Google Search [✓ Search This Site] Full Web

Site Map
HOME

The Heart of Forth
Stacks and Data Space
Logic and Comparison
Arithmetic

Build Your Own Forth

In a sense, you build your own Forth every time you program, extending the system to meet your own particular needs. But the flexibility of Forth does not end there. Forth is all about simplicity and openness. Some Forths have a metacompiler which enables you to alter the system itself, producing a new kernel better suited to your own requirements. Many people have written their own Forths from scratch.

The relative ease of writing Forths has lead to a bewildering variety. It has been said that "when you've seen one Forth - you've seen one Forth" and that the most useful Forths are also the least portable. This document is an attempt to avoid that Tower of Babel, by setting out in simple terms the principles that underlie any Forth. To that end I have given the primitive definitions in the form of code for a pseudo-assembler, not for any particular chip. I am assuming that readers are conversant with the instruction set of their own particular chips, and - if not experienced Forth programmers - at least comfortable with Forth syntax. The idea is that assembly-savvy people can use it to write (and share) interpreters for their own favourite hardware and build to a complete Forth that performs identically on several systems, or have a common language for discussing the differences.

If it is important that your Forth should be portable and compatible with other Forths, you will need to study The ANS Standard and in particular, check it against John Hayes' test suite to test ANS Standard Systems You may not want or need a Standard Sytem, but if you don't know the rules, how can you tell when you need to break them?

HOME Site Map TOP