

F.I.G FORTH



F.I.G. FORTH
MIX VERSION BY KATHY
PRODUCED BY K. WICK
ARTWORK SQUARE

THIS VERSION IS PART OF THE COMPANY OF THE FORTH INTEREST GROUP
P.O. BOX 1105, SAN CARLOS, CA 94070

A SYNTAXsoft PRODUCTION

SYNTAXsoft, 4, BLACK STREET, BOWLEY, B010 1UH

A SYNTAXsoft PRODUCTION



F.I.G FORTH

F.I.G. FORTH

MIX VERSION BY KEITH JONES.
PRODUCED BY K. HOOK
ARTWORK SYNART

THIS VERSION IS PROVIDED THROUGH THE COURTESY OF THE FORTH INTEREST GROUP
P.O. BOX 1105. SAN CARLOS, CA 94070

Provided through the courtesy of the FORTH INTEREST GROUP,PO Box 1105.San Carlos,CA 94070.

Thanks to :

Trevor for help and ideas.

Vaughan for patience,sustenance and understanding.

The aim of this manual is to give the new user a basic introduction to FORTH. It is not a complete teaching aid, as there are many books on the market which can do this better than I. The manual is divided into three parts. The first part describes the fig-FORTH words - a word in FORTH is the name of any definition. It is not to be confused with its meaning in machine code for a sixteen bit value.

The second section is devoted to the MTX extensions and the final part is devoted to the screen editor.

FORTH is a strange language, but it is fast, compact and efficient. I hope you have fun using it. At present work is commencing on a cross compiler so that your definitions can run independently of the main body. I will be pleased to answer any questions or enquires about this program, or FORTH in general. All mail should be addressed to me c/o GENPAT.

Next to each word you will see symbols e.g. next to FILL are the symbols `addr` `quan` `b` ---. Anything to the left of the three dashes is expected to be on the stack for the word to execute. Symbols to the right of the dashes, if any, are values left on the stack after the word has been executed. All the symbols are explained in the key before the definitions.

NUMBERS

FORTH deals with numerical values in four different ways. It will accept signed integers in the range of -32,768 to +32,767. Unsigned integers (0 to 65,535) are accepted and both of these types of numbers are called single length values. Double length numbers are either signed (-2,147,483,648 to +2,147,483,647) or unsigned (0 to 4,294,967,295).

MATHS

FORTH uses post-fix maths, also known as Reverse Polish Notation. It is worth working with the maths until you are fully aware of its operation.

To use post-fix you first place the values on the stack and then supply the maths operator, e.g. "4+2" would be "4 2 +", "1/2" would be "1 2 /". A space must be typed between each number and all operations. It may take some time to get used to this way of working, but it is faster as computers use post-fix within their own floating point maths.

The above examples were easy, but what about an expression like " $((10+3)*5)/2$ ". This is just as easy as the above, it just involves more operators! The stack is a last in first out stack which grows down in memory, so if you type 3 <RET> 4 <RET> the stack has 4 at the top and 3 below it. When you type + <RET> the maths interpreter takes 4 and 3 off the stack, adds them together and leaves the result, 7, on top of the stack. So now to our more complex equation you would type 10 <RET> 3 <RET> + <RET> 5 <RET> * <RET> 2 <RET> / <RET> and this would leave the result on top of the stack. You don't need to type all the <RET>'s in of course, $10\ 3\ +\ 5\ *\ 2\ /\ <RET>$ would leave the same result on the stack. Don't forget that FORTH is integer only maths. Floating point can be done in FORTH, but it slows everything down. FORTH does provide MOD operations, which are explained in the FORTH definitions section of this manual.

CASE CONSTRUCT

An additional feature of MTX-FORTH is the CASE statement. This replaces multiple IF...ELSE...THEN statements and makes the listing easier to follow.

For instance :

```
: EXAMPLE
DUP 0=IF ." ZERO " ELSE
DUP 1=IF ." ONE " ELSE
DUP 2=IF ." TWO " ELSE
DUP 3=IF ." THREE " ELSE

THEN THEN THEN THEN DROP ;
```

Can be replaced by ;

```
: EXAMPLE
0 OF ." ZERO " ENDOF
1 OF ." ONE " ENDOF
2 OF ." TWO " ENDOF
3 OF ." THREE " ENDOF
ENDCASE ;
```

As you can see, this is much more compact and easier to follow. If it fails all tests then it will not execute any of the statements.

SYNTAX

Syntax in FORTH will seem curious at first. Having to type 4 VS rather than VS 4 seems silly and unnecessary, but this is not so. Some commands need a value on the stack before they can operate, hence the number must come first.

The same syntax applies to conditional statements e.g. the BASIC statement IF A=1 THEN "condition1" ELSE "condition2" translates in FORTH to A 1 = IF "condition1" else "condition2" THEN. THEN in FORTH can also be called ENDIF.

ERROR MESSAGES

Should the compiler find an error at any point then it clears all stacks and returns an error message with a numerical value. These values mean;

MSG#	Meaning
0	Word does not exist.
1	Stack is empty.
2	Dictionary full.
3	Incorrect address mode.
4	Definition for this word already exists.
6	Outside disc range (Pages 0-4).
7	Stack full.
17	Use in a colon word only.
18	Execution only.
19	Condition not paired.
20	Definition not finished.
21	In protected dictionary.
22	Use only when LOADING.
23	Off current editing screen.

24 Declare vocabulary.

! n addr --- LO
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 LO
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 LO
Terminates numeric output conversion by dropping d, leaving the text address and character count suitable for TYPE.

#BUF --- n
A constant which leaves the number of buffers on the stack.

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

' --- addr P,LO
Leaves the parameter field address of dictionary word nnnn. As a compiler 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,LO
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 paranthesis 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 test for loop completion. See +LOOP.

(ABORT) C
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 unconvertable digit. Used by NUMBER.

* n1 n2 --- prod LO
Leaves the signed product of two signed numbers.

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

*/MOD n1 n2 n3 --- n4 n5 LO
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 LO
Leave the sum of n1 and n2.

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

+ - n1 n2 --- n3
Apply the sign of n2 to n1, which is left in 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,LO
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 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 compile 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 --- LO
Store n into the next available dictionary memory cell, advancing the dictionary


```
OBRANCH      f ---          C2
The run-time procedure to conditionally branch .If f is false (zero),the following in-line
parameter is added to the interpretative 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.
```

```
2I          d addr ---
Stores 32 bits of d at addr.d is split into two 16-bit values (nlow and nhigh) with nhigh
stored at addr and nlow stored at addr+2.
```

```
2@          addr --- d
Fetches 32-bit value from addr.d is placed on the stack as two 16-bit values (nlow and nhigh).
```

```
2DROP      d ---
Drops a 32-bit value from the stack.
```

```
2DUP      d --- d d
Duplicates a 32-bit value.
```

```
2SWAP     d1 d2 --- d2 d1
Will swap two 32-bit values when on the stack.
```

```
:          P,E,LO
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,LO
Terminate a colon-definition and stop further compilation. Compiles the run-time ;S.
```

```
;CODE     P,C,LO
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,LO
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          LO
Leave a true flag if n1 is less than n2; otherwise leave a false flag.
```


ABS n --- u LO
 Leave the absolute value of n as u.

AGAIN addr n --- (compiling) P,C2,LO
 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 --- LO
 Add the signed number to the dictionary pointer DP. May be used to reserve dictionary space or re-originate memory. n is with regard to computer address type (byte or word).

AND n1 n2 --- n2 LO
 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,LO
 A user variable containing the current number base used for input and output conversion.

BEGIN --- addr n (compiling) P,LO
 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 will always occur.
 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,LO
 A user variable containing the block number being interpreted. If zero, input is being taken from the terminal input buffer.

BLOCK n --- addr LO
 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 which ever buffer was least recently written. If the

block occupying that buffer has been marked as updated, it is rewritten to disc before block n is read into the buffer. See also BUFFER, R/W UPDATE FLUSH.

- BRANCH** C2,LO
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 and REPEAT.
- BUFFER** n --- addr
Obtain the next memory buffer, assigning it to block n. If the contents of the buffer is marked as updated, it is written to the disc. The block is not read from disc. The address left is the first cell within the buffer for data storage
- CI** b addr ---
Store 8 bits at address.
- C/L** --- n
Constant which leaves the number of characters per line on the stack. By FORTH convention this is 64
- C,** b ---
Store 8 bits of b into the next available dictionary byte, advancing the dictionary pointer.
- CO** addr --- b
Leave the 8 bit contents of memory address
- CASE** --- n
Must be used within a colon definition. For use see introduction. The value on the stack is for compiler error checking.
- CFA** pfa --- cfa
Convert the parameter field address of a definition into its code field address
- CMOVE** from to count ---
Move the specified number of bytes beginning at address from to address to. The contents of address from is moved first preceding toward high memory.
- COLD**
The cold start procedure to adjust the dictionary pointer to the minimum standard and restart via ABORT.
- 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 --- LO
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 onto the stack.
- CONTEXT** --- addr U,LO
A user variable containing a pointer to the vocabulary within which dictionary searches will first begin.
- COUNT** addr1 --- addr2 n LO
Leave the byte address addr2 and byte count n for a message text beginning at address addr1. It is preassumed 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 words parameter field. The new word is created in the CURRENT vocabulary.
- CSP 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. 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.
- 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,LO

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 the 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> LO

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,LO

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 the output column location of a decimal point,in user generated formatting.The default value on single number input is -1.

DROP n ---
Drop the number from the stack

DUP n --- n n LO
Duplicate the value on the stack.

EDITOR
Make current vocabulary EDITOR,for screen editing see section 3

ELSE addr1 n1 --- addr2 n2
 (compiling) P,C2,LO
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 effect.

At compile time ELSE replaces 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 --- LO

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

EMPTY-BUFFERS LO

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.

IN --- addr LO
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 sequentially 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 message echoing the name with a " ?" will be given. Text input will be taken according to the convention for WORD. If a decimal point is found as part of a number, a double number value will be left. The decimal point has no other purpose than to force this action. See NUMBER.

KEY --- c LO
Leave the ascii value of the next terminal key pressed.

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

LEAVE C,LO
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 unchanged, 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.

LINK n ---
If n>0 then anything which is printed on the screen will also be sent to the printer attached to the centronics interface.

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

LIT --- n C2,LO
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,LO
If compiling, then compile the stack value of 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 --- LO
Begin interpretation of screen n. Loading will terminate at the end of the screen or at ;s. See ;S and -->.

LOOP addr n --- (compiling) F,C2,LO
 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 LO
 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 un-available).

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

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

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

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

NOOP
 A FORTH no operation. Basically does nothing !

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 in DPL, but no other effect occurs. If numeric conversion is not possible, an error message will be given.

OF n --- f
 Used within a colon-definition. The value n is supplied by the user and leaves a flag on the stack for conditional branching.

OFF
 Turns the cursor off

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

ON
 Turns the cursor on

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.

P! b port# ---
 Outputs byte b to port#. So 255 2 P! is equivalent to OUT (2), 255

P@ port# --- b
 Equivalent to IN (port#).

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.

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.

QUERY
 Input 80 characters of text (or until a "return") from the operators 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 operators terminal. No message is given.

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

R# --- addr
 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 all error checking.

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

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

USER n --- LO
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,LO
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 through multiple vocabularies.

VOCABULARY E,L
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 this 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.

WHERE n1 n2 ---
When attempting to LOAD a screen if an ERROR is detected then n1 and n2 are left on the stack. WHERE uses these values to print the line and an indication of the error. It then makes EDITOR the current vocabulary.

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 a Boolean flag f. If f is true (non-zero), WHILE continues execution of the true part through 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 (OBRANCH) 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 name. It must be between 1 and 31, with a default value of 31. The name character count and its natural characters are saved, up to the value of 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 character c is found, storing the packed character string beginning at the dictionary 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, otherwise from the disc block stored in BLK. See BLK, IN.

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

[P,L1
Used in a colon-definition in the 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 the 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 [.

MEMOTECH SPECIFIC COMMANDS

All symbols used are the same as in the MEMOTECH manual, reference section, as are the results of the command

ADJSPR p n v ---

ATTR p state ---

CLS

COLOUR p n ---

CRVS s h w y x t n ---

CSR x y ---

CTLSPR p x ---

GENPAT d1 d2 d3 d4 d5 d6 d7 d8 n p ---

GR x y p --- f

This is an equivalent of the MTX GR\$. The flag will be 1 if the pixel at x y is on, and 0 if off.

INK colour ---
 INKEY --- c
 Will leave the ascii value of any key pressed during the execution of the word. If no key
 pressed will leave a value of 0. N.B. This word has an extremely fast read time.
 LINE x1 y1 x2 y2 ---
 MVSPR p n d ---
 PAPER colour ---
 PLOT x y ---
 SOUND channel freq volume ---
 SPK --- c
 Equivalent to SPK\$: Will read the characters at the cursor location, and leave this as an ascii
 value. So its format is :-
 x y CSR SPK.
 SPRITE n pat xp yp xs ys col ---
 VIEW dir dis ---
 VS n ---

MTX FORTH only uses VS 4 and VS 5. All others may be redefined.

SECTION 3. SCREEN EDITOR.

 TEXT c ---
 Accept following text to PAD. c is delimiter.
 ?LINE n --- addr
 Will leave address of line n of current screen.
 R# --- addr
 A user variable which contains the offset of the editing cursor from the start of the screen.
 #LOCATE --- n1 n2
 From the cursor position leave the line number (n2), and the offset into the line (n1).
 #LEAD --- line-addr offset
 #LAG --- cursor-addr till EOL
 -MOVE addr n ---
 Move a line of text from addr to line n of current screen.
 H n ---
 Hold numbered line at PAD.
 E n ---
 Erase line with n blanks.

S n ---
Spread at line n. Line n and following lines move down one. Line n becomes blank.

D n ---
Delete line n but hold in PAD.

M n ---
Move cursor by a signed amount and then print it. N.B. The cursor is represented by the underline character _.

T n ---
Type line n and save it in PAD

L
List current screen

REP
Replace line n with text in PAD.

P n ---
Put the following text on line n.

INS n ---
Spread at line n and INSert text from PAD.

TOP
Position cursor at top of current screen.

CLEAR n ---
Clear current screen. Can be used to select screen n for editing.

FLUSH
See section 1 for definition.

COPY n1 n2 ---
Copy screen n1 to screen n2.

-TEXT addr1 count addr2 --- flag
True flag if text at addr1 matches text at addr2. Count is the number of characters to search from starting at addr2.

MATCH cursor-addr till-EOL str-addr str-count --- tf
 cursor-count-till-end-of-matching-text

Match the text at str-addr with all strings on the cursor line forward from the cursor. The values left allow the cursor to be updated either to the end of the matching text or to the start of the next line.

1LINE --- f
Scan the cursor line for a match to PAD text. f represents boolean flag. If true then the cursor will be placed at text, if false then cursor will be placed at start of next line.

FIND
Search for match of text in PAD, from the cursor position to the end of the screen. If no match is found then cursor is positioned at top of screen and an error message is given.

DELETE n ---
Delete n characters prior to the cursor.

- N Find next occurrence of PAD text.
- F Input following text to PAD and search for a match from the cursor position to the end of the screen.
- B Back up cursor by text in PAD.
- X Delete next occurrence of following text.
- TILL Delete on cursor line until end of following text.
- C Spread at cursor and copy the following text at the cursor line.

PROGRAMMED BY KEITH JONES.
PRODUCED BY SYNTAXSOFT.

..n15

The code for FORTH is located between #4100 and #6100. The save program is therefore:-

```
LD HL,#4100 ;START OF BLOCK.
LD DE,#2000 ;LENGTH OF BLOCK
XOR A
LD (#FD67),A
LD (#FD68),A
CALL #0AAE
RET
```

N.B. Should you exit from FORTH then type <CTRL> L in order to clear VS 0. You may then list the program, where two lines are contained which will permit a jump to WARM or COLD start. Do not type <RET> straight away as this may overwrite some of the FORTH code.

In the section on Memotech commands, the syntax for the CRVS command has been changed to

```
CRVS      n t x y w h s ---
```

as per the Memotech manual.

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.
LO	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.