CHAPTER 2

Introduction to Microprocessor-Based Control

OBJECTIVES

After studying this chapter, you should be able to:

- Understand what a microprocessor is, what it does, and how it works.
- Understand the concepts of RAM and ROM computer memory and how memory is accessed via the address and data buses.
- Understand how parallel and serial data interfaces work.
- Perform relevant calculations pertaining to analog-to-digital converters and digital-to-analog converters.
- Understand the principles of digital controller software.
- Recognize and describe the characteristics of the various types of available digital controllers, that is, microcontrollers, single-board computers, programmable logic controllers, and personal computers.

INTRODUCTION

The digital integrated circuit (IC) called a microprocessor [Figure 2.1(a)], has ushered in a whole new era for control systems electronics. This revolution has occurred because the microprocessor brings the flexibility of program control and the computational power of a computer to bear on any problem. Automatic control applications are particularly well suited to take advantage of this technology, and microprocessor-based control systems are rapidly replacing many older control systems based on analog circuits or electromechanical relays. One of the first microprocessor-based controllers made specifically for control applications was the programmable logic controller (PLC), which is discussed later in this chapter and in Chapter 12. A microprocessor by itself is not a computer; additional components such as memory and input/output circuits are required to make it operational. However, the microcontroller [Figure 2.1(b)], which is
a close relative of the microprocessor, does contain all the computer functions on a single IC. Microcontrollers lack some of the power and speed of the newer microprocessors, but their compactness is ideal for many control applications; most so-called microprocessor-controlled devices, such as vending machines, are really using microcontrollers. Some specific reasons for using a digital microprocessor design in control systems are the following:

- Low-level signals from sensors, once converted to digital, can be transmitted long distances virtually error-free.
- A microprocessor can easily handle complex calculations and control strategies.
- Long-term memory is available to keep track of parameters in slow-moving systems.
- Changing the control strategy is easy by loading in a new program; no hardware changes are required.
- Microprocessor-based controllers are more easily connected to the computer network within an organization. This allows designers to enter program changes and read current system status from their desk terminals.
In this chapter, we will present the basic concepts of a microprocessor- and microcontroller-based system with particular emphasis on control system applications. It is by no means an in-depth treatment, but enough to make the rest of the text more meaningful.

In the first sections of this chapter the basic concepts of microprocessor hardware and operation are introduced (these concepts also apply to microcontrollers). I have included this material because the student of modern control systems should have at least a general knowledge of how the microprocessor performs its job.

2.1 INTRODUCTION TO MICROPROCESSOR SYSTEM HARDWARE

A computer is made up of four basic functional units: the central processing unit (CPU), memory, input, and output (I/O). The central processing unit does the actual computing and is composed of two subparts: the arithmetic logic unit and control sections (Figure 2.2). The arithmetic logic unit (ALU) performs the actual numerical and logic calculations such as addition, subtraction, AND, OR, and so on. The control section of the CPU manages the data flow, such as reading and executing the program instructions. If data require calculations, the control section hands it over to the ALU for processing. In a microprocessor-based computer, the microprocessor is the CPU.

Figure 2.2
A block diagram of a microprocessor-based computer.
Digital data is in the form of **bits**, where each bit has a value of either 1 or 0. Digital circuits usually use 5 Vdc to represent logic 1 and 0 Vdc to represent logic 0. Eight bits together is called a **byte**. A microprocessor handles digital data in **words**, where a word may be 8, 16, or 32 bits wide. For example, an 8-bit microprocessor has a byte-sized word, with a maximum decimal value of 255. (Computers represent numbers in the **binary number system**; for example, 11111111 binary = 255 decimal.) The rightmost bit in a binary number has the least value (usually 1) and is called the **least significant bit (LSB)**. The leftmost bit represents the highest value and is called the **most significant bit (MSB)**. The conversion between binary and decimal can be performed directly with most scientific calculators or manually using the technique shown in Example 2.1. To express values larger than 255, two or more words are put together. In this text, we will assume 8-bit microprocessors are used unless otherwise stated.

**EXAMPLE 2.1**

Find the decimal value of the 8-bit binary number 10110011.

**SOLUTION**

Each bit in the binary number has a different value, or weight. The LSB has a weight of 1. The bit to the left of the LSB has a weight of 2, the third bit has a weight of 4, and so on, with the weight doubling for each bit up to 128 for the MSB. To find the value of an 8-bit number, you can set up a chart (shown below) and then sum the values that correspond to the 1s in the binary number.

<table>
<thead>
<tr>
<th>MSB</th>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th>LSB</th>
<th>Bit weights</th>
<th>Binary number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1 x 1 =     1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 x 1 =     2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 x 0 =     0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8 x 0 =     0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16 x 1 =    16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32 x 1 =    32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64 x 0 =    0</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128 x 1 = 128</td>
<td></td>
</tr>
<tr>
<td></td>
<td>179</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

The **memory** section of the computer is a place where digital data in binary form (1s and 0s) are stored. Memory consists of cells organized in 8-bit groups. Each byte
is given a unique numeric address, which represents its location just as a street address represents the location of a house. Data are written into memory and read out of memory, based solely on their address. In a particular memory circuit, the addresses might start at 1000 and run consecutively to 2000. Figure 2.3 diagrams a section of memory. Note that the first byte of data has a decimal value of 2 (00000010 = 2 decimal) and an address of 1000.

Computers usually have two kinds of addressable memory. The first is random-access memory (RAM), which allows the computer to read and write data at any of its addresses (it is also called read/write memory, or RWM). All data in this type of memory are lost when the power is turned off and is called volatile memory (an exception is designs where RAM is kept “alive” with a small battery). The second type of memory is read-only memory (ROM), which is similar to RAM except that new data cannot be written in; all data in ROM are loaded at the factory and cannot be changed by the computer. This memory does not lose its data when power is turned off and is called non-volatile memory. Most microprocessor systems have both RAM and ROM. RAM is used for temporary program storage and as a temporary scratch-pad memory for the CPU. ROM is used to store programs and data that need to be always available. Actually, many computers use an EPROM (erasable programmable read-only memory) or an EEPROM (electrically erasable programmable ROM) instead of a ROM for long-term memory. EPROMs can be erased with a strong UV light and reprogrammed. EEPROMs can be erased and reprogrammed electrically. Disk drives also store digital data but in a form that must be processed before they are accessible to the microprocessor.

The input/output (I/O) section of the computer allows it to interface with the outside world. The input section is the conduit through which new programs and data are entered into the computer, and the output section allows the computer to communicate its results. An I/O interface is called a port. An input port is a circuit that connects input devices to the computer; examples of input devices are keyboards, sensors, and switches. An output port is a circuit that connects the computer to output devices. Examples of output devices are indicator lamps, actuators, and monitors. Input/output is discussed in more detail in the next section.

Referring again to Figure 2.2, we see that the blocks are connected by three lines labeled address bus, data bus, and control bus. The address bus is a group of wires that carries an address (in binary form) from the CPU to the memory and I/O circuits. The
need for memory to receive addresses has already been discussed, but you may wonder why I/O ports need addresses. It turns out that all I/O ports are assigned addresses and are treated essentially like memory locations by the CPU. The CPU outputs data to the outside world by sending them to a port address. When the circuitry of the designated output port detects its assigned address on the address bus, it opens and allows data to pass from the data bus to whatever is connected to the port. There are two ways that I/O addressing is done. Some microprocessors use what is called memory-mapped input/output, where an I/O address is treated just like another memory address. Other microprocessors treat I/O addresses completely separate from memory addresses.

The data bus is a group of eight wires that carries the actual numerical data from place to place within the computer. Figure 2.2 shows how the data bus interconnects all blocks. Data flow in both directions on the data bus. For example, input data enters through the input port and proceeds through the data bus to the CPU. If the CPU needs to store these data, it will send them back through the data bus to memory. Data to be outputted are sent (by the CPU) through the data bus to the output port. If the data bus connects to all blocks, how do the data know which block to go to? The answer is the address system. For example, when the CPU sends data to memory, it does it in two steps: First, it puts the destination memory address on the address bus; second, it puts the data on the data bus. When the designated memory detects its own address, it “wakes up” and takes the data from the data bus. The other blocks connected to the buses will ignore the whole sequence because they were not addressed. A good analogy here is the phone system, where the phone number is analogous to the memory address. Even though thousands of phones may be connected to the system, when you dial a number, only the designated phone rings. The beauty of the bus system is that it is expandable. Memory or addressable I/O units can be added to the system by simply connecting them to the buses.

The control bus (see Figure 2.2) consists of timing and event-control signals from the CPU. These signals are used to control the data flow on the data bus. For example, one of the control signals is the read/write (R/W) line. This signal informs the memory if the CPU wishes to read existing data out of memory or write new data into memory. Non-memory-mapped machines have a memory-I/O control line. This signal informs the system if the current data exchange involves memory or an I/O port. In general, the control bus is not as standardized as are the address and data buses.

### 2.2 INTRODUCTION TO MICROPROCESSOR OPERATION

The microprocessor works by executing a program of instructions. Creating the program is similar in concept to programming in BASIC, C, or any other high-level computer language. Each type of microprocessor has its own instruction set, which is the set of commands that it was designed to recognize and obey. Microprocessor instruc-
tions are very elemental and specific, and it usually takes more than one to accomplish what a single, high-level language instruction would. Many microprocessor instructions simply move data from one place to another within the computer; others perform mathematical or logic operations. Still another group of instructions control program flow, such as jumping forward or backward in the program. Each instruction in the instruction set is assigned its own unique operation code, (which is typically 8 bits long and referred to as the op-code). The CPU uses this 8-bit number to identify the instruction.

All microprocessors have at least one accumulator [Figure 2.4(a)], which is a data-holding register in the CPU. The accumulator acts as a “staging area” for data. It is common for data coming to the CPU to go first to the accumulator, where it can be operated on. Similarly, most data leaving the CPU exits from the accumulator. Mathematical operations usually store the result in the accumulator. Many of the instructions involve the accumulator in one way or another.

A machine language program is a list of instructions (in op-code form) for the microprocessor to follow. Before the program can be executed, it must first be loaded sequentially into memory. The op-code for the first instruction is loaded at the first address location, the op-code for the second instruction is loaded next in line, and so on.

Figure 2.4(b) shows a section of memory with a short program loaded in. The program listing includes the address, op-code, mnemonic, and a brief explanation. (A mnemonic is an English abbreviation of an instruction. A program listing using only mnemonics is called assembly language.) The program in Figure 2.4(b) directs the CPU to get 1 byte of data from input port 01, add 1 to it, and send the result to output port 02. Before execution can start, the address of the first instruction must be loaded into the program counter. The program counter is a special address-storage register that the CPU uses to keep track of where it is in the program, much like a bookmark. The program counter always holds the address of the next instruction to be executed. Once the

---

**Figure 2.4**

The CPU uses an accumulator and a program counter to execute a simple program.

<table>
<thead>
<tr>
<th>Adr</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>DB IN 01</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>02</td>
<td>3C INRA</td>
</tr>
<tr>
<td>03</td>
<td>D3 OUT 02</td>
</tr>
<tr>
<td>04</td>
<td>02</td>
</tr>
<tr>
<td>05</td>
<td>76 HLT</td>
</tr>
</tbody>
</table>

Op codes are for Intel 8085 in hexadecimal. In this case, hexadecimal is used as a shorthand form of binary.
CHAPTER 2

microprocessor is activated, execution of the program is completely automatic. The execution process is a series of *fetch-execute cycles*, whereby the microprocessor first fetches the instruction from memory and then executes it. The following are the specific steps the microprocessor would go through to execute the program of Figure 2.4(b):

1. The microprocessor fetches the first instruction from memory. It knows where to find the instruction because its address is in the program counter.

2. Once in the CPU, the op-code is decoded to see which instruction it is, then the proper hardware is activated to execute this instruction. In the example program of Figure 2.4(b), the first instruction (IN 01) is 2 bytes long. The first byte of the instruction is the op-code, telling the CPU to input data from a port. The second byte of the instruction tells the CPU which port to read from. Execution of this instruction causes data from input port 01 to travel along the data bus to the accumulator. Also, the program counter advances to 02 (the address of the next instruction). Execution of the first instruction is now complete.

3. The next fetch-execute cycle starts, this time fetching the instruction from address 02. The new instruction (INR A) is “increment the accumulator,” so the accumulator is sent to the ALU to be incremented (add 1) and the result put back in the accumulator. The program counter advances to 03, which is the address of the next instruction.

4. The next fetch-execute cycle starts, this time fetching the instruction from address 03. The instruction (OUT 02) is executed, causing the accumulator data to be sent to output port 02.

5. The final instruction is fetched. It is a “halt,” which causes the microprocessor to cease operating and go into a wait mode.

2.3 INTERFACING TO A MICROPROCESSOR CONTROLLER

An important part of any control system is the link between the controller and the real world. For a digital controller, data enter and exit through a parallel interface or through a serial interface. Both data formats are discussed next.

The Parallel Interface

The *parallel interface* transfers data 8 bits (or more) at the same time, using eight separate wires. It is essentially an extension of the data bus into the outside world. The parallel interface is ideal for inputting or outputting data from devices that are either on or off. For example, a single limit switch uses only one input bit, and an on-off signal to a motor requires only one output bit. These 1-bit signals are called *logic variables*, and eight such signals can be provided from a single (8-bit) port. This concept will be expanded on later in this section.
In other applications, the controller may use a parallel interface to connect to an analog device—for example, driving a variable-speed DC motor. In such a case, the binary output of the controller must first be converted into an analog voltage before it can drive the motor. This operation is performed by a special circuit called a digital-to-analog converter.

**Digital-to-Analog Conversion**

The digital-to-analog converter (DAC) is a circuit that converts a digital word into an analog voltage. It is not within the scope of this text to describe the internal workings of the DAC, but a general understanding of the operating parameters is appropriate.

Figure 2.5 shows the block diagram of a typical 8-bit DAC. The input is an 8-bit digital word. The output is a current that is proportional to the binary input value and must be converted to a voltage with an op-amp. A stable reference voltage ($V_{ref}$) must be supplied to the DAC. This voltage defines the maximum analog voltage—that is, for a digital input of 11111111, $V_{out}$ is essentially $V_{ref}$. If the input is 00000000, the $V_{out}$ will be 0 Vdc. For all values in between, the output voltage is a linear percentage of $V_{ref}$. Specifically, the output voltage for any digital input (for the 8-bit DAC) is

$$V_{out} = \frac{{\text{input} \times V_{ref}}}{{256}}$$

(2.1)

where

- $V_{out}$ = DAC output analog voltage
- input = decimal value of the binary input
- $V_{ref}$ = reference voltage to the DAC

**EXAMPLE 2.2**

An 8-bit DAC has a $V_{ref}$ of 10 V. The binary input is 10011011. Find the analog output voltage.
CHAPTER 2

An important consideration of digital to analog conversion is resolution. The resolution of a DAC is the worst case error that is introduced when converting between digital and analog. This error occurs because digital words can only represent discrete values, as indicated by the stair-step diagram in Figure 2.5. For example, the maximum value of an 8-bit number is 255 decimal, which means there are 255 possible “steps” of the output voltage. The difference between steps is the value of the least significant bit (LSB). Because the smallest increment is one step, the resolution (for 8-bit data) is 1 part in 255, or 0.39%. This resolution is adequate for many applications, but if more is needed, two (or more) 8-bit ports can be used together. Two ports provide 16 bits of data. The maximum decimal value of 16 bits is 65,535. Being able to divide an analog number into 65,535 parts means that each part will be much smaller, so we can more precisely represent that number.

EXAMPLE 2.3

A computer uses a DAC to create a voltage that represents the position of an antenna. The antenna can rotate 180° and must be positioned to within 1°. Can an 8-bit port be used?

SOLUTION

The resolution required is 1 part in 180. Because 8 bits provide a resolution of 1 part in 255, an 8-bit port is certainly adequate. In fact, we have a choice: We could have the LSB = 1°, in which case the input values would range from 0 to 180, or we could equate 180° with 255, which makes the LSB = 0.706°. The latter makes maximum use of the 8 bits to give a better resolution, but if the system really doesn’t need it, the clear, simple relationship of LSB = 1° is desirable.
Figure 2.6 shows a data sheet for an 8-bit DAC (DAC0808). This device comes as a 16-pin DIP (dual in-line package) and uses an external op-amp (such as the LF351), two resistors, and a capacitor to complete the circuit. It requires plus and minus power-supply voltages. The time to complete a conversion is a fast 150 ns (nanoseconds). (The circuit shown in Figure 2.6 has a $V_{\text{ref}}$ of 10 Vdc.)

### National Semiconductor

**DAC0808, DAC0807, DAC0806 8-Bit D/A Converters**

**General Description**

The DAC0808 series is an 8-bit monolithic digital-to-analog converter (DAC) featuring a full scale output current settling time of 150 ns while dissipating only 33 mW with ±5V supplies. No reference current (IREF) trimming is required for most applications since the full scale output current is typically ±1.56 of 255 IREF/256. Relative accuracies of better than ±0.19% assure 8-bit monotonicity and linearity while zero level output current of less than 4 μA provides 8-bit zero accuracy for IREF ≥ 2 mA. The power supply currents of the DAC0808 series are independent of bit codes, and exhibits essentially constant device characteristics over the entire supply voltage range.

The DAC0806 will interface directly with popular TTL, DTL, or CMOS logic levels, and is a direct replacement for the MC1408/1408. For higher speed applications, see DAC0800 data sheet.

**Features**

- Relative accuracy: ±0.19% error maximum (DAC0808)
- Full scale current match: ±1 LSB typ
- 7 and 6-bit accuracy available (DAC0807, DAC0806)
- Fast settling time: 150 ns typ
- Noninverting digital inputs are TTL and CMOS compatible
- High speed multiplying input slew rate: 8 mA/μs
- Power supply voltage range: ±4.5V to ±18V
- Low power consumption: 33 mW @ ±5V

**Block and Connection Diagrams**

![Block Diagram]

**Typical Application**

![Typical Application Diagram]
Analog-to-Digital Conversion

An analog-to-digital converter (ADC) is a circuit that converts an analog voltage into a digital word. A typical ADC consists of a single IC with a few support components. Analog-to-digital conversion is a more complicated process than for the DAC, and the hardware requires some conversion time, which is typically in the microsecond range. The conversion time required depends on the type of ADC, the applied clock frequency, and the number of bits being converted. Figure 2.7 shows a block diagram for an 8-bit ADC. The input \( V_{in} \) can be any voltage between 0 V and \( V_{ref} \). When \( V_{in} \) is 0 Vdc, the output is 00000000; when \( V_{in} \) is \( V_{ref} \), the output is 11111111 (255 decimal). For input voltages between 0 and \( V_{ref} \), the output increases linearly with \( V_{in} \); therefore, we can develop a simple ratio for the ADC:

\[
\frac{\text{output}}{V_{in}} = \frac{255}{V_{ref}} \quad \text{(for 8 bits)}
\]

Solving for output gives the following:

\[
\text{Output} = \frac{V_{in} \times 255}{V_{ref}} \quad (2.2)
\]

where

- output = decimal output value of an 8-bit ADC
- \( V_{in} \) = analog input voltage to the ADC
- \( V_{ref} \) = ADC reference voltage

To start the conversion process, a start-conversion pulse is sent to the ADC. The ADC then samples the analog input and converts it to binary. When completed, the ADC activates the data-ready output. This signal can be used to alert the computer to read in the binary data.

**Figure 2.7**

Analog-to-digital converter (ADC) block diagram.
Figure 2.8
The data sheet for the ADC0804, an 8-bit analog-to-digital converter.
(Courtesy of National Semiconductor Corp.)

Analog-to-Digital Converters

ADC0801, ADC0802, ADC0803, ADC0804 8-Bit \( \mu \)P Compatible A/D Converters

General Description

The ADC0801, ADC0802, ADC0803, ADC0804 are CMOS 8-bit, successive approximation A/D converters which use a modified potentiometric ladder—similar to the 256R products. They are designed to meet the NSC MICROBUS™ standard to allow operation with the 8080A control bus, and TRI-STATE® output latches directly drive the data bus. These A/Ds appear like memory locations or I/O ports to the microprocessor and no interfacing logic is needed.

A new differential analog voltage input allows increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

Features

- MICROBUS (8080A) compatible—no interfacing logic needed
- Easy interface to all microprocessors, or operates “stand alone”
- Differential analog voltage inputs
- Logic inputs and outputs meet \( T^2L \) voltage level specifications
- Works with 2.5V (L333) voltage reference
- On-chip clock generator
- 0V to 5V analog input voltage range with single 5V supply
- No zero adjust required
- 0.3” standard width 20 pin DIP package

Key Specifications

- Resolution 8 bits
- Total error \( \pm 1/4 \) LSB, \( \pm 1/2 \) LSB and \( \pm 1 \) LSB
- Conversion time 100 \( \mu \)s
- Access time 135 ns
- Single supply 5 Vdc
- Operates ratiometrically or with 5 Vdc, 2.5 Vdc, or analog span adjusted voltage reference

Typical Applications

Connection Diagrams
Figure 2.8 shows a data sheet for an 8-bit ADC (ADC0804). Packaged as a 20-pin DIP, this device can operate on a single 5-Vdc power supply and requires an external resistor and capacitor to complete the ADC circuit. The start-conversion pulse is applied to pin 3 (WR), and the data-ready signal comes from pin 5 (INTR). This particular ADC can be connected in a free-running mode where it performs one conversion after the other as fast as it can. Notice also that the pin labeled Vrefa (pin 9) must be set at half of the actual Vref. For example, if the requirements call for an analog voltage range of 0-5 Vdc, then pin 9 would be set to 2.5 Vdc. The time to complete a conversion is approximately 100 μs (micro-seconds), making it almost 700 times slower than the DAC0808 discussed earlier.

A Control System Using Parallel Ports

Figure 2.9 shows a position control system using a microprocessor-based controller with parallel ports. This particular system has one output port and three input ports (each port has its own address). The output port is partitioned: Six bits are converted in a DAC to provide the analog motor-drive signal, the seventh bit specifies motor direction (1 = clockwise, 0 = counterclockwise), and the eighth bit turns an audio alarm if some emergency situation is detected. The first input port inputs the set-point data, the second inputs the ADC data from the sensor, and the third inputs various 1-bit logical variables. In this case, the system has three front-panel switches as well as two limit

EXAMPLE 2.4

An 8-bit ADC has a $V_{\text{ref}}$ of 7 Vdc; the analog input is 2.5 Vdc. What is the binary output of the ADC?

SOLUTION

The output is an 8-bit word that has a maximum decimal value of 255 (decimal) when $V_{\text{in}} = V_{\text{ref}}$. Therefore, an analog input voltage ($V_{\text{in}}$) of 7 Vdc would be converted to 255 decimal. Using this set of I/O data, we can develop a ratio and then use that to find the output for the specific input of 2.5 Vdc:

$$\frac{\text{output}}{V_{\text{in}}} = \frac{255}{7 \ \text{Vdc}}$$

Solving for output gives the following:

$$\text{Output} = \frac{2.5 \ \text{Vdc} \times 255}{7 \ \text{Vdc}} = 91$$

The result is 91 decimal = 01011011 binary. This is the output that would appear on the eight output lines of the ADC.
Figure 2.9
A control system using parallel interface.

Microprocessor controller
Output port
Adr = 00

Analog Switches to input set point in binary.
(Gnd = logic 0
open = logic 1)

Input port
Adr = 01

Input port
Adr = 02

Input port
Adr = 03

DAC
Motor control
Motor
Load

Direction

Switches

Sensor

Limit switches

Front panel

Start
Stop
Go-to position
switches. The limit switches are used as a “back up” to detect it if the load has gone out of its designated range.

Operation of the system proceeds as follows: The controller inputs the data from port 03 to determine if the start (or stop) button has been pressed. If the start button has been pressed, then the set point is read in from port 01 and the digitized sensor data is read in from port 02. Based on its control strategy, the controller outputs to port 00 a binary word representing the motor-control voltage. This digital data is converted to an analog voltage with the DAC. This entire sequence is repeated over and over until the stop button is pushed.

The Serial Interface

In a serial interface, the data are sent 1 bit after the other on a single wire. There are a number of good reasons for doing this. First, the cabling is simpler because only two wires are needed (at a minimum), those being “data” and “return.” Second, shielding a small group of wires, which is often necessary in an electrically noisy industrial environment, is easier. Third, serial data can make use of existing single-channel data lines such as the telephone system (which may require using a modem). For these reasons, serial data transfer is usually recommended for distances greater than 10-30 ft.

Because data always exist in a parallel form inside the computer, it must be converted to serial data before coming out the serial port. This is accomplished with a special parallel-to-serial converter IC called a universal asynchronous receiver transmitter (UART). On the other end of the line, a receiver must convert the serial data back into parallel data, which is done with another UART. Figure 2.10 shows the basic serial data circuit.

Serial data are classified as being either synchronous or asynchronous. Synchronous data require that the data bytes be sent as a group in a “package.” It is used in sophisticated communication systems that move a lot of data and will not be further discussed here. Asynchronous data transfer is the more common (but slower) type of serial transfer and allows for individual bytes to be sent when needed.

Figure 2.10
Components in a serial interface circuit.
Figure 2.11 shows the standard format for asynchronous serial data. First, a start bit is sent, then the data (LSB first), then a parity-error checking bit, and finally the stop bit(s). Some variation is allowed to this format, but both transmitter and receiver must use the same format. The other important parameter in serial transmission is the number of bits sent per second (frequently called the baud rate, although the term is technically incorrect in most cases). Standard bit rates are 300 bps (bits per second), 1200 bps, 2400 bps, 9600 bps, 14,400 bps, 28,800 bps, 33,600 bps, and 57,600 bps. Serial data transmission is much slower than parallel transmission. At 300 bps, it takes almost 37 ms to transmit 1 byte of data, compared to less than a microsecond for parallel—this is thousands of times slower. Still, for many applications, particularly process control, the longer data-transfer times are not a problem.

**RS 232**

In order to make the serial interface practical, a set of specifications called the RS-232 standard was established. Officially, the **RS-232 standard** specifies the serial data interface between **data terminal equipment (DTE)** and **data communication equipment (DCE)**. A common application of RS-232 is the interface between a PC and the modem, in which case the computer is the DTE and the modem is the DCE [see Figure 2.12(a)]. A **modem** is a device that converts digital data into audio tones so that it can be transmitted over the telephone lines. As shown in Figure 2.12, the RS-232 interface consists of seven signals; the serial data is sent on pin 2 and received on pin 3; the other signals, such as “Request to send” and “Clear to send,” are used to confirm that the two units are ready to communicate. The RS-232 standard specifies connector types, signal names, pin numbers, and voltages. In practice, the RS-232 standard can be applied to any serial interface as long as one unit acts as a DTE and the other as a DCE. If two DTE units need to interface with each other—for example, a PC to a PC—a special cable called a **null modem** or **crossover cable** is used. RS-232 is commonly used in the control field when two units need to exchange data—for example, to connect a PC to a local control unit for the purpose of downloading a new control program, as illustrated in Figure 2.12(b).

RS-232 serial data transfer is somewhat more complicated than parallel data transfer, but it offers advantages such as two-wire communications and a universally accepted interface. The hardware to handle serial data is standardized, readily available, and reliable.
Chapter 2

Figure 2.12
The RS-232 serial interface.

(a) Interface between data terminal equipment (DTE) and data communication equipment (DCE). Serial data are transferred on pins 2 and 3; the other signals control the flow of data.

(b) Using an RS-232 serial cable to connect a PC to a controller

Networking

Probably the most common use of serial data is networking. More and more, networks are being used to interconnect all the units and devices in the control system. Network cabling differs depending on the type of local area network (LAN), but most use the generalized bus system diagrammed in Figure 2.13. Typically, each unit on the net has a unique address number and also address detection cirquitry. When one unit wants to talk to another unit, it first broadcasts the address of the unit it wants to talk to (serially, of course, on the signal wire) and then sends the data (seri-
ally), which consists of some number of bytes. All units on the net will receive the address, but only the intended receiver will activate and then read in the data. The interface between the network cable and the PC is done through a commercially available interface expansion card called a network interface card (NIC). Other devices on the net, such as control units, would require a special interface circuit, which may be built in or available as an external module. Control system networks are discussed further in Chapter 12.

### 2.4 INTRODUCTION TO CONTROLLER PROGRAMMING

It is beyond the scope of this text to present a detailed discussion of how to program a microprocessor in machine language. Still, it is useful to investigate in a general way what the software must do. A digital controller is a computer operating in real time. This means that the program is running all the time—repeatedly taking in the newest sensor data and then calculating a new output for the actuator.

The basic structure of a controller program is a loop. In a loop structure, the same sequence of instructions is executed over and over again, and each pass through the loop is called an iteration, or scan. Figure 2.14 shows a generalized controller program, and an explanation of the program follows:

1. The program reads in the set-point data (recall that the set point is the desired position of the controlled variable). This data could be read in from an input port or from memory.
2. The program directs the computer to read (from a sensor) the actual value of the controlled variable.
3. The actual data are subtracted from the set point to get the error.
4. Based on the error data, the computer calculates a new actuator control signal.
5. The new output is sent to the actuator.
6. The program loops back to step 1 and starts over again.

The time it takes for the computer to execute one pass through the loop determines the time interval between input readings (known as the sampling rate). If this interval is too long, the computer may not get an accurate picture of what the controlled variable is really doing (see Chapter 11 for a discussion of aliasing). Execution of the loop can be accelerated by specifying a faster computer or streamlining the program. In other situations, the computer must pause and wait. For example, a pause might be inserted to give an operator time to make some adjustment or to allow time for a motor to “spin down.” This is done by inserting time-delay loops in the program. A time-delay loop is simply a do-nothing, “wheel-spinning” loop where the computer is instructed to count up to some large number. Using this technique, we can make the program pause for any length of time—from a few microseconds to hours. If a time-delay loop is inserted in the main program loop (as shown in Figure 2.14), the effect is to slow the cycle time for the main loop. This is sometimes done to force matching of the sample rate to some predetermined value.

At one time, people thought that the best and most efficient microprocessor programs were those written directly in assembly language—that is, the programmer would directly select the machine language instructions. Today, sophisticated programs (called...
compilers) can convert a program written in a high-level language, primarily C, into very efficient machine language. High-level languages use English-sounding words and a set of powerful commands to specify simple and complicated programming operations with a minimum of instructions. Using a high-level language to write programs for a microprocessor offers big advantages, such as more compact program listing, ease of writing equations, and more comprehensible documentation. Also, programs written in a high-level language can be compiled to run on any model of microprocessor.

2.5 MICROPROCESSOR-BASED CONTROLLERS

Single-Chip Microcomputers (Microcontrollers)

A microprocessor by itself is not a computer. To be functional, the microprocessor must be connected to other integrated circuits that provide the memory and I/O capability. A microcontroller is a computer on a single IC, designed specifically for control applications. It consists of a microprocessor, memory (both RAM and ROM), I/O ports, and possibly other features such as timers and ADCs/DACs. Having the complete controller on a single chip allows the hardware design to be simple and very inexpensive. Microcontrollers are showing up increasingly in products as varied as industrial applications, home appliances, and toys. In such uses as these, they are called embedded controllers because the controller is located physically in the equipment being controlled.

The main difference between microprocessors and microcontrollers is that microprocessors are being designed for use in microcomputers where greater speed and larger word size are the driving requirements, whereas microcontrollers are evolving toward reduced chip count by integrating more hardware functions on the chip. Most control applications do not need the 32-bit word size and 500-MHz (megahertz) speed of the newer microprocessors. Eight or 16 bits and 1 MHz will work just fine in many applications, and the single-chip microcontroller costs much less.

Another difference between microprocessors and microcontrollers concerns the instruction set. The microprocessor tends to be rich in instructions dealing with moving data into and out of memory. The microcontroller has fewer memory-move instructions and more bit-handling instructions. The reason for the lack of memory-move instructions is that the microcontroller typically has only a small amount of RAM, which it uses only as a "scratch pad." The additional bit-handling instructions were included because they are so useful in control system applications. For example, in a control system, each separate bit of a parallel output word might control a different device, such as a motor or indicator light. The bit-handling instructions allow the software to turn one device easily on or off without affecting the others.

The Motorola 68HC11 is a popular 8-bit microcontroller that has 256 bytes of RAM, 8K of ROM, and 512K bytes of EPROM (see Figure 2.15a). It also has five 8-bit ports with built-in serial data transfer and ADC capability. Another common 8-bit
Figure 2.15
Block diagrams of microcontrollers.

(a) Motorola 68HC11 microcontroller block diagram

(b) Intel 8051 microcontroller block diagram

(c) PIC 16C72 microcontroller block diagram
microcontroller is the Intel 8051, which has 128 bytes of RAM and 4K bytes of ROM, four parallel data ports, and a serial port (see Figure 2.15b). For control applications, these hardware arrangements usually are adequate: ROM is used to store the control program, and RAM is used as data registers and a “scratch pad.” The I/O signal lines can usually be connected directly to the microcontroller without additional port circuitry. Software is typically written in C++ or some other language (including assembly language) and then converted into machine language with a compiler or assembler program. The machine language program would then be loaded into the microcontroller’s ROM or EPROM.

Another popular microcontroller is the PIC from Microchip Technology. For example, the PIC16Cxx family of 8-bit microcontrollers is a low-cost, versatile product that has found wide acceptance [see Figure 2.15(c)]. There are a wide range of options, including ROM, EPROM, EEPROM, ADCs, Timers, and serial ports. The PIC uses a slightly different architecture from the 68HC11 and 8051 in that the ROM (or EPROM) that contains the program connects to the CPU with its own 14-bit bus, whereas the regular data bus is 8 bits. Allowing 14 bits for the program memory means that all instructions are just one word long. The device has three I/O ports, but many of the I/O bits can be used in different ways (such as for an on/off switch or ADC input), depending on how they are programmed.

Finally, another product called the BASIC Stamp from Parallax Inc. is usually considered a microcontroller, although it is actually a very small circuit board with a few ICs and pins. The whole circuit board plugs into an IC socket, as though it were an IC (see Figure 2.16). What makes the BASIC Stamp somewhat unique is that it has an onboard BASIC program interpreter. A program can be written in BASIC on a PC and then directly downloaded into the Stamp’s EEPROM through a RS-232 port. No assembler or compiling operation is required. There are now other microcontrollers on the market that can be programmed in BASIC.

In summary, a wide variety of microcontrollers are available. At the low end are the 4-bit models, which are more than adequate for appliances and toys. These tend to be large-volume, low-cost applications. Eight-bit microcontrollers (such as the 68HC11 and 8051 mentioned earlier) are very popular because 8 bits turn out to be a convenient size for both numeric and character data. At the high end, 16- and 32-bit microcontrollers are available for control systems requiring sophisticated, high-speed
calculating power for such applications as complicated servomechanisms, avionics, or image processing.

Single-Board Computers

Single-board computers are off-the-shelf microprocessor-based computers built on a single printed-circuit card (Figure 2.17). They come in many configurations, but in general they use a standard microprocessor such as the Zilog Z80, the Intel x86 family, the Motorola 68000, or a microcontroller. They also include memory ICs (both RAM and ROM), I/O capability, and perhaps special interface circuits such as ADCs or DACs. Single-board computers are manufactured by major microprocessor producers such as Intel and Motorola as well as many other smaller companies. Some single board computers are designed to plug into a PC as an expansion card. The obvious advantage of using a ready-made microprocessor board is that it eliminates design- and board-testing time. This is particularly important in small-volume production or one-of-kind systems.

Programmable Logic Controllers

A programmable logic controller (PLC) is a self-contained microprocessor-based unit, designed specifically to be a controller. The PLC includes an I/O section that can
interface directly to such system components as switches, relays, small motors, and lights. Developed in the late 1960s to replace relay logic controllers, PLCs have evolved to be able to handle sophisticated motion control applications. PLCs come in various sizes and capabilities; Figure 2.18 shows a selection of PLCs. The big difference between PLCs and the other devices discussed in this section is that the PLC has the microprocessor, ports, and power supply built into a package that has been ruggedized for an industrial environment. Installation is very easy because in many cases the sensors and actuators can be connected directly to the PLC. Once installed, the microprocessor program is downloaded into the PLC from some source such as a personal computer. The PLC manufacturer usually supplies software to facilitate the programming operation. This software allows the user to write a program with line-by-line instructions, or it can convert a relay logic-wiring diagram (ladder diagram) directly

Figure 2.18
Programmable logic controllers. (Allen-Bradley products courtesy of Rockwell Automation).
into a PLC program. Multiple PLCs in a plant can be networked so the individual units can be monitored and programmed from a single station. This is a form of distributed computer control (DCC) discussed in Chapter 1. PLCs are discussed in detail in Chapter 12.

**Personal Computers Used in Control Systems**

The availability of relatively low-cost, off-the-shelf personal computers (PCs) has made them an attractive alternative for small, one-of-kind control applications. Control system software packages are commercially available for the PC that run under DOS and Windows. These programs are adaptable and allow the user to tailor the software to fit the control application, essentially turning a PC into a PLC (although not as rugged). Most of these packages use interactive graphics to link animation with changing process values. Some programs have provisions to mathematically simulate the process being controlled to help optimize the controller coefficients.

A standard PC comes with **expansion slots**, which are circuit-card connectors emanating from the *motherboard* (main board) of the computer. **Expansion cards** plug into these slots and form a bridge between the computer and the outside world. Many different types of interface cards are available, such as I/O serial and parallel data ports.

Figure 2.19

Multi-function I/O board, includes ADC, DAC, and digital I/O. (Courtesy of Omega Engineering, Inc.)
ADCs, DACs, and computer-controlled output relays, to name a few. Figure 2.19 shows an example of an interface expansion card.

Historically, data-acquisition and control functions were kept separate. Controllers ran the process, and other instruments measured and recorded the result. The concept of having a single PC perform both tasks seems logical; after all, the PC can use its computing ability first as the controller and then tabulate system performance data. These data can be stored on disk and/or displayed on the monitor.

A potential problem may arise because the controller must operate in real time. If a computer is to control a process and monitor it at the same time, the data-reduction process must not take so long as to interfere with the control duties; a control response can't wait. One way to overcome this problem is to divide the control and data-acquisition tasks among multiple processors. Using the PC as the master computer, a separate microprocessor on an expansion card can perform data collection uninterrupted. One type of I/O controller card has slots for three smaller boards. These smaller boards have various combinations of analog and digital I/O ports and counter-timers. Some boards are available with solid-state relays, which can be used to directly control AC and DC motors.

A PC with I/O expansion cards often costs less than a stand-alone computerized control system. The cards do not need a separate enclosure and use the PC's power supply, keyboard for input, and monitor for display. Also, using a standard PC means that programs can be developed on another compatible computer, eliminating process downtime.

Numerous manufacturers are selling rugged PCs that can survive in harsh industrial environments. These computers typically use a membrane-type keyboard (the keyboard appears as one continuous sheet of flexible plastic) and have sealed cases and filters covering the air vents. Some models of these computers are rack-mountable and contain their own battery-backup power supply.

**SUMMARY**

A microprocessor is a digital integrated circuit that performs the basic operations of a computer. Microprocessors are used extensively as the basis of a digital controller. Digital control systems are advantageous because digital data can be transferred and stored virtually error-free, and the control strategy can be changed by simply reprogramming.

A computer consists of four basic functional units: (1) the CPU (microprocessor), which executes the programmed instructions and performs the calculations; (2) the memory, which stores the program and data; (3) input; and (4) output. Input/output interfaces the computer to the outside world. A microprocessor-based computer interconnects these units with three groups of signals called buses. The address bus carries the address of the data to be processed. The data bus carries the data, and the control bus carries timing and control signals. Computers handle data as groups of binary bits. Many microprocessor-based controllers handle data in 8-bit groups called a byte.

A microprocessor has a set of instructions that it can execute (called the instruction set). Each instruction is identified by a digital code called the operation code (op-code).
A program consists of a list of these op-codes stored in memory. The microprocessor automatically fetches the instructions from memory and executes them, one by one.

A digital controller may have two kinds of data interfaces: parallel and serial. The parallel interface is the most straightforward system, where all 8 bits are sent at the same time on eight separate wires. In the serial interface, data is sent 1 bit after the other on a single wire. Serial data transfer is better for longer distances.

Many control systems use components that require an analog signal interface; therefore, the signals to or from the digital controller must be converted with an ADC (analog-to-digital converter) or a DAC (digital-to-analog converter). Both circuits are available in IC form.

The digital controller program has a standard format. First, it reads the set point and sensor values. Then it subtracts these values to determine the system error. Based on the error value, it next calculates the appropriate actuator response signal and sends it out. Then it loops back to the beginning of the program and executes the same set of instructions over and over.

Microprocessor-based controllers come in a number of standard forms. A microcontroller includes a microprocessor, memory, and input/output all on a single IC. A single-board computer is an off-the-shelf microprocessor-based computer, assembled onto a single printed circuit board. A programmable logic controller (PLC) is a self-contained unit specifically designed to be a controller. A personal computer (PC) is a general-purpose, self-contained computer; however, with the addition of interface expansion cards, a PC becomes a very adaptable and cost-effective controller.

**GLOSSARY**

**accumulator** A temporary, digital data-storage register in the microprocessor used in many math, logic, and data-moving operations.

**ADC** See analog-to-digital converter.

**address** A number that represents the location of 1 byte of data in memory or a specific input/output port.

**address bus** A group of signals coming from the microprocessor to memory and I/O ports, specifying the address.

**ALU** See arithmetic logic unit.

**arithmetic logic unit (ALU)** The part of the CPU that performs arithmetic and logical operations.

**analog-to-digital converter (ADC)** A device (usually an IC) that can convert an analog voltage into its digital binary equivalent.
assembly language: A computer program written in mnemonics, which are English-like abbreviations for machine-code instructions.

baud: The rate at which the signal states are changing; frequently used to mean “bits per second.”

bit: The smallest unit of digital data, which has a value of 1 or 0.

byte: An 8-bit digital word.

central processing unit (CPU): The central part of a computer, the CPU performs all calculations and handles the control functions of the computer.

core: A group of timing and control signals coming from the microprocessor to memory and I/O ports.

crossover cable: See null modem.

CPU: See central processing unit.

DAC: See digital-to-analog converter.

data bus: A group of signals going to and from the microprocessor, memory, and I/O ports. The data bus carries the actual data that are being processed.

data communication equipment (DCE): One of two units specified by the RS-232 standard (for serial data transfer); the DCE is usually a modem.

data terminal equipment (DTE): One of two units specified by the RS-232 standard (for serial data transfer). The DTE is usually the computer.

DCE: See data communication equipment.

digital-to-analog converter (DAC): A circuit that translates digital data into an analog voltage.

download: To transfer a computer program or data into a computer (from another computer).

DTE: See data terminal equipment.

embedded controller: A small microprocessor-based controller that is permanently installed within the machine it is controlling.

expansion card/slot: An expansion card is a printed circuit card that plugs into an expansion slot on the motherboard of a personal computer (PC). The expansion card usually interfaces the PC to the outside world.

fetch-execute cycle: A computer cycle where the CPU fetches an instruction and then executes it.

input/output (I/O): Data from the real world moving in and out of a computer.

I/O: See input/output.
instruction set  The set of program commands that a particular microprocessor is
designed to recognize and execute.

iteration  One pass through the computer program being executed by the digital con-
troller; each iteration “reads” the set-point and sensor data and calculates the output to
the actuator.

LAN  See local area network.

least significant bit (LSB)  The rightmost bit of a binary number. Can also mean the
smallest increment of change.

logical variable  A single data bit in those cases where a single bit is used to repre-
sent an on-off switch, motor on-off control, and so on.

local area network (LAN)  A system that allows multiple units to communicate with
each other, all sharing the same interconnection wire.

LSB  See least significant bit.

machine language  The set of operation codes that a CPU can execute.

memory  The part of the computer that stores digital data. Memory data is stored as
bytes, where each byte is given an address.

memory-mapped input/output  A system where I/O ports are treated exactly like
memory locations.

microcontroller  An integrated circuit that includes a microprocessor, memory, and
input/output; in essence, a “computer on a chip.”

microprocessor  A digital integrated circuit that performs the basic operations of a
computer but requires some support integrated circuits to be functional.

most significant bit (MSB)  The leftmost bit in a binary number.

mnemonic  An English-like abbreviation of an operation code.

modem  A circuit that converts serial data from digital form into tones that can be sent
through the telephone system.

MSB  See most significant bit.

nonvolatile memory  Computer memory such as ROM that will not lose its data when
the power is turned off.

null modem  A cable that allows two DTE units to communicate with each other (see
RS-232).

operation code (op-code)  A digital code word used by the microprocessor to iden-
tify a particular instruction.

parallel interface  A type of data interface where 8 bits enter or leave a unit at the
same time on eight wires.
PC See personal computer.

**personal computer (PC)** A microprocessor-based, self-contained, general-purpose computer (usually refers to an IBM or compatible computer).

**PLC** See programmable logic controller.

**port** The part of a computer where I/O data lines are connected; each port has an address.

**program counter** A special address-holding register in a computer that holds the address of the next instruction to be executed.

**programmable logic controller (PLC)** A rugged, self-contained microprocessor-based controller designed specifically to be used in an industrial environment.

**RAM** See random-access memory.

**random-access memory (RAM)** Sometimes called read/write memory, a memory arrangement using addresses where data can be written in or read out; RAM loses its contents when the power is turned off.

**read-only memory (ROM)** Similar to RAM in that it is addressable memory, but it comes preprogrammed and cannot be written into; also, it does not lose its data when the power is turned off.

**read/write (R/W) line** A control signal that goes from the microprocessor to memory.

**real time** Refers to a computer that is processing data at the same time that the data are generated by the system.

**resolution** In digital-to-analog conversion, the error that occurs because digital data can only have certain discrete values.

**ROM** See read-only memory.

**RS-232 standard** A serial data transmission standard that specifies voltage levels and signal protocol between a DTE (computer) and a DCE (modem or other device).

**R/W** See read/write line.

**sampling rate** The times per second a digital controller reads the sensor data.

**scan** See iteration.

**serial interface** A type of interface where data are transferred 1 bit after the other on a single wire.

**single-board computer** A premade microprocessor-based computer assembled onto a single printed-circuit card.
time-delay loop A programming technique where the computer is given a "do-nothing" job such as counting to some large number for the purpose of delaying time.

UART See universal asynchronous receiver transmitter.

universal asynchronous receiver transmitter (UART) A special purpose integrated circuit that converts data from parallel to serial format and vice versa.

volatile memory Computer memory such as RAM that will lose its data when the power is turned off.

word A unit of digital data that a particular computer uses; common word sizes are 4, 8, 16, and 32 bits.

EXERCISES

Section 2.1

1. Briefly describe the functions of the ALU, control unit, CPU memory, and input/output.

2. What steps does the microprocessor take to read data at address 1020? (Specify the actions of the address bus and data bus in your answer.)

3. Briefly define address bus, data bus and control bus.

4. Use the method shown in Example 2.1 to find the decimal value of the binary number 01011101.

5. Use the method shown in Example 2.1 to find the decimal value of the binary number 11011010.

Section 2.2

6. What is a microprocessor instruction set, and how is it different from a high-level language such as BASIC?

7. A certain microprocessor has a simple instruction set shown below.

<table>
<thead>
<tr>
<th>Instruction Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>76</td>
</tr>
<tr>
<td>C6*</td>
</tr>
<tr>
<td>D6*</td>
</tr>
<tr>
<td>3C</td>
</tr>
<tr>
<td>3D</td>
</tr>
<tr>
<td>3E*</td>
</tr>
</tbody>
</table>

*These instructions use two bytes.

What number would be in the accumulator after the program shown below was run?

```
76
C6
D6
3C
3D
3E
```
Section 2.3

8. Temperature values from $-20^\circ F$ to $120^\circ F$ are input data for a microprocessor computer. Are 8 bits sufficient? If so, what is the resolution?

9. Explain the function of the following: parallel data port and serial data port.

10. Serial data are sent at 1200 bps using the format of Figure 2.11, with one stop bit. How long would it take to send 1000 bytes of data?

11. An 8-bit DAC has a reference voltage of 9 V. The binary input is 11001100. Find the analog output voltage.

12. The binary data from the computer in a certain application are expected to go from 00000000 to only 00111111. These data are the input of a DAC. The analog output should go 0-5 V. Find the DAC reference voltage necessary to make this happen.

13. An 8-bit ADC has a reference voltage of 12 V and an analog input of 3.7 V. Find the binary output.

14. The binary output of an ADC should have the range 00000000-11111111 corresponding to an input of 0-6 V. Find the necessary reference voltage.

Section 2.4

15. What is real-time computing, and is it necessary for control systems?

16. Describe the basic steps in a control program scan (loop).

17. At some point in the program it is desired to have the computer wait 5 s for an operator response. How would this delay be accomplished in software?

18. A program contains 150 instructions, and the average execution time per instruction is 2 $\mu$s. Find the sample rate of this program.

Section 2.5

19. What is a microcontroller, and what are some differences between a microcontroller and a microprocessor?

20. What is a programmable logic controller?

21. You want to use a personal computer to control a simple robot arm. The arm has two joints, an elbow and a wrist. Each joint has a DC motor and a position sensor that outputs a DC voltage. You already have a "plain vanilla" PC; make a list of what you would need to acquire to make this system work.

22. Compare and contrast the following: a microprocessor, a microcontroller, a programmable logic controller, and a personal computer.