# RPL Description

RPL is a Stack based Threaded language, similar to FORTH.

It uses R0 as the stack top, and R8 as the data stack pointer. (R8) is the second value on the stack. Other stack and register usage is as normal.

## RPL() command

Compiles the tokens within the brackets. Begins with *push link* and ends with *pop link ; ret.* Returns the initial address (set up by code).

You can execute via SYS <address>. In this scenario R0 (TOS) points to the A variable

It is perfectly possible to mix RPL and assembler, though not in the same word. Assembler routines have to be careful with R0 and R8 however (and, obviously R14).

## Dictionary

The dictionary is fixed. Entries can either be tokens, identifiers or a mix. Additionally each dictionary entry can be executed or a call to it compiled. The dictionary is in backward length order and passes at the first match.

Dictionary entries are as follows :

|  |  |
| --- | --- |
| **Offset** | **Contents** |
| +0 | Address of next dictionary item, or 0 if this is the end of the dictionary |
| +1 | Address of code routine |
| +2 | Number of words in token to compare (0..3) execute (15) |
| +3 | First word of token. |
| … | Last word of token (may also be the first). |

## Tokens

|  |  |  |
| --- | --- | --- |
| **Type** | **Code** | **Notes** |
| Line End | 0000 | Causes an error as the closing bracket is missing |
| String | 0100-01FF | Generates code to push the string address (the following word) on the stack. |
| Token | 2000-3FFF | Searches dictionary for that token, executes or compiles call accordingly. |
| , | - | Ignored, used for syntactic seperation |
| Const/Shift | - | Prefixes a constant to indicate range 32768-65535 |
| Identifier | 4000-7FFF | Checks to see if in dictionary, if so executes or compiles subroutine call. If not, it identifies as a variable using the normal assignment code, and checks what follows. If it is an & it loads the variable, a ^ stores the variable, and () calls its value as a subroutine. The () is optional and for syntactic clarity. |
| Constant | 8000-FFFF | Compiles code to push constant on stack. |

Note that defined words must be legal constants, and words in the dictionary must be either known tokens or legal constants. Combinations are not allowed (e.g. +xy). Commas are sometimes required for token series. For example > = (with a space) will tokenise correctly but list as >=, so if you edit the line it will go back in as >=