stack.  See DO.

(FIND)      addr1  addr2  ---  pfa  b  tf   (ok)
            addr1  addr2  ---   ff         (bad)
       Searches the dictionary starting at the
       name field address addr2, matching to
       the text at addr1.  Returns parameter
       field address, length byte of name field
       and boolean true for a good match.  If
       no match is found, only a boolean false
       is left.

(LINE)      n1  n2  ---  addr  count
       Convert the line number n1 and the
       screen n2 to the disc buffer address
       containing the data.  A count of 64
       indicates the full line text length.

(LOOP)                                        C2
       The run-time procedure compiled by LOOP
       which increments the loop index and
       tests for loop completion.  See LOOP.

(NUMBER)    d1  addr1  ---  d2  addr2
       Convert the ascii text beginning at
       addr1+1 with regard to BASE.  The new
       value is accumulated into double number
       d1, being left as d2.  addr2 is the
       address of the first unconvertible
       digit.  Used by NUMBER.

*           n1  n2  ---  prod                 L0
       Leave the signed product of two signed
       numbers.

*/          n1  n2  n3  ---  n4               L0
       Leave the ratio  n4 = n1*n2/n3
       where all are signed numbers.  Retention
       of an intermediate 31 bit product per-
       mits greater accuracy than would be
       available with the sequence:
           n1  n2  *  n3  /

*/MOD       n1  n2  n3  ---  n4  n5           L0
       Leave the quotient n5 and remainder n4
       of the operation  n1*n2/n3.  A 31 bit
       intermediate product is used as for */.
```

```
+          n1  n2  ---  sum                  L0
       Leave the sum of n1+n2.

+!         n  addr  ---                      L0
       Add n to the value at the address.
       Pronounced "plus-store".

+-         n1  n2  ---  n3
       Apply the sign of n2 to n1, which is
       left as n3.

+BUF       addr1  ---  addr2  f
       Advance the disc buffer address addr1 to
       the address of the next buffer addr2.
       Boolean f is false when addr2 is the
       buffer presently pointed to by variable
       PREV.

+LOOP            n1  ---  (run)
           addr  n2  ---  (compile)      P,C2,L0
       Used in a colon-definition in the form:
            DO  ...  n1  +LOOP
       At run-time, +LOOP selectively controls
       branching back to the corresponding DO
       based on n1, the loop index and the loop
       limit.  The signed increment n1 is added
       to the index and the total compared to
       the limit.  The branch back to DO occurs
       until the new index is equal to or
       greater than the limit (n1>0), or until
       the new index is equal to or less than
       the limit (n1<0).  Upon exiting the
       loop, the parameters are discarded and
       execution continues ahead.

       At compiled time, +LOOP compiles the
       run-time word (+LOOP) and the branch
       offset computed from HERE to the address
       left on the stack by DO.  n2 is used for
       compile time error checking.

+ORIGIN    n  ---  addr
       Leave the memory address relative by n
       to the origin parameter area.  n is the
       minimum address unit, either byte or
       word.  This definition is used to access
       or modify the boot-up parameters at the
       origin area.

,          n  ---                            L0
       Store n into the next available
       dictionary memory cell, advancing the
       dictionary pointer.  (comma)

-          n1  n2  ---  diff                 L0
       Leave the difference of n1-n2.

-->                                        P,L0
       Continue interpretation with the next
       disc screen.  (pronounced next-screen).

-DUP       n1  --  n1      (if zero)
           n1  --  n1  n1  (if non-zero)      L0
       Reproduce n1 only if it is non-zero.
       This is usually used to copy a value
       just before IF, to eliminate the need
       for an ELSE part to drop it.

-FIND      ---  pfa  b  tf   (found)
           ---   ff          (not found)
       Accepts the next text word (delimited by
       blanks) in the input stream to HERE, and
       searches the CONTEXT and then CURRENT
       vocabularies for a matching entry.  If
       found, the dictionary entry's parameter
       field address, its length byte, and a
       boolean true is left.  Otherwise, only a
       boolean false is left.

-TRAILING     addr  n1  ---  addr  n2
       Adjusts the character count n1 of a text
       string beginning address to suppress the
       output of trailing blanks.  I.e. the
       characters at addr+n1 to addr+n2 are
       blanks.

.          n  ---                             L0
       Print a number from a signed 16 bit
       two's complement value, converted
       according to the numeric BASE.  A trail-
       ing blank follows.  Pronounced "dot".

."                                          P,L0
       Used in the form:
               ." cccc"
       Compiles an in-line string cccc
       (delimited by the trailing ") with an
       execution procedure to transmit the text
       to the selected output device.  If exe-
       cuted outside a definition, ." will
       immediately print the text until the
       final ".  The maximum number of
       characters may be an installation
       dependent value.  See (.").

.LINE      line  scr  ---
       Print on the terminal device, a line of
       text from the disc by its line and
       screen number.  Trailing blanks are
       suppressed.

.R         n1  n2  ---
       Print the number n1 right aligned in a
       field whose width is n2.  No following
       blank is printed.

/          n1  n2  ---  quot                  L0
       Leave the signed quotient of n1/n2.

/MOD       n1  n2  ---  rem  quot             L0
       Leave the remainder and signed quotient
       of n1/n2.  The remainder has the sign of
       the dividend.
```

0 1 2 3   --- n

These small numbers are used so often that it is attractive to define them by name in the dictionary as constants.

0<   n --- f   L0

Leave a true flag if the number is less than zero (negative), otherwise leave a false flag.

0=   n --- f   L0

Leave a true flag if the number is equal to zero, otherwise leave a false flag.

0BRANCH   f ---   C2

The run-time procedure to conditionally branch. If f is false (zero), the following in-line parameter is added to the interpretive pointer to branch ahead or back. Compiled by IF, UNTIL, and WHILE.

1+   n1 --- n2   L1

Increment n1 by 1.

2+   n1 --- n2

Leave n1 incremented by 2.

:   P,E,L0

Used in the form called a colon-definition:

```
    : cccc    ...    ;
```

Creates a dictionary entry defining cccc as equivalent to the following sequence of Forth word definitions '...' until the next ';' or ';CODE'. The compiling process is done by the text interpreter as long as STATE is non-zero. Other details are that the CONTEXT vocabulary is set to the CURRENT vocabulary and that words with the precedence bit set (P) are executed rather than being compiled.

;   P,C,L0

Terminate a colon-definition and stop further compilation. Compiles the run-time ;S.

;CODE   P,C,L0

Used in the form:

```
    : cccc   ....    ;CODE
         assembly mnemonics
```

Stop compilation and terminate a new defining word cccc by compiling (;CODE). Set the CONTEXT vocabulary to ASSEMBLER, assembling to machine code the following mnemonics.

When cccc later executes in the form:

```
    cccc   nnnn
```

the word nnnn will be created with its execution procedure given by the machine code following cccc. That is, when nnnn is executed, it does so by jumping to the code after nnnn. An existing defining word must exist in cccc prior to ;CODE.

;S   P,L0

Stop interpretation of a screen. ;S is also the run-time word compiled at the end of a colon-definition which returns execution to the calling procedure.

<   n1 n2 --- f   L0

Leave a true flag if n1 is less than n2; otherwise leave a false flag.

<#   L0

Setup for pictured numeric output formatting using the words:

```
    <#  #  #S  SIGN  #>
```

The conversion is done on a double number producing text at PAD.

<BUILDS   C,L0

Used within a colon-definition:

```
    : cccc  <BUILDS  ...
            DOES>    ...   ;
```

Each time cccc is executed, <BUILDS defines a new word with a high-level execution procedure. Executing cccc in the form:

```
    cccc  nnnn
```

uses <BUILDS to create a dictionary entry for nnnn with a call to the DOES> part for nnnn. When nnnn is later executed, it has the address of its parameter area on the stack and executes the words after DOES> in cccc. <BUILDS and DOES> allow run-time procedures to be written in high-level rather than in assembler code (as required by ;CODE).

=   n1 n2 --- f   L0

Leave a true flag if n1=n2; otherwise leave a false flag.

>   n1 n2 --- f   L0

Leave a true flag if n1 is greater than n2; otherwise a false flag.

>R   n ---   C,L0

Remove a number from the computation stack and place as the most accessible on the return stack. Use should be balanced with R> in the same definition.

?   addr ---   L0

Print the value contained at the address in free format according to the current base.

?COMP

Issue error message if not compiling.

?CSP

Issue error message if stack position differs from value saved in CSP.

?ERROR   f n ---

Issue an error message number n, if the boolean flag is true.

?EXEC

Issue an error message if not executing.

?LOADING

Issue an error message if not loading.

?PAIRS   n1 n2 ---

Issue an error message if n1 does not equal n2. The message indicates that compiled conditionals do not match.

?STACK

Issue an error message if the stack is out of bounds. This definition may be installation dependent.

?TERMINAL   --- f

Perform a test of the terminal keyboard for actuation of the break key. A true flag indicates actuation. This definition is installation dependent.

@   addr --- n   L0

Leave the 16 bit contents of address.

ABORT   L0

Clear the stacks and enter the execution state. Return control to the operator's terminal, printing a message appropriate to the installation.

ABS   n --- u   L0

Leave the absolute value of n as u.

AGAIN   addr n --- (compiling)   P,C2,L0

Used in a colon-definition in the form:

```
    BEGIN  ...  AGAIN
```

At run-time, AGAIN forces execution to return to corresponding BEGIN. There is no effect on the stack. Execution cannot leave this loop (unless R> DROP is executed one level below).

At compile time, AGAIN compiles BRANCH with an offset from HERE to addr. n is used for compile-time error checking.

ALLOT   n ---   L0

Add the signed number to the dictionary pointer DP. May be used to reserve dictionary space or re-origin memory. n is with regard to computer address type (byte or word).

AND   n1 n2 --- n3   L0

Leave the bitwise logical and of n1 and n2 as n3.

B/BUF   --- n

This constant leaves the number of bytes per disc buffer, the byte count read from disc by BLOCK.

B/SCR   --- n

This constant leaves the number of blocks per editing screen. By convention, an editing screen is 1024 bytes organized as 16 lines of 64 characters each.

BACK   addr ---

Calculate the backward branch offset from HERE to addr and compile into the next available dictionary memory address.

BASE   --- addr   U,L0

A user variable containing the current number base used for input and output conversion.

BEGIN   --- addr n (compiling)   P,L0

Occurs in a colon-definition in form:

```
    BEGIN  ...  UNTIL
    BEGIN  ...  AGAIN
    BEGIN  ...  WHILE  ...  REPEAT
```

At run-time, BEGIN marks the start of a sequence that may be repetitively executed. It serves as a return point from the corresponding UNTIL, AGAIN or REPEAT. When executing UNTIL, a return to BEGIN will occur if the top of the stack is false; for AGAIN and REPEAT a return to BEGIN always occurs.

At compile time BEGIN leaves its return address and n for compiler error checking.

BL   --- c

A constant that leaves the ascii value for "blank".

BLANKS   addr count ---

Fill an area of memory beginning at addr with blanks.

BLK   --- addr   U,L0

A user variable containing the block number being interpreted. If zero, input is being taken from the terminal input buffer.

BLOCK   n --- addr   L0
Leave the memory address of the block buffer containing block n. If the block is not already in memory, it is transferred from disc to whichever buffer was least recently written. If the block occupying that buffer has been marked as updated, it is re-written to disc before block n is read into the buffer. See also BUFFER, R/W, UPDATE, FLUSH.

BLOCK-READ
BLOCK-WRITE   These are the preferred names for the installation dependent code to read and write one block to the disc.

BRANCH   C2,L0
The run-time procedure to unconditionally branch. An in-line offset is added to the interpretive pointer IP to branch ahead or back. BRANCH is compiled by ELSE, AGAIN, REPEAT.

BUFFER   n --- addr
Obtain the next memory buffer, assigning it to block n. If the contents of the buffer are marked as updated, it is written to the disc. The block is not read from the disc. The address left is the first cell within the buffer for data storage.

C!   b addr ---
Store 8 bits at address. On word addressing computers, further specification is necessary regarding byte addressing.

C,   b ---
Store 8 bits of b into the next available dictionary byte, advancing the dictionary pointer. This is only available on byte addressing computers, and should be used with caution on byte addressing minicomputers.

C@   addr --- b
Leave the 8 bit contents of memory address. On word addressing computers, further specification is needed regarding byte addressing.

CFA   pfa --- cfa
Convert the parameter field address of a definition to its code field address.

CMOVE   from to count ---
Move the specified quantity of bytes beginning at address from to address to. The contents of address from are moved first proceeding toward high memory. Further specification is necessary on word addressing computers.

COLD
The cold start procedure to adjust the dictionary pointer to the minimum standard and restart via ABORT. May be called from the terminal to remove application programs and restart.

COMPILE   C2
When the word containing COMPILE executes, the execution address of the word following COMPILE is copied (compiled) into the dictionary. This allows specific compilation situations to be handled in addition to simply compiling an execution address (which the interpreter already does).

CONSTANT   n ---   L0
A defining word used in the form:
    n CONSTANT cccc
to create word cccc, with its parameter field containing n. When cccc is later executed, it will push the value of n to the stack.

CONTEXT   --- addr   U,L0
A user variable containing a pointer to the vocabulary within which dictionary searches will first begin.

COUNT   addr1 --- addr2 n   L0
Leave the byte address addr2 and byte count n of a message text beginning at address addr1. It is presumed that the first byte at addr1 contains the text byte count and the actual text starts with the second byte. Typically COUNT is followed by TYPE.

CR   L0
Transmit a carriage return and line feed to the selected output device.

CREATE
A defining word used in the form:
    CREATE cccc
by such words as CODE and CONSTANT to create a dictionary header for a Forth definition. The code field contains the address of the word's parameter field. The new word is created in the CURRENT vocabulary.

CSP   --- addr   U
A user variable temporarily storing the stack pointer position, for compilation error checking.

D+   d1 d2 --- dsum
Leave the double number sum of two double numbers.

```
D+-         d1  n  ---  d2
```
Apply the sign of n to the double number d1, leaving it as d2.

```
D.      d  ---                              L1
```
Print a signed double number from a 32 bit two's complement value. The high-order 16 bits are most accessible on the stack. Conversion is performed according to the current BASE. A blank follows. Pronounced D-dot.

```
D.R         d  n  ---
```
Print a signed double number d right aligned in a field n characters wide.

```
DABS        d  ---  ud
```
Leave the absolute value ud of a double number.

```
DECIMAL                                     L0
```
Set the numeric conversion BASE for decimal input-output.

```
DEFINITIONS                                 L1
```
Used in the form:

```
    cccc  DEFINITIONS
```
Set the CURRENT vocabulary to the CONTEXT vocabulary. In the example, executing vocabulary name cccc made it the CONTEXT vocabulary and executing DEFINITIONS made both specify vocabulary cccc.

```
DIGIT       c  n1  ---  n2  tf  (ok)
            c  n1  ---  ff      (bad)
```
Converts the ascii character c (using base n1) to its binary equivalent n2, accompanied by a true flag. If the conversion is invalid, leaves only a false flag.

```
DLIST
```
List the names of the dictionary entries in the CONTEXT vocabulary.

```
DLITERAL   d  ---  d    (executing)
           d  ---       (compiling)         P
```
If compiling, compile a stack double number into a literal. Later execution of the definition containing the literal will push it to the stack. If executing, the number will remain on the stack.

```
DMINUS      d1  ---  d2
```
Convert d1 to its double number two's complement.

```
DO          n1   n2  ---    (execute)
            addr  n  ---    (compile)     P,C2,L0
```
Occurs in a colon-definition in form:

```
    DO  ...  LOOP
    DO  ...  +LOOP
```
At run time, DO begins a sequence with repetitive execution controlled by a loop limit n1 and an index with initial value n2. DO removes these from the stack. Upon reaching LOOP the index is incremented by one. Until the new index equals or exceeds the limit, execution loops back to just after DO; otherwise the loop parameters are discarded and execution continues ahead. Both n1 and n2 are determined at run-time and may be the result of other operations. Within a loop 'I' will copy the current value of the index to the stack. See I, LOOP, +LOOP, LEAVE.

When compiling within the colon-definition, DO compiles (DO), leaves the following address addr and n for later error checking.

```
DOES>                                       L0
```
A word which defines the run-time action within a high-level defining word. DOES> alters the code field and first parameter of the new word to execute the sequence of compiled word addresses following DOES>. Used in combination with <BUILDS. When the DOES> part executes it begins with the address of the first parameter of the new word on the stack. This allows interpretation using this area or its contents. Typical uses include the Forth assembler, multi-dimensional arrays, and compiler generation.

```
DP          ---  addr                     U,L
```
A user variable, the dictionary pointer, which contains the address of the next free memory above the dictionary. The value may be read by HERE and altered by ALLOT.

```
DPL         ---  addr                     U,L0
```
A user variable containing the number of digits to the right of the decimal on double integer input. It may also be used to hold output column location of a decimal point, in user generated formatting. The default value on single number input is -1.

DR0
DR1
Installation dependent commands to select disc drives, by presetting OFFSET. The contents of OFFSET are added to the block number in BLOCK to allow for this selection. Offset is suppressed for error text so that it may always originate from drive 0.

DROP   n ---   L0

Drop the number from the stack.

DUMP   addr a ---   L0

Print the contents of n memory locations beginning at addr. Both addresses and contents are shown in the current numeric base.

DUP   n --- n n   L0

Duplicate the value on the stack.

ELSE   addr1 n1 --- addr2 n2
(compiling)   P,C2,L0

Occurs within a colon-definition in the form:

IF ... ELSE ... ENDIF

At run-time, ELSE executes after the true part following IF. ELSE forces execution to skip over the following false part and resumes execution after the ENDIF. It has no stack.

At compile-time ELSE emplaces BRANCH reserving a branch offset, leaves the address addr2 and n2 for error testing. ELSE also resolves the pending forward branch from IF by calculating the offset from addr1 to HERE and storing at addr1.

EMIT   c ---   L0

Transmit ascii character c to the selected output device. OUT is incremented for each character output.

EMPTY-BUFFERS   L0

Mark all block-buffers as empty, not necessarily affecting the contents. Updated blocks are not written to the disc. This is also an initialization procedure before first use of the disc.

ENCLOSE   addr1 c ---
addr1 n1 n2 n3

The text scanning primitive used by WORD. From the text address addr1 and an ascii delimiting character c, is determined the byte offset to the first non-delimiter character n1, the offset to the first delimiter after the text n2, and the offset to the first character not included. This procedure will not process past an ascii 'null', treating it as an unconditional delimiter.

END   P,C2,L0

This is an 'alias' or duplicate definition for UNTIL.

ENDIF   addr n --- (compile)   P,C0,L0

Occurs in a colon-definition in form:

IF ... ENDIF
IF ... ELSE ... ENDIF

At run-time, ENDIF serves only as the destination of a forward branch from IF or ELSE. It marks the conclusion of the conditional structure. THEN is another name for ENDIF. Both names are supported in fig-FORTH. See also IF and ELSE.

At compile-time, ENDIF computes the forward branch offset from addr to HERE and stores it at addr. n is used for error tests.

ERASE   addr n ---

Clear a region of memory to zero from addr over n addresses.

ERROR   line --- in blk

Execute error notification and restart of system. WARNING is first examined. If 1, the text of line n, relative to screen 4 of drive 0, is printed. This line number may be positive or negative, and beyond just screen 4. If WARNING=0, n is just printed as a message number (non disc installation). If WARNING is -1, the definition (ABORT) is executed, which executes the system ABORT. The user may cautiously modify this execution by altering (ABORT). fig-FORTH saves the contents of IN and BLK to assist in determining the location of the error. Final action is execution of QUIT.

EXECUTE   addr ---

Execute the definition whose code field address is on the stack. The code field address is also called the compilation address.

EXPECT   addr count ---   L0

Transfer characters from the terminal to address, until a "return" or the count of characters have been received. One or more nulls are added at the end of the text.

FENCE   --- addr   U

A user variable containing an address below which FORGETting is trapped. To forget below this point the user must alter the contents of FENCE.

```
FILL       addr  quan  b  ---
       Fill memory at the address with the
       specified quantity of bytes b.

FIRST      ---  n
       A constant that leaves the address of
       the first (lowest) block buffer.

FLD        --- addr                            U
       A user variable for control of number
       output field width.  Presently unused in
       fig-FORTH.

FORGET                                      E,L0
       Executed in the form:
           FORGET cccc
       Deletes definition named cccc from the
       dictionary with all entries physically
       following it.  In fig-FORTH, an error
       message will occur if the CURRENT and
       CONTEXT vocabularies are not currently
       the same.

FORTH                                       P,L1
       The name of the primary vocabulary.
       Execution makes FORTH the CONTEXT
       vocabulary.  Until additional user
       vocabularies are defined, new user
       definitions become a part of FORTH.
       FORTH is immediate, so it will execute
       during the creation of a colon-defini-
       tion, to select this vocabulary at
       compile time.

HERE       ---  addr                          L0
       Leave the address of the next available
       dictionary location.

HEX                                           L0
       Set the numeric conversion base to
       sixteen (hexadecimal).

HLD        ---  addr                          L0
       A user variable that holds the address
       of the latest character of text during
       numeric output conversion.

HOLD       c  ---                             L0
       Used between <# and #> to insert an
       ascii character into a pictured numeric
       output string.  E.g.  2E  HOLD  will
       place a decimal point.

I          ---  n                           C,L0
       Used within a DO-LOOP to copy the loop
       index to the stack.  Other use is imple-
       mentation dependent.  See R.

ID.        addr  ---
       Print a definition's name from its name
       field address.

IF         f  ---        (run-time)
           ---  addr  n  (compile)      P,C2,L0
       Occurs in a colon-definition in form:
           IF (tp) ... ENDIF
           IF (tp) ... ELSE (fp) ... ENDIF
       At run-time, IF selects execution based
       on a boolean flag.  If f is true (non-
       zero), execution continues ahead thru
       the true part.  If f is false (zero),
       execution skips till just after ELSE to
       execute the false part.  After either
       part, execution resumes after ENDIF.
       ELSE and its false part are optional; if
       missing, false execution skips to just
       after ENDIF.

       At compile-time IF compiles 0BRANCH and
       reserves space for an offset at addr.
       addr and n are used later for resolution
       of the offset and error testing.

IMMEDIATE
       Mark the most recently made definition
       so that when encountered at compile
       time, it will be executed rather than
       being compiled.  I.e. the precedence bit
       in its header is set.  This method
       allows definitions to handle unusual
       compiling situations, rather than build
       them into the fundamental compiler.  The
       user may force compilation of an
       immediate definition by preceding it
       with [COMPILE].

IN         ---  addr                          L0
       A user variable containing the byte
       offset within the current input text
       buffer (terminal or disc) from which the
       next text will be accepted.  WORD uses
       and moves the value of IN.

INDEX      from  to  ---
       Print the first line of each screen over
       the range from, to.  This is used to
       view the comment lines of an area of
       text on disc screens.

INTERPRET
       The outer text interpreter which sequen-
       tially executes or compiles text from
       the input stream (terminal or disc)
       depending on STATE.  If the word name
       cannot be found after a search of
       CONTEXT and then CURRENT it is converted
       to a number according to the current
       base.  That also-failing, an error mes-
       sage echoing the name with a " ?" will
       be given.  Text input vill be taken ac-
       cording to the convention for WORD. If a
       decimal point is found as part of a num-
       ber, a double number value will be left.
       The decimal point has no other purpose
       than to force this action.  See NUMBER.
```

KEY   --- c   L0
Leave the ascii value of the next terminal key struck.

LATEST   --- addr
Leave the name field address of the topmost word in the CURRENT vocabulary.

LEAVE   C,L0
Force termination of a DO-LOOP at the next opportunity by setting the loop limit equal to the current value of the index. The index itself remains un-changed, and execution proceeds normally until LOOP or +LOOP is encountered.

LFA   pfa --- lfa
Convert the parameter field address of a dictionary definition to its link field address.

LIMIT   --- n
A constant leaving the address just above the highest memory available for a disc buffer. Usually this is the highest system memory.

LIST   n ---   L0
Display the ascii text of screen n on the selected output device. SCR contains the screen number during and after this process.

LIT   --- n   C2,L0
Within a colon-definition, LIT is automatically compiled before each 16 bit literal number encountered in input text. Later execution of LIT causes the contents of the next dictionary address to be pushed to the stack.

LITERAL   n ---   (compiling)   P,C2,L0
If compiling, then compile the stack value n as a 16 bit literal. This definition is immediate so that it will execute during a colon definition. The intended use is:

```
: xxx   [ calculate ]  LITERAL  ;
```

Compilation is suspended for the compile time calculation of a value. Compilation is resumed and LITERAL compiles this value.

LOAD   n ---   L0
Begin interpretation of screen n. Loading will terminate at the end of the screen or at ;S. See ;S and -->.

LOOP   addr n --- (compiling)   P,C2,L0
Occurs in a colon-definition in form:

```
DO  ...  LOOP
```

At run-time, LOOP selectively controls branching back to the corresponding DO based on the loop index and limit. The loop index is incremented by one and compared to the limit. The branch back to DO occurs until the index equals or exceeds the limit; at that time, the parameters are discarded and execution continues ahead.

At compile-time, LOOP compiles (LOOP) and uses addr to calculate an offset to DO. n is used for error testing.

M*   n1 n2 --- d
A mixed magnitude math operation which leaves the double number signed product of two signed numbers.

M/   d n1 --- n2 n3
A mixed magnitude math operator which leaves the signed remainder n2 and signed quotient n3, from a double number dividend and divisor n1. The remainder takes its sign from the dividend.

M/MOD   ud1 u2 --- u3 ud4
An unsigned mixed magnitude math operation which leaves a double quotient ud4 and remainder u3, from a double dividend ud1 and single divisor u2.

MAX   n1 n2 --- max   L0
Leave the greater of two numbers.

MESSAGE   n ---
Print on the selected output device the text of line n relative to screen 4 of drive 0. n may be positive or negative. MESSAGE may be used to print incidental text such as report headers. If WARNING is zero, the message will simply be printed as a number (disc unavailable).

MIN   n1 n2 --- min   L0
Leave the smaller of two numbers.

MINUS   n1 --- n2   L0
Leave the two's complement of a number.

MOD   n1 n2 --- mod   L0
Leave the remainder of n1/n2, with the same sign as n1.

MON
Exit to the system monitor, leaving a re-entry to Forth, if possible.

MOVE addr1 addr2 n ---
Move the contents of n memory cells (16 bit contents) beginning at addr1 into n cells beginning at addr2. The contents of addr1 are moved first. This definition is appropriate on word addressing computers.

NEXT
This is the inner interpreter that uses the interpretive pointer IP to execute compiled Forth definitions. It is not directly executed but is the return point for all code procedures. It acts by fetching the address pointed by IP, storing this value in register W. It then jumps to the address pointed to by the address pointed to by W. W points to the code field of a definition which contains the address of the code which executes for that definition. This usage of indirect threaded code is a major contributor to the power, portability, and extensibility of Forth. Locations of IP and W are computer specific.

NFA pfa --- nfa
Convert the parameter field address of a definition to its name field.

NUMBER addr --- d
Convert a character string left at addr with a preceding count, to a signed double number, using the current numeric base. If a decimal point is encountered in the text, its position will be given in DPL, but no other effect occurs. If numeric conversion is not possible, an error message will be given.

OFFSET --- addr U
A user variable which may contain a block offset to disc drives. The contents of OFFSET are added to the stack number by BLOCK. Messages by MESSAGE are independent of OFFSET. See BLOCK, DR0, DR1, MESSAGE.

OR n1 n2 --- or L0
Leave the bit-wise logical or of two 16 bit values.

OUT --- addr U
A user variable that contains a value incremented by EMIT. The user may alter and examine OUT to control display formatting.

OVER n1 n2 --- n1 n2 n1 L0
Copy the second stack value, placing it as the new top.

PAD --- addr L0
Leave the address of the text output buffer, which is a fixed offset above HERE.

PFA nfa --- pfa
Convert the name field address of a compiled definition to its parameter field address.

POP
The code sequence to remove a stack value and return to NEXT. POP is not directly executable, but is a Forth re-entry point after machine code.

PREV --- addr
A variable containing the address of the disc buffer most recently referenced. The UPDATE command marks this buffer to be later written to disc.

PUSH
This code sequence pushes machine registers to the computation stack and returns to NEXT. It is not directly executable, but is a Forth re-entry point after machine code.

PUT
This code sequence stores machine register contents over the topmost computation stack value and returns to NEXT. It is not directly executable, but is a Forth re-entry point after machine code.

QUERY
Input 80 characters of text (or until a "return") from the operator's terminal. Text is positioned at the address contained in TIB with IN set to zero.

QUIT L1
Clear the return stack, stop compilation, and return control to the operator's terminal. No message is given.

R --- n
Copy the top of the return stack to the computation stack.

R# --- addr U
A user variable which may contain the location of an editing cursor, or other file related function.

R/W   addr blk f ---
The fig-FORTH standard disc read-write linkage. addr specifies the source or destination block buffer, blk is the sequential number of the referenced block, and f is a flag for f=0 write and f=1 read. R/W determines the location on mass storage, performs the read-write and performs any error checking.

R>   --- n   L0
Remove the top value from the return stack and leave it on the computation stack. See >R and R.

R0   --- addr   U
A user variable containing the initial location of the return stack. Pronounced R-zero. See RP!

REPEAT   addr n ---   (compiling)   P,C2
Used within a colon-definition in the form:

    BEGIN ... WHILE ... REPEAT

At run-time, REPEAT forces an unconditional branch back to just after the corresponding BEGIN.

At compile-time, REPEAT compiles BRANCH and the offset from HERE to addr. n is used for error testing.

ROT   n1 n2 n3 --- n2 n3 n1   L0
Rotate the top three values on the stack, bringing the third to the top.

RP!
A computer dependent procedure to initialize the return stack pointer from user variable R0.

S->D   n --- d
Sign extend a single number to form a double number.

S0   --- addr   U
A user variable that contains the initial value for the stack pointer. Pronounced S-zero. See SP!

SCR   --- addr   U
A user variable containing the screen number most recently referenced by LIST.

SIGN   n d --- d   L0
Stores an ascii "-" sign just before a converted numeric output string in the text output buffer when n is negative. n is discarded, but double number d is maintained. Must be used between <# and #>.

SMUDGE
Used during word definition to toggle the "smudge bit" in a definition's name field. This prevents an uncompleted definition from being found during dictionary searches, until compiling is completed without error.

SP!
A computer dependent procedure to initialize the stack pointer from S0.

SP@   --- addr
A computer dependent procedure to return the address of the stack position to the top of the stack, as it was before SP@ was executed (e.g. 1 2 SP@ @ . . . would type 2 2 1).

SPACE   L0
Transmit an ascii blank to the output device.

SPACES   n ---   L0
Transmit n ascii blanks to the output device.

STATE   --- addr   L0,U
A user variable containing the compilation state. A non-zero value indicates compilation. The value itself may be implementation dependent.

SWAP   n1 n2 --- n2 n1   L0
Exchange the top two values on the stack.

TASK
A no-operation word which can mark the boundary between applications. By forgetting TASK and re-compiling, an application can be discarded in its entirety.

THEN   P,C0,L0
An alias for ENDIF.

TIB   --- addr   U
A user variable containing the address of the terminal input buffer.

TOGGLE   addr b ---
Complement the contents of addr by the bit pattern b.

TRAVERSE   addr1 n --- addr2

Move across the name field of a fig-FORTH variable length name field. addr1 is the address of either the length byte or the last letter. If n=1, the motion is toward hi memory; if n=-1, the motion is toward low memory. The addr2 resulting is the address of the other end of the name.

TRIAD   scr ---

Display on the selected output device the three screens which include that numbered scr, beginning with a screen evenly divisible by three. Output is suitable for source text records, and includes a reference line at the bottom taken from line 15 of screen 4.

TYPE   addr count ---   L0

Transmit count characters from addr to the selected output device.

U*   u1 u2 --- ud

Leave the unsigned double number product of two unsigned numbers.

U/   ud u1 --- u2 u3

Leave the unsigned remainder u2 and unsigned quotient u3 from the unsigned double dividend ud and unsigned divisor u1.

UNTIL   f ---   (run-time)
    addr n ---   (compile)   P,C2,L0

Occurs within a colon-definition in the form:

```
    BEGIN  ...  UNTIL
```

At run-time, UNTIL controls the conditional branch back to the corresponding BEGIN. If f is false, execution returns to just after BEGIN; if true, execution continues ahead.

At compile-time, UNTIL compiles (0BRANCH) and an offset from HERE to addr. n is used for error tests.

UPDATE   L0

Marks the most recently referenced block (pointed to by PREV) as altered. The block will subsequently be transferred automatically to disc should its buffer be required for storage of a different block.

USE   --- addr

A variable containing the address of the block buffer to use next, as the least recently written.

USER   n ---   L0

A defining word used in the form:

```
    n  USER  cccc
```

which creates a user variable cccc. The parameter field of cccc contains n as a fixed offset relative to the user pointer register UP for this user variable. When cccc is later executed, it places the sum of its offset and the user area base address on the stack as the storage address of that particular variable.

VARIABLE   E,L0

A defining word used in the form:

```
    n  VARIABLE  cccc
```

When VARIABLE is executed, it creates the definition cccc with its parameter field initialized to n. When cccc is later executed, the address of its parameter field (containing n) is left on the stack, so that a fetch or store may access this location.

VOC-LINK   --- addr   U

A user variable containing the address of a field in the definition of the most recently created vocabulary. All vocabulary names are linked by these fields to allow control for FORGETting thru multiple vocabularies.

VOCABULARY   E,L1

A defining word used in the form:

```
    VOCABULARY  cccc
```

to create a vocabulary definition cccc. Subsequent use of cccc will make it the CONTEXT vocabulary which is searched first by INTERPRET. The sequence "cccc DEFINITIONS" will also make cccc the CURRENT vocabulary into which new definitions are placed.

In fig-FORTH, cccc will be so chained as to include all definitions of the vocabulary in which cccc is itself defined. All vocabularies ultimately chain to Forth. By convention, vocabulary names are to be declared IMMEDIATE. See VOC-LINK.

VLIST

List the names of the definitions in the context vocabulary. "Break" will terminate the listing.

```
WARNING     ---  addr                        U
       A user variable containing a value
       controlling messages.  If = 1 disc is
       present, and screen 4 of drive 0 is the
       base location for messages.  If = 0, no
       disc is present and messages will be
       presented by number.  If = -1, execute
       (ABORT) for a user specified procedure.
       See MESSAGE, ERROR.

WHILE           f  ---   (run-time)
           ad1  n1  ---  ad1  n1  ad2  n2   P,C2
       Occurs in a colon-definition in the
       form:
           BEGIN  ...  WHILE (tp)  ...  REPEAT
       At run-time, WHILE selects conditional
       execution based on boolean flag f.  If
       f is true (non-zero), WHILE continues
       execution of the true part thru to
       REPEAT, which then branches back to
       BEGIN.  If f is false (zero), execution
       skips to just after REPEAT, exiting the
       structure.

       At compile time, WHILE emplaces
       (0BRANCH) and leaves ad2 of the reserved
       offset.  The stack values will be
       resolved by REPEAT.

WIDTH       ---  addr                        U
       In fig-FORTH, a user variable containing
       the maximum number of letters saved in
       the compilation of a definition's name.
       It must be 1 thru 31, with a default
       value of 31.  The name character count
       and its natural characters are saved, up
       to the value in WIDTH.  The value may be
       changed at any time within the above
       limits.

WORD        c  ---                          L0
       Read the next text characters from the
       input stream being interpreted, until a
       delimiter c is found, storing the packed
       character string beginning at the dic-
       tionary buffer HERE.  WORD leaves the
       character count in the first byte, the
       characters, and ends with two or more
       blanks.  Leading occurrences of c are
       ignored.  If BLK is zero, text is taken
       from the terminal input buffer, other-
       wise from the disc block stored in BLK.
       See BLK, IN.

X
       This is a pseudonym for the "null" or
       dictionary entry for a name of one
       character of ascii null.  It is the
       execution procedure to terminate
       interpretation of a line of text from
       the terminal or within a disc buffer,
       as both buffers always have a null at
       the end.

XOR         n1  n2  ---  xor                 L1
       Leave the bitwise logical exclusive-or
       of two values.

[                                           P,L1
       Used in a colon-definition in form:
           : xxx  [  words  ]  more  ;
       Suspend compilation.  The words after
       [ are executed, not compiled.  This
       allows calculation or compilation
       exceptions before resuming compilation
       with ].  See LITERAL, ].

[COMPILE]                                    P,C
       Used in a colon-definition in form:
           : xxx  [COMPILE]  FORTH  ;
       [COMPILE] will force the compilation
       of an immediate definition, that would
       otherwise execute during compilation.
       The above example will select the FORTH
       vocabulary when xxx executes, rather
       than at compile time.

]                                             L1
       Resume compilation, to the completion of
       a colon-definition.  See [.
```