HERE (-- addr)

Places the contents of DP (pointer to the next available dictionary location) onto the parameter stack. Indicates size of COMPILED image from the beginning of the kernel. The next word that is compiled will go "here".

```
CREATE FOO HERE  ( start simple array )
23 , 987 , 1722 ,  ( lay in some data )
HERE SWAP - CELL/ CONSTANT NUM_FOO
( figure out how many cells are in FOO )
```

HEX (--)

Sets user BASE to 16 decimal . All number input and output conversions will be done in hex .

```
DECIMAL 10 HEX . ( will display  A  )
```

Related Words: DECIMAL BINARY COMMAS BASE $

HICASE (-- n) "hi case"

HICASE is a CONSTANT used to determine the desired characteristics of output ASCII case. See OUTPUT-CASE.

HIDEMODULE (-- , <modulename>)

See the chapter on MODULES.

HOLD (char --)

Add the ASCII character to the number being converted to a text string. Used by # to store ASCII characters in memory.

```
: .TIME ( time -- )
    S->D <# # # ASCII : HOLD #S #> TYPE ;
1032 .TIME  ( prints 10:32 )
```

Related Words: <# # #S #> COMMAS

I (-- index)

Place current DO LOOP index on stack .

NOTE: this is the only approved method of retrieving the index value of the currently running DO-LOOP.

```
: PRINT-0-TO-9  ( -- )  10 0 DO  I .  LOOP CR ;
```

Related Words: J IK DO LOOP +LOOP -DO -LOOP

ICON?

```
ICON_LIB
ICON_NAME
```

Manage the Amiga Icon Library. See :Library

ID. (nfa --) "i d dot"

Print the name of the Forth word whose NFA is on the stack. COUNT TYPE won't work because of the smudge bits, etc.

Standards: fig '79 '83. Called .NAME or .ID in some '83 systems.

```
' NEGATE >NAME ID.
```

IF (flag --)

Execute the following code if the flag is true (non-zero). If the flag is false, then skip forward to the next ELSE or THEN. IF may be used during COMPILE MODE only, and will generate an error if used otherwise.

```
: TEST  ( n -- , print based on N )
    DUP 10 >
    IF . ."  is bigger than 10!" CR
    ELSE . ." is small." CR
    THEN
;
```

IF is an IMMEDIATE word and will execute immediately in COMPILE MODE. It has no effect on the data stack, but will store data needed to resolve the branch on a special stack called the "User Stack".

Compile time: user stack: (-- if-flag address)

Related Words: ELSE THEN = TRUE BEGIN ?PAIRS

IF-NOT (flag --)

Operates identically to IF , except the boolean flag satisfying the run-time branch is reversed.

Standards: JForth unique. Simply a faster and smaller NOT IF .

IF>ABS (relative-addr -- absolute-addr) "if to abs"

Convert address UNLESS it is NULL, ie. zero. This is typically used with Amiga routines that use an address or a NULL. The NULL has a special meaning and should not be converted.

Related Words: >ABS >REL IF>REL

IF>REL (absolute-addr -- relative-addr) "if to abs"

Convert address UNLESS it is NULL, ie. zero. NULL is often used as a flag so it must be preserved, not converted.

Related Words: >ABS >REL IF>ABS

IFLEAVELONG (-- var-addr)

See the chapter on CLONE.

IK (-- 3rd-loop-out-index)

Retrieve the index of the loop in which it is nested 3 deep. If I is for the current loop and J is for the next outer loop, then IK is for the loop next further out.

Standards: JForth unique

```
: EXAMPLE  ( -- )
    3 0
    DO   4 0
       DO  5 0
          DO  IK . J . I . CR
          LOOP
       LOOP
    LOOP ;
```

In the above example, the leftmost number will change the most slowly because it is the index of the

outer loop.

IMMEDIATE (--)

Changes the word just compiled to be immediate. The programmer, by making a definition IMMEDIATE, sets the word to be executed, rather than compiled, when found within a colon definition. This is useful for extending the compiler.

```
: PRINT-HERE  ( -- , prints current DP, even while compiling )
    HERE .
; IMMEDIATE
: FOO 23 DUP PRINT-HERE + ;  ( will print HERE now )
```

To cause an IMMEDIATE word to be compiled, it should be preceded with [COMPILE] .

```
: FANCY-PRINT-HERE  ( -- , fancier version )
    ." Here = " [COMPILE] PRINT-HERE
;
```

IMMEDIATE? (nfa -- flag)

Given the name-field-address of a dictionary header, return whether it is compiled as an IMMEDIATE definition.

```
' LITERAL >NAME IMMEDIATE?  ( -- true  )  .  64  ok
' DROP    >NAME IMMEDIATE?  ( -- false )  .  0  ok
```

INCLUDE (<filename> --)

Input Forth code from the given file. This is used to compile source code from a file. If an INCLUDE command is encountered in a file it will include that file then continue with the original file. If you want to INCLUDE a file that has spaces in the name, then put the filename in double quotes.

```
INCLUDE RAM:X  ( compile what's in RAM:X )
INCLUDE "RAM:SPACEY FILE"
```

INCLUDE? (<name> <filename> --) "include question"

INCLUDE the given file if the Forth word named is not already compiled. This is useful if you need a word from a file, and want to compile it if it's not already loaded.

```
INCLUDE? MSEC JU:MSEC
INCLUDE? GR.INIT JU:AMIGA_GRAPH
```

If you are using ANEW in a file, you should place the INCLUDE? commands before you call ANEW .

Related Words: EXISTS? FILEWORD

INIT-USP (--) (? --US--) "init u s p"

Initialize the user stack. JForth incorporates a third stack mainly for use in compiling DO LOOP and other control structures.

Standards: JForth unique.

Related Words: US@ >US US>

INITCLONE (--)

See the chapter on CLONE.

INITIALIMAGESIZE (-- var-addr)

See the chapter on CLONE.

INLINE (--)

INLINE is an immediate compiler directive used to inform the compiler that the user wishes this word to always be compiled inline; i.e. the entire body of the definition (minus any trailing RTS) will be copied to HERE when later compiled. This avoids the overhead of a subroutine call and is thus faster.

INLINE may only precede the ; marking the end of a colon definition, and will cause the compiler to verify that the word may safely be compiled in this manner.

If so, the word will be marked INLINE by setting bit 31 in the size-field. Once marked, the word will unconditionally be compiled inline when referenced in later colon definitions.

If not, a warning message is issued, informing the operator that the word cannot be compiled inline, and the word will be set to CALLED .

Standards: JForth unique.

```
: EXAMPLE  ( -- )  ROT OVER +    INLINE ;
: FOO 23 34 45 EXAMPLE * . ;
( EXAMPLE will be expanded inline in FOO, not called. )
```

Related Words: ; BOTH

INLINE+ (n --) (addr -- addr+n)

INLINE+ adds the top of the data stack to the value on return stack. INLINE+ is one of a set of 4 words used to improve the readability and transportability of programs that use inline data.

INLINE@ (-- inline-data-address) "inline fetch"

INLINE@ returns the address of the inline data.

INLINE+ INLINE> >INLINE (INLINE is not related)

INLINE> (absolute-address --R--)
(-- inline-data-address)

INLINE> takes the return stack absolute address, converts it to a relative address, then moves it to the data stack.

Related Words: INLINE+ INLINE@ >INLINE (INLINE is not related)

INTERPRET (--)

This is a DEFERed word that is used to process a line of Forth input. See (INTERPRET).

Related Words: $INTERPRET QUIT COMPILING? INTERPRET? QUERY (INTERPRET)

INTERPRETING? (-- flag)

Returns true if STATE is zero, ie. interpreting, false if STATE is true, ie. compiling. When you are inside a colon definition, STATE is true. Usually used in IMMEDIATE words that behave differently when interpreting or compiling.

Related Words: STATE ?COMP

INTUITION? intuition question
INTUITION_LIB intuition lib

```
INTUITION_NAME  intuition name
```

Used for managing the INTUITION Library. See :LIBRARY.

IS (cfa <name> --) "is"

IS provides the mechanism for setting the action vector for a DEFERed word. See DEFER.

```
' DROP IS EMIT
( sets the deferred word EMIT to execute DROP )
```

Related Words: DEFER CFA,

J (-- index)

Used in nested DO LOOPs. Gets the index of the DO LOOP one level above.

```
: TESTJ   2 0 DO  3 0 DO I . J . LOOP LOOP ;
TESTJ  ( Prints    0 0 1 0 2 0  <CR> 0 1 1 1 2 1 )
                   I J I J I J       I J I J I J
```

Related Words: I IK DO LOOP

KEY (-- char)

KEY is used by programs to obtain single-character input; this is taken from the AmigaDOS file-pointer stored in the variable CONSOLEIN, waiting until a key has been pressed.

KEY is a DEFERred execution word and is normally set to (KEY). Any word that replaces (KEY) should call FLUSHEMIT to force out any stored up characters.

```
: PAUSE ." Hit key to continue: " KEY DROP ;
```

If you clone a program with KEY, you may want to open a RAW: window to get immediate response. See CONSOLE! .

Related Words: ?TERMINAL EXPECT EMIT (KEY)

LATEST (-- nfa)

Returns the name-field of the most recently created dictionary header in the CURRENT vocabulary.

```
: FOO ;    LATEST ID.  ( Prints "FOO" )
```

Related Words: CURRENT HERE

LAYERS?

```
LAYERS_LIB
LAYERS_NAME
```

Standards: JForth internal

Used to manage the LAYERS library. See :LIBRARY .

LEAVE (--)

LEAVE a DO LOOP. Used in the form : ... DO ... LEAVE ... LOOP ; At run time LEAVE causes immediate exit from the loop with which LEAVE is associated .

```
: SCAN-MAX  ( max  -- , scan until found or max reached. )
    0 DO
        IS-FOUND?  ( -- true_if_found )
        IF ." Found at " I . CR LEAVE ( quit loop )
        THEN
    LOOP ;
```

```
Related Words: DO LOOP +LOOP RETURN
```

LIBVERSION (-- addr)

LIBVERSION is a user-variable, accessed by the XXX? functions (where XXX is the name of an existing LIBRARY, GRAPHICS? for example).

The JForth Library Manager normally opens the latest version of an existing library, but will settle for any version number.

To specify a particular version, the programmer should place the desired version number in LIBVERSION, just prior to calling the appropriate XXX? word.

Standards: Amiga unique

```
Related Words: :LIBRARY DOS
```

LIMIT (-- addr)

LIMIT is applicable only to the BLOCK or SCREEN environment, and returns the address of the end of the virtual buffer area.

```
     PREV @ LIMIT =   \ check to see if time to 'circle' around
```

```
Related Words: FIRST  BLOCK  BUFFER  (LIMIT)
```

LINESFILLV "lines fill virtual"

(file-pointer memory-block max-char-count -- #chars-actually-read)

This word is used to fill a memory area with ASCII text from a specified file. The file should contain standard EOL characters; the area is filled to a line boundary. The memory block MUST have been previously allocated via ALLOCBLOCK.

LINESFILLV accepts 3 parameters:

1) A file-pointer specifying the file to read.

2) The address of a memory-block to put the data read from the file.

3) The byte-count of the memory block area.

LINESFILLV is used by the READLINE function. See READLINE and the chapter on File I/O.

Standards: JForth unique

```
Related Words: OPENFV READLINE FWRITE FREAD FSEEK
```

LINK> (lfa -- cfa) "link to c f a"

LINK> converts the link-field-address of a word to the corresponding code-field-address.

```
     ' TASK  >LINK     ( derives the lfa of TASK )
     ( lfa -- ) LINK>  ( puts it back to the cfa of TASK )
```

```
Related Words: >LINK  >NAME  NAME> VLINK>'
```

LIT (n --inline--) (-- n) "lit"

LIT is compiled when the compiler must create code that serves to push a literal value to the stack at run-time. The value to be pushed is compiled in the dictionary after a call to LIT .

Standards: fig '79 83

```
Related Words: LITERAL INLINE@
```

LITERAL

Compile mode: (n --) Pops the value n from the stack, and compiles it such that it will be pushed at run-time.

Interpret mode: (n -- n) LITERAL has no effect when INTERPRETING.

LITERAL allows certain calculations to be performed once at compile time and stores only the result, along with a run-time operator, LIT, to push the result to the stack. This frees us from having to perform that same calculation repeatedly at run-time just to keep pushing the same value to the stack.

NOTE: LITERAL should not be used if the data to be compiled represents the address of a JForth entity. See the discussion of ALITERAL in the CLONE chapter.

Standards: fig '79 83

```
: TEN-DOZEN  ( -- ten-dozen )  [ 10 12 * ]  LITERAL   ;
```

Related Words: [ ] COMPILE IMMEDIATE LIT

LOAD (screen# --)

LOAD a screen of Forth code. LOAD is only applicable to the BLOCK or SCREEN environment; this word is not available in the standard JForth system...the source file JU:BLOCK must be compiled in ASCII-file format to gain access to LOAD.

LOAD causes interpretation to be redirected to the specified SCREEN of the file being pointed to by the user-variable SCR-FILE.

LOADing will continue to the next SCREEN of the file if the operator --> is encountered during interpretation of the current SCREEN.

Otherwize, LOAD will return when the end of the current SCREEN is reached, at which time interpretation will continue from wherever the LOAD command was invoked.

Standards: '83

Related Words: BLK >IN LIMIT BLOCK FIRST

LOCASE

LOCASE is a CONSTANT used to determine the desired characteristics of output ASCII case . See OUTPUT-CASE.

LONGCFA, (cfa --) "long c f a comma"

LONGCFA, is a DEFERred word; it normally executes (LONGCFA,) which acts to compile a long absolute JUMP-SUBROUTINE to the cfa on the stack.

Of the 3 types of calls available to the JForth compiler, this is the least preferred as it is slightly larger and slower and requires relocation.

For these reasons, LONGCFA, is only invoked by the compiler when the distance from the destination address is too great for the preferred methods to be practical.

Standards: JForth internal

```
\ works, but wastes time and space.
: HARD-VLIST ( -- ) [ ' VLIST LONGCFA, ] ;
```

Related Words: CFA, (LONGCFA,)

LOOP (--)

This is used with DO to form a loop that executes a specified number of times. Each time through

the LOOP the index is incremented until it reaches the limit, at which point execution proceeds past the LOOP command. Use +LOOP if you want to add more than 1 to the loop index. See DO .

```
: HI ( -- , print HIYA 10 times)
    10 0 DO
        CR ." HIYA "
    LOOP ;
```

Related Words: DO +LOOP -LOOP

LPLACE (string-addr string-cnt destination-addr --)

LPLACE is functionally identical to PLACE, except it will unconditionally preserve ASCII-case across the move. See PLACE.

LWORD (char -- addr)

LWORD is functionally identical to WORD, except it will unconditionally preserve ASCII-case across the move. See WORD.