**Flat Language Notes**

Rule #1. When call all words compile themselves.

This means one of three things.

1. The word is prefixed by a call to a routine that compiles a call to it. It pops the address of the returns stack and compiles call <xxxx>.
2. The word is prefixed by ld b,<size> and a call to a routine that copies that code into program memory. The address of the code is got as before, by popping the address of the return stack.
3. The word does something else entirely – like if for example.

Executing words

Words are executed by compiling their code into a buffer, adding a ret on, and calling them in context. Some words will not work, because they either compile code themselves (, and c,) or they manipulate the return stack.

Security

Some words of type 2 – that copy code rather than a call to code – can cause problems – for example they manipulate the return stack. It is therefore a compiler convention that bit 7 of the size byte, if set, means the word cannot be executed.

**Source code**

Source code is divided into 512 byte pages (the size of an edit area). They consist of ‘tags’ (value $80-$87) which colour the word, followed by the word name in ASCII. The word is ended by the next byte having bit 7 set. The end of the tag list is indicated by $FF (which itself ends a word).

**Colouring Scheme**

Source code is coloured by what you want to do with the text.

**Red words ($82)**

Red words are normal defining words ; like : in FORTH. They define the current position as a word in the dictionary and add the call to the routine that compiles the word itself. If you want the word to do something else then user the green (or yellow) word does> which copies the Here at current definition start address (+28) to the here address (+0)

**Green words ($84)**

Green words are normal compiled words. Green words simply call the word which compiles its own code to execute itself. See Rule # 1. Green numbers compile the code to copy A to B, and load the number into A.

**Yellow words ($86)**

Yellow words execute using the green word code, but it is placed in a temporary buffer rather than the main code area, and called ‘in context’. These are only used in source code to do inline calculations and things like that, though *every* word execute from the command line is a yellow word. Note that words which are ‘copied’ and have bit 7 of the length bit set, should not be allowed to be executed.

**White words**

White words are comments and are ignored.

**Format Notes**

**Dictionary**

|  |  |
| --- | --- |
| **Offset** | **Contents** |
| +0 | Offset to next entry, or 0 if end of table |
| +1 | Page number where word i |
| +2 | Low byte of word address |
| +3 | High byte of word address |
| +4 | Bits 0..5 length of name. Bit 6,7 are zero. |
| +5 | First character of name (7 bit ASCII) |