separate them from the more general-purpose embedded processors. Since the microcontroller does not need to generate signals to external memory, the device pins are available for I/O. These pins are grouped as *ports*, and each pin may be an input or an output. In our example system, one pin might turn on the pool pump relay. Another pin might allow the processor to monitor the water level sensor.

Most microprocessor manufacturers make a controller with internal memory and external pins for controlling I/O devices. While it is impossible to list all the variations and subtleties of these devices here, a brief list of typical devices follows:

Manufacturer	*Processor*	*I/O Pins*
Intel and others	8031/8051 family	32
Microchip	PIC17C42	33
Motorola	68HC05 family	varies
Zilog	Z8 (Z86E40)	32
Signetics/Philips	83C751	19
Atmel	AT90S8515	32

This list cannot describe all the tradeoffs among the various parts. Some of these parts include a bidirectional serial interface, for example, but you must give up two port pins to use it. Some have internal timers that use a port pin for certain functions. Some have high-current and open-drain outputs that are ideal for driving relay or solenoid coils with no additional driver hardware. The specific IC that is ideal for your application depends on the application.

When counting I/O pins, make sure that you take into account the use of internal functions, such as serial ports and timers, that restrict the use of certain pins. Although we'll discuss this in more detail in Chapter 2, keep in mind that some of these parts support external RAM or ROM, but using that capability takes anywhere from 8 to 19 I/O pins to access the external memory.

Interfaces Required

The entire point of an embedded processor is to interact with some piece of real-world hardware. Not only must the hardware be in place to handle the interface, the processor must be fast enough to perform whatever processing must be done on the data. In a single-chip system, processor selection may be highly dependent on the interface requirements. For example, the Microchip PIC17C42 has two pulse-width modulation (PWM) outputs that simplify design of such things as antilock braking systems and motor servos. One caveat: Study the data sheets carefully. Many processors have limitations that are not immediately obvious. You might find, say, that the serial port is specified as being able to operate at a certain

maximum baud rate, but careful examination of the data sheet may reveal that not all modes of operation are available at the maximum rate.

Determining whether a particular processor can keep up with the interface requirements is not always easy. Unfortunately, there is no magic formula to determine this. I have frequently resorted to writing part of the code for an interface just to be sure that the processor has enough capacity.

Memory Requirements

Determining the memory requirements is an essential part of embedded system design. If you overestimate the memory required, you may select an unnecessarily expensive solution. If you underestimate it, you risk project delays while the system is redesigned. Since memory comes only in sizes that are addressable with digital bits, such as 8K × 8, 32K × 8, and so on, you need not estimate memory requirements down to the last byte. You do need to ensure that you have enough memory, however.

RAM RAM is fairly straightforward to estimate. The number of variables plus the sum of all internal buffers, FIFOs (first in, first out), and stacks is the amount of RAM required. Many single-chip microcontroller ICs are limited to less than 1024 bytes (1K, or 1 kilobyte) of internal memory. If the memory goes beyond what is internally available, then external RAM must be added. However, this requires the use of I/O pins to address the added memory and often defeats the purpose of using a single-chip controller.

One caution is important: Some microcontrollers have restrictions on RAM usage, such as the need to use part of the internal RAM for banks of internal registers. For a couple of examples, look at the 8031, which has 128 bytes of internal RAM. The 8031 has four register banks that use 32 bytes of that, leaving 96 usable bytes of RAM. If your application needs only one or two register banks, the rest is available for general use. The 8052 processor has 256 bytes of general-purpose RAM, but the upper 128 bytes are accessible only by using indirect addressing. The Atmel AVR90S8515 has 32 general-purpose registers, but only 16 can be used with the immediate data instructions.

The amount of RAM required also will vary with the development language used. Some inefficient compilers use enormous amounts of RAM.

ROM The amount of ROM required for a system is the sum of the program code and any ROM-based tables required. Examples of ROM tables are step motor ramp tables, data translation lookup tables, and indirect branch tables. The tables usually are straightforward to estimate. The difficult part is estimating the code size. Estimates of code size become more accurate with increasing experience, usually gained by being wrong. However, it is important to remember that being precise is