On a large project, these costs usually are minimal. On a small project, they can drive development costs above what the product will produce in sales.

Hardware and Software Requirements

If the product specifications or requirements definition is the goal for the product, the hardware and software requirements are the goal for the detailed design. These requirements start with definition of the user interface and system functionality. In the example system, the complete system definition (see Appendix A) specifies what must be done and how the user operates the device.

From the system definition, a hardware interface is defined. The most productive method of defining the hardware is to start with the requirements—what the hardware must have. This is tied to the system specifications because the hardware must support whatever functionality the system has. In the example system, the time must be displayed. Given the constraints of the system (the timer will not be connected to a PC, for example, so a CRT display is out), it came down to two choices: light-emitting diode (LED) and liquid crystal display (LCD). Even though an LCD would be more readable, I chose the LED display because the timer will be exposed to the weather all year, and the LCD displays available at the time had problems with cold temperature.

Some people consider each set of specifications to be a fixed, immutable document. I prefer that the hardware specifications be a record of the design decisions. The first section of the hardware specifications is the requirements. This is given to the hardware engineer and becomes the basis for what he or she does. The requirements should spell out just that—the requirements. How the requirements are met is up to the engineer. Anything that cannot be left to the engineer's discretion should be in the requirements document. Of course, what you leave to the engineer's discretion may be different for a new college graduate than for an engineer with 10 years of experience.

When working as a project engineer on a large project, I like to put a list of the requirements for each microprocessor-based board in the engineering specifications. This single document then becomes the foundation for the individual board specifications.

After the requirements document is completed and while the design is progressing, the hardware specifications are updated to include the specific information that another engineer needs to understand the design and that the software engineer needs to program around the hardware. So, when the board specifications are completed, a preface is added that describes the original requirements (and any updates that occurred as the design progressed) and a description of how

the design was implemented, with all the information the software engineers need to control the hardware. This includes the following:

- Memory and I/O port addresses (including memory maps if appropriate)
- Amount of memory available
- The definition of each bit in each status register
- The use of each bit on each port pin
- An explanation of how peripheral devices are driven (such as the clock frequency input to a timer IC)
- Anything else the software engineer needs to know about the design

On a complex board, I have often had two separate sections in the hardware specifications. The first section describes the hardware and how it works. The second section contains the information the software engineers need to do their job. In a similar fashion, a software requirements document is created that defines what the software must do. In a simple design like the pool timer, this may consist of the system requirements document, which describes the user interface, and the hardware specification, which describes how the hardware works.

A detailed software specification that describes the completed design is less common than the equivalent hardware specification. This occurs for four reasons:

1. The hardware specification is passed to the software engineers so they will know how to manipulate the hardware. Usually, no corresponding "customer" needs to know the technical details of the software, so the need for documentation is not as great.
2. Software is easy to change, so it changes frequently, often whenever someone in marketing thinks up a new feature to add. In some situations, software specifications can be very hard to keep up to date, especially if the software engineers have other, higher priorities.
3. Software usually is the last part of a project to be finished and often not enough time is left at the end of a project to document it. That said, company or customer policy sometimes requires detailed software specifications. For example, defense projects usually require extensive documentation detailing every function that the software performs.
4. The mechanical and electrical requirements are typically testable. Stresses and tolerance stickups and power dissipations can be mathematically tested. With software, it is more difficult to prove that the requirements are correct and that the flowcharts really will produce code that does what was intended. The more complex and detailed the software requirements are, the less likely it is that you can prove the requirements to be correct. For this reason, the software requirements document is likely to be less detailed or even to be omitted entirely from the design process.