

The intent of the set of tutorials presented on this web page is to provide an introduction to or refresher of some of the basic software and hardware tools that should be a part of every embedded developer's toolbox. They are intended to augment not replace the material covered in the text; certainly they cannot take the place of formal training or concentrated self study in each of the areas. There are many many excellent texts that provide either a broad introduction or in depth presentation of the topics. I encourage you to explore and to take advantage of them.

Tutorial 1 – The C Preprocessor

The set of tutorials opens with a brief introduction to the C preprocessor. We begin with the preprocessor language then examine and illustrate each of the major directives. These include working with simple and parameterized macros, conditional compilation, file inclusion, and several other useful tools.

The objective is to introduce some of the tools that the preprocessor provides to make our development task a little easier and to help to make the software we design a little more robust.

Tutorial 2 – C Operators and Flow of Control

In the second tutorial we open with a review the C language arithmetic and logical operators then follow with a brief study of the constructs by which we can alter and control the flow through a program. As the different topics are presented, we suggest techniques and methodologies that can help one to learn and to practice good coding style and to design and develop more robust programs.

The purpose of this tutorial is to provide a short review/refresher of the C arithmetic and relational operators as well as the looping constructs supported by the language.

Tutorial 3 – Data Structures and Algorithms

In this tutorial, we look at several basic data structures and algorithms that find utility in many embedded applications. These comprise the *linked list*, the *queue*, and the *stack* data structures and the four algorithms, *linear search*, *binary search*, *selection sort*, and *quicksort*.

The goal in presenting / reviewing this material is to revisit these fundamental data structures and algorithms and then to use the process of designing them as a platform to stress a more formal approach to the design of hardware and software modules in general. The design approach is a distillation and reemphasis of that presented in Chapter 9 of the text.

Tutorial 4 – Combinational Logic and Memory Devices

In this tutorial we review of the basics of *Boolean algebra*, *combinational logic*, and *storage devices*. We use Boolean algebra to formulate and manipulate logical equations then we implement the resulting equations using the AND, OR, and NOT *gates*. We present the Karnaugh map as an easy to use graphical logic reduction tool for small problems.

We then move from combination logic to *latches* and *flip-flops*. In our review of these devices, we begin with the basic storage device, add capabilities to support greater control over when and how the device changes state in response to its inputs, then develop the characteristic equations describing the behavior of the most common latch and flip-flop types.

The purpose of this tutorial is offer a review of basic combinational logic and storage devices. Today, much of the implementation and application of such devices is found in

arrayed logics or Very Large Scale Integrated Circuits (VLSI). None-the-less, the underlying concepts remain and they remain important.

Tutorial 5 – Sequential Circuits – Finite State Machines

In this tutorial, we build on an assumed fundamental understanding of sequential circuits to examine the development and optimization of general purpose *finite state machines*. Such machines find application in controlling and coordinating the activity of most modern microprocessors, microcontrollers, and microcomputers as well as more complex digital systems.

Like the discussion of combinational logic circuits, the goal of the tutorial is to serve primarily as a review of basic sequential circuits and their role in many contemporary embedded designs.

In putting these tutorials together, every effort has been made to ensure the quality and accuracy of the text. As will happen, despite these efforts, errors sometimes still manage to find their way in. If you do discover an error, large or small, please send email to jkp@u.washington.edu describing what you've found.

Thanks and Cheers,
James K. Peckol