EMC (electromagnetic interference/electromagnetic compatibility) certification, safety agency approval (UL/IEC), and environmental specifications (temperature, humidity, salt spray, and so on).

Although we'll discuss this further in Chapter 7, one problem with specifying requirements is verifying them. It is easy to determine whether the product meets the EMI/EMC requirements—you can run tests to prove it. But how do you prove you've met the requirement for "minimum switches and knobs"? Thus, keep in mind the problem of verification when specifying requirements.

A complex system may have another level of documentation, which I usually refer to as the *Engineering Specification.* This document describes the approach that will be used to implement the design, including which boards will be included and how the functions are partitioned onto those boards. I will return to this information later, in Chapter 8. For now, assume that we have a simple product, which makes this intermediate document unnecessary.

After the requirements are defined, the next step is to determine whether a microprocessor is the best choice. For the pool timer, it is fairly obvious that a microprocessor is the easiest way to do the job. Some other systems are not so obvious. The following questions can help determine whether a microprocessor is justified:

- At what speed must the inputs and outputs be processed or updated? Although the clock rates are ever increasing, there is a practical upper limit to the speed at which a microprocessor can read an input or update an output and still do any real work. At the time of this writing, an update rate of a few hundred kHz is a practical upper limit for a simple microprocessor system with few processing demands and running on a fast processor or digital signal processor (DSP). If the system must do significant processing, buffer manipulation, or other computing, the potential update rate will decrease.
- Is there a single integrated circuit (IC) or a programmable logic device (PLD) that will do the job? If so, a microprocessor is probably not justified.
- Does the system have a lot of user I/O, such as switches or displays? If so, a microprocessor usually makes the job much easier.
- What are the interfaces to other external systems? If your system must talk to something else using Synchronous Data Link Control (SDLC) or some other complex communication protocol, a microprocessor may be the only practical choice.
- How complex is the computational burden on the system? Modern electronic ignition systems, for example, have so many inputs (air sensors, engine rpm, and so on) with complex relationships that few choices other than a microprocessor are suitable.
- Will the design need to be changed once it is finished, or will the requirements be changing as the design progresses? Is there a need for customization of the product or for special versions? Any of these requirements makes a microprocessor attractive due to the flexibility of implementing functionality in firmware.

Fortunately, the job of the system designer is becoming easier. Microprocessor costs are coming down as speed and performance rise. Even simple microprocessors are capable of handling tasks that were limited to dedicated hardware just a few years ago. When you include very fast processors (such as low-cost DSPs), the range of potential applications that can be performed with a microprocessor is wider than ever.

Processor Selection

Suppose you decide to use a microprocessor for your new widget. What steps do you take to select the processor to be used? Fortunately, for all but a very few applications, more than one right solution is possible because several microprocessors can meet the requirements. As with most real-world engineering decisions, the selection consists of a series of tradeoffs between cost and functionality. The specific selection process will depend on the complexity of the finished product, but the following items must be taken into consideration:

- Number of I/O pins required
- Interfaces required
- Memory requirements
- Number of interrupts required
- Real-time considerations
- Development environment
- Processing speed required
- ROMability
- Memory architecture
- Power requirements
- Environmental requirements
- Life cycle costs
- Operator training/competence
- The "real" requirements

Number of I/O Pins

In a minimum-cost system, component count is a major factor in the final product cost. These systems generally use a single-chip microprocessor with internal ROM and RAM. There is a convention to identify these parts as *microcontrollers,* to