Part IV
The Yerk Glossary

This is a glossary of Yerk words, including only those dictionary entries which are not classes. Strictly speaking, classes themselves are Yerk words, but since they are discussed amply elsewhere in this manual, they are not covered thoroughly here. Also excluded are module names, and a variety of low level words used internally by Yerk.

SOME CONVENTIONS AND DEFINITIONS

Unless states otherwise, the stack comment displays contents of the PARAMETER (data) STACK. The parameters are ordered from left to right, with rightmost being top of the stack.. parameters to the left of the hyphens reflect the local stack frame upon entry to the word, and those to the right reflect the parameters left when the word returns to its caller.

You may find these definitions useful:

memory word	16-bit number.
long memory word	32-bit number.
byte	8-bit number (high 24 bits ignored).
fcb	file control block; usually the private data of a File object, it is an extended File Manager parameter block. It is used in Yerk for file, volume, and device I/O.
top of stack	the 32-bit number in the topmost cell of the Parameter Stack.

The symbols used in the stack comments are:

symbol	description
addr	memory address.
b	8-bit byte (high 24 bits zero).
bool	boolean flag; 0=false, 1=true
chr	7-bit ASCII character (high 25 bits zero).
cfa	code field address, or compilation address of a word.

d	signed double number, that is, a 64-bit signed integer; most significant 32 bits are on top of stack.
T	true boolean flag (<>0).
F	false boolean flag (=0).
fcb	address of file control block.
len	usually, length of a string.
lfa	link field address.
n	signed single number, that is, a 32-bit signed integer.
yerk	parameter(s) used by Yerk itself during compilation, etc.
nfa	name field address.
pfa	parameter field address.
ud	unsigned double number, that is, a 64-bit unsigned integer.
un	unsigned single number, that is, a 32-bit unsigned integer.
w	16-bit number (higher 16 bits are ignored).
^base	"pointer-to-base", base address of an object's private data, same as pfa of the object.
^class	"pointer-to-class", pfa of a class.
^obj	"pointer-to-object", pfa of an object, same as ^base.
:	The colon will appear *after* the stack comment of some words. It indicates that tokens are expected in the input stream. The token(s) required by the word are listed after the colon.
fptr	A pointer to a 10-byte floating point number on the float heap.
<u>token</u>	<u>description</u>
word	name of a word currently in the dictionary.

name	name for a new entry into the dictionary.
_name	name of a toolbox routine. The underscore should *not* be typed in the actual name.
op	name of a multiple code field word. "op" is usually a value, vect, local variable or named input parameter. "->" is an example of a word which requires an "op" in the input stream.

! "store" (n addr --)

Stores 32-bit value **n** at the memory location specified by **addr.**

See also: c! w! 2! m! mw! 0! @

!csp (--)

Saves the current Parameter Stack address in the Value CSP . This word is used in conjunction with ?CSP to determine if the parameter stack has become unbalanced due to a compilation error.

!fname (addr len fcb --)

Stores a string as the file object specified by **fcb**. Clears all other fcb fields.

" "quote" Compile: (--) : textString"
Run: (-- addr len)

At compile time, compiles the text which follows in the input stream into the dictionary as a str255-format string literal. The text is delimited by a second " or a carriage-return. At run time, puts the counted string literal at buf255, a temporary buffer, and leaves its address and length on the stack. The first" must be followed by a blank; the first character of the string literal is the character after this blank.

Example: : printMsg "Finishing report." type ;

0->printMsg <return>
Finishing report. 0->

\ " used at run time:
0->" Finishing graph." type <return>
Finishing graph. 0->

This is an immediate word.
See also: ."

"sharp" (d -- quotient)

Converts a digit into an ASCII character. # is used within a <#...#> sequence to convert an unsigned double number into a text string. # divides d by the current number base to obtain the least significant digit and appends the converted digit to the string being built from high to low memory starting one byte below Pad.

Example: 6 0 <# # # #> type
\ Puts 6 on the stack as a double number and prints:
06 0->

2223243 0 <# # # # # $ 2d hold # # # #> type

\ will print:
222-3243 0->

#> "sharp-bracket" (d -- addr len)

Terminates conversion of a double number into a text string. Leaves the beginning address and length of the converted string.

See also: <#

#s "sharp-s" (d -- 0 0)

Converts an unsigned double number into a text string. #s is used within a <#...#> sequence. #s repeatedly calls # until d is converted completely.

Example: 2223243 0 <# # # # # $2d hold #s #> type
\ will print (compare with Example for #):
222-3243 0->

$ "dollar" (--) : hex#

Converts the word in the input stream into a number treating it as a hexadecimal value, and then executes LITERAL. If used within a program definition, compiles the number as a literal, to be placed on the stack at run time; otherwise puts it onto the stack immediately.

Example: $ ae7f \ Hex AE7F will be put on the stack.

: aWord ... $ 23da ... ;
\ Hex 23DA will be compiled into aWord.

This is an immediate word.

$= (addr1 len1 addr2 len2 -- result)

Compares the two strings and returns the following result:

-1 -- str1 < str2
0 -- str1 = str2

1 -- str1 > str2

See also: s= sort

' **"tick"** **Compile:** **(--) : word**
Run: **(-- pfa)**

Finds the pfa of the next word in the input stream. If used within a program definition, compiles the **pfa** of the word as a literal to be placed on the stack at run time; otherwise puts it onto the stack immediately. Don't use ' in a word or method

definition inside a module, because tick can compile a 16-bit address that is not relocateable. You may, however, use 'c.

Example: ' doGraph \Puts the pfa of doGraph on the stack

: aWord ... ' doGraph ... ;
\ Compiles pfa of doGraph as a literal

This is an immediate word.
See also: 'c

'c **"tick-c"** **Compile:** **(--) : word**
Run: **(-- cfa)**

Finds the cfa of the next word in the input stream. If used within a program definition, compiles the **cfa** as a literal to be placed on the stack at run time; otherwise puts it onto the stack immediately. 'C may be used inside modules.

Example: \ 'c is commonly used to initialize a vect to be used for vectored execution.

'c doPict -> provec \ Puts the cfa doPict into]
\ vect provec.

: aWord ... 'c doPict ... ;
\ Compiles the cfa of doPict into aWord's
\ definition as a literal.

This is an immediate word.
See also: 'cfas

'cfas **Compile:** **(n --) : word1 word2 ... wordn**
Run: **(n -- cfa1 cfa2 ... cfan)**

Finds the cfa of the next **n** word in the input stream. If used within a program definition, compiles the **cfa's** of the next **n** words as literals to be placed on the stack at run time; otherwise puts them onto the stack immediately. Note that **n** must be on the stack at compile time.

Example: \ Put the cfa's of the 4 action handler words for Window on stack.

```
4 'cfas  close enact draw content

: aWord ...
<[4]> 'cfas   close enact draw content
...  ;
\  The cfa's of the 4 words will be compiled into the definition of
\   aWord .  The 4 is flanked by <[and]> to put it on the stack instead
\  of compiling it as a literal.
```

This is an immediate word.

'code **Compile: (--) : word**

Run: (-- cf-contents)

Gets the contents of the code field (i.e., the address of the word's run-time code) of the next word in the input stream. If used within a program definition, compiles the **cf-contents** as a literal to be placed on the stack at run time; otherwise puts it onto the stack immediately.

This is an immediate word.

'type **(-- n) : type**

Takes four ASCII characters out of the input stream and packs them into one 32-bit number on the stack. The left-most character becomes the highest order byte.

Example: 'type ABCD .h <return>
41424344 0-> \ ASCII values in hex for ABCD

This is an immediate word.

(**(--)**

Denotes beginning of a comment. At least one space must follow (Stack comments are usually in parentheses in Yerk.

Example: (This is a comment in parentheses)

This is an immediate word.

((findm)) **Successful: (selhash ^class -- 1cfa T)**
Unsuccessful: (selhash ^class -- F)

Performs a search to find a method specified by selhash, a selector hash value, in the superclass chain of the class specified by ^class. If successful, leaves a true flag and the 1cfa of the method; otherwise leaves a false flag. ((findm)) is the primitive used by (findm).

(.") **(--) : textString"**

This is the run-time primitive compiled by ." when it is used in compile state. Expects a string to be compiled into the dictionary immediately following its own cfa. A standard Forth primitive and probably not otherwise useful.

(;m) **Parameter Stack: (--)**
Method Stack: (pN...p0 #parms ^obj --)

This is the run-time primitive compiled by ;m at the end of a method definition. (;m) removes the parameters and the object address left by the message just executed, then transfers execution to the next higher level by popping the return address of the calling word or method off the return stack.

See also: ;s

(@) **(addr -- n t or f)**

Checks before fetching with @ to make sure the memory belongs to the application
heap (that it is within heapBot and heapTop).

(abs) **Parameter Stack: (-- abs:^base)**
Method Stack: (^base -- ^base)

Returns the absolute address of the data of the object whose relative address is on top of the methods stack. You can use (abs) inside a method for Toolbox calls that require the address of the private data of a Yerk object. See also the Abs: method of class Object.

Example:
```
\ These methods are in class Rect.
:M INVERT:  (abs) call inverRec ;
:M PAINT:   (abs) call paintRec ;
```

See also: ^base

(bs) **(--)**

"Prints a backspace on the Macintosh screen. (bs) moves the cursor back by six pixels and prints the cursor if cursor display has been turned on with +curs. (bs) will not go left of 8 (local coordinates).

(close) **(fcb -- ioResult)**

Calls the File Manager Close to close the file specified **fcb**, the address of the file control block. If successful, **ioResult** is zero; otherwise, it is non-zero. (close) is the primitive used by the Close: method of class File.

(cr) **(--)**

Simulates a CR for a quickdraw screen. (CR) is the primitive used by CRVEC to print a carriage return on the screen.

(de) **(objaddr or cfa --)**

Decompiles the object or word whose addr or cfa is on the stack. Useful for decompiling Value pointers to objects. If a value is a pointer to an object, de' will automatically decompile its contents.

See also: de' dode

(delete) **(fcb -- ioResult)**

Calls the File Manager routine Delete to delete the file specified **fcb**, the address of the file's file control block. If successful, **ioResult** is zero; otherwise, it is non-zero. (delete) is the primitive used by the Delete: method of class File.

(emit) **(c --)**

Prints a character on the Macintosh screen using the QuickDraw routine DrawChar. (emit) is the primitive used by emit (which calls EMITVEC) to print a character on the screen.

(fdos) **(fcb trap# -- ioResult)**

Allows register-based access to Operating System routines requiring a parameter block such as File Manager routines. **Fcb** is the address of a file control block and **trap#** is the trap number of the system call. If successful, **ioResult** is zero; otherwise, it is non-zero.

```
Example:    myfcb setname "myFile"     \ Initializes myfcb, a File object.

            myfcb  $ A00C (fdos)              \ Calls the File Manager routine GetFileInfo
                                              \ (trap# = A00C); the signature, type, etc. of
                                              \ myFile are now in the data of myfcb.
```

(find) **Successful: (strAddr nfa -- pfa len T)**
Unsuccessful: (strAddr nfa -- F)

Preforms a dictionary search starting from **nfa** on the stack. (find) looks for a match on the str255 string at **strAddr**. The search ends when a 0 link field is encountered, the link field of the first entry. If successful, at exit, the top of stack contains a true flag, the second cell contains the byte length of the name field, and the third cell contains the pfa of the "found" dictionary entry. (find) is the primitive used by Find.

(findm) **(selhash ^class -- 1cfa)**

Performs a high speed search to find a method specified by **selhash**, a selector hash value, in the superclass chain of the class specified by **^class**. If

successful, leaves the **1cfa** of the method. If unsuccessful, prints message: "method not available for class CCCCC" and aborts.

(flush) **(w:eventMask w:stopMask --)**

With one cell on the stack, the high word being an **eventMask** and the low word being a **stopMask**, flush the corresponding events from the event queue.

See also: (post)

(gestalt) **Successful** **(gtype -- n 0)**
Unsuccessful **(-- err)**

(gestalt) is the primitive used by GESTALT. It either returns the gestalt code n and a 0, or an error code which will be negative. A -1 means that the mac you are using doesn't support the gestalt calls.

See also: gestalt

(install) **(addr len --)**

Sets this string as the name of the application which the nucleus will automatically load when it is double clicked. Used by the word Finalsave.

(intrp) **(--)**

Sequentially executes or compiles text from the input stream depending on state. If a word name is not found in both the context and current vocabularies, attempts conversion to a number. If conversion fails, issues an error message echoing the name followed by a "?". This word is normally executed through the vector interpret.

(key) **(-- keycode)**

(key) is the default action for the system vector key and should normally be executed by that word.

See also: key

(loop) **(--)**

This is the run-time procedure for LOOP. At compile time, LOOP compiles the cfa of (loop) into the definition being compiled.

(loop+) **(--)**

This is the run-time procedure for +LOOP. At compile time, +LOOP compiles

the cfa of (loop+) into the definition being created.

(lseek) **(fcb mode offset -- ioResult)**

Calls the File Manager routine SetFPos to position the file pointer of the file specified by **fcb** as specified by **mode** and **offset**. If successful, **ioResult** is zero; otherwise, it is non-zero. (lseek) is the primitive used by the moveTo: method of class File.

(make) **(fcb -- ioResult)**

Calls the File Manager routine Create to create the file specified by **fcb**. If successful, **ioResult** is zero; otherwise, it is non-zero. (make) is the primitive used by the New: method of class File.

(number) **(d addr -- d char-addr)**

Converts the character string at addr+1 to a double precision number, accumulating the result into **d**; the byte at addr contains the string's length but is ignored. Conversion proceeds until a non-digit is encountered. **char-addr** is the address of the first non-convertible character encountered. (number) is the primitive used by number.

(open) **(fcb mode -- ioResult)**

Calls the File Manager routine to open the file specified by **fcb** using the access mode **mode**. If successful, **ioResult** is zero; otherwise, it is non-zero. (open) is the primitive used by the Open: method of class File.

(post) **(eventCode eventMsg --)**

Post the corresponding events to the event queue.

See also: (flush)

(read) **(fcb count bufaddr -- ioResult)**

Calls the File Manager routine Read to read **count** bytes into **bufaddr** from the file specified by **fcb**. If successful, **ioResult** is zero; otherwise, it is non-zero. (read) is the primitive used by the Read: method of class File.

(save) **(--)**

A primitive used by save in the process of saving the Yerk dictionary.

(type) **(addr len --)**

Calls the QuickDraw routine to DrawText to print the string whose address and length are on the stack. (type) is the primitive used by type to print a string on the screen.

(write) **(fcb count bufaddr -- ioResult)**

Calls the File Manager routine Write to write **count** bytes into **bufaddr** from the file specified by **fcb**. If successful, **ioResult** is zero; otherwise, it is non-zero. (write) is the primitive used by the Write: method of class File.

(^elem) **Parameter Stack: (n -- addr)**
Method Stack: (-- addr:e(n))

Returns the address of the n-th element of an indexed object. (^elem) is the primitive used by ^elem, which also does range-checking.

*** "star" (n1 n2 -- n1*n2)**

Multiplies the two signed numbers leaving their signed product.

See also: m* u* 2* f* <<

---

*/ **"star-slash"** **(n1 n2 n3 -- (n1*n2)/n3)**

Multiplies the two signed numbers, then divides the double number product by the signed number **n3**, leaving a signed quotient. Division by zero and overflow are not checked for.

Example: 1220 34 100*/ \ Compute 34% of 1220
. 414 0->

---

*/mod **(n1 n2 n3 -- rem (n1*n2)/n3)**

Same as */ except that the second cell contains the remainder of the division.

+ **(n1 n2 -- n1+n2)**

Adds two signed numbers, leaving their signed sum.

See also: 1+ 2+ 4+ 8+ f+

+! **"plus-store"** **(n addr --)**

Adds **n** to the memory location at **addr**. In other words you can use +! to increment memory location by **n**.

++1 **(incVal idx --)**

A primitive used in its class object for incrementing a bArray value. It adds **incVal** to the cell of the array specified by **idx**.

++2 **(incVal idx --)**

A primitive used in its class object for incrementing a wArray value. It adds **incVal** to the cell of the array specified by **idx**.

++4 **(incVal idx --)**

A primitive used in its class object for incrementing an Array value. It adds **incVal** to the cell of the array specified by **idx**.

++> **"increment" (n --) : op**

This prefix operator adds the number on the stack to the operand following it. The operand may be a value, named input parameter, or local variable.

```
Example:    30 value xx       \ Define and initialize value xx.
            17 ++> xx         \ Add 17 to xx.
            xx . 47 0->       \ Print current contents of xx.

            \ The following (dummy) word shows ++> operating on a named
            \ input parameter and a local variable.
            : aWord { x \ counter -- result }
            BEGIN
                    x 0> WHILE
                    -3 ++> x            \ Add -3 to x, named input parm.
                    ...         \ more code
                    1 ++> counter     \Increment counter, local variable
            REPEAT  ;
```

This is an immediate word.
See also: ->

+- **(n1 n2 -- n1)**

Sets the sign of **n1** to that of **n2**.

```
Example:    -340 20 +- .      \ Change sign of -340 to sign of 20
            340 0->
```

+base **(rel-addr -- base-addr)**

Adds the contents of the 68000's register A3, Yerk's base pointer, to the address on the stack converting it to an absolute address. All addresses within Yerk are relative to register A3. You can use +base to prepare addresses for Toolbox calls

---

+curs **(--)**

Turns on display of the cursor on the screen.

+docs **(--)**

Turns on file marking and enables display of the source definition of a word compiled after filemarking was turned on. The source file must be in the search path. +docs is best used during development, but should be turned off (-docs) to preserve space when you compile your final code.

See also: -docs see fm rl mforget /// srcName findFMark

+echo (--)

Turns on echoing to the screen of source files being loaded from disk. You may want to use +echo when loading a newly created source file to help spot any code snagging compilation. Echo is toggled by the keyboard short cut: Command-O

See also: -echo decho

+file (--)

Turns on echoing to a file. The default filename is 'logfile' and will be located in the Yerk Folder. Holding the option key down when executing this word will bring up a standard get box for you to provide a different file name. You may want to use +file when debugging or you just want to save things to a file.

See also: -file

+loop

Compile: **(yerk --)**
Run: **(i --)**

Used to end a DO-LOOP structure. At run time, the top of the stack contains the increment to be added to the loop index. If the increment is positive, a branch back to the loop body is executed if Index is greater than or equal to the limit. If the increment number is negative, a branch back to the loop body is executed if the Index is greater than the limit. No parameters are left at exit. +loop is an immediate, compile-time-only word. Use +loop instead of loop when you need a loop increment other than 1 or -1.

```
Example:   10 0          \ 10 = limit; 0 = loop index
           DO aWord      \ some code
           anotherWord   \ ..
           2 +LOOP       \ increment loop index by 2
```

This is an immediate word.

+print (--)

Enables printer output and turns on echoing to the printer. (Echoing to printer means printing on the printer whatever text appears on the screen.) +print vectors system vector pemitvec to permit, pcrvec to pcr, ptypevec to ptype,

and echovec to echo. (Echo executes pemitvec and emitvec.) Accordingly, after you execute +print, any text emitted or typed on the screen is also printed on the printer. Also, you can send characters directly to the printer with pemitvec and ptypevec from within your program. Print echo is toggled by the keyboard short cut: Command-P.

+range (--)

Enables runtime range checking of indexed objects.

+rdepth (--)

Turns on return stack depth checking used for method execution. You might want to turn this off (using -rdepth) during completion routine executions, and then use +rdepth to turn it back on for normal use.

See also: -rdepth ?rdepth

, **"comma"** **(n --)**

Compiles **n** into the next available dictionary location and advances the dictionary pointer.

See also: c, w, s, 0, f,

,exec **(cfa --)**

State-smart execute. If used within a program definition, compiles the **cfa** as a literal to be executed at runtime; otherwise executes it immediately. This is not an immediate word. It is useful in building compiler words which conditionally compile other words.

See also: execute

- **(n1 n2 -- n1-n2)**

Subtracts two numbers, leaving their signed difference.

See also: 1- 2- 4- f-

-1 **(-- -1)**

Puts a -1 onto the stack. -1 is provided to save space and compilation time.

-> **"into"** **(n --) : op**

This prefix operator stores the number on the stack into the operand following it. The operand may be a value, vect, sysvect, named input parameter, or local variable.

```
Example:   0 value curRoom              \ Define value curRoom.
           room  ->  curRoom            \ put current room in
                                        \   dictionary into curRoom.
           0 vect provec                \ Define vect provec .
           'c doGraph -> provec         \ Put cfa of doGraph into provec.
           provec                       \ doGraph is executed.
```

This is an immediate word.
See also: ++>

-base **(abs-addr -- rel-addr)**

Subtracts the contents of the 68000's register A3, Yerk's base pointer, from the absolute address on the stack converting it to its relative address. All addresses within within Yerk are relative to register A3. You can use -base to convert an absolute address returned by a Toolbox call into a Yerk-compatible relative address.

-curs (--)

Turns off display of the cursor on the screen.

-docs (--)

Turns off file marking and disables display of the source definition of a word compiled after filemarking was turned on. Saves space in the dictionary.

See also: +docs see fm rl mforget /// srcName findFmark

-dup **If zero: (n1 -- n1)**
If non-zero: (n1 -- n1 n1)

Duplicates the number on the stack if it is non-zero.

-echo (--)

Turns off echoing to screen of source files being loaded from disk. Echo is toggled by the keyboard short cut: Command-O.

See also: +echo decho

-file (--)

Turns off echoing to a file.

See also: +file

-print (--)

Turns off the printer output and echoing to the printer. You can use +print to turn it all on again. Print echo is toggled by the keyboard short cut: Command-P

-range (--)

Disables runtime range checking. Can be used to speed up already debugged code. This word is the inverse of +range.

-rdepth (--)

Turns off return stack depth checking used for method execution. You might want to use this word during completion routine executions, and then use +rdepth to turn it back on for normal use.

See also: +rdepth ?rdepth

-trailing **(addr len -- addr len')**

Suppresses trailing blanks before outputting a string with type. It examines the character string **addr** and reduces the character count so that trailing blanks are not output by a subsequent type.

See also: padBL

. **"dot"** **(n --)**

Prints n using the current number base as the conversion radix.

Example: 12 14 + . <return> \ Print number on top of stack.
26 0->

See also: d. u. e. e.r <#

." **"dot-quote"** **(--) : textString**

Prints the **textString**. If used within a program definition, compiles the address of its run-tine primitive, (."), the length of the string, and the string itself into the dictionary to be printed at run time; otherwise prints it immediately.

Example: \ ."used at run time:
."Bit-mining, the mother of the Information Age." <return>
Bit-mining, the mother of the Information Age. 0->

\ ."used at compilation time:
: aWord ... ."`I am a string." ... ;

\ The string will be printed when aWord is executed.

This is an immediate word.
See also: type

..end **"dot-dot-end" (--)**

An analog of ;S used at the end of a codeField handler in an mCFA word. See Part II.4.

This is an immediate word.
See also: do.. prefix codefields build

.classes **(--)**

Prints the name, data length and element width for each class defined in the current dictionary. Classes which are not indexed will show a width of zero.

See also: @dlen @width

.classname (^obj --)

Prints the name of this object's class. Primarily used by Yerk internals.

.cur (--)

Draw the cursor at the current position.

.d (n --)

Prints **n** in decimal, regardless of the current number base.

See also: decimal

.h (n --)

Prints **n** in hexadecimal, regardless of the current number base.

See also: hex

.mods (--)

Lists the names of existing modules and their load status. If the module is loaded into the heap it's load address will be printed; if not, a zero. Locked modules are indicated.

.name (nfa --)

Prints the name of the word whose name field address is specified, substituting three question marks if the name length is >16. This word can be used in place of id. when it is not 100% certain whether or not you are pointing to the header of a real word.

See also: id.

.path (--)

Prints the pathList for file i/o.

.pause (--)

Prints the pause message.

See also: ?pause

.r **(n w --)**

Prints **n** using the current number base, right-justified in a field **w** characters wide. The entire number is printed even if it exceeds field width.

Example: 17 5 .r <return>
bbb17 0-> \ The b's represent blanks.

.room **(--)**

Prints the status of memory usage. Shows the total memory usage of the dictionary, usage by non-purgeable blocks and largest purgeable block in the heap.

.s **(--)**

Prints the contents of the three stacks in decimal and hexadecimal.

Example: Parameter Stack: (--Empty Stack--)
Return Stack:
16286 $ 3F9E
16744 $ 4168
Methods Stack: (--Empty Stack--)

.sys **(--)**

Loads the sysenvirons module to prints information about the current mac system. You may also get the sysenvirons record pointer to use this info within your program.

See also: .sys hascolor hasFPU appletalkOn getEnv

.val **(n w --)**

Prints **n** using the current number base, right-justified in a field **w** characters wide, and plus two spaces. The entire number is printed even if it exceeds field

width. .val is the same as .r, except that two spaces are printed after the string.

Example: 175 .val <return>
bbb17bb 0->

.W **(--) : word**

Finds the next word in the input stream and dumps the 100 bytes in the dictionary after the words entry.

Example:
.w dup <return>
Dump from address: 23F6
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF

23DO 23 F0 68 00 4E F3 78 00 83 44 55 50 00 00 23 CE ..h.N.x..DUP..#.

/ **"slash" (n1 n2 -- n1/n2)**

Divides two signed numbers leaving their signed quotient.

See also: m/ u/ 2/ f/ >> mod/mod

// **"load" (--) : filename**

Loads and compiles source code from a text file on disk. The text is echoed to the screen if echoing to the screen has been enabled to +echo. Pauses if space bar is pressed; resumes if space bar is pressed again. Restores keyboard input on any error or end-of-file. The source file being loaded may itself initiate loading a file as in "// myFile". Source file loads may be nested up to six levels deep. The File objects (fcb's) associated with each file being loaded are in the indexed object loadFile. The names of the files in loadFile are printed when abort is executed.

Example: // dungeons \ Loads source file "dungeons".

\ The following is an Example: of a "load" file that automatically loads several
\ source files; compiles a module, and saves the dictionary when it is loaded.

```
1 // Mod
2 // Imports
3 Module Tool
4 // Event
5 // QD
7 // Window
8 // forget.load
9 Save yerk.com
10 Become yerk
```

See also: sony external profile +echo<"

/// **(--) : filename**

Import word that is enabled after +docs is set. If documentation is on, this word

will forget all words up to the filename and then will reload all the files in the load sequence from filename to here (when you executed ///). Note that the file list is forgotten when the load is completed or is aborted (by hitting a key).

See also: +docs -docs see rl fm mforget srcName findFmark

/mod **"slash-mod" (n1 n2 -- rem n1/n2)**

Divides two signed numbers leaving the remainder and signed quotient.

Example: 34 11 /mod . . <return>

3 1 0-> \ 11 goes into 34 3 times with a remainder of 1.

---

0 "zero" (-- 0)

Puts a zero on the stack. It is provided to save space and compilation time.

0! (addr --)

Stores a 0 at the memory location specified by **addr**.

0, (--)

Compiles at a 0, an empty cell, into the dictionary.

0.0 (-- fptr)

Puts the value "zero" into the floating heap and returns it's pointer. It is provided to save space and compilation time.

0= (n -- bool)

Examines the signed number **n** and leaves a true flag if **n** is equal to zero; otherwise, leaves a false flag. 0= is identical to NOT.

Example: -9 0= . <return> \ Is -9 equal to zero;
0 0-> \ zero = false.

See also: f0=

0< (n -- bool)

Examines the signed number **n** and leaves a true flag if **n** is less than zero; otherwise, leaves a false flag.

Example: 57 > . <return> \ Is 57 greater than zero;
1 0-> \ non-zero = true.

See also: f0>

0branch **(bool --)**

Compiled into definitions by the Yerk words ELSE, AGAIN, LOOP, +LOOP and REPEAT. Changes the value of IP to cause the branch to occur when bool is false (zero). For a more complete explanation of the Forth elements of Yerk, see The Forth Encyclopedia, cited in the bibliography at the end of the manual.

---

1 **(-- 1)**

Puts a 1 on the stack. It is provided to save space and compilation time.

1+ **(n -- n+1)**

Adds one to the number on the stack.

1- **(n -- n-1)**

Subtracts one from the number on the stack.

1.0 **(-- fptr)**

Puts the value "one" into the floating heap and returns it's pointer. It is provided to save space and compilation time.

1/x **(fptr -- fptr)**

Computes the reciprocal of the floating point number pointed to by **fptr**.

---

2 **(-- 2)**

Puts a 2 on the stack. It is provided to save space and compilation time.

2! **(d addr --)**

Stores a double number into the two long memory words specified by **addr**; that is, this is a 64-bit store.

2* **(n -- n*2)**

Multiplies **n** by 2. This is a fast signed integer multiply by 2, implemented as an arithmetic 32-bit left shift.

See also: <<

2** **(exp -- 2^exp)**

Raises 2 to the **exp** power.

2+ **(n -- n+2)**

Increments **n** by 2.

2- **(n -- n-2)**

Decrements **n** by 2.

2/ **(n-- n/2)**

Divides **n** by 2. This is a fast signed integer divide by two, implemented as an arithmetic 32-bit right shift.

See also: >>

2@ **"two-fetch"** **(addr -- d)**

Fetches the 64-bit value at **addr** and puts it on the stack as a double number.

2drop **(d --)**
(n1 n2 --)

Drops a double number or two 32-bits numbers on the stack.

2dup **(d -- d d)**
(n1 n2 -- n1 n2 n1 n2)

Duplicates the double number or two 32-bits numbers on the stack.

2over **(d1 d2 -- d1 d2 d1)**

Duplicates the double number below the double number on the stack.

2swap **(d1 d2 -- d2 d1)**

Exchanges the top two double numbers.

3+ **(n -- n+3)**

Increments **n** by 3.

---

4 **(-- 4)**

Puts a four on the stack. It is provided to save space and compilation time.

4* **(n -- 4*n)**

Multiplies **n** by 4. This is a fast signed integer multiply by 4, implemented as two 32-bit arithmetic left shifts.

See also: <<

4+ **(n -- n+4)**

Increments **n** by 4.

4- **(n -- n-4)**

Decrements **n** by 4.

8* **(n -- 8*n)**

Multiplies n by 8. This is a fast signed integer multiply by 8, implemented as three 32-bit arithmetic left shifts.

See also: <<

8+ **(n -- n+8)**

Increments **n** by 8.

8- **(n -- n-8)**

Decrements **n** by 8.

: **"colon"** **Compile:** **(--) : name**
Run: **(--)**

Creates a dictionary entry for the next word in the input stream, sets compile mode, and stores the current value of the parameter stack pointer in CSP for subsequent error-checking. Once in compile mode, Yerk compiles the cfa's of subsequent words from the input stream which are not immediate words into the dictionary entry of the word just created. Immediate words are executed as encountered. If a word cannot be found, Yerk's interpreter attempts to convert it into a literal using the current number base and then to compile it. If conversion to literal fails, prints message: "not found" and aborts.

Example: : aWord word1 word2 word3 ... ;

This is an immediate word.

See also: create state exit immediate

:CLASS **(--) : name**

Begins compilation of a class. The next word in the input stream is the name of the class to be compiled. You end class compilation with ;class.

Example: :CLASS Two PanelWind
...
;CLASS

This is an immediate word.
See also: <Super <Indexed cstate

:CODE (--) : name

Begins compilation of a 68000 Code definition using the Yerk assembler. You end :code compilation with ;code. This can only be used within a file and not from the interpreter. Read the Assembler section in the manual.

This is an immediate word.
see also: :mcode

:f (--) : name

Begins compilation of a word which was previously forward-referenced. :f creates a headerless entry for the word and then patches the previous entry to point to the newly compiled definition. Forward-referencing is useful when a word is to be used before it can be defined. You end the compilation with ;f.

```
Example:    forward setControl          \ forward reference setControl
                  ...                   \ more code
            :f setControl ... ... ;f    \ compile setControl
```

See also: forward

:M (--) : name

Begins compilation of a method within a class definition. The next word in the input stream is a selector (always ending in ':'). You end method compilation with ;m.

```
Example:    :CLASS TwoPanelWind <Super window
                  ...                   \ more code
            :M SETPANES:                        \ Compile definition of method
                  ...                   \ code for method
            ;M
                  ...                   \ more methods
```

This is an immediate word.

:mcode **(--) : name**

Begins compilation of a 68000 Code method definition within a class using the Yerk assembler. You end :mcode compilation with ;mcode. This can only be used within a file and not from the interpreter. Read the Assembler section in the manual.

This is an immediate word.
See also: :code

:module (--) : name

Begins compilation of a relocatable module. Instantiation of a class defined in a module can occur only within that module. A reference in a module or in the dictionary to an object defined in another module must be late-bound (enclosed within brackets). Don't use ' inside a word or method definition in a module, because the address returns may not be valid when the module is later relocated. You end compilation of a module with ;module. The name following :Module must be predefined in the resident dictionary using FROM.

```
Example:    :module   myUtils          \ Begin compilation of myUtils
                ...
                : utilWord1 ... ;
                ...
                size: [anIdxObj]      \ anIdxObj is defined in another module, so
                                      \ message to it is late-bound.
                ...
            ;module
```

See also: from

:PROC (--) : name

Begins compilation of a word that to the Toolbox behaves like a Pascal procedure or function. You can use a :proc word when a Toolbox routine requires a procedural argument.

```
Example:    \ This proc word is from file ctlWind
            \ Procedure to be executed when a control is being tracked.
            ( ctlHndl int:part -- )
            :PROC  ctlProc word0 swap ctlExec ;PROC
```

See also: initproc initproc +rdepth -rdepth ?rdepth

; "semicolon" (--)

Terminates a colon definition and compiles its own runtime procedure ;s. Prints message: "definition not finished" and aborts if the parameter stack

pointer is different from when compilation was started.

This is an immediate word.

;f **(--)**

Ends compilation of a previously forward-referenced word.

This is an immediate word.

;M **(--)**

Ends compilation of a method within a class definition. ;m compiles (;m), its own run-time procedure. At run time, (;m) removes the data left on the methods stack by the object that received the message.

This is an immediate word.

;module (--)

Ends compilation of a module.

;PROC (--)

Ends compilation of a proc word, which appears as a Pascal procedure or function to the Toolbox.

This is an immediate word.

;S

Parameter Stack: **(--)**
Return Stack: **(addr --)**

This is the primitive that ; compiles at the end of a colon definition. ;s transfers execution to the next higher level by popping the return address of the calling word or method off the return stack.

< **(n1 n2 -- bool)**

Compares two signed numbers. If **n1** is less than **n2**, leaves a true flag; otherwise, leaves a false flag.

See also: d< 0< u< f<

<" **(--) : filename**

This is the primitive loader. Use this when rebuilding Yerk from below "Files" which contains the definition for //.

<# **(d --)**

Prepares for conversion of a double number into a text string for printing. <# is paired with #>.

See also: # #s hold sign

<< "shift-left" (n exp -- n*(2^exp))

Multiplies **n** by 2 raised to the signed **exp** power. << is implemented as exponential arithmetic left-shifts on **n**.

Example: 3 2 << . <return> \Multiplies 3 by 2^2, i.e., left shift 3 (0011) by 2 bits
12 0-> \Result is 12 (1100).

<= **(n1 n2 -- bool)**

Compares two signed numbers. If **n1** is less than or equal to **n2**, leaves a true flag; otherwise, leaves a false flag.

<> **(n1 n2 -- bool)**

Compares two signed numbers. If **n1** is not equal to **n2**, leaves a true flag; otherwise, leaves a false flag.

See also: f<>

<builds **(--)**

A compiler word that defines the compile time behaviour of a new word. Usually used in conjunction with DOES> which defines the runtime behavior of the word. Probably only usable by FORTH gurus.

Example:
```
\ define a compiler for a one dimensional byte array
: 1barray  <builds allot does> + ;
\ create a 1-d byte array of 10 elements called bob
10 1barray bob   \ usage:  3 bob c@ or 22 3 bob c!
```

See also: does>

<indexed **(width --)**

Sets the **width** of the bytes of the elements of an indexed class.

Example:
```
\ Array objects will have indexed elements of four bytes.
:CLASS Array <Super Object  4<indexed
        ...
;CLASS
```

<super **(--) : class**

Stets the superclass of the object being defined to the name following <super. Objects of the class being defined will inherit all the instance variables and methods of the superclass, except for the methods of the superclass which may be redefined in the current class.

Example:
```
\ Oval objects inherit all the ivars and methods of Rect.
:CLASS Oval <super Rect
        ...
```

This is an immediate word.

<var **([#elems] ^class --)**

Compiles an instance variable dictionary entry. A primitive used in class compilation. Requires the **#elems** field only if the class pointed to is indexed.

<[**(--)**

Ends compile state to resume run state. Sets state = 0. Used in conjunction with]>.

Example:
```
: aWord ...
<[3]> 'cfas  doGraph doPict doText ... ;
\ <[ is used to put the interpreter in run state temporarily because
\ 'cfas needs the count of cfa's to compile on the stack; ]> resumes
\ compilation.
```

This is an immediate word.

= **(n1 n2 -- bool)**

Compares two signed numbers. If **n1** is equal to **n2**, leaves a true flag; otherwise, leaves a false flag.

See also: d= 0= s= f= <>

> **(n1 n2 -- bool)**

Compares two signed numbers. If **n1** is greater than **n2**, leaves a true flag; otherwise, leaves a false flag.

See also: d> 0> s> f>

>= **(n1 n2 -- bool)**

Compares two signed numbers. If **n1** is greater than or equal to **n2**, leaves a true flag; otherwise, leaves a false flag.

>> "shift-right" (n exp -- n/(2^exp))

Divides **n** by 2 raised to the **exp** power. >> is implemented as exponential arithmetic right-shifts on **n**.

Example: 12 2 >>. <return> \ Divide 12 by 2^2, i.e., right-shifts 12 (1100) by
\ 2 bits.
3 0-> \ Result is 3 (0011).

>body (cfa -- pfa)

Derives the parameter field address, **pfa**, from the code field address, **cfa**, of the word.

>float **(n -- fptr)**

Takes a 32-bit number off the stack and converts it to a floating point number on the float heap, and returns a pointer to the floating point number on the stack.

>link **(cfa -- lfa)**

Derives the link address, **lfa**, from the code field address, **cfa**, of the word.

See also: lfa

>name **(cfa -- nfa)**

Derives the name field address, **nfa**, from the code field address, **cfa**, of the word.

See also: nfa

>origin **(x y --)**

Calls the QuickDraw routine SetOrigin to set the origin of the current grafPort.

>ptr **(handler -- ptr)**

Dereferences a Toolbox **handle** returning a relative pointer. A handler is a pointer to a pointer. By convention, in Yerk a handler's address is absolute and a pointer's address is relative to A3. You can use >ptr when a Toolbox call returns a handle from which you need to derive an address that Yerk can work with.

>r "to-r" **(n --)**

Removes (pops) the number on the parameter stack and places (pushes) it onto

the Parameter stack. Use of >r must be balanced with >r in the same definition. You should use the return stack with great care. In general, use named input parameters and local variables instead; they are easier to use and a lot friendlier.

>str255 **(addr1 len addr2 -- abs:addr2)**

Converts an addr-len format string into a str255-format string, putting the string at **addr2**, a buffer, and leaves the absolute address of the string.

>uc **(abs:addr len --)**

Converts a string into a somewhat mixed format to uppercase. Does the actual conversion in place. See use in class string for an example of how to use this primitive word in a higher-level word.

? **(addr --)**

Prints the number at **addr** as a text string using the current using base by fetching the contents of addr and executing . ("dot").

?cfa **(addr -- addr bool)**

Tests the address on the stack to see if it a cfa of a yerk word.

?class **(--)**

Checks to see if the interpreter is in class compilation mode, that is if cstate is set to true. If cstate is set to false issues error message #115 and executes an abort.

?comp **(--)**

Prints message: "compilation only" if not compiling, that is, if state is not zero.

?csp **(--)**

Prints message: "definition not finished" if the contents of the parameter stack pointer differs from that stored in value csp when : began compilation.

?dp **(--)**

Checks for condition that the dictionary has grown into the heap. If it has, aborts with the message: out of room". Primarily used by Yerk internals.

?error **(bool --) : resID**

Prints the string with resource ID of **resID** if the boolean on the stack is true. The string must be in a currently open resource file or in Yerk.rsrc.

Example: notPrime ?error 532 \ Prints string with ID 532
\ if notPrime is true.

This is an immediate word.

?event **(eventMask -- bool)**

Calls the Toolbox Event Manager routine EventAvail to test for the event(s) specified by **eventMask**. If an event is available, leaves a true flag; otherwise, leaves a false flag.

Example: mouseDnMask ?event \ Test if a mouse-down event is
\ available in the event queue.

See also: @event-msg get-event

?exec (--)

Issues an error message if not in interpret state.

?idx (--)

This vector executes the word responsible for range-checking in indexed object methods. If range-checking has been turned on with +range, ?idx is vectored to ?ixRange; if range-checking has been turned off with -range, the vect ?idx is vectored to null, Yerk's "do-nothing" word. You can turn off range-checking within the methods of indexed objects if you need the extra speed this provides.

?IsClass **(pfa -- pfa bool)**

Returns the **pfa** and a true flag if the pfa belongs to a class, that is, the pfa is the pointer to a class (^class); otherwise, returns the **pfa** and a false flag.

Example: ' Window ?IsClass . <return>
1 0-> \ Returns 1 as true flag

?IsHandle **(addr -- bool)**

Returns a true if the addr on the stack is a valid handle. False if not. THIS IS NOT 100% reliable, but it's in the dictionary for testing statistically how good it is. Use it with care. The method VALID: of class HANDLE uses it.

?IsObj **(pfa -- pfa bool)**

Returns the **pfa** and a true flag if the pfa belongs to an object, that is, the pfa is the pointer to an object (^obj, or equivalently, ^base); otherwise, returns the **pfa** and a false flag.

Example: ' fWind ?IsObj . <return>
1 0-> \ Returns 1 as a true flag.

?IsVect **(pfa -- pfa bool)**

Returns the **pfa** and a true flag if the pfa belongs to an object vector (a value or vect); otherwise, returns the **pfa** and a false flag.

Example: ' windVec ?IsVect . <return>
1 0-> \ Returns 1 as a true flag.

?IxObj **Parameter Stack: (--)**

Method Stack: (^base --^base)

Prints message: "not indexed" if an object is not indexed. ^**base**, the pointer to the object's data, is on the methods stack.

?IxRange

Parameter Stack: (index -- index)
Method Stack: (^base --^base)

Prints message: "not indexed" if an object is not indexed. ^**base**, the pointer to the object's data, is on the methods stack. ?IxRange prints message "index out of bounds" if the index on the parameter stack is not in the range (0 <= **index** < limit: idxObj).

See also: ?idx

?key

Successful: (-- keysword modsword T)
Unsuccessful: (-- F)

Executes the Next: method of object fEvent to see if a key event is pending in the event queue. If a key event is available, ?key returns **keysword**, **modsword** and a true flag. If no key is available, ?key returns a false flag only. This is defined in the optional source Interval.See Part III.4, Events.

See also: key-evt key

?lead

(-- #pixels)

Returns the pixel increment used by the system when skipping to the start of a new text line. The calculation is 120% of the point size of the current font. It is possible to change this factor:

```
Example:    130 ' ?lead 24 + !          \ Set spacing factor to 130%
```

This figure should always be greater than 100% to provide for lower case descenders and spacing between lines. Primarily used by CR.

See also: tSize

?lines

(-- #lines)

Returns the number of character lines which will fit into the font window at the current font size. The result is based, in part, on ?lead.

See also: tSize ?lead

?mlock **(cfa -- bool)**

Returns true if the module is locked.

Example: 'c grepmod ?mlock

See also: mlock munlock

?mod (cfa -- bool)

This word checks to see if the cfa is a module cfa.

?new (handle --)

This word checks to see if the **handle** has been initialized, that is to say whether or not it is non-zero. If the **handle** equals zero, it issues error message # 153 and executes an abort.

?num (addr -- d t or f)

Converts a character string at **addr**+1 to a double precision number; the byte at **addr** contains the string's length but is ignored. Conversion proceeds until a blank is encountered - the string of digits must end with a blank - any character which is not a digit, decimal point or minus sign will fail to convert. The position of the last decimal point encountered (if any) is left in DPL. If conversion fails, a false is left on the stack. This is the primitive used by number.

See also: digit dpl (number) @val number

?pairs (n1 n2 --)

Prints message: "unpaired conditionals" if **n1** does not equal **n2**. Message indicates that compiled conditionals do not match, such as an IF without a THEN.

?pause (--)

Checks whether the user wants to stop or pause or stop. Prints the message: "Paused -<Space Bar> to continue>>>". If the next key pressed is NOT a space bar of CR, executes an abort.

?range

Parameter Stack: (index -- index)
Method Stack: (^base --^base)

Prints class error message "index out of bounds" if the index is not in the range (0 <= **index** < limit: idxObj). **^base**, pointer to the object's data, is on the methods stack.

?rdepth

(--)

Prints message: "Return Stack Overflow" and executes an abort if the return stack is too close to its maximum depth.

Example: : test ... ?rdepth ... ;
\ Test determines if the return stack is nested too deeply as the result
\ of a recursive routine, for example.

See also: +rdepth -rdepth

?stack **(--)**

Prints message: "empty stack" and executes an abort if underflow has occurred on the parameter stack. Underflow may occur if a word or method expects more parameters than are provided.

?terminal **(-- bool)**

Preforms a 40 ?Event. This word is used in class Timer, class Mouse, and in the utility word ?pause to test for a keyboard event.

See also: ?event ?pause

@ **"fetch"** **(addr -- n)**

Fetches the number at **addr** and puts it on the stack.

Example:
```
39 variable temp      \ Define variable temp
temp @ . <return>     \ Fetch and print the contents of temp.
39 0->
```

See also: c@ w@ 2@ m@ mw@ ? !

@dlen **(pfa -- len)**

Returns the length of the object's private data.

See also: @width

@event-msg **(-- msg)**

Fetches the message of the latest event from the event record in object. fEvent, the Event object in the nucleus.

See also: ?event get-event

@pfa **(-- pfa) : word**

Executes find on the next word in the input stream. If successful, leaves the pfa of the word; otherwise, prints message: "not found" and aborts.

```
Example:    ( -- ^class)        \Fetches the pfa (^class) of a class
            : getClassPtr  @pfa ?IsClass not
```

```
abort" Arguments not in class.";
```

@val **(-- val) : val**

Takes the next word in the input stream, converts it to a number and leaves the value on the stack. An example of a word which uses @val is $.

See also: number

@width **(^class -- width)**

Returns the width of the elements of indexed class.

Example: ' Array @width

See also: @dlen

@word **(-- addr) : word**

Takes the next word in the input stream, stores it as a text string converted to upper-case at here, and leaves its address.

Example:
```
( -- )
: test@word
        " The word was "
        @word count   \   leaves addr len for type.
        type ;

test@word insanely <return>
The word was INSANELY 0->
```

See also: word" @val

@xy **(-- x y)**

Leaves the current position of the cursor in global coordinates.

See also: gotoxy

---

abort (--)

Clears the parameter and methods stacks, stops interpreting the current input line, enters run state, and sets the number base to decimal. Then, executes the word in abortvec. Returns control to the keyboard (unless abortvec prevents this) printing the current prompt, usually "0->".

See also: quit

---

abort" **(bool --) : textString"**

Prints the text string and executes an abort, if the boolean is true. Useful in debugging and error reporting. At least one space must follow the first quote. Can be used in colon definition only.

```
Example:    : printReport ...
            printFlag not abort" Printer not on"
                ... ;           \ printFlag is true if the printer is on.
```

This is an immediate word.

---

abortvec (--)

This is the system execution for abort. If abort is executed and abortvec is vectored to CLEAN2, the contents of the file stack, loadFile, are printed and control is returned to the keyboard, providing access to the Yerk interpreter. If abort is executed and abort is vectored to a user-defined error handler, it is executed. AbortVec may be used with QuitVec to seal off the user of an application from the interpreter.

---

abs (n -- un)

Leaves the absolute value of **n**.

See also: dabs fabs

---

actv-evt (-- 0)

Handles activate/deactivate events for object fEvent. It sends its Window object a message to execute its enable: method: if the window is being deactivated, sends its Window object a message to execute its disable: method. This word can be redefined through the use of :f or by substitution of vector 8 in fEvent.

See also: disk-evt key-evt mouse-evt null-evt upd-evt

---

actw (-- ^obj)

Actw is a Value that points to the active window. You may treat actw as the

active window object.

---

addm **Parameter Stack: (i --)**
Method Stack: (n -- n+i)

Adds the number on the parameter stack to the number on the methods stack.

See also: copym exgm dropm dupm popm pushm

---

addTask **(cfa --)**

Adds the **cfa** to taskList. TaskList contains the cfas of up to four words that object fEvent executes as background tasks whenever it receives a null event. (You must load optional file "Tasks" to use this facility.)

Example: 'c dispClock addtask
\ Adds displaying clock as one of the background tasks.

See also: killtask dotasks tasklist

---

again

Compile: **(yerk --)**
Run: **(--)**

At run time, transfers execution to the corresponding BEGIN. This is an unconditional loop. You can exit only by doing r> drop one level below, or by transfer of control to another portion of the application because of a become.

Example:
```
( -- ) \ Listen to events, dropping the key events.
: listen
BEGIN key      \ key actually listens to all events
               \ returning only after key-down events.
drop
AGAIN ;
```

This is an immediate word.

---

alert!

(n1 n2 -- n)

Import word from AlertMod that will bring up an alert from your resource file with ID of n1 and type (0,1,2) n2. You may up to four buttons in the alert, and the item# (0-4) is returned after clicking the alert closed.

---

align

(addr -- even-addr)

Increments **addr** by one if it is odd, leaving the next address. You can use align to make sure the 68000 will not try to access memory beginning at an odd address, which can cause a fatal error #2.

---

allot

(n --)

Adds **n** to the dictionary pointer dp. You can use allot to reserve **n** bytes of dictionary space.

```
Example:   0 variable printBuff      \Creates header for a 256-byte
                                     \ buffer and first 4 bytes.
           252 allot                 \ Allots next 252 bytes.

           printBuff <return>
           1->                       \ Leaves address of buffer.
```

See also: reserve bytes

---

and (n1 n2 -- n3)

Leaves the bitwise AND of **n1** and **n2** as **n3**. And works as a logical and if you want to use **n1**, **n2**, **n3** as booleans (non-zero=true; zero=false).

```
Example:  10 6 and . <return>      \   1010 AND 0110
          2 0->                    \ = 0010

          10 6 and . <return>      \ true AND true
          2 0->                    \ non-zero=true
```

See also: or xor Land

---

annuity (fptr:r fptr:n -- fptr)

Computes annuity(r,n)=$[1-(1+\mathbf{r})^{-\mathbf{n}}]/\mathbf{r}$, where **r** is the interest rate and **n** is the number of periods. Annuity is more accurate than the straightforward computation of the expression above using basic arithmetic operations and exponentiation. The annuity function is directly applicable to the computation of present and future values of ordinary annuities.

See also: compound

---

antilog (fptr -- fptr)

Computes the antilog (base 10) of fptr.

See also: log

---

appleEvt (-- 0)

AppleEvt is a vector that handles Apple Events. You may set this in your application to handle the events as you see fit. The default is drop.

See also: hlEvt

---

applemen (-- addr)

Leaves the address of the object that represents the menu under the apple in the

upper left corner of the menu bar. Where the desk accessories are stored.

---

appletalkOn **(-- bool)**

Import word from env mod that returns true if appletalk is on.

---

arccos **(ftpr -- ftpr)**

Computes $\cos^{-1}(x)$ of the floating point number pointed to by **ftpr**.

---

arcsin **(ftpr -- ftpr)**

Computes $\sin^{-1}(x)$ of the floating point number pointed to by **ftpr**.

---

arctan **(ftpr -- ftpr)**

Computes $\tan^{-1}(x)$ of the floating point number pointed to by **ftpr**.

---

arrowCurs **(--)**

Sets the cursor to the standard arrow cursor.

See also: ibeamCurs crossCurs watchCurs plusCurs

---

asc>bin **(addr len -- n)**

Takes the number string on the stack and converts it to a binary integer.

See also: bin>asc

---

ascii **Compile: (--) : chr**
Run (-- c)

If used within the program definition, compiles the ASCII value of the next character in the input stream as a literal to be placed on the stack at run time; otherwise puts the ASCII value onto the stack immediately.

Example:
```
ascii X . <return>
88 0->   \ ASCII value of upper-case x.
```

This is an immediate word.

---

asmCall **(addr len --)**

Assembles the trap call to the toolbox routine named by the string.

Example:
```
\ This code word is from the file "date"
( secs dateRec -- )
Creates secs2date                \ Converts seconds to date.
        popA0                    \ pop register A0
        popD0                    \ pop register D0
        " Secs2Date" asmCall
next,                            \ End compilation.
```

See also: call

---

at1

Parameter Stack: **(i -- e(i))**
Method Stack: **(^base -- ^base)**

Fetches the **i**-th one byte element of an indexed object whose **^base**, pointer to the objects data, is on the methods stack. Uses "at1", the optimized fetch for indexed objects with one-byte elements.

Example:
```
\ This method is from bArray.
(i --e(i) )
:M AT:  ?idx  at1  ;M
```

---

at2

Parameter Stack: **(i -- e(i))**
Method Stack: **(^base -- ^base)**

Fetches the **i**-th 2-byte element of an indexed object whose **^base**, pointer to the objects data, is on the methods stack. Uses "at2", the optimized fetch for indexed objects with 2-byte elements.

Example:
```
\ This method is from wArray.
(i --e(i) )
:M AT:  ?idx  at2  ;M
```

---

at4

Parameter Stack: **(i -- e(i))**
Method Stack: **(^base -- ^base)**

Fetches the **i**-th one byte element of an indexed object whose **^base**, pointer to the objects data, is on the methods stack. Uses "at4", the optimized fetch for indexed objects with 4-byte elements.

Example:
```
\ This method is from Array.
(i --e(i) )
:M AT:  ?idx  at4  ;M
```

---

atof

Successful: **(addr len -- fval T)**
Unsuccessful: **(addr len -- F)**

Converts a string into a floating point number.

base (-- base)

This value contains the current number base used for the input and output conversions. Note that the middle characters of the Yerk prompt changes to a '?' for all bases other than 10 and 16. For 10 the character is '-'; for 16 the character is '$'.

```
Example:  2 -> base                  \ Make the number base 2.
          1011  110 + .    \ Add 11 and 6.
          10001  0?>       \ = 17
```

become (--) : word

Executes the word following it as the highest level word in the application. Before executing the word, become clears all three stacks. Become gives you a clean slate when you jump from one part of the application to another, via a menu choice. For Example:

```
Example:  : optMen.disp become display ;
          \ optMen.disp is the handler word for selection display in the
          \ Options menu.  It transfers control to the high-level word display.
```

This is an immediate word.

beep (dur --)

Beeps the Macintosh's speaker for **dur** ticks (60ths of a second).

begin Compile: (-- yerk)
Run: (--)

Used in colon definition in the form:

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

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

This is an immediate word.

begin-dp (-- addr)

This variable contains the contents of dp, the user dictionary pointer,when Yerk starts up, before a saved user dictionary is loaded.

Example: begin-dp @ .h <return> \ Fetch and print in hex.
49B2 0-> \ Lowest address in the dictionary above the nucleus.

---

bin>asc **(n -- addr len)**

Takes the binary integer off the stack and converts it to a string.

See also: asc>bin

---

binType **(-- BIN)**

This constant leaves the file type (four ASCII characters) of Yerk modules.

See also: save TYPE txTYPE

---

bl **(-- 32)**

This constant leaves the ASCII value for "blank" on the stack.

---

blanks **(addr count --)**

Fills memory with blanks starting at **addr** for **count** number of bytes.

Example: buf255 256 blanks \ Fills the default string buffer with blanks.

See also: padBL erase fill

---

bldvec **(--)**

This is the system execution vector for the word that builds objects from classes. You should not alter the contents of bldvec, because it is solely for Yerk's internal use.

---

body> **(pfa -- cfa)**

Derives the code field address, **cfa**, form the parameter field address, **pfa**, of the word.

See also: >body cfa

---

bool **(bool -- toolbool)**

Converts a Yerk boolean into a Toolbox boolean. A Toolbox boolean is 16 bits wide with the lower 8 bits and the higher 8 bits zero for false, non-zero for true.

Example: true bool \ Convert boolean constant true
\ to a Toolbox boolean

---

bottom **(-- #pixel)**

Returns the coordinate of the bottom of the front window.

---

branch **(--)**

Compiled into the definitions by the Yerk control words: ELSE, AGAIN, LOOP, +LOOP, and REPEAT. Causes an unconditional branch to occur. Changes the value

of the IP to cause the branch to occur. For a more complete explanation of the Forth elements of Yerk, see The Forth Encyclopedia, cited in the bibliography at the end of this manual.

---

buf255 **(-- addr)**

Leaves the address of a 256-byte buffer used for converting an addr-len format string into a Toolbox-compatible str255 string. You can use this buffer as a temporary workspace, provided this does not interfere with the string processing.

```
Example:   " Mary had a little lamb"  \ Define a string literal.
           . . <return>               \ Print addr len.
           22,20039 0->
           buf255 . <return>          \ Print addr of buffer.
           20038 0->
```

See also: str255

---

build **(--)**

Defines the compile-time behavior of a word that itself defines multiple-cfa words. See Part II.4 for more detail. See "codefields" for example.

This is an immediate word.
See also: prefix codefields do.. ..end

---

buttonID **(-- 0)**

Returns the toolbox constant that indicates a control is actually a button. This word is defined in the file "ctl", and is an optional part of the system.

See also: checkID radioID useWfont vsID

---

bye **(--)**

Closes open files and returns control to the Finder via ExitToShell. Bye is executed when Quit is selected from the File menu.

---

bytes **(n --)**

Allocates **n** bytes as a named instance variable of an object. The instance variable's class is Object. You can use bytes to map a Toolbox data structure as an object when named access to some fields in the data structure is not needed.

Example:
```
\ These are named instance variables of class Grafport.
:CLASS  Grafport  <SuperObject
        16 Bytes graf1    \Unmapped
        Rect Portrect
        44 Bytes graf3    \ Unmapped
```

```
Var Text1          \ font, face,
Var Text2          \mode, size
32 Bytes graf2     \ Unmapped
```

See also: allot reserve

---

c! **(b addr --)**

Stores byte **b** at **addr**.

---

c+! **(i addr --)**

Adds the 8-bit increment **i** to the memory byte at **addr**.

---

c, **(b --)**

Stores byte **b** into the next available dictionary byte, and advances the dictionary pointer by one.

---

c@ **(addr -- b)**

Fetches the byte at addr and puts it on the stack.

---

call **(parameters -- results) : _name**

Calls a routine in the Macintosh ROM. Parameters are passed to the routine on the stack. A function's result is returned on the stack. With call you can access ALL Toolbox and Operating Systems routines. See Part II.4, section "Calling the Toolbox".

Example:

```
( -- bool ) \ Mouse button still down?
: stillDown?
        word0          \ Puts 16 zero bits on stack for function result.
        call stillDown          \ Toolbox Event Manager function
                \  to detect mouse-down.
        word0 ;         \ Pad result with 16 zero bits to
                \ make a (32-bit) cell on stack.
```

See also: asmCall
This is an immediate word.

---

caller **(-- ^obj)**

This is a Vect that holds the cfa of CopyM. It is defined in the Dialog source and late binds to the dialog. Ex: close: caller.

---

case **(--)**

Case provides a Pascal-like CASE conditional structure. If a particular clause is valid, then the words between OF and ENDOF for that clause are executed; execution resumes as the word after ENDCASE. If no clause is valid, then the words (if any) between the last ENDOF and the ENDCASE are executed; execution resumes at the word after ENDCASE. However, since each clause in the case statement leaves the argument on the stack, ENDCASE consumes the top stack argument. If the statements between the last ENDOF and the ENDCASE leave anything on the stack, the original argument must be SWAPped back to the top of the stack. Otherwise the intended result will be thrown away and the case argument will be returned.

```
Example:    : testCase ( c -- )  \ A character is on top.
            CASE
                    ascii 0 ascii 9 RANGEOF 0 ENDOF                    \
if digit, return 0
                    ascii A OF 1 ENDOF      \ if A, return 1
                    ascii B OF 2 ENDOF      \ if B, return 2
                    ascii C OF 3 ENDOF      \ if C, return 3
                    4 swap                  \ otherwise return 4 (see above)
                    ENDCASE  ;
```

See also: of rangeof endof endcase select{
This is an immediate word.

---

cfa **(pfa -- cfa)**

Derives the code field address, **cfa** , from the parameter field address, **pfa**, of the word.

```
Example:    'aWord cfa      \ "tick" leaves the pfa; cfa converts it to a cfa;
                            \  easier way is "'c aWord".
```

See also: body> link> name>

---

cfaLen **(-- 4)**

This constant leaves a 4 on the stack, the length of a cfa in the Yerk system.

---

cflush **(--)**

Flush the program cache on a 600x0 chip. You should call this whenever you modify an executable part of the program.

---

checkID **(-- 1)**

Returns the toolbox constant that indicates that a given control is a checkbox. This word is defined in the file ctl, and is an optional part of the system.

See also: buttonID radioID useWfont vsID

---

classErr" **(bool --) : resID**

ClassErr" allows an object print an error message from within a method, specifying the object's address and class. ClassErr" will print the message if the boolean on top is true. The string printed by classerr" must be in the resource file "yerk.rsrc".

Example: :M READIT: ... read: theFile classErr" 150 ... ;M
\ 150 is the resource ID of the string to be printed on error.
\ An error message may be:
Error from class ARRAY ::A2FE File read failed

This is an immediate word.

---

clean1 **(--)**

Cleans up the filelist object when an abort occurs.

---

clean2 **(--)**

Cleans up the window related information when an abort occurs. Calls clean1.

---

cleanFloat **(--)**

Cleans up the floating point heap when an abort occurs. Calls clean2.

---

closeall **(--)**

Closes all open windows except the fwind. Each window, when open, has an instance variable that is set true. This value is saved with the snapshot, so that when you then try to load the new image, the windows that were open will think they are still open. Version 3.64 takes care of this instance variable in the background (without closing the windows), but to be safe, you should execute this word before saving a snapshot of your work.

---

clrFCB **(fcb --)**

Clears (zeros) the **fcb**.

Example: ffcb clrFCB \ Clear the default file object

---

cls (--)

Clears the current window (or grafport) and puts the cursor in the upper left corner.

See also: home

---

cmove (addr1 addr2 count --)

Copies **count** bytes starting at **addr1** to **addr2.** This uses the Mac Toolkit call BlockMove, and can handle moving from high to low, or low to high memory.

Example:
```
\ Copy the 64 bytes (characters) at here to buf255.
here buf255 64 cmove
```

---

codefields (n --)

Defines **n** as the number of cfa's a multiple-cfa data structure will have. See Part II.4.

Example:
```
\ This is a possible implementation of value; the actual
\  implementation is more efficient
2 prefix ->       \ Define a prefix operator for value.
1 prefix ++>      \ Define another prefix operator.
3 codefields      \ A value will have 3 cfa's.

' -> Do.. ! ..End          \ 2cfa is the store operation.
' ++> Do.. +! .. End      \ 1cfa is the increment operation.
Do.. @ ..End     \ 0cfa is the fetch operation (default).

: value  build ,  ..End     \ Define the compilation behavior of value
```

---

colcode (-- addr)

This constant returns the address for the code which executes colon definitions; the traditional FORTH equivalent is doCol.

See also: fval mopdCode valCode vectCode

---

cold (--)

Initializes Yerk to its startup state, as if you had just opened the Yerk user dictionary you are working on. You can execute cold to remove all dictionary entries since Yerk started up.

Example:
```
cold <return>                \ Restart Yerk.
Macintosh Yerk version X.X
Bytes available: XXXXXX
```

0->

---

command? (-- bool)

Returns true if the command key is held down when the word is executed.

See also: option? shift? ctl?

---

compile (--)

Compile is useful in describing compile-time behavior for a word that extends Yerk's language model. Compile builds the word following it in the dictionary when the word containing Compile is executed (generally during compilation of another word). All of Yerk's control structures, such as IF, BEGIN and DO are implemented using Compile.

```
Example:    : myCompiler ... compile aWord ... ;                    \
definition of a compiling word
            Immediate                       \ compiler words are immediate

            : xxx ... myCompiler;           \ myCompiler executes
```

See also: [compile]

---

compound **(fptr:r fptr:n -- fptr)**

Computes compound(r,n)=$(1 + \mathbf{r})^{\mathbf{n}}$, where **r** is the interest rate and **n** is the number (perhaps nonintegral) of periods. When the rate is small, compound gives you a more accurate computation than does the straightforward computation of $(1 + \mathbf{r})^{\mathbf{n}}$ by addition and exponentiation.

See also: annuity

---

constant **(n --)**

Constant creates a word, initializing its contents to n. When the word is executed, it puts n on the stack. You can use constant to define (you guessed it) constants that you'll need later in your program. This will help you understand your program when you read it later.

```
Example:    1024 constant oneKay    \ Define constant
            oneKay . <return>       \ Print oneKay
            1024 0->
```

See also: scon fcon value variable

---

contBot **(-- #pixel)**

Returns the coordinate of the bottom of the content of the region of the font window.

See also: contTop

---

contTop **(-- row)**

Returns the coordinate of the top of the content of the region of the font window.

See also: contBot

---

copym **Parameter Stack: (-- n)**
Method Stack: (n -- n)

Pushes a copy of the number on the methods stack onto the parameter stack.

See also: addm exgm dropm dupm popm pushm ^base

---

cos **(fptr -- fptr)**

Computes cos(x) of the floating point number pointed to by **fptr**.

---

cot **(fptr -- fptr)**

Computes cot(x) of the floating point number pointed to by **fptr**.

---

count **(addr -- addr+1 count)**

Converts a str255-format string (0th byte is count byte) to an addr-len format string.

```
Example:   bl word      \ Input a blank-delimited word.
           here         \ The address of the str255-format string
           count type   \ Convert to addr len format and print
```

See also: buf255

---

cr **(--)**

"Prints" a carriage-return/linefeed (cr/lf) on the screen. If system vector pcrvec is vectored to pcr, by executing +print, the cr sends a cr/lf to the printer.

See also: pcr ?lead scroll

---

create **(--) : name**

Creates a header for the definition of a word. The header consists of a field name, link field, and code field. The code field contains the address of the next field, the parameter field. Create leaves the dictionary pointer pointing to the presumptive) parameter field. You can use CREATE to build code words, words that contain machine code at their parameter field. CREATE is actually

the system vector that is defaulted to the primitive (CREATE). You should never have to change this vector.

Example:
```
\ This code word is from file "date".
(secs dateRec -- )
Create secs2date                \ Converts seconds to a date.
        popA0                   \ pop register A0
        popD0                   \ pop register D0
        "Secs2Date" asmCall
next,                           \ End compilation.
```

See also: sCreate

---

crossCurs (--)

Changes the cursor to the predefined cross form.

See also: ibeamcurs pluscurs watchcurs cursor arrowcurs

---

crvec (--)

System Vector whose default is (cr).

See also: cr (cr) prvec

---

csp (-- addr)

This value is the stack pointer position when compilation of a colon word or class is begun. Csp is used for compilation error-checking.

See also: !csp ?csp

---

cstate "c-state" (-- bool)

This value is a boolean which is true while a class is being compiled. It is analogous to state which is true while a colon word is being compiled.

See also: c[?class state

---

ctlexec (part# ctlhandle --)

This is an optional word defined in the source file "ctlwind". Executes the Exec: method for the control object. **Part#** is toolbox indicator for what part or type of control was clicked on.

---

ctlhit? (^wind -- bool)

Determines if a content click is a hit on the control' area, and if so, processes that click by tracking the control, if appropriate, and then executing the control's handler. **^wind** is a pointer to the owner window. The **bool** returned indicates whether or not the word did the processing on the click. This is an

optional word defined in the source file "ctlwind".

---

ctlproc **(^wind -- bool)**

This is a :proc word and is passed as the completion routine for control handling. This is an optional word defined in the source file "ctlwind".

---

curs **(-- bool)**

This value is the boolean which controls displaying of the cursor. If curs is true, the cursor is displayed; if curs is false the cursor is hidden.

Example: : -curs false -> curs ;

See also: +curs -curs

---

cursor **(cursid --) : name**

This is a simple data type defined as a mcfa defining word with a single action. The definition is in the file QD, and is a good example of mcfa usage. It is used to define the words crosscurs, watchcurs, pluscurs, and ibeamcurs, which change the cursor in use. Cursid indicates the resource id number to use for the cursor definition. Types one through four inclusive are used by the screen.

See also: crosscurs ibeamcurs pluscurs watchcurs arrowcurs

---

cvtClip **(--)**

This vector may be filled by your application to take care of converting the clip during suspend or resume events. Default is null.

See also: suspend resume mouseMoved inForeGround

---

c[**(--)**

Stops compilation within a Class definition to resume run state. Sets cstate = 0. Used in conjunction with]c.

This is an immediate word.

---

d+ **(d1 d2 -- dsum)**

Adds two double numbers leaving their sum.

---

d. **(d --)**

Prints a double number using the current number base as the conversion radix.

---

d.r **(d w --)**

Prints a double number using the current number base, right-justified in a field **w** characters wide. The entire number is printed even if it exceeds the field width.

---

d< **(d1 d2 -- bool)**

Compares two signed double numbers. If **d1** is less than **d2**, leaves a true flag; otherwise, leaves a false flag.

---

d= **(d1 d2 -- bool)**

Compares two signed double numbers. If **d1** is equal to **d2**, leaves a true flag; otherwise, leaves a false flag.

---

d> **(d1 d2 -- bool)**

Compares two signed double numbers. If **d1** is greater than **d2**, leaves a true flag; otherwise, leaves a false flag.

---

dabs **(d -- ud)**

Leaves the absolute value of a signed double number.

Example:
```
-47 0      \ Put -47 on the stack as a double number.
dabs d. <return>          \ Convert to absolute value and print it..
47 0->
```

---

dblTicks **(-- dblTicks)**

Leaves the maximum number

Example:
```
'aWord cfa      \ "tick" leaves the pfa; cfa converts it to a cfa;
                \  easier way is "'c aWord".
```

See also: body> link> name>

---

de' **(--) : word**

Decompiles a yerk definition. Uses the configuration in the demod dialog box.

See also: dode (de)

---

decho **(-- bool)**

This value is the boolean that controls echoing to screen of source loaded from disk. If true, echoing to screen is turned on.

Example: : +echo true -> decho ;

See also: +echo -echo

---

decimal (--)

Sets the base, the numeric conversion radix, to 10. You can put decimal at the beginning of each source file to be sure the base is 10.

See also: .d base hex

---

default{ (--)

Begins the default clause of a select{...}select conditional structure. The words within default{ and }end are executed if none of the preceding clauses are valid.

Example:
```
\ Prints "zero", "one", "two , or "none of the above".
(n --)
: testSel
select{
        0 Is{." zero" }end
        1 Is{." one"  }end
        2 Is{." two"  }end
        default{." none of the above"
}select ;
```

This is an immediate word.
See also: select{ }select is{ }is

---

deg2rad (ftpr -- fptr)

Converts the floating point number pointed to by **ftpr** from degrees to radians.

---

depth (-- n)

Leaves the number of 32-bit cells occupying the stack (before **n** was placed on top of the stack).

---

dfa (^class -- dfa)

Derives the data field address, **dfa** which contains the length of the non-indexed data associated with the class), from the **^class** or pfa of a given class definition.

---

die (n --)

Prints a string of ID **n** in the resource chain, prints it, then aborts.

See also: abort"

---

digit

Successful: **(c base -- n2 T)**
Unsuccessful: **(c Base -- F)**

Converts the ASCII character **c** using base into its binary equivalent **n2**, and leaves a true flag; if conversion fails, leaves a false flag only. Digit is a primitive used by (number).

```
Example:  54          \ ASCII for '6'.
          10          \ Conversion base.
```

```
digit .. <return>\ Convert to digit.
1  6  0->          \ The 1 is a true flag.
```

---

dirFind **(n fcb --)**

Calls the HFSDispatch Procedure of the Mac File Manager. **n** is the call number and **fcb** is the file parameter block. See IM-IV.

---

disfw **(--)**

The disable action for fwind. Disables various menus items.

See also: enfw

---

disk-evt **(-- 0)**

This word is called whenever a disk insert event occurs. The default definition prints a mount message on the screen. This word can be redefined through the use of :f or by substitution of vector 7 in fEvent.

See also: actv-evt key-evt mouse-evt null-evt upd-evt

---

dispose **(addr --)**

Releases the nonrelocatable block of heap pointed to by th cell whose address is on the stack and zeroes the cell.

This is an immediate word.

---

dispose> **(--) : block**

A prefix that releases a nonrelocatable block of heap whose pointer is stored in a Yerk data structure. Dispose> can operate on values, named input parameters, local variable, and modules. Must be used within a word; cannot be used in the interpret state.

Example:
```
free . <return>
2023 0->
```

```
: getRid dispose> myMod ; <return>
getRid free . <return>
3564 0->          \ Heap space occupied by module has
                  \ been made available.
```

---

dlgWind (-- 1)

This constant is the resource ID for a window having no title bar and a double click outline.

See also: docwind rndwind

---

dliteral

Compile: (d --)
Run: (-- d)

At compile time, compiles the double number on the stack as a literal into the definition being compiled. When the definition is executed later, **d** is put on the stack.

Example: : testDlit <[5467203 2234435]> dliteral ;
\ testDlit puts 9,596,825,255,504,963 on top.

This is an immediate word.

---

dnegate

(d -- d)

Leaves the two's compliment of a double precision number.

---

do

Compile: (-- yerk)
Run: (n1 n2 --)

At run time, DO begins repetitive execution of the sequence of words up to LOOP or +LOOP. The loop limit is **n1** and the initial value of the loop index is **n2**. Upon reaching LOOP, the index is incremented by one (upon reaching +LOOP, the index is incremented by the number on the stack). The DO loop is executed until the index equals or exceeds the limit. When the loop index equals or exceeds the limit, the loop parameters are discarded and execution continues after LOOP or +LOOP.

Example: 10 0 DO i . LOOP \ Prints from 0 to 9.

See also: i j leave
This is an immediate word.

---

do'

(--) : class

Lists all objects of class **class**.

See also: doolist

---

do.. **"do-dot-dot" (--)**

Begins the definition of one of the operations (cfa's) of a multiple-cfa word. Used in conjunction with ..End. See part II.4 for more detail.

Example:
```
\ This is a possible implementation of value; the actual
\ implementation is more efficient.
2 prefix ->                 \ Define a prefix operator for value.
1 prefix ++>                \ Define another prefix operator.

3 codefields                \ A value will have 3 cfa's.
' -> Do.. ! ..End                   \ 2cfa is the store operation.
```

```
'++> Do.. +! ..End          \ 1cfa is the increment operation.
Do.. @  ..End               \ 0cfa is the fetch operation (default).

: value  build, ..End       \ Define the compilation behavior of
```
value.

---

docWind **(-- 0)**

This constant is the resource ID of a document window.

See also: dlgwind rndwind

---

dode **(--)**

Brings out the demod dialog box for word decompilation.

See also: de' (de)

---

does> **(--)**

Compiling word paired with <builds. Defines the run time behavior of a word defined by the new compiling word.

See also: <builds

---

doinDlg **(addr len -- addr len T or F)**

Import word from Indlg module. Takes the string on the stack, display it in a dialog for keyed entry. If the OK button is hit, returns the string the user entered in the TEdit box and a True. If the Cancel button is hit, returns a False. See the Module section in the Manual.

---

doolist **(--)**

Lists members of a class. Uses the configuration in the demod dialog box.

See also: do'

---

dosave (--)

Takes the current image name and saves a copy in the yerk folder.

See also: (save) save saver

---

doTasks (--)

Executes the task in taskList. Tasklist contains the cfa's of up to four words that fEvent executes when it receives a null event. This provides a means of executing

tasks in the background when Yerk is in its event loop. (You must load optional file "Tasks" to use this facility.)

See also: addtask killtask tasklist

---

dp **(-- addr)**

This value is the dictionary pointer; it is the address of the next free memory location above the dictionary. The number may be read by Here and altered by Allot.

See also: begin-dp flen room msize here

---

dpl **(-- n)**

Returns the number of digits to the right of the decimal on double integer input. This may be used to compute the output column location of a decimal point in user generated formatting. The default value on single number inputs is -1.

See also: number interpret ?num

---

drop **(n --)**

Discards the number on top of the stack.

See also: 2drop fdrop

---

dropm **Method Stack: (n --)**

Drops the number on the methods stack.

See also: addm copym exgm dupm popm pushm

---

dump **(addr n --)**

Dumps the contents of memory starting at **addr** for **n** bytes in a hex and ASCII format.

Example: $ be98 20 dump <return>
Dump from address:BE98

```
              0  1  2  3  4  5  6  7...          0123456789ABCDEF

      BE90  A9 1D 00 00 BB    .....                ..........
      BEA0  1E 72 BC E3 00    .....                .r...j.....
```

---

dup (n -- n n)

Duplicates the number on top of the stack.

See also: 2dup fdup -dup

---

dupm **Methods Stack: (n -- n n)**

Duplicates the number ob top of the methods stack.

See also: addm copym exgm dropm popm pushm

---

e **(-- fptr)**

This is the constant which places the value for **e** on the floating point heap and returns its pointer.

See also: pi ln(1 0) 0.0 1.0

---

e. **(fptr --)**

Prints the number pointed to by **fptr** in scientific notation, using 26 digits.

See also: e.r f.r

---

e.r **(fptr w --)**

Prints the number pointed to by **fptr** in scientific notation, in a field **w** characters wide.

See also: e. f.r

---

echovec **(--)**

This system vector is executed in Yerk to echo keystrokes. When Yerk starts up, echovec is vectored to (emit) which prints a character on the screen only. Selecting Echo to Printer from the Yerk menu vectors echovec to echo, which prints a character on the screen and printer (this is the same as +print).

---

edname **(-- addr len)**

Returns the STR resource with ID 99 stored in the Yerk.rsrc file. This is used to call a desk accessory editor of your choice. Just set the string using ResEdit to

be your own desk accessory editor name.

---

else

Compile: (yerk -- yerk)
Run: (--)

At run time, if the boolean on top of the stack before the IF is true, the words between if and else are executed; otherwise, the words between else and then are executed.

Used in this format: IF ... ELSE ... THEN

This is an immediate word.

---

emit (chr --)

Prints the character (lower 8 bits of the number) on the stack to the screen or printer by executing the system vectors emitvec and pemitvec. Whether the screen and/or printer are used depends on the contents of emitvec and pemitvec. When Yerk starts up, emitvec is vectored to (emit) and pemitvec is vectored to drop. So, the default action of emit is to print a character on the screen only. Selecting Echo to Printer from the Yerk menu vectors echovec to echo, which prints a character on the screen and the printer (this is the same as +print). The system value OUT is incremented for each character transmitted.

Example: ascii t emit <return>
t 0->

See also: space pemit +print

---

emitvec (--)

This system vector contains the cfa of the word that preforms character output to the primary device, usually the screen. When Yerk starts up, emitvec is vectored to (emit), the primitive that prints a character on the screen using the DrawChar routine in QuickDraw. All input/output in Yerk is vectored via system vectors such as emitvec so that you can tailor I/O to your needs.

```
Example:  'c (plotEmit)    \ Get cfa of primitive that sends a character to a
                           \ hypothetical plotter.
          -> emitvec       \ Put it in emitvec

          45 emitvec       \ Send a character to the plotter
```

---

enclose (addr c -- addr n1 n2 n3)

Parses beginning at **addr** for a text string delimited by character **c**. Leaves the following parameters:

addr --address from which parsing began;
n1 --offset to the beginning of the string;
n2 --offset to the end of the string; and

n3 --offset to the first unexamined character.

Enclose will not proceed past an ASCII null, treating it as an unconditional delimiter.

See also: parse

---

endcase **(n --)**

Ends a case conditional structure. Used in conjunction with CASE.

This is an immediate word.

---

endof (--)

Ends each clause in a case conditional structure. Used in conjunction with OF.

This is an immediate word.

---

enfw (--)

The enable handler for fwind. Enables various menu items.

See also: disfw

---

eof (-- -39)

This constant is the ioResult returned by Macintosh ROM routines for an end-of-file condition.

---

erase (addr n --)

Clears (fills with zeros) memory starting at **addr** for **n** bytes.

Example: buf255 256 erase \ Erase buf255

See also: blanks fill

---

errbeep (--)

Sounds the standard error tone. Defined as: 5 beep.

---

errlog (--)

Execute this word if you want to revector the output to send a stack dump to a file that will be created and named "ErrLog". If the file already exists, nothing happens.

---

exam (--)

Import word from the exam module. Brings out a window to display a hex and ascii dump of the application's heap.

---

exec> **(--) : word**

This prefix operator executes the contents of the operand following it as the cfa of a word. The operand may be a named input parameter, local variable, value, vect, or sysvect. You don't need to prefix a vect or sysvect with exec>

Example: : doGraph { functionCfa \a b -- }
...

```
exec> functionCfa   \ The cfa of a function to be graphed is passed
                    \ as a named input parameter; it is executed
                    \ when needed
... ;
```

This is an immediate word.

---

execute **(addr --)**

Executes the word whose compilation address (code field address) is on the stack.

Example:
```
'c myFunc value func \ Value func contains cfa of myFunc
func execute         \ Func puts cfa on top; execute executes it
                     \ Another way  is "exec>func".
```

See also: ,exec

---

exgm **Parameter Stack: (n1 -- n2)** **Method Stack: (n2 -- n1)**

Swaps the top two numbers on the methods and parameter stacks.

See also: addm copym dropm dupm popm pushm

---

exit **(--)**

Terminates execution of the current word or method, and returns control to the next higher word on the return stack. Exit may be used in IF ... ELSE ... THEN, BEGIN ... REPEAT, BEGIN ... WHILE ... UNTIL, CASE ... ENDCASE, or Select{ ... }select structures. Exit may not be used inside a DO loop.

Example:
```
: testExit { doneFlag -- }
doneFlag          \ Exit if true
IF  ."Quitting." exit
ELSE ." Entering doGraph." doGraph
THEN
cr ."Graph plotted". ;
```

```
true testExit <return>
Quitting. 0->

false testExit <return>
Entering doGraph.
Graph plotted. 0->
```

This is an immediate word.

---

exp **(fptr -- fptr)**

Computes e^x for the floating point number pointed to by **fptr**.

---

exp1 **(fptr -- fptr)**

Computes $e^{x}-1$ for the floating point number pointed to by **fptr**.

---

exp2 **(fptr -- fptr)**

Computes 2^{x} for the floating point number pointed to by **fptr**.

---

exp21 **(fptr -- fptr)**

Computes $2^{x}-1$ for the floating point number pointed to by **fptr**.

---

expect **(addr count --)**

Transfers characters from the keyboard to memory starting at **addr** until a return is received or **count** characters have been input. Characters are echoed to the primary and secondary devices by executing the system vector echovec. When Yerk starts up, echovec is vectored to (emit), so that characters are echoed to the screen only. Expect adds a null at the end of the text string.

```
Example:  buf255 10 expect <return>          \ Expect 10 chars at buf255
          abcdefghij  0->
          buf255 10 type <return>  \ Type the 10 chars at buf255
          abcdefghij  0->
```

See also: expvec

---

expvec **(--)**

This system vector allows a user-installable expect routine. The default routine is expect. For a more complete discussion of system vectors, look in Part II.4.

---

extend **(n -- n)**

Extends the sign of a 16-bit number, making it a 32-bit signed number. You can use extend to handle results from Toolbox calls which return 16-bit signed integers.

Example: $ FF23 \ -221 as a 16-bit signed int, but on the stack as 32bit
extend \ Sign-extend it.
dup .h cr. <return>
FFFFFF23 \ -221 as a 32-bit signed number.
-221 0-> \ Print it as a decimal.

See also: i->|

---

f* **(fptr0 fptr1 -- fptr0)**

Multiplies two floating point numbers on the float heap and leaves a single floating point result with its pointer on the stack.

---

f+ **(fptr0 fptr1 -- fptr0)**

Adds two floating point numbers on the float heap and leaves a single floating point result with its pointer on the stack.

---

f, **(fptr --)**

Compiles a 10-byte floating point number off the float heap into the next available dictionary location, and advances the dictionary pointer. Removes the pointer to the float heap from the stack.

---

f- **(fptr0 fptr1 -- fptr0)**

Subtracts two numbers on the float heap leaving the result. Leaves a pointer to the result on the stack.

---

f.r **(fptr w d --)**

Prints the floating point number pointed to by **fptr** without exponents, in a field **w** wide and with **d** decimal places. If there are more than **d** decimal places, it rounds.

See also: e. e.r

---

f/ **(fptr0 fptr1 -- fptr0)**

Divides the floating point number pointed to by **fptr0** by the one pointed to by **fptr1** leaving the result. Leaves a pointer to the result on the stack.

---

f0< **(fptr -- bool)**

Compares the floating point number with zero. If less than zero, leaves a true flag; otherwise, leaves a false flag.

---

f0= **(fptr -- bool)**

Compares the floating point number with zero. If equal to zero, leaves a true flag; otherwise, leaves a false flag.

---

f0> **(fptr -- bool)**

Compares the floating point number with zero. If greater than zero, leaves a true flag; otherwise, leaves a false flag.

---

f2drop **(fptr fptr --)**

Drops the 2 floating point numbers.

See also: 2drop

---

f2dup **(fptr1 fptr2 -- fptr1 fptr2 fptr1 fptr2)**

Duplicates the two floating point numbers on the stack.

See also: 2dup

---

f< **(fptr0 fptr1 -- bool)**

Compares two floating point numbers on the float heap. If the one pointed to by **fptr0** is less than the one pointed to by **fptr1**, leaves a true flag; otherwise, leaves a false flag.

---

f<= **(fptr0 fptr1 -- bool)**

Compares two floating point numbers on the float heap. If the one pointed to by **fptr0** is less than or equal to the one pointed to by **fptr1**, leaves a true flag; otherwise, leaves a false flag.

---

f<> **(fptr0 fptr1 -- bool)**

Compares two floating point numbers on the float heap. If the one pointed to by **fptr0** is not equal to the one pointed to by **fptr1**, leaves a true flag; otherwise, leaves a false flag.

---

f= **(fptr0 fptr1 -- bool)**

Compares two floating point numbers on the float heap. If they are equal, leaves a true flag; other-wise, leaves a false flag.

---

f> **(fptr0 fptr1 -- bool)**

Compares two floating point numbers on the float heap. If the one pointed to by **fptr0** is greater than the one pointed to by **fptr1**, leaves a true flag;

otherwise, leaves a false flag.

---

f>= **(fptr0 fptr1 -- bool)**

Compares two floating point numbers on the float heap. If the one pointed to by **fptr0** is greater than or equal to the one pointed to by **fptr1**, leaves a true flag; otherwise, leaves a false flag.

---

fAbs **(fptr -- fptr)**

Takes the floating point number pointed to by **fptr** and returns its absolute value.

---

false **(-- 0)**

Leaves a false flag, zero.

See also: true not

---

fcall **(fcb -- result) : _name**

Special Call for low-level file manager routines in the Macintosh ROM. Parameters are passed to the routine on the stack. A function's result is returned on the stack. Sets a call up using (fdos).

This is an immediate word.
See also: asmcall call (fdos)

---

fCon **(fptr --) : name**

FCon creates a word, initializing its contents to the floating point value pointed to by the pointer on the stack. When the word is executed, it puts the floating point number on the float heap, and returns its pointer to the stack.

See also: fValue

---

fDrop **(fptr --)**

Deallocates a floating point value on the float heap and drops its pointer from the stack.

---

fDup **(fptr0 -- fptr0 fptr1)**

Duplicates a floating point value on the float heap and returns its pointer from the stack.

---

fence **(-- addr)**

This value is the address below which forget will not operate. The default value of fence is the name field address of the last word in the nucleus, so that you don't forget only words in the user dictionary.

---

fEvent **(-- ^base)**

This event is the default (and should be the only) Event object used by Yerk. It exists in the nucleus. See Part III.4, Events.

---

fFcb **(-- ^base)**

This File object is the default I/O parameter block used by Yerk and exists in the nucleus. If you need a parameter block, don't use fFcb but instead use the New: method of class loadFile, the stack of fcb's. See the III.2 Files.

---

fFind **(-- bool)**

Leaves a true if the string at here is a floating point number.

---

file-install **(fcb type sig --)**

Sets the file type and signature of the file specified by fcb. Each of type and signature is 4 ASCII characters as one 32-bit number.

---

fillInit **(--)**

Initializes the file objects. This word is intended to be part of the objinit vector, which initializes system objects as a startup.

---

fill **(addr n b --)**

Fills memory with byte **b** starting at **addr** for **n** bytes.

Example:
```
\ This is a possible definition of erase.
: erase  { addr n -- }
          addr n 0 fill  ;    \ Fill memory with 0's.
```

See also: erase blanks

---

find **Successful: (-- pfa len T) : word**
UnSuccessful: (--F)

Takes the word out of the input stream and searches for it in the dictionary. Returns a boolean to indicate if it found it or not. If so, returns the **pfa** and the **len**gth of the reference in the dictionary.

See also: (find) sFind

---

findFmark **(addr -- cfa T or F)**

Import word that returns the **cfa** of the filemark above the input **addr** if +docs has been enabled.

See also: +docs -docs fm rl mforget /// see srcname

---

find-method **(selhash objpfa -- objpfa 1cfa)**

Performs a high speed search to find a method specified by the **selhash**, a selector hash value, in the superclass chain of the object specified by **objpfa**. If successful, leaves **objpfa** and the **1cfa** of the method. If unsuccessful, it aborts and prints message:

```
Msg#  108  :method not found in class
XXXX  ::
```

---

find-window **(Tpoint -- region ^window)**

Calls the Window Manager routine FindWindow to find the window in which a mouse-down event occurred at Tpoint, a Toolbox point in global coordinates. Leaves the **region** of the window in which the mouse-down occurred and ^**window**, the pointer to the window object's private data.

---

fInfo **(-- ptr)**

Leaves a relative pointer to the Finder information area. The following code will scan the information block, and print the Vref#, file type, version and file name for each file passed by the Finder.

```
: .fInfo    CR fInfo                    \ get pointer to Finder Information
block
            dup w@ ." Documents to be "
            IF ." Printed ELSE ." Opened:" THEN
            4+ dup 2- w@ 0              \ get number of files passed
            DO CR
               dup w@ .                 \ print Vref#
               2+ dup @ sp@ 4 type drop  \ print file type
               4+ dup c@ .                  \ print file version
               2+ count 2dup type       \ print file name
               + align                  \ ...advance to next file
            LOOP drop ;
```

---

firstchr **(-- char)**

Returns the **char** stored at here + 1.

---

flen **(-- len)**

Leaves the length of the user dictionary to be saved (dp - [begin-dp]). Flen is used by save.

---

fLit **(-- fptr)**

Takes ten bytes from the next dictionary address, allocates an entry on the float heap, and returns its pointer on the stack. Within a colon definition, fLit is automatically compiled before each floating point number encountered in the input text.

---

fLiteral

Compile: **(fptr --)**
Run: **(-- fptr)**

At compile time, compiles the floating point number pointed to by the number on the stack. This word is an immediate, and so executes within a colon definition. The run-time action of fLiteral is provided by fLit.

This is an immediate word.
See also: fLit, literal

---

float>

(fptr -- n)

Takes the floating point number pointed to by **fptr** and converts it to a 32-bit number leaving the result on the stack.

---

fltMem

(-- ^obj)

The floating point heap object. Default has room for 100 floats. You can check how much room you have by sending the room: message to it.
Example: room: fltmem .
98

---

fm

(--)

Import word that forgets all definitions up to the last filemark. +docs must be enabled.

See also: +docs -docs see rl mforget /// srcName findFmark

---

fmax

(fptr1 fptr2 -- fptr)

Returns the maximum value of the two floats on the stack.

---

fmin

(fptr1 fptr2 -- fptr)

Returns the minimum value of the two floats on the stack.

---

fmod

(fptr1 fptr2 -- fptr)

Returns the remainder of the division of **fptr1** by **fptr2**, rem f1/f2.

---

fNegate **(fptr -- fptr)**

Takes the floating point number pointed to by **fptr** and returns its complement.

---

fNumber **Successful: (addr -- fptr T)**
UnSuccessful: (addr -- F)

Attempts to convert the str255 format string at **addr** to a floating point number. If successful returns its pointer and a true flag; otherwise returns a false flag.

---

forget **(--) : word**

Deletes the word from the dictionary and ALL entries above it (i.e., in higher memory) . Issues message "in protected dictionary" if the word is below fence, and aborts.

```
Example:    forget doMyAppl              \ Deletes doMyAppl and all words
                                         \ above it in the dictionary
            forget (emit) <return>       \ Try to forget a nucleus word.
            (EMIT)? in protected dictionary
            File Stack:
            Token=(EMIT)  ::
              0->
```

---

forward **(--) : name**

Creates a word which is to be defined later with :f, the special colon compiler for forward-referenced Yerk words, because :f makes the word behave like a colon definition. If you forget to define a forward-referenced word, the word issues the message: "Unresolved forward reference to: AWORD at address: XXXX" and aborts. To forward reference an object, define a value having the same name as the object-to-be. Later create the object as you would normally.

```
Example:    forward  drawSpiral      \ Forward reference the word.
                ...                  \ More code which may call
                                     \   drawSpiral.
            :f  drawSpiral ... ;f    \ Define drawSpiral.

            0 value pandora          \ Forward reference an object.
                ...                  \ More code including refer-
            :Class Cube              \   ences to pandora, and
              <Super Object          \   definition of pandora's
                ...
            ;Class                   \   class Cube.
```

```
...
Cube pandora                    \ Object pandora finally created.
```

---

fOver **(fptr0 n -- fptr0 n fptr1)**

Creates a copy on the floating point heap of the floating point value pointed to by the second number down on the stack. Returns a pointer to the new value on the stack

---

fpInit **(--)**

This is the startup word to initialize the floating point heap. You must include this in your application startup word if you use floating point. See the definition of 'yerk' in the source Frontend.

fprect (-- addr)

This is a system object of class rectangle used in calculation for scrolling fWind.

free (-- n)

Calls the Memory Manager routine MemFree to find the total number of bytes available on the application heap. Free is useful in debugging the compilation and loading of modules and Yerk objects that allocate space on the heap. Because of fragmentation, you may not be able to allocate a single block as large as the value returned by Free.

Example: free . <return>
7094 \ 7094 total bytes available.

freeBlk (-- n)

Calls the Memory Manager routine MaxMem to find the size in bytes of the largest block available on the application heap. MaxMem compacts the heap and purges all purgeable blocks before returning its result. Freeblk is useful in debugging the compilation and loading of modules and Yerk objects that allocate space on the heap.

Example: freeblk . <return>
3024 \ Largest block available.

from (--)

This construct is used to define modules, so that the exported words are properly located in the dictionary. See Part II.4 for full description of the us of modules.

Used in the form: From modName Import{ word1 word2 word3 }

See also: import{ ;Module :Module module

frontWind **(-- ^obj)**

Calls the system Toolbox routine FrontWindow. Leaves a pointer to the frontmost window of your application on the stack.

---

fvalCode **(-- addr)**

This constant returns the address of the code executed by fValue

See also: colCode modCode valCode vectCode

---

fValue **(fptr --) : name**

General purpose data variable for floating point values. See Part III.11.

Used in the form: 1.2 fValue foo

See also: Value

---

fWind **(-- ^obj)**

This is the default Window object and is the window you normally see when using Yerk. fWind exists in the nucleus.

---

g->l **(Tpoint -- Tpoint')**

Converts a Toolbox point (higher 16 bits y, lower 16 bits x) in global coordinates into a Toolbox point in the local coordinates of the current grafport.

See also: l->g

---

gestalt

Successful (-- n 0) : word
Unsuccesful (-- err)

Takes the next word in the stream and converts it to a gestalt selector, then calls (gestalt). An error is a negative number, with -1 meaning gestalt is not available on the machine you are using. State smart word.

This is an immediate word.
See also: (gestalt)

---

get-ctl-obj **(ctlHandle -- ^obj)**

Given a handle to a control record, returns the object associated with the control. Assumes that the control has been initialized be Yerk using Set-ctl-obj.

---

get-event **(eventMask -- eventType)**

Calls the Toolbox Event Manager routine GetNextEvent repeatedly until an event matching **eventMask** is detected. Leaves the type of the first non-null event detected.

See also: ?event @event-msg ?terminal

---

getA3A5 **(-- a3 a5)**

Returns the current values of registers A3 and A5. Used to stuff completion routines with the current values. You should not need to use this since a high level word 'InitProcs' uses it.

---

getEnv **(-- addr)**

An import word from the env module. Returns the address of the sysenvirons record.

---

getEvent **(-- bool)**

Calls the Toolbox Event Manager routine GetNextEvent (and SystemTask) or WaitNextEvent (depending on the Rom version of your Mac) to get the next event from the event queue into the private data of the Event object. Leaves a true flag if an event that the user's application should handle is available; otherwise, leaves a false flag. GetEvent is used by the next: method of object fEvent.

---

getHsize **(handle -- size)**

Obtains this **handle**'s **size**. This word is used by class handle; you should normally use the Size: method of that class.

See also: setHsize

---

getPtrSize **(ptr -- size)**

Obtains this relative pointer's size.

See also: newPtr growPtr

---

getMtxt **(addr len --) NO LONGER SUPPORTED**

Loads an entire menu bar from a file. GetMtxt loads the text and handler cfas of each menu's items, and Initializes and "wakes up" each Menu object and object menuBar so that the Toolbox is aware of it. See Part III.6 on menus. THIS IS NO LONGER SUPPORTED IN VERSIONS 3.64 and higher. You now load menus from a resource file.

---

getPtxt **(addr len --)**

Fills the pathlist object with the paths specified in the named file. See Part III.10 on Files

---

getRes **(resID -- handle) : type**

Calls the toolbox routine GetResource, returning the resource's handle; or zero if not found. GetRes can be used interactively, as well as within colon definitions.

This is an immediate word.
See also: (GetRes)

---

getScrap **(-- len)**

Transfers the desk scrap Text contents to 'parmstr'. Returns the length of the desk scrap transfered. Used in copying and pasting...you should not need to call this word.

---

getString **(resID -- addr len)**

Takes the resource ID of a STR resource and returns an addr-len pair. The string must be in the currently open chain of resource files.

---

global **(-- n) : name**

Using this word, you can get the value of a global quantity by name. This is defined in the Yerk module TOOL.BIN.

```
Example:    : topOfMem
                    global memTop  ; \ get the global variable 'memTop'
                                   \   on the stack.
```

This actually compiles the absolute address of 'memTop' ($108) and '-base' in the definition of 'topOfMem'. So if you decompile 'topOfMem', you would see the definition is:

```
: TOPOFMEM 264 -BASE ;
```

This is an immediate word.

See also: konstant

---

gotoxy **(x y --)**

Positions the cursor at the coordinate point on the stack.

See also: @xy

grayRgn (-- l t r b)

Returns the coordinates of the rectangle bounding the region of all graphic screens installed on the Mac.

growPtr (n ptr --)

Calls the Memory Manager routine SetPtrSize to increase the size of the nonrelocatable block on the application heap whose relative pointer is on the stack by **n** bytes.

See also: getPtrSize newPtr

growZone (--)

This vector holds the cfa of a word to be executed when the system needs more heap space. You may want to assign GrowZone a new cfa if you allocate heap space that could be deallocated in favor of a later heap request. See Part II.4.

---

hasColor **(-- bool)**

An import word from module env. Returns true if the Mac has color quickdraw.

---

hasFPU **(-- bool)**

Returns true if the Mac has a Floating Point Processor.

---

hasGestalt **(-- bool)**

Returns true if the Mac has the Gestalt Manager.

See also: gestalt

---

hash **(addr -- hashVal)**

Hashes the str255-format name string at addr into a 16-bit hash value. Names of selectors and named instance variables are hashed in Yerk to conserve memory.

```
Example:    " Macintosh"  \ The name.
            str255 -base      \ Address of string literal.
            hash . <return>                    \ Print 16-bit hash value.
            24481 0->
```

---

hc' **(--) : classname**

Import word from util module. Takes the classname in the stream and lists the class hierarchy in the front window.

Example: hc' window
OBJECT
GRAFPORT
WINDOW

See also: hier

---

heap> **(-- ^obj) : classname**

Heap> takes a classname out of the input stream and dynamically allocates an object of that class. Heap> returns a pointer to the object on the heap. If there was insufficient space to create the object, it returns a zero. The memory allocated is a non-relocatable block.

---

heapBot **(-- addr)**

Returns the lowest address in memory of the Yerk application heap.

---

heapTop **(-- addr)**

Returns the highest address in memory of the Yerk application heap.

---

here **(-- addr)**

Leaves the address of the next available dictionary location. Here is often used as the location for temporary string manipulation.

```
Example:   here            \ Next available dictionary location.
           begin-dp @      \ Beginning dictionary location.
           - . <return>    \ Print size of user dictionary.
           30210 0->       \ This is equivalent to flen.
```

---

hex **(--)**

Sets the value base, the number conversion radix, to 16. (Hex is short for hexadecimal.)

See also: .h $ base decimal

---

hfs? **(-- bool)**

Returns true if the disk volume that you booted from has a hierarchical file system. This is archaic, since Yerk may not even run on such old systems anymore.

---

hGetstate **(handle -- state)**

Gets this **handle**'s **state**. This word is used by class handle - you should normally use the getState: method of class handle.

See also: lock unlock hSetState

---

hier (--)

Import word from util module. Brings a dialog up to enter a classname, and lists the hierarchy of that class in the front window.

See also: hc'

---

hl-Evt (-- 0)

Event handling word for high level events. You supply handlers to the two vectors that this calls: AppleEvt and hlEvt.

See also: AppleEvt hlEvt

---

hld **(-- addr)**

This value is the address of the latest Hold character during numeric output conversion.

---

hlEvt **(--)**

Vector that you fill with the cfa of a high level event handler.

See also: hl-evt appleEvt

---

hold **(c --)**

Inserts a character into an ASCII string being built by an <# ... #> sequence.

Example:
```
2223243 0 <# # # # # $ 2d hold # # # #> type
\ 2D is hex for an ASCII dash '-'.
222-3243 0->
```

---

home **(--)**

Moves the cursor to the upper left-hand corner of the current grafport.

---

hSetState **(state handle --)**

Sets the state of the **handle** with **state.** This word is used by class handle - you should normally use the setState: method of class handle, having originally gotten the state by calling getState: method of handle.

See also: lock unlock hGetState

---

hwpAvail **(-- bool)**

Returns true if there the hwPriv trap is available on the Mac you're using.

---

i (-- n)

Copies the current contents if the innermost DO loop index onto the stack. Used within a DO ... LOOP structure. Note that i never equals the loop limit. So, if the loop limit is 10 and the initial value of the index is 0 (as in 10 0 DO ... LOOP), i will attain the values 0 through 9, inclusive.

Example: 16 0 DO i . LOOP \ Will print:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0->

---

i! (n --)

Stores **n** at the memory location pointed to by i, the innermost DO loop index.

Example:
```
\ This method fills an indexed object of 4-byte elements with the cfa of null, Yerk's "do-nothing" word.
:M CLEAR: limit: self 4* idxdata: self + \ upper limit
    ixaddr: self \ lower limit
    DO nullcfa i! 4
    +LOOP ;M
```

---

i+ (n -- n+i)

Adds i, the innermost DO loop index, to the number on the stack.

---

i- (n -- n-i)

Subtracts i, the innermost DO loop index, from the number on the stack.

---

i->l (16-bit-cell -- 32-bit-cell)

Converts the 16-bit half-cell on top of stack into a full 32-bit Yerk cell, extending the sign bit. i->l is useful in handling the result from Macintosh ROM routines that return a 16-bit signed integer on the stack. i->l differs from Extend in that i->l pushes two bytes onto the stack while extending the sign, whereas extend only extends the sign of a 16-bit integer contained in the 32-bit cell on top of stack, converting it into a 32-bit signed integer. Word0, a related word, pushes two zero bytes onto the stack.

See also: extend unpack makeint word0

---

i@ (-- n)

Fetches the number at the memory location pointed to by i, the innermost DO loop index.

Example:
```
\ This is a possible definition of type.
```

```
: type { addr  count \ temp -- sum }
        0 -> temp                   \ Initialize sum temporary.
        addr count +  addr          \ Loop index cycles through
        DO i@ ++> temp LOOP\  all memory to be summed.
        temp   ;                    \ Leave sum on top.
```

---

ibeamcurs (--)

Changes the operating cursor to the I-Beam cursor, used in editing.

See also: crosscurs pluscurs watchcurs cursor arrowcurs

---

ic! **(c --)**

Stores the character on top of stack at the memory location pointed to by i, the innermost DO loop index.

Example:
```
\ This is a possible definition of erase.
: erase { addr  count -- }
        addr count +  addr          \ Loop index cycles through
        DO  0 ic! LOOP   ;          \  all memory to be erased.
```

---

ic@ **(-- c)**

Fetches the character at the memory location pointed to by i, the innermost DO loop index.

Example:
```
\ This is a possible definition of type.
: type { addr  len -- }
        addr len +  addr            \ Loop index cycles through
        DO ic@ emit LOOP   ;                                    \
all memory to be typed.
```

---

id. **(nfa --)**

Prints a definition's name given its name field address on the stack.

Example:
```
 'c gotoxy        \ Cfa of gotoxy.
>name             \ Convert to nfa.
id. <return>      \ Print name.
GOTOXY 0->
```

See also: traverse .name

---

idxBase **Parameter Stack: (-- idxbase)**
Methods Stack: (^base -- ^base)

Leaves the base address of the indexed data of the object whose **^base**, pointer to the object's data, is on the methods stack.

Example:
```
\ This method is in class Object.
```

```
( -- addr ) ( Leave address of 0th indexed element. )
:M IXADDR: idxbase  ;M
```

---

if

Compile: (-- yerk)
Run: (bool --)

At compile time, IF reserves space at addr for a branch offset, leaving an address and a pairing value for error checking. At run time, if the boolean on top of stack is true, the words between IF and ELSE are executed; otherwise, the words between ELSE and THEN are executed. If there is no ELSE and the boolean is false, execution resumes at the first word following THEN.

Used in the form: IF ... THEN or
IF ... ELSE ... THEN

This is an immediate word.
See also: exit leave then else

---

imagename (-- ^obj)

A string object that holds the name of the current document you are working on.

---

immediate (--)

Marks the most recently defined word so that it is executed when encountered in a colon definition, instead of being compiled into it. Internally, marking a word as immediate means setting its precedence bit. [compile] can be used to force compilation of an immediate word.

Example:
```
: aWord ... ; immediate
\ aWord will be executed instead of being compiled
\ when it appears in a definition.
```

---

immediates (n --)

Same as immediate, but marks the last n defined words as immediate. Useful in defining several immediate import words.

---

import{ (--) : word1 word2 … wordn

Used in defining modules in the following way:

```
From modName import{ word1 word2 word3 }
```

See Part II.4 for a full description of defining modules.

---

in (-- offset)

This value is the byte offset within the text input buffer from which text is to be

accepted. IN is used and updated by word as it parses a word from the text input buffer and transfers it to Here as a str255-format string (0th byte is count byte).

---

inForeGround **(-- b)**

This value tells the user whether the application is in the foreground (true) or background (false).

See also: suspend resume mouseMoved cvtClip

---

inFront **(-- -1)**

This constant specifies to applicable Window Manager routines that the window in question is to be the front window on the desktop.

---

initFont (--)

Sets the current grafport's font, face, size, and mode. These are initialized as follows: font, Monaco (4); face, plain (0); size, 9; mode, srcCopy (0).

---

initProc (cfaProc --)

Loads the current A3 and A5 register values into the specified :Proc word.

See Also: initProcs :Proc

---

initProcs (--)

Loads the current A3 and A5 register values into the all :Proc words in the dictionary. You must execute this word in your startup word.

See Also: initProc :Proc

---

install (--)

Runs the install procedure, which is exactly the same as selecting install from the menu bar. See Part II.5 for a complete description.

---

interpret (--)

This vector invokes the outer text interpreter. By default, this vect executes (intrp).

---

intSwap (w1:w2 -- w2:w1)

Given a long word on the stack, this word will swap the high word with the low word.

---

isAppWindow (ptr -- ptr bool)

Inspects the windowKind of the window pointed to by ptr and returns a true flag if the window belongs to your application. It is used for processing events correctly if your application is processing in the background.

---

is{ (--)

Begins each case in a select{ conditional structure. Used in conjunction with }end. The select{ conditional structure is like the case structure, except that the indices must

start at 0 and must be contiguous. The advantage of using select{ is that it executes faster than case. See Select{ for example.

This is an immediate word.

---

j (-- n)

Copies the index of the next outer DO loop onto the stack. May be used only within a nested DO loop. Note that j never equals the outer loop's loop limit. So, if the loop limit is 10 and the initial value of the index is 0 (as in 10 0 DO ... LOOP), j will attain the values 0 through 9, inclusive.

```
Example:  ( -- )   \ Print multiplication tables from 2 to 10.
          : tables
          11 2                          \ Outer loop limits; index is j
          DO  11 1                      \ Inner loop limits; index is i
                 DO  i  j  *  .  space
                 LOOP                   \ Print element.
                 cr                     \ Carriage return after each table.
          LOOP  ;
```

---

key (-- chr)

Leaves the ASCII value of the next keystroke by executing keyvec. Key listens and handles other events, such as mouse-down and activate events, while waiting for a key-down event. Note that there is also a key defined in the nucleus, called (key), which only knows about key-down and mouse-down events. You should never need to use the nucleus (key).

```
Example:  \ The highest level code of an application could be:
          : runMyAppl
            BEGIN   key                 \ Listen to ALL events, returning
only
                                        \  after a key-down event.
                 key: [ frontWindow ]
                                        \ Send the character to the active Window
                                        \  object for processing by key: method.
            AGAIN  ;                    \ Do it again.  This is an infinite loop.
```

See also: rekey

---

key! **(--)**

Resets the keyboard to use key and not to do mouse processing.

See also: rekey

---

key-evt **(-- keyword modsword T or F)**

Handles key-down events for object fEvent. Leaves **keyword**, the msg instance variable of fEvent in which the lowest byte gives the ASCII code and the next higher

byte gives the key code, **modsword**, the mods instance variable of fEvent which gives the state of the mouse button and the modifier keys at the time of the event, and a true flag. The true flag signals to the Next: method of fEvent that a keystroke has been detected and so fEvent may stop listening to events and return control to the word that executed key. A false flag is left if the key event occurs on a window that is not part of your application (while using multiFinder). This word can be redefined through the use of :f or by substitution of vector 3 in fEvent.

See also: actv-evt disk-evt mouse-evt null-evt upd-evt app4-evt

---

keyVec (--)

This system vector is the execution vector for key. It contains the cfa of the word that gets a character from the keyboard. At system startup, keyvec is vectored to the "smart" key that listens to all events.

---

killHandle **(handle --)**

Calls the Memory Manager routine DisposHandle to release the block whose **handle** is on the stack. The space the block occupied now becomes part of the free space of the application heap. You should normally use the release: method of class Handle.

---

killPtr **(ptr --)**

Calls the Memory Manager routine DisposPtr to release the space occupied by the nonrelocatable block whose relative pointer is on the stack. The space the block occupied now becomes part of the free space of the application heap.

---

killTask **(cfa --)**

Removes the word whose **cfa** is on the stack from the task list which fEvent performs when it receives a null event. If the **cfa** is not one of those in the taskList issues message "task not found" and aborts. (You must load optional file "Tasks" to use this facility.)

Example: 'c dispClock killTask
\ Removes displaying clock from the task list.

See also: addtask dotasks tasklist

---

konstant **(-- n) : name**

An import word from the konstant module. Takes the name in the stream and returns the value associated with it as defined in Inside Mac.

See also: global

---

l->g **(Tpoint -- Tpoint')**

Converts a Toolbox point (higher 16 bits y, lower 16 bits x) in local coordinates of the current grafport into a Toolbox point in global coordinates.

See also: g->l

Land (n1 n2 -- b)

Leaves the logical and of **n1** and **n2**. Treats any non-zero **n** as boolean 1.

See also: and Lor Lxor

---

latest (-- nfa)

Leaves the name field address of the topmost word in the current vocabulary.

Example:
```
\ Compile a word into dictionary.
: newWord ... ;
latest id. <return>      \ Print its name string.
NEWWORD 0->
```

---

leave (--)

Forces termination of a DO loop at the next iteration by setting the loop limit equal to the current value of the index; the index itself is not tampered with. Execution proceeds normally until LOOP or +LOOP is encountered. Leave may appear within other control structures which are nested.

Example:
```
  : testLeave
10 0
DO i .    i 7 =                        \ Leave loop after index=7.
  IF leave THEN
LOOP   ;

testleave <return>
0 1 2 3 4 5 6 7 0->
```

---

lfa (pfa -- lfa)

Derives the link field address, **lfa**, from the parameter field address, **pfa**, of the word.

See also: >link trav

---

limit

Parameter Stack: **(-- limit)**
Method Stack: **(^base -- ^base)**

Leaves the number of indexed elements of the indexed object whose **^base**, pointer to the object's data, is on the methods stack.

Example:
```
\ This method is from class Object
( -- limit )
:M LIMIT:  ?ixObj                \ Is it an indexed object?
```

```
limit  ;M                    \ Get limit
```

---

line **(dh dv --)**

Draws a line on the screen from the current pen position for **dh** and **dv** pixels.

---

lineTo **(h v --)**

Draws a line on the screen from the current pen position to the location given by **h**, **v** .

---

link> **(lfa -- cfa)**

Derives the code field address, **cfa**, from the link field address, **lfa**, of the word.

See also: cfa

---

lit **(-- n)**

Pushes the (32-bit) long memory word at the next dictionary address onto the stack. Within a colon definition, lit is automatically compiled before each single-precision number greater than 65,535 encountered in the input text.

See also: wlit

---

literal **Compile: (n --)** **Run: (-- n)**

At compile time, compiles the number on the stack as a single-precision literal. This word is an immediate, and so executes within a colon definition. The run-time action of literal is provided by either lit or wlit.

Example:
```
: testLit1 <[ 1024 4 * ]> literal . ;
testlit1 <return>                    \ Print 4096.
4096  0->

\ The next definition is equivalent to testLit1.
: testLit2  4096 . ;
```

```
testLit2 <return>                    \ Print 4096.
4096  0->
```

This is an immediate word.
See also: dliteral lit wlit

---

ln **(fptr -- fptr)**

Computes ln(x) for the floating point number pointed to by **fptr**.

---

ln(10) **(-- fptr)**

This is a constant which returns a pointer to the value which corresponds to ln(10). This is useful for computing $\log_{10}$ values.

Example: : log ln log(10) f/;

---

ln1 **(fptr -- fptr)**

Computes ln(x+1) for the floating point number pointed to by **fptr**.

---

load-error **(-- addr)**

This system constant points to a memory location that contains an error number if an error occurred while loading a dictionary image at cold start. Use @ to get error number.

---

loadFile **(-- ^obj)**

A filelist object that allows file nesting up to 6 deep. See the manual section on Files.

---

loCase **(--)**

Sets the compiler to accept upper and lower case characters in words. This allows case sensitivity. The abort word will reset Yerk to uppercase mapping. loCase is used for filemarking, so it is not advisable to ever use this feature.

See also: upCase

---

lock **(handle -- ptr)**

Calls the Memory Manager routine hLock to lock the relocatable block whose **handle** is on the stack. Leaves a relative **pointer** to the locked (now unrelocatable) block. You will most likely use the lock: method of class Handle.

See also: unlock hGetState hSetState

---

locked **(--)**

Prevents a module from being *unlocked* upon completion. This is necessary if you have action vector words stored in a module which remain active after the export word finishes: if LOCKED is not used unpredictable results will occur. Use MUNLOCK when you are completely finished with all code in the module. (LOCKED must be coded somewhere between :MODULE and ;MODULE in your module.)

Example:

```
:Module FooMod
Locked
```

See also: mlock munlock ?mlock

---

log (fptr -- fptr)

Computes $\log_{10}(x)$ for the floating point number pointed to by **fptr**.

---

log2 (fptr -- fptr)

Computes $\log_2(x)$ for the floating point number pointed to by **fptr**.

---

log21 (fptr -- fptr)

Computes $\log_2(x+1)$ for the floating point number pointed to by **fptr**.

---

loop **Compile:** (yerk --) **Run:** (--)

At compile time, compiles (loop) and uses addr to compute an offset to DO; n is used for compiler error checking. At run time, DO begins repetitive execution of a sequence of words delimited by LOOP. 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 forward.

Example:
```
\ Print integers from 0 to 9.
: testDoLoop  10 0 DO i . LOOP    ;
testdoloop <return>
0 1 2 3 4 5 6 7 8 9 0->
```

This is an immediate word.
See also: +loop leave

Lor (n1 n2 -- bool)

Leaves the logical or of **n1** and **n2**. Treats any non-zero **n** as boolean 1.

See also: or Land Lxor

Lxor **(n1 n2 -- bool)**

Leaves the exclusive or of n1 and n2. Treats any non-zero n as boolean 1.

See also: or Land Lor

---

m! **Parameter Stack: (n --)**
Methods Stack: (addr -- addr)

Stores **n** at the long memory word specified by **addr** on the methods stack.

---

m* **(n1 n2 -- d)**

Multiplies **n1** and **n2** leaving their double-precision product on the stack. ("M" here refers to "mixed-precision.")

---

m/ **(d n -- rem quot)**

Divides **n** into **d**, a signed double number, leaving their single-precision remainder and quotient. ("M" here refers to "mixed-precision.")

---

m/mod **(ud1 u2 -- u3 ud4)**

Divides **u2**, an unsigned number, into **ud1**, an unsigned double number, leaving their unsigned single-precision remainder, **u3**, and unsigned double-precision quotient, **ud4**. ("M" here refers to "mixed-precision.")

---

m0 **"m-zero" (-- baseAddr)**

This value is the base address of the methods stack, and is the initial value of the methods stack pointer, the address of the top cell,. R0 is used by mp! to initialize the stack pointer.

```
Example:    m0 .h <return>                  \ Print base of methods stack.
            1561C 0->
```

---

m@ **Parameter Stack: (-- n)**
Methods Stack: (addr -- addr)

Fetches the number at the address on the methods stack and pushes it onto the parameter stack.

---

makeInt **(32-bit-num -- 16-bit-num)**

Drops the higher 16 bits of the number on top of the stack. This is handy in some Toolbox calls that require an Int value.

See also: pack i->l

---

marks (--)

Import word that lists all filemarks if +docs has been enabled.

See also: +docs -docs fm rl mforget /// see srcname findFmark

---

max (n1 n2 -- max(n1,n2))

Leaves the greater of two numbers.

Example: 334 45765 max . <return>
45765 0->

---

maxDict (**maxSize --)**

Sets the maximum allowable dictionary size; the size that the dictionary will actually be set to on a 512K Mac or larger. Extra space is allotted to the heap.

---

maxmem (**-- n)**

MaxMem compacts the current heap zone and purges all purgeable blocks from the zone. It returns as its result the size in bytes of the largest contiguous free block in the zone after the compaction.

See also: .room

---

mdepth (**-- n)**

Leaves the number of cells pushed on the methods stack. You may find mdepth useful in debugging your programs.

---

menubar (**-- addr)**

Returns the address of the system object representing the menubar. This is an object of class mbar.

---

menuId (**-- n)**

Leaves the menu ID number of the last clicked menu.

See also: theMenu mItem

---

mfa (**^class -- ifa)**

Leaves the methods field address of a class whose **^class**, its parameter field address, is on the stack. The methods field of the dictionary entry of a class contains the pointer to the last entry in the class's methods dictionary.

```
Example:        ' Window     \ ^class (pfa) of class Window.
                mfa          \ Ifa of Window.
                dup @        \ Address of methhash of last method.
                swap .h cr .h <return>
                BA3A         \ Mfa
                BAFA 0->             \ Address of last method.
```

---

mforget **(--) : filename**

Import word that forgets all words up to the named filename. +docs must have been enabled.

See also: +docs -docs see fm rl /// srcName findFmark

MFSname **(addr len -- addr' len')**

Converts the given filename to its equivalent MFS filename by stripping out path specifications. This can be used wherever you need the simple filename from the fully qualified path.

---

min **(n1 n2 -- min(n1,n2))**

Leaves the lesser of two numbers.

Example: 334 45765 min . <return>
334 0->

---

mitem **(-- n)**

Global value holding the item number of the last menu click.

See also: theMenu menuID

---

mlock **(cfa --)**

Locks a module that has been loaded into memory, so that it stays there, even if it is not being executed.

Example: grepmod \ Loads the module into memory
'c grepmod mlock \ Locks the module in place.

See also: munlock ?mlock

---

mod **(n1 n2 -- rem)**

Leaves the remainder of the division of **n1** by **n2**, n1/n2.

Example:
```
23 3 mod . <return>      \ 23/3 = 7 + 2/3
2 0->                    \  remainder is 2.
-23 3 mod . <return>
2 0->
23 -3 mod . <return>
-2 0->
-23 -3 mod . <return>
-2 0->
```

See also: /mod

modCode (-- addr)

This constant returns the address of the code executed by a module reference.

See also: colCode fvalCode valCode vectCode

module (--) : modName

The special compiler for modules. Takes a module name from the input stream, compiles it twice to determine what addresses may need to be relocated at load time, and saves it in a binary file.

See also: :module from import{

mouse-evt (-- 0)

Handles mouse-down events for object fEvent. Mouse-evt updates the Mouse object theMouse, and finds the window and the region of the window in which the mouse-down occurred. Then executes the clause of a select{ ... }select conditional structure that matches the region ID, an integer in the range 0..8. The region ID's and the action taken for each are:

ID	region	executes
0	in	desk, null case desk
1	in	menu bar click: method of menuBar
2	in	system window sys
3	in	content region content: method of window, the "meat" of the application
4	in	drag region drag: method of window
5	in	grow region grow: method of window
6	in	go-away region close: method of window if there

	is a valid click	
7	in	zoomBox
	doZoomWind	
8	in	zoomBox
	doZoomWind	

Finally, mouse-evt leaves a zero on the stack for the next: method of fEvent, to signify that a mouse-down is not a key event. This word can be redefined through the use of :f or by substitution of vector 1 in fEvent.

See also: actv-evt disk-evt key-evt null-evt upd-evt app4-evt

---

mouseMoved (--)

This is a vector defaulted to nullcfa. The user may put in his own response to a multifinder mouseMoved event.

See also: suspend resume cvtClip inForeGround

---

moveHi **(handle --)**

Moves the data referred to by **handle** to high memory. This is always called by the lock: method of class handle.

See also: lock unlock

---

mp! (--)

Initializes the methods stack pointer, the address of the top cell, with the contents of value m0, the base address of the methods stack. mp!, in effect, clears the methods stack.

---

mp0...5 (-- parm0...5)

Pushes a copy of the n+3rd methods stack cell (counting the top cell as the first) onto the parameter stack. mpN is used in handling named parameters and local variables. You should have little reason to use it, unless you really want to juggle numbers on the methods and parameter stacks.

---

mp@ (-- addr)

Leaves the current contents of the methods stack pointer, the address of the top cell, on the parameter stack.

---

ms0...5 (n --)

Puts the number on the parameter stack into the n+3rd methods stack cell (counting the top cell as the first). msN is used in handling named parameters and local variables. You should have little reason to use it unless you really want to juggle numbers on the methods and parameter stacks.

---

mSelect (point -- item# menuID)

Calls the menu manager to track a menu selection. Used primarily by the Click: method of class mBar.

---

msg# (--) : resID

At compile time takes the **resID** out of the input stream and compiles a literal

and an error-handling primitive into the current word. At run time prints the string resource associated with the given **resID** as an advisory message. Does not execute an abort.

This is an immediate word.

---

msize **(-- addr)**

This system constant points to the number of bytes allocated in the Yerk dictionary. Use @ to get #bytes.

---

munlock **(modpfa --)**

Unlocks a module that has been locked into memory.

See also: mlock ?mlock

---

mw! **Parameter Stack: (n --)** **Methods Stack: (addr -- addr)**

Stores the low-order 16-bits of the number on the parameter stack into the memory word at the **addr** on the methods stack.

---

mw@ **Parameter Stack: (-- n)** **Methods Stack: (addr -- addr)**

Fetches the (16-bit) memory word at the address on the methods stack and pushes it onto the parameter stack.

---

myCtl **(-- ^ctl)**

When used inside of a control action handler, this word returns a pointer to the actual control record. For an example of its usage, see Part III.7. This word is not loaded into the pre-compiled system and is defined in the file "Ctl".

---

myDoc **(-- addr len)**

Leaves an addr-len pair representing the string that is the name of the start-up document.

Example: myDoc type
yerk.com 0->

---

n>count **(nfa -- addr len)**

Leaves the address and length of a name field, suitable for use by "type".

---

name> **(nfa -- cfa)**

Derives the code field address, **cfa**, from the name field address, **nfa**, of the word.

See also: cfa

---

need (n --)

Releases modules until n bytes are available.

See also: release purge ovblock

---

negate **(n -- -n)**

Leaves the two's complement of a number. Negate changes the sign of a number.

Example: 35 negate . <return> \ Print -35.
-35 0->

See also: dnegate

---

nevent **(--)**

Vector called inside of interpret: file. During recompile of Yerk, it is set to a simpleaction, but after recompiling Yerk, it is automatically set to (nevent). (nevent) allowscompiling to occur in the background which is not allowed while recompiling Yerk.

---

newHandle **(n -- handle)**

Calls the Memory Manager routine NewHandle to allocate a new relocatable block of **n** bytes from the application heap and leaves a **handle** to the block. The **handle** is the absolute address of the master pointer to the block; or zero if the call fails. The block is initially marked unlocked and unpurgeable.

See also: >ptr

---

newPtr **(n -- ptr)**

Calls the Memory Manager routine NewPtr to allocate a new nonrelocatable block of **n** bytes from the application heap and leaves a relative pointer to the block. The value of the **ptr** will be negative if the call fails.

---

next, **(--)**

Compiles the value of next at the end of a code definition into the dictionary. Useful in defining code words. See Part II.4.

This is an immediate word.
See also: popD0 popA0 pushD0 pushA0

---

nfa **(pfa -- nfa)**

Derives the name field address, **nfa**, from the parameter field address, **pfa**, of the word.

See also: >name

---

nmenu **(--)**

Reads the yerk menu from the yerk.rsrc file and draws the new menubar.

---

not **(bool1 -- bool2)**

Examines a boolean on top of stack and leaves a true flag if it is false; otherwise, leaves a false flag. Not is identical to 0=.

Example: -45 not . <return> \ -45, being non-zero, is a boolean true.
0 0-> \ 0 is a boolean false.

---

npath **(--)**

Reads the pathlist file 'npath.txt' from the yerk folder.

See also: .path

---

null **(--)**

Does absolutely nothing. Null is Yerk's "do-nothing" word, equivalent to a no-operation in other languages. You can plug nullcfa, a predefined constant containing the cfa of null, into an element of an object that needs cfa's of handler words, if you want nothing to be done by a particular handler.

See also: nullcfa

---

null-evt **(-- 0)**

Handles null events for object fEvent, *while the window is enabled*, by calling the operating system's routine systemTask.

See also: actv-evt disk-evt key-evt mouse-evt upd-evt tasklist

---

nullCfa **(-- cfa:null)**

This constant is the **cfa** of null, Yerk's "do-nothing" word. You can plug nullcfa into an element of an object that needs cfa's of handler words, if you want nothing to be done by a particular handler. In fact, objects of classes X-array,

Menu, Window, and others when created "wake up" with many of their indexed elements filled with nullcfa. This is so that if, for example, a menu selection transfers control to an element of the Menu object which has not been initialized, you won't end with a system error box on your screen.

Example:
```
\ This is the classinit: ("wake-up") method for X-array.
:M  CLASSINIT:  limit 0        \ Stuff all elements
        DO nullcfa i to: self LOOP                    \
with nullcfa.
;M
```

See also: dropcfa

---

nullOSstr (-- addr)

Leaves a pointer to a str255-format string of length 0.

---

nullVal (-- 0)

Leaves a zero on the stack. NullVal is an optional word in file "Ctl".

---

number (addr -- d)

Converts a character string at **addr**+1 to a double precision number; the byte at **addr** contains the string's length but is ignored. Conversion proceeds until a blank is encountered - the string of digits must end with a blank - any character which is not a digit, decimal point or minus sign will cause an error. The position of the last decimal point encountered (if any) is left in DPL. If conversion fails, an error message is issued. If number is difficult to use in your application try the more general (number) or ?num.

See also: digit dpl (number) @val ?num

---

objInit (--)

This system vector is executed when Yerk starts up. It is initialized with the word YERK, which loads the menu bar and initializes Yerk's internal data.

---

objLen

Parameter Stack: **(-- len)**
Method Stack: **(addr -- addr)**

Returns the data length of the object whose address is on the methods stack. Used as the primitive of the Length: method of class object.

---

of

Match: **(n1 n2 --)**
No match: **(n1 n2 -- n1)**

Begins a clause in a case conditional structure. OF compares the case index, **n1**, with the OF index, **n2**. If they are equal, OF drops both and the words between the OF and the corresponding ENDOF are executed. If they are not

equal, OF drops the OF index and execution continues after the corresponding ENDOF.

This is an immediate word.
See also: case

---

ok (--)

Prints the Yerk prompt, commonly 0->. The "0" means there are no values on the stack, the "-" means the current base is decimal, and the ">" is always present. The "-" changes to a "$" if the base is hexadecimal, and to "?" if the base is anything else.

Example: 345 -65765 hex ok <return>
2> 2>
\ There are two numbers on the stack, and the base is hex.
\ The first prompt is printed by the ok you typed in, the second by the
\ ok in quit

---

opennr (--)

Opens Yerk's resource file "yerk.rsrc" as the current resource file.

---

openResFile (addr len --)

Opens the file indicated by the addr-len pair as the current resource file.

---

or (n1 n2 -- n3)

Leaves the bit-wise or of **n1** and **n2** as **n3**. Or works as a logical or if you want to use **n1**, **n2**, and **n3** as booleans (non-zero = true; zero = false).

Example: 9 4 or . <return> \ Or 12 (1001) with (0100).
13 0-> \ Result: 13 (1101).

true false or . <return> \ Or two booleans.
1 0-> \ Result: 1 (nonzero = true).

See also: xor and Lor

---

os-evt (-- 0)

This word is called whenever a multifinder type event occurs (suspend or resume). The two vectors that will be executed upon a suspend or resume event are 'suspend' and 'resume'. You may set these vectors in your own application. They default to null. There is also a 'mouseMoved' vector that will execute if the mouse was detected to have moved. The clipboard may be converted with the cvtClip vector. See Inside Multifinder. This word can be redefined through the use of :f or by substitution of vector 16 in fEvent.

See also: suspend resume cvtclip mousemoved

---

out (-- #chars)

This value is incremented by emit, type, space, and spaces after each character transmitted. You can change and examine OUT to control formatting of output.

```
Example:    0 -> out                        \ Initialize out to zero.
            ." All's well in the bit mines." cr out . <return>
            All's well in the bit mines.
            32 0->                          \ 32 characters output since out was zeroed.
```

---

ovblock **(size -- ptr)**

Ovblock attempts to get a non-relocatable block of heap of the **size** requested. It will compact memory, purge heap objects marked purgeable, and execute the vector GrowZone, if necessary to get the requested space. If it was unable to get the requested space it signals an error #121.

See also: need

---

over **(n1 n2 -- n1 n2 n1)**

Copies the second number on the stack to the top of stack.

Example: 17 2 over . . . <return>
17 2 17 0->

See also: 2over pick

---

pack **(x y -- x:y)**

Packs two 32-bit numbers into one 32-bit number. Only the lower 16 bits of x and y are used. You can use pack to convert a coordinate point into a Toolbox-compatible point.

See also: makeint unpack

---

pad **(-- addr)**

Leaves the address of the text output buffer, a temporary area of memory of 256 bytes. Yerk uses PAD to hold information (text, numbers) for intermediate processing. You may use it to the same end, but be aware that output words like emit and type may interfere by using pad as well.

---

padBL **(addr len --)**

Pads a str255-format string with blanks and adjusts its count byte accordingly. **Addr** is the address of the 0th byte, the count byte, of the string and **len** is its desired blank-padded length.

Example:

```
0->" Bit-mining stocks rose today."    \ Put string at buf255.
0->2drop                               \ Drop the addr and len.
0->buf255 50 padbl                     \ Pad with blanks to 50 chars.
0->buf255 count type <return>          \ Convert to addr-len format and
                                       type it.
Bit-mining stocks rose today.bbbbbbbbbbbbbbbbbbbb  0->
                                       \ The b's mean blanks.
```

See also: blanks -trailing

---

paramText (addr0 len0 addr1 len1 addr2 len2 addr3 len3 --)

Sets global text-substitution strings for Dialogs through a call to ParamText. Thereafter, dialogs will automatically substitute strings 0 through 3 for the character sequences "^0" through "^3". This word is optional and is included in the file "Dialog".

---

parmStr (-- ^obj)

A string object used for various tasks.

---

parse **(addr chr -- addrnext addr len)**

Parses text starting at **addr** scanning for the **char** producing the address for the next parse and the string parsed. Will not parse past an ascii null character, treating it as a unconditional delimeter.

See also: enclose

---

pat **(--)**

Displays the palette of system pen patterns.

---

patch **(--) : oldWord newWord**

Patch changes all references to oldWord to references to newWord. So if a word had oldWord in its definition, now newWord will be executed instead of oldWord. Patch only works if both oldWord and newWord are colon definitions (words compiled with : and ;) and DO NOT have local variables or named input parameters.

Example: patch doGraph doGraph2
\ This changes all earlier references to doGraph to the revised word
\ doGraph2.

This is an immediate word.

---

path **(-- ^obj)**

This value contains the sArray object being used by the OPEN: method fo finding files on an HFS system.

---

pcr **(--)**

Sends a carriage return and a line feed to the printer.

---

pcrvec **(--)**

This system vector is the execution vector for pcr. When Yerk starts up, pcrvec is vectored to null, Yerk's "do-nothing" word. +print vectors pcrvec to pcr so that you or Yerk can send a carriage return and linefeed to the printer.

---

pemit **(chr --)**

Sends a character to the printer.

Example:: ascii Q pemit \ Print a 'Q' on the printer.

---

pemitvec **(--)**

This system vector is the execution vector for pemit. When Yerk starts up, pemitvec is vectored to drop. +print vectors pemitvec to pemit so that you or Yerk can send a character to the printer.

---

pfa **(nfa -- pfa)**

Derives the parameter field address, **pfa**, from the name field address, **nfa**, of a word.

See also: >body

---

pi **(-- fptr)**

This is a constant which returns a pointer the the floating point value for π.

---

pick **(n -- val)**

Copies the **n**-th stack cell (not counting **n**) onto the top of the stack. The top of stack is the first stack cell. Thus 1 pick is equivalent to dup, and 2 pick is equivalent to over. Pick is useful for references beyond the second item.

Example: 49 14 78 3 pick .s \ Put three numbers on stack,
\ pick the third cell, and print
\ contents of stacks.
Parameter Stack:

```
49  $    31
78  $    4E
14  $     E
49  $    31
Return Stack:
16286  $   3F9E
16744  $   4168
Methods Stack:(--Empty Stack--)
4->
```

---

pluscurs (--)

Changes the cursor to the predefined cursor type plus, used in spreadsheets.

See also: crosscurs ibeamcurs watchcurs cursor arrowcurs

---

popA0 (--)

Compiles the hexcode to move a value from the data stack to register A0. For a fuller explanation of the use of this word and the next three, see Part III.4.

This is an immediate word.
See also: pushA0 create next,

---

popA1 (--)

Compiles the hexcode to move a value from the data stack to register A1. This word is in the optional file "pops".

This is an immediate word.
See also: pushA1 create next,

---

popD0 (--)

Compiles the hexcode to move a value from the data stack to register D0.

This is an immediate word.
See also: pushD0 create next,

---

popD1 (--)

Compiles the hexcode to move a value from the data stack to register D1. This word is defined in the optional file "pops".

This is an immediate word.
See also: pushD1 create next,

---

popm **Parameter Stack: (-- n)** **Methods Stack: (n --)**

Pops the number on the methods stack and pushes it onto the parameter stack.

See also: pushm addm copym exgm dropm dupm

---

popPort **(portAddr --)**

Pops the **portAddr** on the stack to restore it as the current grafPort.

See also: pushPort

---

prefix **(value --) : name**

This word is used in the definition of multiple cfa words to define a prefix that will call into effect one of the pre-defined behaviors for a mCFA word. The name taken from the input stream is the prefix to be executed, and the value taken from the stack indicates which cfa is to be executed. Note: the 0CFA is the default behavior for a mCFA word. See the example in Part II.4.

See also: codefields do.. ..End build

---

ptype **(addr len --)**

Prints an (**addr len** format) string on the printer.

---

ptypevec **(--)**

This is the execution vector for ptype. When Yerk starts up, typevec is vectored to 2drop, so that it does nothing. +print vectors ptype to ptypevec, so you can send a string to the printer.

---

purge **(--)**

Purges all unlocked yerk modules from the application heap.

---

pushA0 **(--)**

This word compiles in the code to move a value from register A0 to the data stack. This word is only useful inside of a code word definition. For a fuller explanation of the use of this word and the next three, see the description in Part II.4.

This is an immediate word.
See also: popA0 create next,

---

pushA1 **(--)**

This word compiles in the code to move a value from register A1 to the data stack. This word is an optional word, defined in the file "pops".

This is an immediate word.
See also: popA1 create next,

---

pushD0 (--)

This word compiles in the code to move a value from register D0 to the data stack.

This is an immediate word.
See also: popD0 create next,

---

pushD1 (--)

This word compiles in the code to move a value from register D1 to the data stack. This word is an optional word, defined in the file "pops".

This is an immediate word.
See also: popD1 create next,

---

pushm **Parameter Stack: (n --)** **Methods Stack: (-- n)**

Pops the number on the parameter stack and pushes it onto the methods stack.

See also: popm addm copym exgm dropm dupm

---

pushPort (-- **portAddr**)

Pushes the **portAddr** onto the stack for restoration later.

See also: popPort

---

query (--)

Inputs up to 128 characters from the keyboard until a carriage is typed. Query puts the acquired text at the address TIB and sets the value IN to zero.

---

quit (--)

Returns control to the Yerk interpreter or an installed application. Quit executes quitvec, which must be vectored to the highest level word of an installed application. If an application is not installed, quit clears the return stack, sets state to 0 (stops interpreting), and prints the Yerk prompt, usually 0->. Quit is the highest level word in Yerk.

See also: abort

---

quitvec (--)

This system vector is the execution vector for quit. If quit is executed and quitvec is vectored to null, the default word, control is returned to the keyboard, providing access to the Yerk interpreter. If quit is executed and quitvec is vectored to an application-defined word, control is transferred to that word, sealing the user from the Yerk interpreter. The word quitvec is vectored to is usually the highest level word of the application which starts the application. Install, the word that you use to prepare a final application, assumes that the contents of quitvec and abortvec have been vectored to the highest level word and the error handler of the application.

Example: 'c myAppl -> quitvec

---

r **(-- n)**

Copies the top number on the return stack onto the parameter stack. You should use the return stack with great care. Use named input parameters and local variables instead; they are easier to use and a lot friendlier.

---

r0 **"r-zero" (-- baseAddr)**

This value is the base address of the return stack, and is the initial value of the return stack pointer, the address of the top cell. R0 is used by rp! to initialize the stack pointer.

Example: r0 .h <return> \ Print base of return stack.
1516C 0->

---

r> **"r-from" (-- n)**

Pops the number on top of the return stack and pushes it onto the parameter stack. You should use the return stack with great care. Use named input parameters and local variables instead; they are easier to use and a lot friendlier.

---

rad2deg **(fptr -- fptr)**

Converts the floating point number pointed to by **fptr** from radians to degrees.

---

radioID **(-- 2)**

Returns a system constant to indicate that a given control is a radio button. Not defined in the pre-built system, this is an optional word in the file "ctl".

See also: buttonID checkID vslD

---

random **(n -- rand#)**

Generates a pseudo-random number in the range 0..**n**-1 . **n** should be less than or equal to 65535.

```
Example:  ( -- )                    \ Print a random number in range 0..99 .
          : rand100  100 random .  ;
          rand100 rand100 rand100 <return>
          56 23 78 0->
```

---

rangeof

Compile: **(--)**
Run: **(val lo hi --)**

Used in the same way as OF in a CASE statement, this word provides conditional execution if **val** is within the range **lo** to **hi**.

This is an immediate word.

---

rdepth (-- n)

Leaves the number of cells occupied on the return stack. You may find rdepth useful in debugging your programs.

---

readFP (--)

Reads a fresh copy of the floating point code from yerk.rsrc, and installs it into yerkFP.com or any of its clones. This is necessary because of how Apple decided to 'improve' floating point performance in system 7.0.1 and 7.1. It changes your code!! So, if you save a snapshot of your code and relaunch, things don't work. To work around this, whenever you save a snapshot of your development work, yerk automatically executes 'readFP' and restores a pristine floating point code before writing todisk. You should never have to execute this word.

---

recoverHndl (ptr -- hndl)

Calls the toolkit routine to return the handle to the given pointer. No error is returned. So you have to be careful with this one. The valid: method for class handle uses this call as well as other tests to try to determine if a given handle is known to the system.

See also: ?ishandle

---

rekey (--)

Once this word is executed, all events are handled by the Yerk system. This word is executed by the system vector objinit on startup.

See also: key!

---

release (--)

The cfa of this word is what is normally kept by the vector GrowZone. Release disposes of the heap memory allocated to Yerk system modules. If you change the contents of GrowZone, whatever word you install in the vector should call release to dispose of the Yerk modules, in case the space they

occupy on the heap is needed.

---

repeat

Compile: **(yerk --)**
Run: **(--)**

At run time, repeat forces an unconditional branch to the first word after begin.

Example:
```
: aWord   { x y \ done? -- result }
          ...       \ Initialization code.
          BEGIN
                    done?                   \
          Done is a boolean.
          WHILE \ Execute words up to repeat
                    firstWord               \
          if done? is true.
                    secondWord
                    …
```

```
                                        REPEAT ;                        \
                                        Branch to word after begin.
```

This is an immediate word.

---

reserve (n --)

Allocates and erases **n** bytes in dictionary at here.

Example: 200 reserve \ Reserve 200 bytes at here.

See also: allot

---

reserve**Mem** (n --)

ReserveMem creates free space for a block of **n** contiguous bytes at the lowest possible position in the current heap zone. It will try every available means to place the block as close as possible to the bottom of the zone, including moving other blocks upward, expanding the zone, or purging blocks from it. Note that ResrvMem doesn't actually allocate the block.

See also: moveHi lock unlock

---

reset (--)

Causes the Mac to reboot. Executing this word has exactly the same effect as pressing the RESET button on the side of the machine.

---

restPort (--)

This word restores the previously saved grafport, so that screen output is again written to that grafport. The inverse of this word is SavePort. These words only have a single location to save the port in, so calling SavePort twice, followed by RestPort twice, will not restore the original port. It is best to use pushPort and popPort.

See also: savePort pushPort popPort

---

resume (--)

This is a vector defaulted to nullcfa. The user may put in his own response to a multifinder resume event.

See also: suspend mouseMoved cvtClip inForeGround

---

returnToModal (--)

Call this word after handling an enabled item in a modal dialog.

---

rl (--)

Import word that forgets all words up to the last mark and reloads the last file. +docs must have been enabled.

See also: +docs -docs see fm mforget /// srcName findFmark

---

rndWind **(-- 16)**

This constant is the resource ID of a document window with round corners.

See also: docWind dlgWind

---

room **(-- room)**

Leaves the number of bytes left in the dictionary. You can run the Install utility to change this.

```
Example:  room .    \ Print the number of bytes left.
          16543 0->       \ On a Mac II, this number will
                  \  be MUCH larger.
```

---

rot **(rhymes with boat)** **(n1 n2 n3 -- n2 n3 n1)**

Rotates "left" the top three stack cells, so that the third cell becomes the first.

See also: pick

---

round **(fptr -- fptr)**

Rounds the floating point number pointed to by **fptr** and returns the result.

---

rp! **(--)**

Initializes the return stack pointer, the address of the top cell, with the contents of value r0, the base address of the return stack. rp!, in effect, clears the return stack.

---

rp@ **(-- addr)**

Leaves the current contents of the return stack pointer, the address of the top cell, on the parameter stack.

---

rsrc **(-- ptr) : name**

Rsrc is an mCFA word with a single cfa.

Example: 'type STR 113 rsrc joe

Later referencing joe produces a relative pointer to the string on the heap. Two cautions in the use of rsrc, 1) the resource may by relocated and your pointer no longer valid if you do anything that disturbs the heap between accessing the resource and using it, 2) you must ascertain that the appropriate resource file is open.

Example: opennr
joe count type
prefix token not found 0->

See also: spList

---

s, (toggle-mask -- latest)

Takes the text string at here, Exclusive-ORs the byte at here with **toggle-mask**, and compiles the string into the dictionary. Returns the new 'latest' (see latest).

---

s->d (n -- d)

Sign-extends a single-precision number to form a double-precision number.

```
Example: -221              \ -221 as a 32-bit signed integer.
         s->d      \ Sign-extend to 64-bit signed integer.
         2dup hex u. u. cr          \ Print double number in hex.
         decimal d. <return>        \ Print double number in decimal.
         FFFFFFFF FFFFFF23    \ -221 as double number in hex.
         -221 0-> \ -221 as double number in decimal.
```

---

s0 "s-zero" (-- baseAddr)

This value is the base address of the parameter stack, and is the initial value of the parameter stack pointer, the address of the top cell,. S0 is used by sp! to initialize the stack pointer.

```
Example: s0 .h <return>      \ Print base of parameter stack.
         14FDC 0->
```

---

s= (addr1 len1 addr2 len2 -- bool)

Compares two (addr len format) strings and leaves a true flag if they are equal; otherwise, leaves a false flag.

```
Example: scon str1 "Simon"    \ Define two string constants.
         scon str2 "Samson"
```

```
str1 str2 s= . <return>          \ Are they equal?
0 0->                \ 0 means false.
```

See also: (s=) $=

---

save (--)

Saves to disk the current image of the user dictionary, the part of the dictionary above the nucleus. You can find the saved image in the yerk folder.

Example: save yerk.com \ Write to disk an image of current user dictionary
\ as yerk.com.

See also: saver stdsave

---

saveNuc (--)

This is the equivalent of a SAVE for the nucleus; writes the Yerk nucleus out to disk. Use this after you have made a patch to the nucleus. This word is defined in the Install Module. *Be certain you have a backup of the nucleus before using* ***saveNuc.***

---

saveport (--)

Calls the QuickDraw routine GetPort to get a pointer to the current grafport's record on the heap into the variable thePort. With savePort you can save the current status of a grafport, and then restore it later with restPort. It is best to use popPort and pushPort.

See also: restPort pushPort popPort

---

saver (--)

Saves to disk the current image of the user dictionary, the part of the dictionary above the nucleus. You can find the saved image in the same folder as the document you launched (not necessarily the yerk folder). Gets the name from 'imagename'. Useful for quick saves while building up an image.

See also: save stdsave

---

saveSig (-- YERK)

Leaves the signature of the current application (whatever the nucleus has as an owner signature); (four ASCII chars in one stack element). If you change the ownership of the Yerk nucleus, 'saveSig' will tell you what it is.

---

saveType (-- COM)

Leaves the file type of a saved Yerk dictionary; (four ASCII chars in one stack

element).

---

sCon (--) : name "textString"

Defines string constants that return their address and length when executed. Be sure to leave only one space between the end of the scon's name and the first quote, or a null string will result. Do not use an extra space between the opening quote and your textString. Use scon if you need a string more than once; use string literals, instead, if you only need a string once.

Example:
```
scon str1 " Federation of "
scon str2 " Bit-miners Unions."
str1 type str2 type <return>
Federation of Bit-miners Unions.
```

See also: ascii

---

sCreate **(addr len --)**

An interface to Create; takes an **addr-len** pair representing a string. In the following example sCreate creates a dummy word for reference, then compiles the source file selected by the user. Subsequently, Forget task would clear the program from memory. (Module compilation works this way.)

Example:
```
: load new: loadfile " TASK" sCreate
txType 1 stdGet: topfile interpret: topfile
close: topfile drop remove: loadfile ;
```

ScreenBits **(-- l t r b)**

Leaves the dimension coordinates of your computer's main display. This is useful in targeting your software for various screen sizes.

---

scroll **(dh dv --)**

Moves the image in the current grafport **dh** pixels horizontally and **dv** pixels vertically. Scroll is chiefly used by cr and will not work under all circumstances.

---

see **(--) : wordname**

Import word that will bring up a scrollable window to display the named word if +docs has been enabled and was enabled when the dictionary was created. All dictionary words except module words are support for now. The scrollable window is not yet editable, but will be in the future.

See also: +docs -docs fm rl mforget /// srcName findFmark

---

select{ **(n --)**

Begins a select{ ... }select conditional structure. The select{ conditional structure is like the case structure, except that the indexes must start at 0 and

must be contiguous. The advantage of using select{ is that it executes faster and compiles smaller code than case, and for this reason you should use it whenever applicable. Cannot be used within method definitions.

Example:
```
( n -- ) \ Print zero, one, two, or none of the above.
: testSelect
        Select{
        0 is{ ." Zero" }end
        1 is{ ." One"  }end
        2 is{ ." Two"  }end
        Default{ ." None of the above"
}select  ;
```

This is an immediate word

---

set-ctl-obj **(^obj ctlHandle --)**

Puts **^obj**, the relative pointer to a control object, into the control record whose handle is **ctlHandle**.

---

set-file **(fcb --)**

Puts the absolute address of a filename into the ioNamePtr field of the file control block at address **fcb**.

---

setHsize **(handle size --)**

Sets this **handle**'s **size**. This word is used by class handle - you should normally use the setSize: method of class handle.

See also: getHsize

---

setName **(fcb --) : " filename"**

Sets the filename of the **fcb** to the quoted string after setname in the input stream.

Example: myFcb setName " Helter Skelter"

---

sFind **Successful: (addr len -- pfa len T)**
Unsuccessful: (addr len -- F)

An interface to the primitive (find) that takes an **addr-len** pair representing a string. SFind does not map your string to upper case, so it will not find lower case versions of your word.

Example: " HERE" sfind . . .
1 132 6982 0->

See also: (find)

---

shift? **(-- bool)**

Returns true if the shift key is held down when the word is executed.

See also: optkon? command? ctl?

---

sign **(sign d -- d)**

Sign is normally used within a pictured numeric output expression to place a sign immediately to the left of a converted numeric character string.

Example: -1 45 0 <# #s sign #> type <return>
-45 0->

See also: <# # #s #> hold

---

sin (fptr -- fptr)

Computes sin(x) of the floating point number pointed to by **fptr**.

---

smudge (--)

Makes the current word being defined unfindable. This is to prevent reentrancy.

Example: : bye smudge kill: iwin drop bye ; \ redefine 'bye' to kill the port
\ and call the older bye.

---

sort (ixAddr #elem compCFA --)

Sorts a list of 4 byte elements. **ixAddr** is the base address of the 0th element of the list object. **#elem** is the number of items in the list to be sorted. **compCFA** is the CFA of the comparison word for use in the sort. The stack comment for the comparison word must be (val1 val2 -- result). The word must take two 4-byte values, val1 and val2, and return the following result:

-1 -- val1 < val2
0 -- val1 = val2
1 -- val1 > val2

Example: : varComp -dup IF dup abs / THEN ;

```
: varSort { theObj compWord -- }
        theObj ?IsObj not Abort" argument is not an object."
        ixaddr: theObj limit: theObj 'c varComp
        sort ;   \ do it !
```

By designing the comparison word appropriately, data of an arbitrary kind can be sorted. To sort strings, for example, each element in the list object should be a pointer to a string; accordingly the comparison word would be based on

$=. Sort is provided as a module. See Part II.6 - Utility Modules.

See also: $=

---

sp! **(--)**

Initializes the parameter stack pointer, the address of the top cell, with the contents of value s0, the base address of the return stack. sp!, in effect, clears the parameter stack.

see also: mp! rp!

---

sp@ **(-- addr)**

Leaves the current contents of the parameter stack pointer, the address of the top cell, on the parameter stack.

See also: mp@ rp@

---

space (--)

Emits an ASCII blank character.

---

spaces (n --)

Emits **n** ASCII blank characters.

---

spList (-- ptr)

Leaves a pointer to the system's pattern list resource.

See also: syspat rsrc

---

sqrt (fptr -- fptr)

Computes √x of the floating point number pointed to by **fptr**.

---

srcCopy (-- 0)

Constant for QuickDraw srcCopy drawing mode.

---

srcName **Successful: (addr -- addr len)** **Unsuccessful: abort**

Import word that returns the filename **addr len** above the input **addr** in the dictionary if +docs has been enabled. If no source name is found, an abort is executed.

See also: +docs -docs fm rl mforget /// see findFmark

---

srcOr (-- 1)

Constant for QuickDraw srcOr drawing mode.

---

srcXor **(-- 2)**

Constant for QuickDraw srcXor drawing mode.

---

state **(-- compile-state)**

This value is the compilation state of the Yerk interpreter. A non-zero value means that the definition of a word, method, or class is being compiled.

See also: cstate ?comp

---

stdload (--)

Defined in source FrontEnd, this word will bring up a the standard getBox for files. The selected text file will be loaded into Yerk.

See also: stdSave

---

stdSave (--)

Defined in source FrontEnd, this word will bring up a the standard saveBox for files. This will save the application image to disk.

See also: stdLoad

---

stillDown? (-- bool)

Calls the Toolbox Event Manager routine stillDown to see if the mouse button is still down.

---

str, (here --)

Compiles the string at HERE into the dictionary. Used by abort" and " when in compile mode.

---

str255 (addr len -- absAddr)

Converts an (addr-len format) string into a str255-format string at buf255 and leaves its absolute address. You can use str255 to prepare strings for calls to Toolbox routines.

See also: buf255 >str255

---

suspend (--)

This is a vector defaulted to nullcfa. The user may put in his own response to a multifinder suspend event.

See also: resume mouseMoved cvtClip inForeGround

---

swap **(n1 n2 -- n2 n1)**

Exchanges the top two single-precision stack values.

See also: 2swap

---

sys (-- 0)

Handles system clicks for object fEvent. A system click occurs when there is a mouse-down event in a system window, like a desk accessory.

---

sysInit (--)

Initializes objects fEvent, fWind, and fEvent when Yerk starts up.

---

syspat (idx -- pattern)

Syspat returns the system **pattern** indicated by the **idx** value which can then be used by a Yerk fill method for a graphical object:

Example:
```
rect box
10 10 200 200 put: box
0 syspat fill: box
```

will black out that region of your screen. Patterns are defined in the resource type PAT#. Common patterns are:

0	black
1	dark grey
2	medium gray
3	light gray

See also: splist

---

tan (fptr -- fptr)

Computes tan(x) of the floating point number pointed to by **fptr**.

See also: arctan

Task (--)

Marker word used during module compilation. You can Forget TASK to restore the dictionary whenever a module compilation fails. Should never be used as a word name, especially within modules.

---

taskList **(-- ^obj)**

This Ordered-col object contains up to four cfa's of words that are executed as background tasks whenever object fEvent receives a null event. You can manipulate taskList with addTask and killTask. This support is optional and is included in file "Tasks".

See also: addtask dotasks killtask

---

temprect (-- ^obj)

This is a Rect object used by Yerk.

---

tFace (face --)

Sets the txFace field of the current grafport to **face**. You can use tface to control the typeface of the current font for the grafport.

See also: tFont tMode tSize initFont

---

tFont (font --)

Sets the txFont field of the current grafport to **font**. You can use tfont to change the font of the current grafport.

Example: 0 tfont \ Sets font to Chicago (=0).

Some fonts: 0 -- Chicago (system font)
1 -- Geneva (application font)
2 -- New York
3 -- Geneva
4 -- Monaco
5 -- Venice
6 -- London
7 -- Athens

See also: tFace tMode tSize initFont ?lead

---

theMenu (-- ^obj)

A value that stores the pointer-to-object of the last menu clicked in.

See also: mitem menuID

---

theMouse (-- ^obj)

This is the default Mouse object. See III.4 Events.

---

then

Compile: **(yerk --)**
Run: **(--)**

Then ends an IF conditional.

Used in the form: IF … THEN
IF … ELSE … THEN

This is an immediate word.

---

thePort **(-- ptr)**

Looks in Yerk's QD globals to determine the current active grafport.

---

thumb **(-- 129)**

This constant is the part ID of the thumb in a scroll bar. See IM Control Manager.

---

tib **(-- addr)**

This value is the address of the terminal input buffer, where characters typed at the keyboard are stored.

---

tMode **(mode --)**

Sets the txMode field of the current grafport to **mode**. You can use tmode to change the text drawing mode of the current grafport. See IM QuickDraw.

See also: tFace tFont tSize initFont

---

to1

Parameter Stack: (n i --)
Methods Stack: (^base -- ^base)

Stores **n** into the **i-**th one-byte element of the object whose **^base**, pointer to the object's data, is on the methods stack. To1 is the optimized store for indexed objects with one-byte elements**.**

---

to2

Parameter Stack: (n i --)
Methods Stack: (^base -- ^base)

Stores **n** into the **i-**th 2-byte element of the object whose **^base**, pointer to the object's data, is on the methods stack. To2 is the optimized store for indexed objects with 2-byte elements**.**

---

to4

Parameter Stack: (n i --)
Methods Stack: (^base -- ^base)

Stores **n** into the **i**-th 4-byte element of the object whose **^base**, pointer to the object's data, is on the methods stack. To4 is the optimized store for indexed objects with 4-byte elements**.**

---

toggle

(addr b --)

Exclusive-ORs the byte at **addr** with mask byte **b**.

TogItem (item# --)

Toggles check boxes and radio buttons. Make TOGITEM the action word for these type items in your dialog. Can only be used while the dialog box is active. This word is defined in the file "dialog" and is an optional part of the system.

ToPad (addr --)

Moves the str255-format string at **addr** to pad+1.

topFile (-- addr)

Topfile is a vect which executes the Last: method of the loadfile object, and allows late-binding to the last element in the load file. It allows the following shorthand to be used if you are using the filelist structure:

```
        open: topfile
instead of:
        open: [ last: loadfile ]
```

trace (--)

Provides a dump of the return stack, printing the pfa's of the words in the order they are nested. You can use trace in debugging your programs.

```
Example:  trace <return>   \ Print the names of nested words.
          67628
          15650  0->
```

trap (trap# --)

Trap is the run-time component of Call. Trap causes the trap of the given number to be actually executed (which may take other arguments from the stack). For a use of the word trap, see the definition of the run time component of cursor in the file "qd".

trav **(execCFA arg --)**

Trav will execute the word represented by **execCFA** for every word in the dictionary, and pass it the argument **arg**. The word to be executed can expect two words on the stack: the argument and the cfa of a word in the dictionary.

Words could be coded this way using trav:

```
: printit { theCfa arg -- }
        theCfa >name count $ 3F and type cr ?pause ;

: newWords 'c printit 0 trav ;
```

traverse **(addr1 n -- addr2)**

Moves across the name field of a definition. **addr1** is the address of either the length byte (the first byte of the name field) or the last byte; **n** = 1 for forward searches (toward high memory), and **n** = -1 for backward searches (toward low memory). Leaves **addr2**, the address of the other end of the name field.

---

true **(-- 1)**

This constant is a boolean. You can use it whenever you need a true flag.

See also: false not

---

trunc **(fptr -- fptr)**

Truncates the floating point number pointed to by **fptr** (rounds toward zero) and returns the result.

---

tSize **(tsize --)**

Sets the txSize of the current grafport. You can use tsize to change the point size of the font of the current grafport.

See also: tFace tFont tMode initFont

---

txType **(-- TEXT)**

Leaves the file type for text files; (four ASCII characters in one stack element).

See also: binType saveType

---

type **(addr len --)**

Prints an (**addr-len** format) string on the screen or printer, by executing the system vectors typevec and ptypevec. When Yerk starts up, typevec is vectored to the primitive (type) that prints a string on the screen, and ptypevec is vectored to 2drop. So the default action of type is to print a string on the

screen only. Type increments value OUT by **len**.

Example: " The Information Age is here." \ Leaves addr len.
type <return>
The Information Age is here. 0->

See also: ." ptype

---

type# **(--) : resID**

TYPE# is a cousin to TYPE except that its text resides in your resource data rather than in the dictionary. At compile time takes the **resID** out of the input stream and compiles a runtime primitive into the current word. At run time prints the string resource associated with the given **resID**. The string can then be changed without recompiling any code.

---

typevec **(addr len --)**

This is the execution vector for type. When Yerk starts up, typevec is vectored to (type), the primitive that prints a string on the screen.

---

u* **"u-star"** **(u1 u2 -- ud)**

Multiplies two unsigned numbers and leaves their unsigned double product.

---

u. **(n --)**

Prints the number on the stack as an unsigned number using the current base.

---

u/ **"u-slash"** **(ud u1 -- u2 u3)**

Divides **u1**, an unsigned number, into **ud**, an unsigned double number, leaving their unsigned remainder, **u2**, and quotient, **u3**.

---

u< **(u1 u2 -- bool)**

Compares two unsigned single-precision numbers. If **u1** is less than **u2**, leaves a true flag (1); otherwise, leaves a false flag (0). Use u< when comparing memory addresses.

---

u> **(u1 u2 -- bool)**

Compares two unsigned numbers. If **u1** is greater than **u2**, leaves a true flag (1); otherwise, leaves a false flag (0). Use u> when comparing memory addresses.

---

ufind **Successful:** **(-- pfa 0 T)**
Unsuccessful: **(-- F)**

This is a system vector that holds the cfa of a special purpose Find word. This Find will be executed before the normal Yerk find. For further information, look under system vectors in Part II.4.

---

unLock **(handle --)**

Unlocks the **handle** given on the stack.

See also: lock hGetState hSetState

---

unpack **(x:y -- x y)**

Unpacks a Toolbox point and puts the two integers on the stack. Unpack is the opposite of pack.

See also: i->I pack

---

until **Compile: (yerk -- yerk)**
Run: (bool --)

At run time, UNTIL controls the conditional branch to the corresponding BEGIN. If boolean is false, execution returns to the first word after BEGIN; otherwise, execution continues forward.

Used in the form: BEGIN ... UNTIL

This is an immediate word

---

upCase **(--)**

Sets the compiler to map all words to upper case. This is the default mode for Yerk which allows case insensitivity. The abort word will reset Yerk to uppercase mapping.

See also: loCase

---

upd-evt **(-- 0)**

Handles an update event for the application by sending a draw: method to the window. This word can be redefined through the use of :f or by substitution of vector 6 in fEvent.

See also: actv-evt disk-evt key-evt mouse-evt null-evt app4-evt

---

useWFont **(-- n)**

This is a constant defined by the toolbox that indicates that a control is to use the same font as its owner window (the application font). The value should be added to the id for the control.

Example: buttonID useWFont + init: aControlObject

would indicate to Yerk that this was a button that was to use the application font. This constant is an optional part of the system and is defined in the file "ctl".

---

valCode (-- addr)

This constant returns the address of the code executed by value.

See also: colCode fvalCode modCode vectCode

---

value **(n --) : name**

Yerk's general-purpose data variable. See Part II.4.

See also: constant variable

---

variable **(n --) : name**

Creates the definition **name** and puts **n** in its parameter field address. When **name** is executed later, its parameter field address is pushed onto the stack, so that a fetch (@) or store (!) can access **n**. For general use, Value is more useful, but Variable can sometimes be handy to provide a Var parameter for the Toolbox.

Example: 12 variable dozen

See also: constant value

---

vect **(cfa --) : name**

Execution variable that can hold and execute the cfa of a Yerk word. See Part II.4.

---

vectCode **(-- addr)**

This constant returns the address of the code executed by vect.

See also: colCode fvalCode modCode valCode

---

vsID **(-- 16)**

Returns a toolbox value to indicate that a given control is to be a scroll bar (either horizontal or vertical). This word is an optional part of the system and is defined in the file "ctl".

See also: buttonID checkID radioID useWfont

W **(-- 16-bit-value) : value**

Pushed a 16-bit value onto the stack. This is more compact that placing a regular literal on the stack followed by MAKEINT. When used within a program definition, compiles the number as a special kind of literal.

See also: word0 makeint pack

W! **(w addr --)**

Stores the 16-bit number **w** into the memory word at **addr**.

W+! **(i addr --)**

Adds the 16-bit increment **i** to the memory word at **addr**.

---

w, **(w --)**

Stores the 16-bit number **w** into the next available dictionary memory word, and advances the dictionary pointer by two bytes.

---

w@ **(addr -- w)**

Fetches the (16-bit) memory word at **addr** and pushes it onto the stack.

---

waitClick **(--)**

Loops until the user either clicks the mouse or presses a key.

---

watchCurs **(--)**

Changes the cursor currently in use to the watch cursor, that is used to indicate that processing is occurring.

See also: crosscurs ibeamcurs pluscurs arrowcurs cursor

---

while **Compile: (yerk -- yerk)**
Run: (bool --)

At run time, if the boolean on top of stack is true, the words between WHILE and REPEAT are executed; otherwise, execution resumes at the first word after REPEAT.

Used in the form: BEGIN ... WHILE ... REPEAT

This is an immediate word.

---

within **(n0 n1 n2 -- bool)**

Checks to see if n0 is within the inclusive limites of n1 and n2.

---

wlit (-- n)

Pushes the (16-bit) memory word at the next dictionary address onto the stack. Within a colon definition, wlit is automatically compiled before each single-precision number less than or equal to 65,535 encountered in the input text.

See also: lit literal

---

wneAvail (-- bool)

Returns true if WaitNextEvent is available on the mac yerk is running on. You should not have to call this.

---

word **(chr --) : text**

Reads characters from the input stream being interpreted until a delimiter **chr** is encountered; stores the text string at the dictionary buffer HERE. Stores the character count in the first byte, the characters in the following bytes, ending with two or more blanks. Leading occurrences of **chr** are ignored.

---

word" **(-- addr) : "textString"**

Word" takes the "-delimited string from the input stream and stores it at HERE, and then returns that address for further processing. Word" does not map to uppercase.

---

word0 **"word-zero" (-- 16-zero-bits)**

Pushes 16 zero bits (hex 0000) onto the stack. You can use word0 to prepare for a Toolbox function call for the result, if the function returns a Toolbox integer.

---

words **(--)**

Prints the names and addresses of all entries in the dictionary beginning with the last entry. Also try keyboard short cut: Command-W.

---

x**y **(fptr:x fptr:y -- fptr)**

Computes $\mathbf{x^y}$ for the floating point arguments provided.

---

xor **(n1 n2 -- n3)**

Leaves the bitwise exclusive-OR of **n1** and **n2** as **n3**. Xor works as a logical xor if you want to use **n1**, **n2**, and **n3** as booleans (non-zero = true; zero = false).

See also: toggle or and Lxor

---

yerk **(--)**

This word executes the startup routines that initialize the menu bar, draw the window, and executes the Sysinit vector.

See also: cold

---

yerk>flt (--)

Places the Yerk interpreter into float mode. Defined in YerkFP.com

---

yerk>int (--)

Places the Yerk interpreter into integer mode. Defined in YerkFP.com.

[(--)

Begins an expression within a late-bound message that computes the address of the object to receive the message.

Example: key: [frontWind] \ frontWind computes the address of the
\ front-most window.

This is an immediate word.
See also: <[c[

[compile] (--) : word

Forces the compilation of an immediate word that would otherwise execute during the compilation of xxx. Here, WHILE will be compiled into the definition of xxx, instead of executing during compilation.

Used in the form: : xxxx [COMPILE] WHILE ;

This is an immediate word.
See also: compile

\ (--)

Begins a comment which is to continue to the end-of-line. In a stack comment in braces, \ flags the following words up to the '--' as local variables.

Example: \ This is an end-of-line comment begun with '\'.

```
: aWord { a b \ temp index -- } ... ;
\ '\' before temp and index indicates they are local vars.
```

This is an immediate word.

] (--)

Ends an expression within a late-bound message that computes the address of the object to receive the message.

Example: key: [frontWind] \ frontWind computes the address of the
\ front-most window.

This is an immediate word.

]> (--)

Enters compile state. Sets state = non-zero.]> is used in conjunction with <[.

Example: : aWord ...
<[3]> 'cfas doGraph doPict doText ... ;

\ <[is used to put the interpreter in run state temporarily because
\ 'cfas needs the count of cfa's to compile on the stack;]> resumes
\ compilation.

]c (--)

Enters class compilation state. Sets cstate = non-zero.]c is used in conjunction with c[.

This is an immediate word.

---

^base

Parameter Stack: **(-- ^base)**
Methods Stack: **(^base -- ^base)**

Pushes the base address of the private data of the object by copying the address from the methods stack.

See also: copym (abs)

---

^class (-- ^class)

Returns a pointer to the class currently being compiled.

---

^elem

Parameter Stack: **(i -- addr)**
Methods Stack: **(^base -- ^base)**

Returns the address of the **i**-th element of the indexed object whose address is on the methods stack.

{ (--)

Begins definition of named instance variables and local variables for a Yerk word or method. Each blank-delimited name following the curly brace will be considered either a named input parameter, which assigns a name to a stack cell, or a local variable, an uninitialized 32-bit cell for the word or method to use as temporary storage. The list of named parameters is separated from the local variables by a backslash. Yerk permits a maximum total of six named parms plus local variables. Two hyphens terminate the list, and everything

following the hyphens up to and including the right curly brace is considered a comment.

Example: : aWord { parmOne parmTwo \ locVar1 locVar2 -- }

This is an immediate word.

} **(--)**

Ends definition of named instance variables and local variables. You can have up to six total of both.

Example: : aWord { parmOne parmTwo \ locVar1 locVar2 -- }
... ; \ more code

}end (--)

Ends a clause in a select{ ... }select conditional structure.

This is an immediate word.

}select (--)

Ends a select{ ... }select conditional structure. The select{ conditional structure is like the case structure, except that the indexes must start at 0 and must be contiguous. The advantage of using select{ is that it executes faster than case, and for this reason you should use it whenever applicable.

Example:
```
( n -- ) \ Print zero, one, two, or none of the above.
: testSelect
Select{
        0 is{ ." Zero" }end
        1 is{ ." One"  }end
        2 is{ ." Two"  }end
        Default{ ." None of the above"
}select  ;
```

This is an immediate word.