you give up the flexibility and programmability of having the design in firmware, at least as far as hardware costs go.

In-Circuit Programming

This is not a consideration for every design, but you sometimes need the capability to program the parts in-circuit—to perform field upgrades of the firmware, for instance. This can be a powerful feature, but the capability (or the lack of it) can affect which processor you choose. To use in-circuit programming, you must have a processor with the program stored in flash memory. You need a way to erase and program the memory without removing the device from the board.

I once developed a system that needed in-circuit programming. The microcontroller I wanted to use was available in an EPROM version, which must be erased using UV light. I needed to program the parts without taking them off the boards. There was a flash version of the microcontroller that could be erased and reprogrammed in-circuit, but it ran at only half the speed of the part I wanted to use. Another version was available with flash memory and was capable of running at the right speed, but it had additional, unneeded features that tripled the cost of the part. I had to compromise on cost or performance or give up in-circuit programming. Check this carefully if you need that capability.

Nonvolatile Storage

Sometimes your application requires internal nonvolatile storage. If you are building a television, you might want to remember what channel the set was on last, even if power is removed. For this, you will need some kind of nonvolatile storage that can be written and read by the processor. Many microcontrollers, such as the PIC and Atmel AT90S series, include a small amount of EEPROM on-chip.

Memory Architecture

Microprocessor memory architectures are divided into two broad camps: von Neumann and Harvard. The von Neumann architecture permits data and code to be intermixed. You can put a data table in PROM with the code, and you can move code to RAM and execute it there. If the code is in RAM, it can modify itself by writing to the code area of RAM.

The Harvard architecture has separate code and data areas. Code executes from PROM (usually), data comes from a separate RAM, and you cannot get data from the code space. Most microprocessors that use the Harvard architecture actually use a *modified* Harvard architecture in which the code and data areas are separate, but a limited ability exists to get data from the code area. This allows tables or other information to be compiled into the code for use at runtime. This usually is

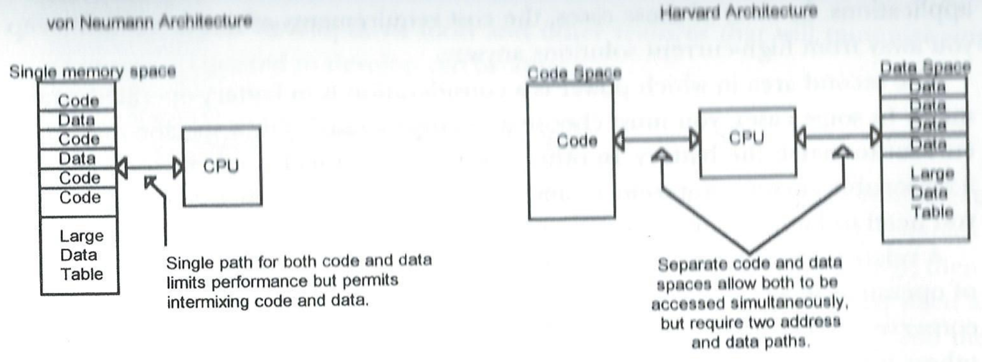
von Neumann Architecture
Single memory space
Code
Data
Code
Data
Code
Code
Large Data Table
CPU
Single path for both code and data limits performance but permits intermixing code and data.
Harvard Architecture
Code Space
Code
CPU
Data Space
Data
Data
Data
Data
Large Data Table
Separate code and data spaces allow both to be accessed simultaneously, but require two address and data paths.

Figure 1.3
Von Neumann versus Harvard Architecture.

implemented with a small number of pointers that can retrieve data from the code space and with the inclusion of immediate instructions where a byte (or word) of data is included in the instruction itself. Many single-chip microcontrollers use the Harvard architecture, among them the 8031, Microchip PIC series, and Atmel AVR90S series. Figure 1.3 shows the relative characteristics of the von Neumann and Harvard architectures.

The advantage of the Harvard architecture is that there are two separate memory areas and often two separate data paths, so code and data can be fetched simultaneously, increasing the throughput of the processor. From an embedded system point of view, the difference between the architectures is important if compiled data tables are needed. For example, a stepper motor controller may have a number of ramp tables embedded in the code space.

If you choose a processor with a modified Harvard architecture, be sure the table lookup features of the instruction set will not bog down the code. If you are considering an 8031 for this application, you will find that it has several registers that can be used as pointers into the data RAM but only one register (DPTR) that can be used as a pointer into the code PROM. An application that must simultaneously use two tables in PROM constantly switches DPTR between the two pointers. One solution to this is to move one or both tables into RAM, but then you must make sure enough additional RAM is available to hold the tables.

Power Requirements

In some designs, power is not an issue—you just put in whatever power supply you need to run whatever the electronics requires. There are two areas in which power is normally an issue. The first is designs that have a power restriction, such as the need to operate from a wall-mounted power supply as found in many consumer