>R "to-r" P
(n --)
Transfers n from the stack to the return stack. Since **>R** changes the return stack, it normally must be paired with a corresponding **R>** before a colon-definition is exited.

?comp "question-comp" PRE
(--)
Generates an error if the compilation state (the value of the **STATE** variabl) is False (execution mode).

?exec "question-execution" PRE
(--)
Generates an error if the compilation state (the value of the **STATE** variable) is True (compilation mode).

?pairs "question-pairs" PRE
(n1 n2 --)
Generates an error if the top two stack items are not equal.

@ "fetch" P
(addr -- n)
Returns a single-length number stored at addr. See also **c@**.

BASE "base" PRE
(-- addr)
A variable containing the number base being used for numerical input and output. Thus :-

```
2 BASE !
```

would select binary input and output. See also **decimal** , **hex** , **binary**.

BL "b-l" PRE
(-- 32)
Puts the ASCII code for a space or blank (decimal 3) on the stack.

C! "c-store" P
(n addr --)
Stores the least significant byte of n into a byte at addr. See also **!** .

C, "c-comma" P
(n --)
Stores the least significant byte of n into a byte at the next available dictionary location (the address returned by **here**), and increments the dictionary pointer by 1 to allocate space for the compiled byte. See also **,** (comma) .

C/L "c-slash-l" PRE
(-- addr)
A variable containing the maximum number of characters per line.

C@ "c-fetch" P
(addr -- n)
Fetches a byte at addr onto the stack. The result is a number whose most significant byte is 0 and whose least significant byte is the fetched byte. See also **@**.

CONTEXT "context" PRE
(-- addr)
Returns the address of a variable that determines the vocabulary to besearched first in the dictionary. See also **CURRENT , definitions , vocabulary , seal .**

CURRENT "current" PRE
(-- addr)
Returns the address of a variable that determines the vocabulary into which new word definitions will be entered See also **CONTEXT , definitions , vocabulary , seal.**

DP "d-p" PRE
(-- addr)
A variable containing the address of the next free location in the dictionary.

FENCE "fence" PRE
(-- addr)
A variable that specifies an address in the dictionary below which words may not be deleted using **forget.**

HLD "h-l-d" PRE
(--)
A variable that contains the byte offset into the text scratch pad where characters are currently being stored.

I "i"
(-- n)
Places the index of a do-loop onto the stack. Can only be used directly and not in a word called from within a do-loop. See also **J.**

IN "in" PRE
(-- addr)
A variable that contains the byte offset into the input stream where interpretation is presently occurring.

INNER "inner" PRE
(-- addr)
A variable containing the code-field address of the inner interpreter.

J "j" P
(-- n)
Used within a nested do-loop in the form, for example :-

do .. do .. J .. loop ... +loop

returns the index of the next outer loop, in this case the loop terminated by **+loop**. Like **I , J** can only be used directly and not in a word called from within the loop.

OUT "out" PRE
(-- addr)
A variable containing a count of the number of characters currently output on a line. See also **C/L , emit.**

R: "r-colon" PRE , E , I
(--)
A defining word that creates a colon-word. Used only in execution mode, in the form :-

:R < name > ... R;

creates a dictionary entry for < name > in the current vocabulary and sets **STATE** to compile mode, causing the following words and numbers from the input stream to be compiled. < name > can be found in the dictionary thus enabling recursion. **R:** sets the search order so that the first vocabulary to be searched is replaced by the compilation vocabulary. The compilationvocabulary is unchanged. See also **R;** , **:** , **;**.

R; "r-semi-colon" PRE , C , I
(--)
Stops compilation of a 'R' colon-definition. Used in the form :-

R: < name > ... R;

Sets **STATE** to execute mode.

R> "r-from" P
(-- n)
Removes n from the return stack and places it on the stack. **R>** generally must be used with a corresponding **>R** to prevent the return stack from becoming unbalanced. See also **R** , **>R**.

R "r" P
(-- r)
Copies n from the return stack to the stack (without changing the return stack). **R** is useful when the return stack is used for temporary storage of numbers within a colon-definition. See also **>R** , **R>**.

STATE "state" PRE
(-- addr)
A variable that contains a value defining the compilation state. A True (-1) value indicates that compilation is occurring and a False (0) value indicates that interpretation is occurring.

TIB "t-i-b" PRE
(-- addr)
returns the address of the start of the text-input buffer (sometimes called the "terminal input buffer"). This is used to receive characters from the keyboard.

WIDTH "width" PRE
(-- addr)
A variable containing the number of characters that will be compiled in the name-field of a definition, thus the maximum length for a word name. **WIDTH** can be used to truncate word names so as to save dictionary space.

["left-bracket" PRE , I
(--)
Sets the system state to execute mode. The text from the input stream is subsequently executed rather than compiled. Used within colon definitions in the form :-

: < name > ... [...] ... ;

allows the words between [and] to be executed when < name > is compiled. See also] , **STATE.**

[compile] "bracket-compile" PRE , I
(--)
Causes an immediate word to be compiled. Used in the form :-

: < namex > ... [compile] < name > ... ;

forces compilation of the immediate word < name >, which would ordinarily be executed even in compile mode. **[compile]** itself is not compiled, in contrast to **compile**, which has a very different function.

\ "backslash" P , I
(--)
Causes the TILE interpreter to ignore the rest of the line of input. Used so that the rest of a line of source code may contain comments. May be used within a colon-definition.

] "right-bracket" PRE , I
(--)
Sets the system state to compile mode. The text from the input stream is subsequently compiled.] is useful for compiling headerless words and for compiling the cfa of words into an array for vectored execution. See also [, **STATE.**

abort "abort" P
(--)
Clears both the parameter and return stacks and sets **STATE** back to execute. No message is displayed, not even " ok ".

abs "absolute" PRE
(n1 -- n2)
Returns the absolute value of the signed number n1, that is, a number of the same magnitude but always with a positive sign.

again "again" PRE , C , I
(--)
Marks the end of an infinite loop. Used in the form :-

: < name > ... begin ... again ... ;

Compiles an unconditional branch back to the corresponding **begin**. When <name> is executed the words between **begin** and **again** willl be repeated without terminating.

allot "allot" PRE
(n --)

Allocates n bytes in the dictionary. That is, the address of the next available dictionary location, the dictionary pointer (returned by **here**), is incremented by n.

and "and" P
(n1 n2 -- n3)
n3 is the bit-by-bit logical "and" of n1 with n2. That is, each bit of n1 is compared with the equivalent bit in n2 and if either bit is 0 a 0 is placed in the corresponding bit position in n3. Thus (in binary) :-

110 100 and

will return binary 100. See also **or** , **xor**.

begin "begin" PRE
(--)
Marks the start of an indefinite loop structure. Used in the form :-

: < name > ... begin ... flag until ... ;

or

: < name > ... begin ... flag while ... repeat ... ;

or

: < name > ... begin ... again ... ;

compiles an indefinite loop structure. When < name > is executed, the words between **begin** and **until** will be repeated until the flag on the stack is true. The words between **begin** and **repeat** will be repeated while the flag on the stack seen by **while** is true. The words after **until** or **repeat** will be executed when either loop is finished. The words between **begin** and **again** will be repeated without terminating.

binary "binary" PRE
(--)
Selects binary notation for numerical input and output.

: binary 2 base ! ;

See also **decimal** , **hex** .

branch "branch" PRE
(--)
When used in the form :-

: < name > ... compile branch ... ;

an unconditional branch operator is compiled. A relative address must be compiled immediately after the compilation address of the unconditional branch operator.

bye "bye" P
(--)
Quit TILE and restore the system to the state just before TILE was invoked.

ca! "c-a-store" PRE
(n --)
Stores n in the code-field address of the last definition in the **CONTEXT** vocabulary.

cfa "c-f-a" PRE
(pfa -- cfa)
Given the parameter-field address of a word, returns the code-field address of that word.

cmove "cmove" PRE
(addr1 addr2 n --)
Copies n bytes beginning at address addr1 to addr2. The move proceeds from low memory toward high memory. If n is zero or negated, nothing is moved. See also **move** .

compile "compile" PRE
(--)
Typically used in the form :-

: < name1> ... compile < name2 > ... ;

When <name1> is compiled, the code-field address of <name2> is compiled in the dictionary following the code-field address of **compile**. <name1> is typically immediate and <name2> is typically not immediate. When <name1> is executed, the code-field address of <name2> is compiled at the top of the dictionary, usually in a colon-definition.

constant "constant" PRE
(n --)
A defining word that creates a single-length constant. Used in the form :-

n constant < name >

creates a dictionary entry for <name> and compiles n into <name>'s parameter field. When <name> is executed, the single-length number n is placed on the stack. See also **variable.**

count "count" PRE
(addr1 -- addr2 n)
Returns the number of characters in a counted string at addr1. addr2 is addr1+1, the start of the text, and n is the length of the text. **count** returns an address and count suitable for use by **type**. thus the phrase

addr count type

will display the counted string stored at addr.

: count dup C@ swap 1+ ;

cr "c-r" PRE
(--)
Sends a carriage return and line feed to the console, moving the cursor to the start of the next line, and resets **OUT**.

create "create" P
(--)
A defining word to create a dictionary entry. Used in the form

create < name >

creates a dictionary entry for <name> without allocating space in <names>'s parameter field, and sets the code-field address to point the parameter-field. Executing <name> without modification to either to code- or parameter-fields will result in an error, thus <name> is normally only used in association with other defining words. See also **<builds , does>**.

d+ "d-plus" P
(d1 d2 -- d3)
Adds d2 to d1 and returns the double-length sum d3.

d+ - "d-plus-minus" PRE
(d1 d2 -- d3)
Returns the absolute value of the double-length signed number d1 if d2 is negative.

d- "d-minus" PRE
(d1 d2 -- d3)
Subtracts d2 from d1 and returns the double-length difference d3.

d. "d-dot" PRE
(d --)
Displays, according to the value of **BASE**, the value of d (including a leading minus sign if negative) at the current display position, followed by a space. See also . (dot).

d.r "d-dot-r" PRE
(d n --)
Displays the value of d right justified in a field n characters wide. A leading minus sign is displayed if d is negative. If n is less than 1, no leading blanks will be supplied and it is not an error if the number of characters required to display d is greater than n.

dabs "d-absolute" PRE
(d1 -- d2)
Returns the absolute value of the signed double-length number d1. That is, d2 will be a double-length number of the same magnitude as d1 but will always have a positive sign.

decimal "decimal" PRE
(--)
Selects decimal notation for input and output. Puts 10 (decimal) into the variable **BASE**.

definitions "definitions" PRE
(--)
Sets the vocabulary into which definitions are compiled to be the same as the first vocabulary in the search order. This is achieved by setting the contents of the variable **CURRENT** equal to that of the variable **CONTEXT**. For example , executing :-

logic definitions

would make subsequent word definitions be placed in the logic vocabulary. See also **vocabulary**.

dminus "d-minus" P
(d1 -- d2)
Reverses the sign of d1 to produce d2. d2 is the twos complement of d1 (i.e. zero minus d1).

do "do" PRE , I , C
(n1 n2 --)
Marks the start of a do-loop. Used in the form :-

: < name > ... n1 n2 do ... loop ... ;

or

: < name > ... n1 n2 do ... increment +loop ... ;

marks the beginning of a do-loop. When < name > is executed, **do** takes both n1 and n2 from the stack and uses n2 as the starting value of the loop index and n1 as the limit that will determine when the do-loop is terminated. See **loop** and **+loop** for the differences in loop termination. Any do-loop is always executed at least once.

does> "does" PRE , I
(-- addr)
Defines the execution-time action of a word created by a defining word. Used in the form :-

: < namex > ... <builds ... does> ... ;

where <builds creates a child dictionary entry, and does> marks the start of the part of <namex>'s definition that determines the execution behaviour of its children. Thus when < namex > is executed in the form :-

< namex > < name >

a dictionary entry is created for the child word <name> and the execution-time action for <name> will be determined by the sequence of words between **does>** and **;** in the definition <namex>. **does>** may be used only within a colon-definition. For example, the definition of **constant** could be specified thus :-

: constant <builds , does> @ ;

See also **<builds** , **;code** , **scode.**

drop "drop" P
(n -- n)
Removes n from the stack.

dup "dupe" P
(n - n n)
Duplicates the single-length number n on the top of the stack.

else "else" PRE , C , I
(--)
Marks the start of an alternative branch. Used in the form :-

: < name > ... flag if ... else ... then ... ;

compiles an unconditional branch operator to guide execution of the word to just after **then**. When < name > is executed, **if** expects a flag on the stack. If the flag is nonzero, the words between **if** and **else** will be executed with a jump to after **then** but if the flag is zero, execution will jump to the words between **else** and **then** with continuation after **then** . See also **if** , **then** .

emit "emit" PRE
(n --)
Sends the character represented by the least significant byte of n to the console and increments **OUT.** see also **cr** , **OUT** .

empty? "empty-question" P
(-- f)
Returns single-length value of True (-1) if the stack is empty otherwise a single-length value of False (0) is returned.

execute "execute" P
(cfa --)
Executes the word whose code-field address is on the stack.

fill "fill" PRE
(addr n1 n2 --)
Fills n1 bytes of memory beginning at addr with the least significant byte of n2. No action is taken if n1 is zero or negative.

forget "forget" PRE
(--)
Used in the form :-

forget <name>

deletes <name> and all words added to the dictionary after <name>, regardless of their vocabulary, if <name> is found in the vocabulary to which definitions are being added. If <name> is a vocabulary link then it is removed and **CONTEXT** and **CURRENT** are reset to point the previous hierarchical vocabulary.

here "here" PRE
(-- addr)
Returns the address of the next available location in the dictionary, the location of the dictionary pointer.

hex "hex" PRE
(--)
Selects hexadecimal notation for input and output.

: hex 16 BASE ! ;