consumer product, functions that cannot be performed in software may simply be left out.

Distributed Processor Systems

We'll cover multiprocessor systems in more detail in Chapter 8. Here, we summarize the tradeoffs involved in choosing a multiprocessor architecture. A distributed processor system might have a single CPU that communicates with a host computer and distributes commands and data to lower-level processors that control motors, collect data from sensors, or perform some other, simpler task. Distributed processor systems have the following advantages:

- The actual processing hardware can be located near the device being controlled or monitored. In large equipment, this may be a real advantage.
- If some of the functionality is optional, the cost of the processor that controls the option can be added or removed with the option.
- In a distributed processor system, each of the distributed CPUs usually can be a lower-performance (cheaper) part than would be required for one central CPU.
- A distributed system can be designed with a better match between the CPU and the task it must perform. In a single-CPU system, the CPU must be fast enough and have enough memory and so forth to perform all the tasks, whether they are simple or complex.
- The code for any given CPU in a distributed system usually is simpler.
- It is easier to determine whether the CPU power is adequate in a distributed system because fewer tasks are being swapped in and out and there is less interaction among the various processing that must be performed. For example, you need not worry about how the motor control function affects the serial interface throughput if the tasks are handled by separate processors.
- Debug of distributed systems can be simpler since each processor performs a limited set of tasks.

The advantages of a single-CPU system are:

- Synchronization, when needed, is easier. For instance, it is easier for a single-CPU system to synchronize motor startup to limit current surge simply by communication between tasks or by scheduling. In a distributed system, such synchronization must be performed by CPU-to-CPU communication or back through a common control CPU.
- All the data is in the same place, making communication with a host or other systems easier. Fewer communication protocols are required to pass data around.
- Since there are fewer oscillators, there usually will be less EMI. On the other hand, a faster processor may be required, operating at a higher frequency and generating a lot of EMI.
- If the design changes so that intertask communication must be added, such as for motor synchronization, a distributed design may require that interfaces be added to each distributed CPU. In a single-CPU design, such a change is likely to be only to the software.
- It is easier to download or update code in a single-CPU system.
- Debug of a single-CPU system may be easier since all the functions are in a single place and all the interactions can be examined. Of course, these interactions as well as the task switching and general complexity of the code can complicate debug as well.
- Fewer development tools are needed since there is only one processor. In a distributed system, the same thing can be achieved by using only one type of CPU; however, this defeats the ability to match the CPU to the task.
- If an RTOS is used, there will be fewer license fees in a single-CPU system. However, a more complex, more expensive RTOS may be required.

With increasing processor power at decreasing cost, I think more single-CPU designs are to be expected. Some designs will take advantage of increased CPU horsepower to add new functions, such as real-time signal processing. But motors and other electromechanical devices are getting no faster, so systems that interact with these devices probably will use fewer, more powerful processors. Complex systems that use a single Pentium-class CPU and a few 8-bit microcontrollers as smart sensors would not be surprising.

Specifications Summary

Let's summarize the contents of the design documents described in this chapter before we look at the actual design in the rest of the book.

The requirements document describes:

- What the design or system is to do
- The user interface, if any
- Any external interfaces to other systems
- What the real world I/O consists of

Hardware specifications (one per board or subsystem) describe:

- The requirements, restated from engineering or requirements documents
- How the hardware implements the functionality
- The software interfaces to the hardware