J (-- index)

Like I, but gives the index of the next outermost DO LOOP. If you're nesting DO LOOPs, you can think of I as "I"nner.

```
: SHOWJ  ( -- )
   5 0 DO  4 0 DO J ( index of outer loop ) . LOOP  LOOP
;
SHOWJ ( prints 0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 )
Here's an example of using I and J together:
: SHOWJ&I  ( -- )
   5 0 DO
      15 10 DO
            J. I .
      LOOP CR
   LOOP
;
SHOWJ&I
( prints: )
0 10 0 11 0 12 0 13 0 14
1 10 1 11 1 12 1 13 1 14
2 10 2 11 2 12 2 13 2 14
3 10 3 11 3 12 3 13 3 14
4 10 4 11 4 12 4 13 4 14
```

KEY (-- char)

Returns the ASCII value of a character typed at the keyboard. If no character is ready, it will wait until one is hit. If a character has already been entered, it will return immediately. Use ? TERMINAL to see if a character is ready. KEY is the primary source of input to HForth. It is deferred.

: TESTKEY (--) ." Hit a key" KEY DUP .HEX EMIT ;

LATEST (-- nfa , name of last word defined)

This is the latest word defined in the dictionary. If the compilation failed LATEST will still point to that word. If you're compiling from a file and compilation stops you can enter this to find out what word had the error.

```
LATEST ID.
```

Related words: CURRENT

LEAVE (--)

Use to escape from a DO LOOP. This is useful if you are using a DO LOOP to search for something and want to stop when you find it. It can also be very useful for debugging purposes.

```
: TEST.LEAVE ( -- )
   100 0
   DO I . CR
      ?TERMINAL IF KEY DROP LEAVE THEN
   LOOP
;
```

LITERAL

LITERAL is an IMMEDIATE word that has a different action depending on whether HForth is compiling or not.

Compile time: (n -- , compile the number for later)

Run Time: (n -- n , no effect)

LITERAL is most often used to save the result of a compile time calculation. The calculation can be performed once and the result saved for later use.

```
: HOURS>SECONDS ( #hours -- #seconds , convert )
   [ 60 ( min/hr ) 60 ( sec/min ) * ] LITERAL *
;
```

This is faster than:

```
: H>S ( hrs -- secs ) 60 60 * * ;
```

If you are saving addresses, they must be saved in a relocatable form. See ALITERAL

Related words: ALITERAL [VALUE CONSTANT VARIABLE

LOOP (-- , terminate a DO LOOP, see DO)

LWORD (char -- addr)

Get the next word from the input stream, just like WORD, but preserve lower case characters.

M* (n1 n2 -- d)

Multiply two single precision numbers and generate a double precision product. The 'M' stands for Mixed precision.

M/ (d n -- rem quotient)

Divide a double precision number D by a single precision number N. Generate single precision results.

MAP (--)

Print information about the memory usage of the system. See the section on HForth internals and dictionary structure for more information.

Related words: CODE-SIZE HERE

MAX (n1 n2 -- n1 | n2 , returns largest)

Compare the top two values on the stack and leave the largest.

Related words: MIN CLIPTO >

MEASURE (<code> -- , benchmarks code)

Interprets the code following MEASURE and reports how long it took to execute. For example, to benchmark DO LOOP , enter:

```
: TDO  0  DO LOOP ;
1000000   MEASURE TDO
```

MIN (n1 n2 -- n1 | n2 , returns smallest)

Compare the top two values on the stack and leave the smallest.

Related words: MAX CLIPTO >

MOD (n1 n2 -- rem)

Divide n1 by n2 and leave the remainder. If you are doing MOD by a **power of two**, it is much faster to AND with N-1.

```
19 8 MOD . ( prints 3 )
19 7 AND . ( prints 3 )
```

Related words: /MOD / AND FL/MOD

MOVE (source destination count -- , moves COUNT bytes)

Moves the count amount of bytes from the source to the destination. This will work even for overlapping memory areas.

N>LINK (nfa -- lfa)

Convert a Name Field Address to a Link Field Address. See section on HForth internals and dictionary structure.

Related words: LINK> PREVNAME NAME>

N>TEXT (number -- addr count)

Converts a number to its ASCII representation.

```
1234 N>TEXT TYPE ( print 1234 )
```

Warning, this will be overwritten by the next number to be printed.

```
1234 N>TEXT .S TYPE ( this won't work )
```

NFA - stands for "name field address"

Used to be a Forth word in the FIG standard. Use >NAME instead. An NFA is the address of where a name string is stored in the dictionary. See the section on HForth Dictionary Structure for more information.

```
'N SWAP 8 DUMP
```

NAME> (nfa -- cfa , convert)

See the section on HForth Dictionary Structure for more information.

NEGATE (n -- -n)

NEGATE has the effect of multiplying by -1 but is much faster.

Related words: ABS XOR

NEW (--)

Set a flag so that FOPEN will create a NEW file on its next call. See the section on file I/O.

NFACALL, (nfa --)

Compile a call to the word whose NFA is on the stack. Calculate the best method for calling: inline macro, BSR, JSR register relative, or calculated JSR. Mostly used internally by the compiler. This is deferred so you can intercept the compiler to perform optimization, etc.

```
: FOO 23 45 [ 'n SWAP NFACALL, ] .S ;
```

NIP (a b -- b)

Deletes second item on stack.

Example: If stack looks like this: 4 6 8, and you say NIP, then the stack will look like this: 4 8.

Related Words: TUCK

NOOP (--)

Null command; does nothing.

NOT (flag -- !flag)

Invert the logical value of a flag. Equivalent to 0=. Warning: some Forths define NOT as -1 XOR. This can be a problem if your flag is 1 because 1 NOT is still true. If you want that other kind of NOT, use COMP.

```
1 NOT . ( prints 0 )
1 COMP . ( prints -2 )
```

NULL (-- 0 , constant)

NUMBER ($string -- d)

Convert a numeric string to a double number. Abort if the numeric string is invalid. Use the current value of the variable BASE as the numeric BASE. If there is a decimal point in the string, place the

number of digits to the right of the decimal point in the variable DPL. Otherwise set DPL to -1. See NUMBER?

```
" 1234" NUMBER D. ( prints: 1234 )
" 12.34" NUMBER D. DPL ?( prints: 1234 2 )
" 123x4" NUMBER D. ( abort because of the 'x' )
```

NUMBER? ($string -- d true | false)

Convert a numeric string to a double number and true. If the conversion fails, return FALSE. See NUMBER

OB.xxx - ODE related word, see ODE chapter in HMSL manual.

OCTAL (-- , set numeric BASE to 8)

ODD! (value odd-address --)

Store a 32 bit value to an odd address without generating an address error on 68000 processors. Not needed on 68030's.

ODD@ (odd-address -- value)

Fetch a 32 bit value from an odd address without generating an address error on 68000 processors. Not needed on 68030's.

OF (value n --)

See CASE.

OFF (addr --)

Set the contents of the given address to FALSE.

```
VARIABLE ALARMS
ALARMS OFF
```

ON (addr --)

Set the contents of the given address to TRUE.

```
VARIABLE ALARMS
ALARMS ON
```

OR (n1 n2 -- n1|n2)

Bitwise OR n1 and n2 together. Very useful for setting bits, combining Boolean flags or packing data. When two bits are ORed together, the result is 1 if one or the other of the input bits is 1. Only if both input bits is false will the result be zero.

```
BINARY  1010  1100 OR .  ( prints 1110 )
HEX A 4 SHIFT 5 OR . ( prints A5 )
```

Related Words: AND XOR NOT

OS.xxx - ODE related word, see ODE chapter in HMSL manual.

OUT (-- addr)

Variable containing the number of characters output since the last CR. This is automatically incremented by EMIT. Reset by CR.

```
OUT @ 60 > IF CR THEN  ( CR if past column 60 )
```

OVER (a b -- a b a)

Make a copy of the second item on the stack, as if **a** leap-frogged OVER **b**.

Related words: PICK DUP SWAP >R {

PAD (-- addr)

Address of a scratch memory area used often by the system for storing strings. PAD is defined as HERE 128 +. Thus PAD will move whenever HERE does.

PARSE (char -- addr count)

Parse from the input string until CHAR is encountered.

PICK (... i2 i1 i0 N -- ... i2 i1 i0 iN)

Pick the Nth item from the stack.

Warning! Older Forths started their numbering with 1 so that 1 PICK was the same as DUP. In F83 and most newer Forths, the numbering starts with 0. Therefore 0 PICK is the same as DUP.

```
321 654 987 369   2 PICK .  ( prints 654 )
```

PREVNAME (nfa -- previous-nfa | 0)

The names in the dictionary are in a linked list. PREVNAME will get the name of the word defined just previous to the input word. PREVNAME can be used to scan the dictionary starting with LATEST. Make sure you stop when PREVNAME returns zero. See section on the HForth dictionary.

```
: FOO ;
: SLOP ;
'N SLOP PREVNAME ID.  ( prints FOO )
```

QUERY (--)

Accept a string from the user and place it in the TIB. The number of characters will be left in the variable #TIB. >IN will be set to zero. The TIB will then be ready for a call to INTERPRET. This word is normally only used internally by the system. Here is a model Forth interpreter loop.

```
: SUBFORTH
   BEGIN CR ." **>"  ( give prompt )
      QUERY INTERPRET
   AGAIN  \ loop forever or until ABORT
;
SUBFORTH   ( enter your own loop )
23 45 + .
ABORT ( back to the old one )
```

QUIT (--)

Lowest level way to stop current processing, reset the system, clear the stacks, and return to the Forth interpreter. In a TURNKEYed system, this will cause the program to terminate. ABORT calls QUIT. QUIT is deferred.

R> (-- n) (n -r-)

Remove a number from the return stack and place it on the data stack. Warning: this must be balanced with a >R so that you don't accidentally remove the subroutine return address. See >R.

R@ (-- n) (n -r- n)

Get a copy of the value on the return stack without changing the return stack. See >R.

RDROP (n -r-)

Remove the top value from the return stack. Careful! See >R.

REL->USE (rel-addr -- addr)

Convert a relocatable address to a useable address. A relocatable address is expressed as an offset from the base of the code memory. A useable address is a 680x0 absolute address.

```
: REL->USE  codebase + ;
```

[Note: In JForth for the Amiga, a useable address is the same as a relocatable address. JForth is said to use "relative addressing" while HForth uses "absolute addressing".]

Related words: USE->REL

REPEAT (--)

Terminate a BEGIN WHILE REPEAT construct. Jump unconditionally to the code immediately following the BEGIN.

```
: VLIST  ( -- print dictionary )
   >NEWLINE LATEST
   BEGIN DUP 0>  \ are we done?
   WHILE DUP ID. CR \ print name
      PREVNAME \ get previous name
   REPEAT \ go back to DUP 0>
;
```

Related words: BEGIN UNTIL IF ?TERMINAL

RETURN (--)

Return immediately from a subroutine (word). This is like EXIT except it is fancier and can be used from within a DO LOOP without crashing. Try to avoid using this if possible. It is used mostly in response to detecting an error.

ROT (a b c -- b c a)

Rotate the third item on the stack to the top.

```
111 222 333
ROT .S ( prints 222 333 111 )
ROT .S ( prints 333 111 222 )
```

RPICK (n -- nth-return-stacxk-value)

This is like PICK but gets the Nth value from the return stack. If you think you need this, you should probably use local variables instead. See { and >R.

S->D (s -- d)

Convert a single precision word to double precision.

Related words: B->S W->S

SAVE-FORTH (--)

Save the current dictionary to the file you started from. If you want to create a new file, make a copy of the original file and start from there. All of the code currently compiled will be in the dictionary the next time you start the Forth.

If you are going to do a SAVE-FORTH, you cannot initialize HMSL. This is because HMSL is too complex to be saved in its running state. When you run HMSL4th, hit 'N' when asked if you want to initialize.

If you want to increase the size of the image, change the values CODE-SIZE and HEADERS-SIZE. They determine the amount of memory allocated for code and name/headers. To see how much you need enter MAP. Here is some sample output from MAP.

```
Compiled Code  =     201280
CodeBase       =    2895172
CODE-SIZE      =     270000
Code Left      =      68688

Compiled Names =      75566
NameBase       =    3165180
HEADERS-SIZE   =     120000
Headers Left   =      44434
Base =         10
```

In this example, we have 68,688 bytes of code defined. If we want more we could change CODE-SIZE from 270,000 to 300,000 which would give us an extra 30,000 bytes. You should also increase HEADERS-SIZE too. In general, HEADERS-SIZE should be roughly half of CODE-SIZE. Here is what we might enter to expand the dictionary.

```
300,000 CODE-SIZE !
150,000 HEADERS-SIZE !
SAVE-FORTH
BYTE
```

When you run the Forth again, you will have more space. You may get a message from MultiFinder saying there is not enough application space. If so, go back to MultiFinder, click once on the HMSL4th icon, then select "Get Info" from the File menu. Then increase the value in the memory box. If you need it, buy more memory.

Related words: TURNKEY

SCAN (addr count char -- addr' count')

Scan for CHAR in the COUNT bytes starting at ADDR. If found, return ADDR' equal to the address of the CHAR and COUNT' equal to the number of characters remaining in the string. If the CHAR cannot be found, COUNT' will be zero.

```
" Frog Peak Music" COUNT  ASCII P SCAN TYPE
\ will print:   eak Music
```

Related words: SKIP PARSE COMPARE

SHIFT (value n --)

Shift VALUE by N bits to the left. If N is negative, shift right. This has the effect of multiplying by 2**N. This is useful when packing or unpacking bit fields.

SKIP (addr count char -- addr' count')

Skip leading occurences of CHAR from the string specified by ADDR and COUNT. ADDR' is the first character not equal to CHAR. COUNT' is the number of remaining characters. This is usually used to remove leading BLANKs.

```
"           Indented Text" COUNT BL SKIP CR TYPE
\ prints: "Indented Text" without leading blanks
```

Related words: SCAN PARSE

SMUDGE (--)

Set a flag in the header of the last word defined so that it cannot be found by FIND. This is called by ':' when a word definition is first started. If the definition completes successfully, then ';' calls UNSMUDGE to make the word callable.

SPACE (-- , print a space)

Same as BL EMIT .

SPACES (n -- , print N spaces)

SPAN (-- addr)

Number of characters input by EXPECT. See EXPECT.

STATE (-- addr)

A variable that controls whether Forth is compiling or interpreting.

STATE = 0 - Interpreting. Words found are executed.

STATE = 1 - Compiling. Words found are compiled except IMMEDIATE words which are executed.

Related words: [ ] : ; LITERAL

SWAP (a b -- b a)

Reverse the order of the top two items on the stack.

```
11 22
SWAP .S ( prints 22 11 )
SWAP .S ( prints 11 22 , back to original )
```