Key to Glossary

This glossary contains descriptions of the most common JForth words. To find words related to a specific function, eg. graphics or assembly, please look in the reference section.

The words are organized in ASCII alphabetical order. Many Forth words start with punctuation. You can find out the alphabetical order for punctuation by looking at the bottom of the page in the glossary.

Each Forth word description has a common format. The first line of the description has the name of the word followed by a stack diagram. The pronunciation of the word sometimes follows.

Following the description of the word will often be an example, indented and in upper case.

If the word is not in the normal JForth image, the file where it can be found will be given.

Any related words will then be listed at the end of the description. Please also see the appendix on words related by function as well.

Understanding Stack Diagrams

Stack diagrams tell you what parameters are passed to a Forth word and what is returned. The input and the output is separated by several dashes. Look at the stack diagram for + as an example.

```
+ ( a b -- a+b , add two numbers together )
```

This means that + takes two numbers, A and B, as input. It leaves A+B as output. The text after the comma is a comment. The rightmost item is on the top of the stack so B is on top in this example.

String Parameters

There are a number of ways to pass a string in JForth.

Parameters that start with a $ are Forth "counted strings". The parameter will be the address of a count byte that will be followed in memory by that number of characters. The Forth string "FROG", for example, will be stored in memory as:

```
" FROG" 5 DUMP  ( reveals 04 46 52 4F 47 )
```

The first byte is the count byte. The 46, 52, 4F, 47 are the ascii characters F R O and G.

Forth also supports NUL terminated strings like 'C'. Here, the address of the first character is passed. The string continues until a zero is encountered.

```
0" FROG" 5 DUMP ( reveals 46 52 4F 47 00 )
```

These strings parameters are referred to with a zero in the name, eg. 0string.

Forth strings can also be passed as an address and a count.

Please examine the stack diagrams for COUNT and 0COUNT as examples.

COUNT ($string -- addr count)

0COUNT (0string -- addr count)

Here are examples of the use of each.

```
" FROG" COUNT TYPE
0" FROG" 0COUNT TYPE
```

Strings can also be passed on the input line. Consider the Forth word FOPEN which expects a filename to follow. FOPEN opens a filename of the given name and returns a file-pointer which can be used to read the file. Here is an example of a call to FOPEN .

```
FOPEN RAM:MYFILE
```

Strings that are passed on the input line are enclosed in angle brackets in stack diagrams. The stack diagram for FOPEN would, therefore, be:

```
FOPEN   ( <filename> -- file-pointer | false )
```

The vertical bar and the FALSE means that if the file cannot be opened, a FALSE will be returned as an error indicator.

Please keep in mind that some words may have an action but may not have any stack activity. The word CR, for example has a stack diagram of:

```
CR ( -- , emit carriage return )
```

Return Stack

There is also a Return Stack in Forth. Words that affect the Return Stack will have a Return Stack Diagram. These will be marked with a --R--. Here is an example.

```
>R   ( n -- , --R-- n , move N to the return stack )
```

Summary

In summary, here is a description of the common parameters you will see:

```
a b      = some values
addr     = address of some memory location
flag     = TRUE (-1) or FALSE (0)
char     = ASCII character
count    = number of things, usually bytes
d        = double precision integer, 64 bit
n        = single precision integer, 32 bit
pad      = address of PAD , a scratch area
u        = unsigned integer
var-addr = address of a variable
$addr    = counted string address
$string  = counted string address
0string  = NUL terminated string address
<text>   = text follows word
<name>   = name of a Forth word
<filename>      = name of a file
|        = OR , denotes alternate parameters
```