CHAPTER 13

Microcontrollers

The microcontroller is essentially a computer on a chip. It contains a processing unit, ROM, RAM, serial communications ports, ADCs, and so on. In essence, a microcontroller is a computer, but without the monitor, keyboard, and mouse. These devices are called *microcontrollers* because they are small (micro) and because they control machines, gadgets, and so on.

With one of these devices, you can build an "intelligent" machine. You write a program on a host computer; download the program into the microcontroller via the USB, parallel, or serial port of the PC; and then disconnect the programming cable and let the program run the machine. For example, in the microwave oven, a single microcontroller has all the essential ingredients to read from a keypad, write information to the display, control the heating element, and store data such as cooking time.

There are literally thousands of different kinds of microcontrollers available. Some are one-time-programmable (OTP), meaning that once a program is written into its ROM (OTP-ROM), no changes can be made to the program. OTP microcontrollers are used in devices such as microwaves, dishwashers, automobile sensor systems, and many application-specific devices that do not require changing the core program. Other microcontrollers are reprogrammable, meaning that the microcontroller's program stored in ROM (which may be EPROM, EEPROM, or flash) can be changed if desired, which is a useful feature when prototyping or designing test instruments that may require future I/O devices.

Microcontrollers are found in bicycle light flashers, data loggers, toys such as model airplanes and cars, antilock braking systems, VCRs, microwave ovens, alarm systems, fuel injectors, exercise equipment, and many other items. They also can be used to construct robots, where the microcontroller acts as the robot's brain, controlling and monitoring various input and output devices, such as light sensors, stepper and servo motors, temperature sensors, and speakers. With a bit of programming, you can make the robot avoid objects, sweep the floor, and generate various sounds to indicate that it has encountered difficulties (such as being low on power or tipped over) or has finished sweeping. The list of applications for microcontrollers is endless, and because they are so widely used, their cost is low.

13.1 Basic Structure of a Microcontroller

Figure 13.1 shows the basic ingredients found within many microcontrollers. These include a CPU, ROM (OTP-ROM, EPROM, EEPROM, of flash), RAM, I/O ports, timing circuitry/leads, interrupt control, a serial port adapter (such as UART or USART), and an ADC/DAC.

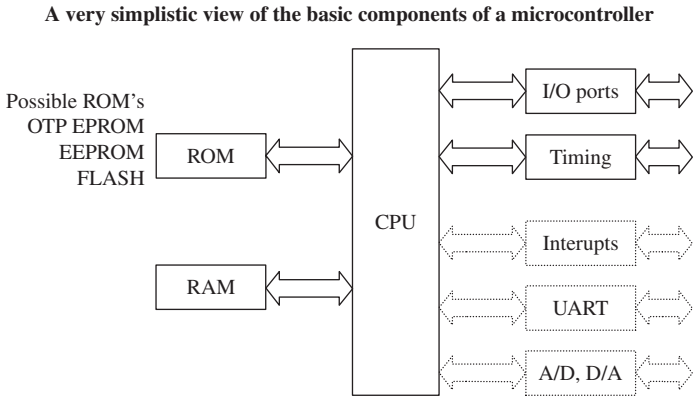

FIGURE 13.1 Possible internal architectures: RISC, SISC, CISC, Harvard, Von-Neuman

The CPU retrieves program instructions that the user programs into ROM, while using RAM to store temporary data needed during program execution. The I/O ports are used to connect external devices that send or receive instructions to or from the CPU.

The serial port adapter is used to provide serial communications between the microcontroller and a PC or between two microcontrollers. It is responsible for controlling the different rates of data flow common between devices. Example serial port adapters found within microcontrollers are the universal asynchronous receiver transmitter (UART) and universal synchronous/asynchronous receiver transmitter (USART). The UART can handle asynchronous serial communications, while the USART can handle either asynchronous or synchronous serial communications. Some microcontrollers take this a step further and include an interface for a Universal Serial Bus (USB) interface on the chip.

An interrupt system is used to interrupt a running program in order to process a special routine called the *interrupt service routine*. This boils down to the ability of a microcontroller to respond to external data that requires immediate attention, such as data conveyed by an external sensor indicating important shutdown information, say, when things get too hot or objects get too close. A timer/counter is used to "clock" the device—to provide the driving force needed to move bits around. Most microcontrollers that come with built-in ADCs and DACs can be used to interface with analog transducers, such as temperature sensors, strain gauges, and position sensors.

13.2 Example Microcontrollers

There are many different families of microcontrollers. Two of the most popular are made by the manufacturers Atmel and Microchip. In this section, we will take a close look at microcontrollers from these two manufacturers.

13.2.1 The ATtiny85 Microcontroller

The Atmel ATtiny85 microcontroller is an 8-pin IC available in both surface-mount and through-hole DIL packages. The device is designed to operate with a minimum of external components. Figure 13.2 shows what is inside one of these little packages.

ATtiny block diagram

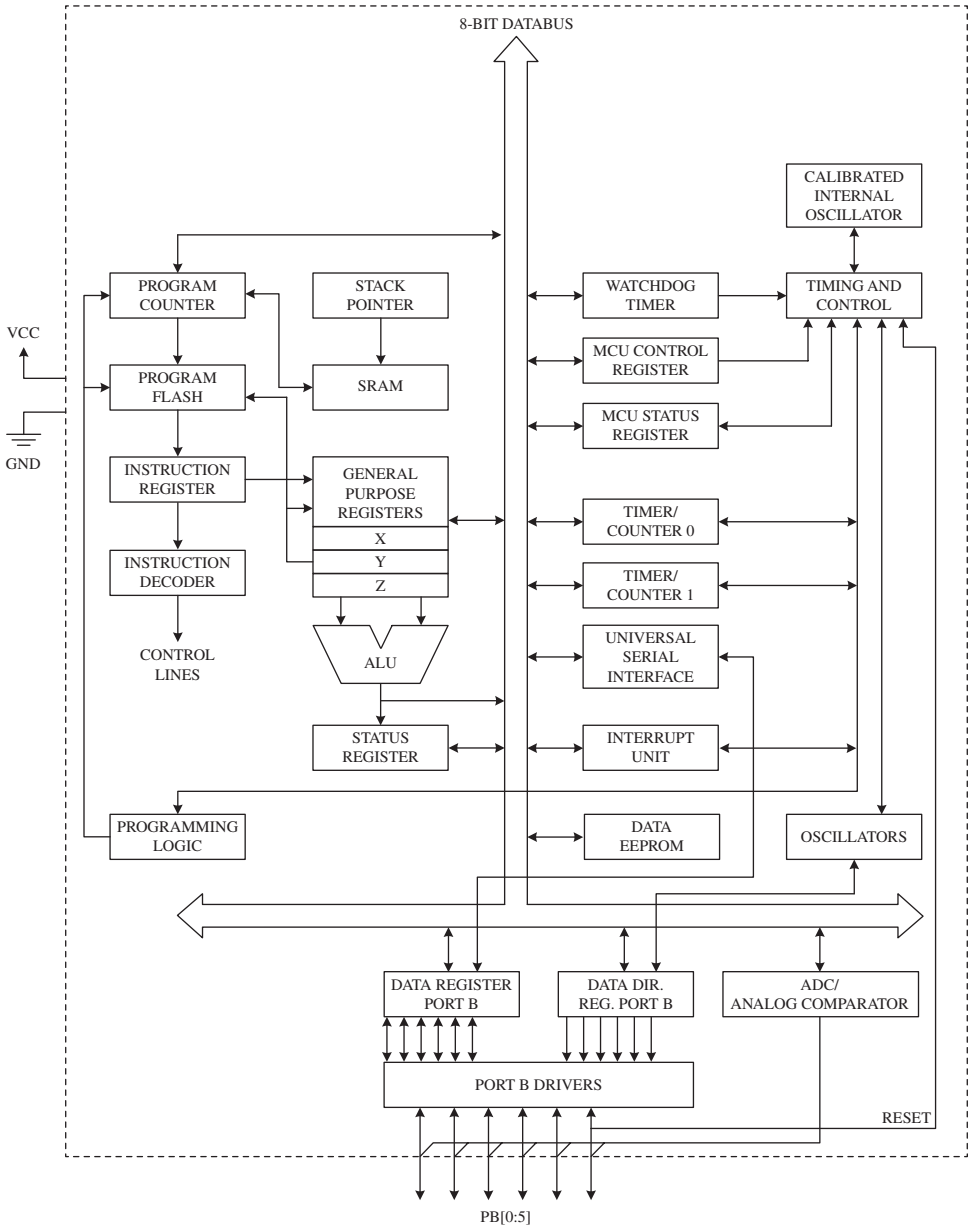

FIGURE 13.2

The ATtiny 85 has three different types of memory:

- 8 kB of flash memory in which the program instructions are stored.
- 256 bytes of SRAM, which is used to contain data during execution of instructions.
- 512 bytes of EEPROM that is used to store nonvolatile data that needs to be retained after a loss of power.

The watchdog timer allows the microcontroller to be put into a sleep mode, in which it consumes negligible power. The watchdog timer will wake up the microcontroller after a certain amount of time has passed.

The device can either use the inaccurate internal oscillator or two of the pins that would otherwise be used for inputs or outputs can be sacrificed to use an external crystal oscillator.

Two timers can be used to generate internal interrupts; that is, to trigger some code to be executed periodically. External interrupts that are triggered by a change in the level at a pin are also possible.

All of the I/O pins can also be used with the internal ADC.

The ATtiny also has a Universal Serial Interface, which can communicate with a number of different types of serial buses including USB, Inter-Integrated Circuit (I^2C), and serial. There is more about these serial protocols in Sec. 13.5.

Minimizing External Components

Figure 13.3 illustrates just how few components you need to make something with an ATtiny. The potentiometer is connected to a pin that will be used as an analog input, which could, for example, be used to control the rate at which the LED flashes.

ATtiny85 LED flasher

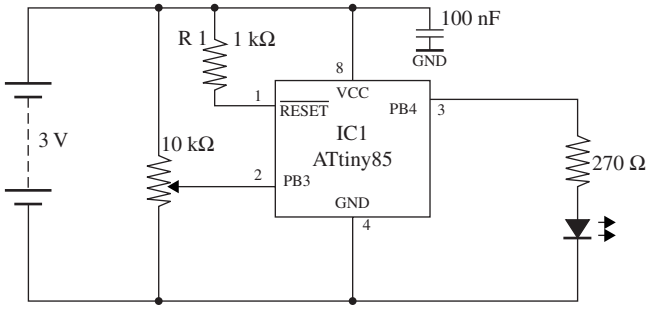

FIGURE 13.3

The resistor R1 could be replaced by a direct connection from the RESET pin to VCC, but by using a resistor here, it becomes possible to have the RESET pin low force a reset—something that is necessary during programming.

The chip will operate from a supply voltage of between 2.7 V and 5.5 V at a clock frequency of 10 MHz or below, making it suitable for running from a 3 V lithium cell or a pair of AA batteries. The clock frequency can be set during programming, and it can also be changed from program code while the ATtiny is actually running. The main reason for controlling the clock frequency is to reduce the power consumption. At 1 MHz, the power can be reduced to just 300 μA and in power-down mode, waiting for an interrupt from the watchdog timer, power consumption is just 0.1 μA.

You may be wondering why we used a microcontroller to create something that we could have made with a 555 timer. Well, why not? The microcontroller is more expensive than a 555 timer, but only around a dollar, and also we can use slightly fewer components with a microcontroller. We also have a great deal more flexibility. Using this same hardware, we could do clever tricks such as turning the LED completely off if the potentiometer is at its most counterclockwise position. And there are three other unused I/O pins that we could do something with.