system, a *double handshake* is used. The circuit connections are the same as those in Figure 9-2. Figure 9-1d shows some example waveforms for a double-handshake input from a peripheral to a microprocessor. Perhaps it will help you to follow these waveforms by thinking of them as a conversation between two people. In these waveforms each signal edge has meaning. The sending device asserts its $\overline{STB}$ line low to ask, "Are you ready?" The receiving system raises its ACK line high to say, "I'm ready." The peripheral device then sends the byte of data and raises its $\overline{STB}$ line high to say, "Here is some valid data for you." After it has read in the data, the receiving system drops its ACK line low to say, "I have the data, thank you, and I await your request to send the next byte of data."

For a handshake output of this type, from a microprocessor to a peripheral, the waveforms are the same, but the microprocessor sends the $\overline{STB}$ signal and the data, and the peripheral sends the ACK signal. In the accompanying laboratory manual we show you how to interface with a speech-synthesizer device using this type of handshake system.

Implementing Handshake Data Transfer

For handshake data transfer, a microprocessor can determine when it is time to send the next data byte on a polled or on an interrupt basis. The interrupt approach is usually used, because it makes better use of the processor's time.

The $\overline{STB}$ or ACK signals for these handshake transfers can be produced on a port pin by instructions in the program. However, this method usually uses too much processor time, so parallel-port devices such as the 8255A have been designed to automatically manage the handshake operation. The 8255A, for example, can be programmed to automatically receive an $\overline{STB}$ signal from a peripheral, send an interrupt signal to the processor, and send the ACK signal to the peripheral at the proper times. The following sections show you how to connect, initialize, and use an 8255A for a variety of handshake and nonhandshake applications.

8255A Internal Block Diagram and System Connections

Figure 9-3, p. 248, shows the internal block diagram of the 8255A. Along the right side of the diagram, you can see that the device has 24 input/output lines. Port A can be used as an 8-bit input port or as an 8-bit output port. Likewise, port B can be used as an 8-bit input port or as an 8-bit output port. Port C can be used as an 8-bit input or output port, as two 4-bit ports, or to produce handshake signals for ports A and B. We will discuss the different modes for these lines in detail a little later.

Along the left side of the diagram, you see the signal lines used to connect the device to the system buses. Eight data lines allow you to write data bytes to a port or the control register and to read bytes from a port or the status register under the control of the $\overline{RD}$ and $\overline{WR}$ lines. The address inputs, A0 and A1, allow you to selectively access one of the three ports or the control register. The internal addresses for the device are: port A, 00; port B, 01; port C, 10; control, 11. Asserting the $\overline{CS}$ input of the 8255A enables it for reading or writing. The $\overline{CS}$ input will be connected to the output of the address decoder circuitry to select the device when it is addressed.

The RESET input of the 8255A is connected to the system reset line so that, when the system is reset, all the port lines are initialized as input lines. This is done to prevent destruction of circuitry connected to port lines. If port lines were initialized as outputs after a power-up or reset, the port might try to output to the output of a device connected to the port. The possible argument between the two outputs might destroy one or both of them. Therefore, all the programmable port devices initialize their port lines as inputs when reset.

We discussed in Chapter 7 how two 8255As can be connected in an 8086 system. Take a look at Figure 7-8 (sheet 5) to refresh your memory of these connections. Note that one of the 8255As is connected to the lower half of the 8086 data bus, and the other 8255A is connected to the upper half of the data bus. This is done so that a byte can be transferred by enabling one device, or a word can be transferred by enabling both devices at the same time. According to the truth table for the I/O port address decoder in Figure 7-16, the A40 8255A on the lower half of the data bus will be enabled for a base address of FFF8H, and the A35 8255A will be enabled for a base address of FFF9H.

Another point to notice in Figure 7-8 is that system address line A1 is connected to the 8255A A0 inputs, and system address line A2 is connected to the 8255A A1 inputs. With these connections, the system addresses for the three ports and the control register in the A40 8255A will be FFF8H, FFFAH, FFFCH, and FFFEH, as shown in Figure 7-16. Likewise, the system addresses for the three ports and the control register of the A35 8255A are FFF9H, FFFBH, FFFDH, and FFFFH.

8255A Operational Modes and Initialization

Figure 9-4, p. 249, summarizes the different modes in which the ports of the 8255A can be initialized.

MODE 0

When you want to use a port for simple input or output without handshaking, you initialize that port in mode 0. If both port A and port B are initialized in mode 0, then the two halves of port C can be used together as an additional 8-bit port, or they can be used individually as two 4-bit ports. When used as outputs, the port C lines can be individually set or reset by sending a special control word to the control register address. Later we will show you how to do this. The two halves of port C are independent, so one half can be initialized as input, and the other half initialized as output.

MODE 1

When you want to use port A or port B for a handshake (strobed) input or output operation such as we discussed in previous sections, you initialize that port in mode 1.

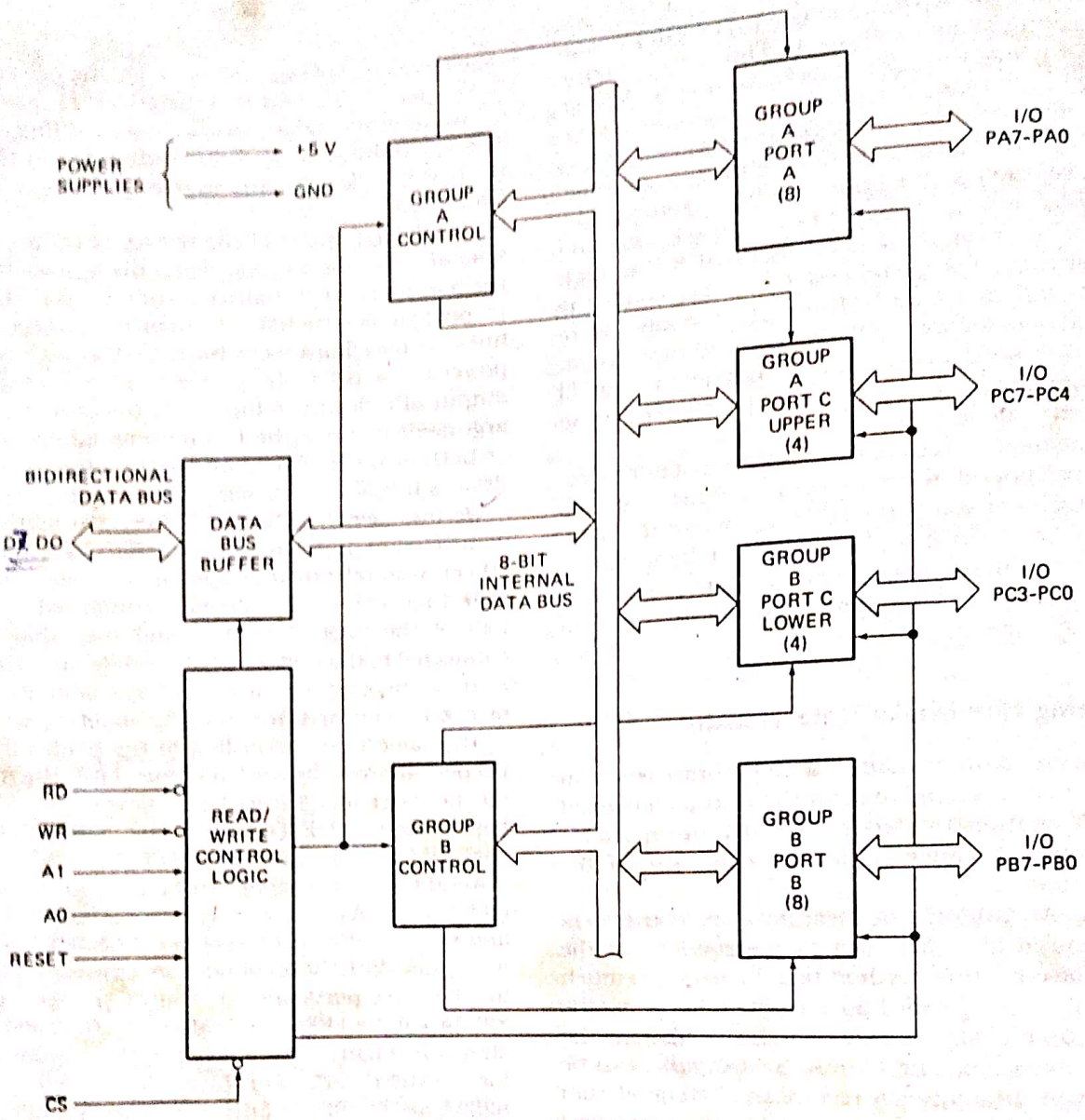


FIGURE 9-3 Internal block diagram of 8255A programmable parallel port device. (*Intel Corporation*)

In this mode, some of the pins of port C function as handshake lines. Pins PC0, PC1, and PC2 function as handshake lines for port B if it is initialized in mode 1. If port A is initialized as a handshake (mode 1) input port, then pins PC3, PC4, and PC5 function as handshake signals. Pins PC6 and PC7 are available for use as input lines or output lines. If port A is initialized as a handshake output port, then port C pins PC3, PC6, and PC7 function as handshake signals. Port C pins PC4 and PC5 are available for use as input or output lines. Since the 8255A is often used in mode 1, we show several examples in the following sections.

MODE 2

Only port A can be initialized in mode 2. In mode 2, port A can be used for *bidirectional handshake* data transfer. This means that data can be output or input on the same eight lines. The 8255A might be used in this mode to extend the system bus to a slave microprocessor or to transfer data bytes to and from a floppy disk controller board. If port A is initialized in mode 2, then pins PC3 through PC7 are used as handshake lines for port A. The other three pins, PC0 through PC2, can be used for I/O if port B is in mode 0. The three pins will be used for port B handshake lines if port B is initialized in mode 1. After you work your way through the mode 1 examples in the following sections, you should have little difficulty understanding the discussion of mode 2 in the Intel data sheet if you encounter it in a system.

Constructing and Sending 8255A Control Words

Figure 9-5 shows the formats for the two 8255A control words. Note that the MSB of the control word tells the 8255A which control word you are sending it. You use the *mode definition control word* format in Figure 9-5*a* to tell the device what modes you want the ports to operate in. You use the *bit set/reset control word* format in Figure 9-5*b* when you want to set or reset the output on a pin of port C or when you want to enable the

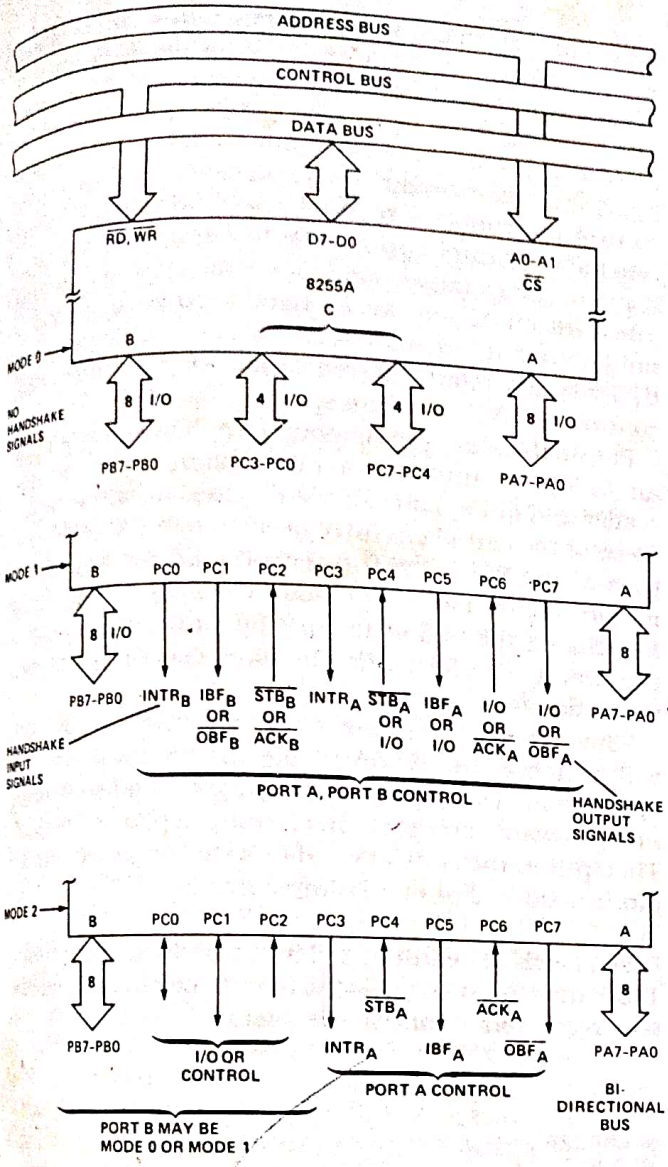


FIGURE 9-4 Summary of 8255A operating modes. (*Intel Corporation*)

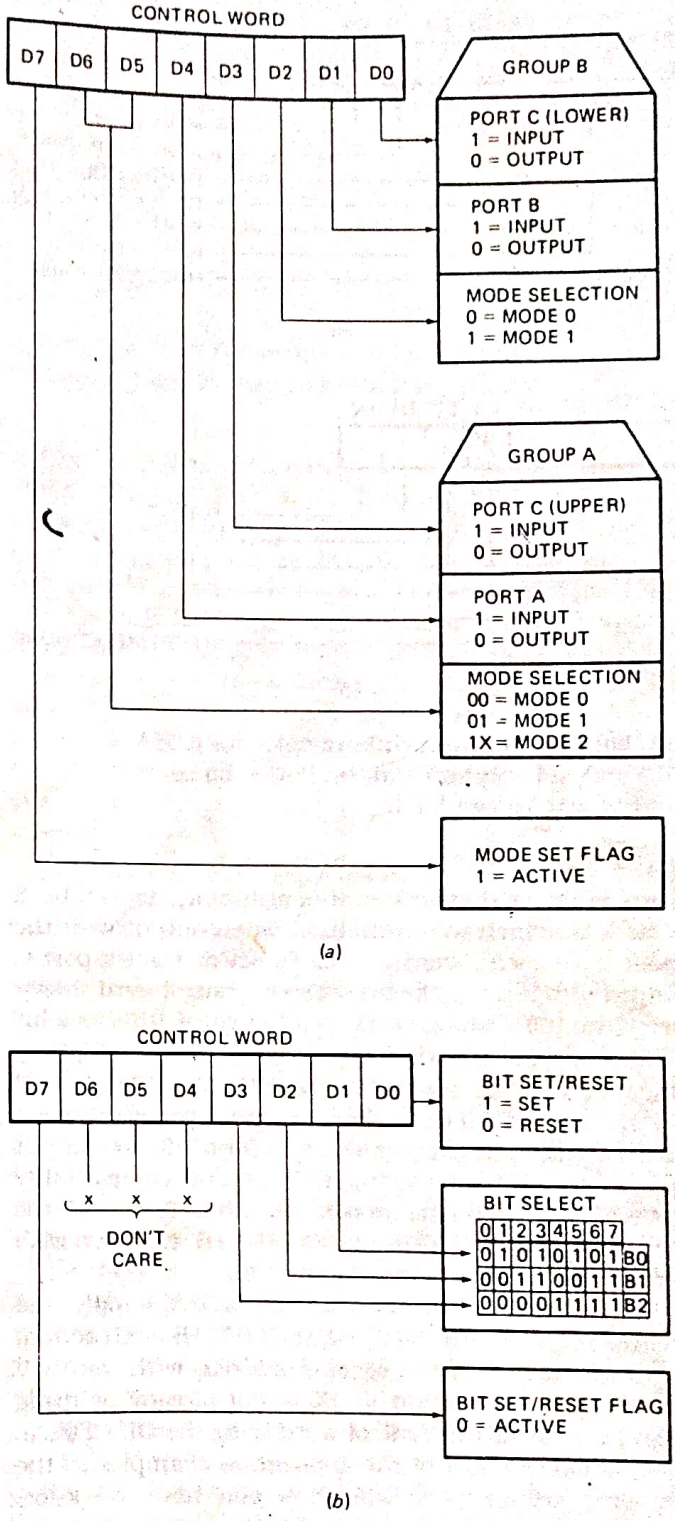


FIGURE 9-5 8255A control word formats. (*a*) Mode-set control word. (*b*) Port C bit set/reset control word.

interrupt output signals for handshake data transfers. Both control words are sent to the control register address of the 8255A.

As usual, initializing a device such as this consists of working your way through the steps we described in the last chapter. As an example for this device, suppose that you want to initialize the 8255A (A40) in Figure 7-8 as follows:

Port B as mode 1 input

Port A as mode 0 output

Port C upper as inputs

Port C bit 3 as output

As we said previously, the base address for the A40 8255A is FFF8H, and the control register address is FFFEH. The next step is to make up the control word by figuring out what to put in each of the little boxes, one bit at a time. Figure 9-6*a*, p. 250, shows the control word which will program the 8255A as desired for this example. The figure also shows how you should document any control words you make up for use in your programs. Using Figure 9-5*a*, work your way through this word to make sure you see why each bit has the value it does.

To send the control word, you load the control word in AL with a MOV AL,10001110B instruction, point DX at the port address with the MOV DX,0FFFEH instruction, and send the control word to the 8255A control register with the OUT DX,AL instruction.

As an example of how to use the bit set/reset control

reference. The D/A converters have a current output, so an op amp is used to convert the D/A output current to a proportional voltage. A FET input amplifier is used here because the input bias current of a bipolar input amp might affect the accuracy of the output. The DAC1208 and DAC1230 have built-in feedback resistors which match the temperature characteristics of the internal current-divider resistors, so all you have to add externally is a 50-Ω resistor for "tweaking" purposes. With a −10.000-V reference as shown, the output voltage will be equal to (the digital input word/4096) × (+10.000 V). Note that the D/A has both a digital ground and an analog ground. To avoid getting digital noise in the analog portions of the circuit, these two should be connected together only at the power supply.

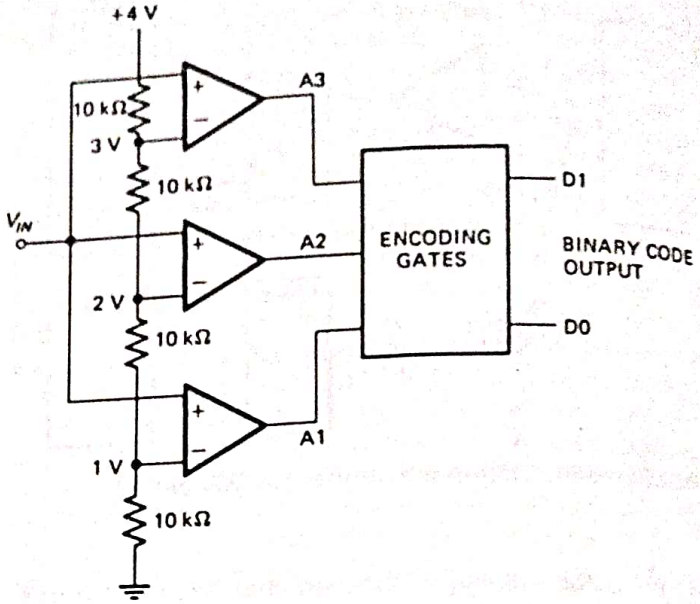


FIGURE 10-18 Parallel comparator A/D converter.

A/D CONVERTER SPECIFICATIONS, TYPES, AND INTERFACING

A/D Specifications

The function of an A/D converter is to produce a digital word which represents the magnitude of some analog voltage or current. The specifications for an A/D converter are very similar to those for a D/A converter. The resolution of an A/D converter refers to the number of bits in the output binary word. An 8-bit converter, for example, has a resolution of 1 part in 256. Accuracy and linearity specifications have the same meanings for an A/D converter as they do for a D/A converter. Another important specification for an A/D converter is its *conversion time*. This is simply the time it takes the converter to produce a valid output binary code for an applied input voltage. When we refer to a converter as *high-speed*, we mean that it has a short conversion time. There are many different ways to do an A/D conversion, but we have space here to review only three commonly used methods, which represent a wide variety of conversion times.

A/D Converter Types

PARALLEL COMPARATOR A/D CONVERTER

Figure 10-18 shows a circuit for a 2-bit A/D converter using *parallel comparators*. A voltage divider sets reference voltages on the inverting inputs of each of the comparators. The voltage at the top of the divider chain represents the full-scale value for the converter. The voltage to be converted is applied to the noninverting inputs of all the comparators in parallel. If the input voltage on a comparator is greater than the reference voltage on the inverting input, the output of the comparator will go high. The outputs of the comparators then give us a digital representation of the voltage level of the input signal. With an input voltage of 2.6 V, for example, the outputs of comparators A1 and A2 will be high.

The major advantage of a parallel, or *flash*, A/D converter is its speed of conversion, which is simply the propagation delay time of the comparators. The output code from the comparators is not a standard binary code, but it can be converted to any desired code with some simple logic. The major disadvantage of a flash A/D is the number of comparators needed to produce a result with a reasonable amount of resolution. The 2-bit converter in Figure 10-18 requires three comparators. To produce a converter with N bits of resolution, you need $(2^N - 1)$ comparators. For an 8-bit conversion, then, you need 255 comparators, and for a 10-bit flash converter, you need 1023 comparators. Single-package flash converters are available from TRW for applications in which the high speed is required, but they are relatively expensive. Flash converters which can do an 8-bit conversion in under 10 ns are currently available.

DUAL-SLOPE A/D CONVERTERS

Figure 10-19*a* shows a functional block diagram of a *dual-slope* A/D converter. This type of converter is often used as the heart of a digital voltmeter because it can give a large number of bits of resolution at a low cost. Here's how the converter in Figure 10-19 works.

To start, the control circuitry resets all the counters to zero and connects the input of the integrator to the input voltage to be converted. If you assume the input voltage is positive, then this will cause the output of the integrator to ramp negative, as shown in Figure 10-19*b*. As soon as the output of the integrator goes a few microvolts below ground, the comparator output will go high. The comparator output being high enables the AND gate and lets the 1-MHz clock into the counter chain. After some fixed number of counts, typically 1000, the control circuitry switches the input of the integrator to a negative reference voltage and resets all the counters to zero. With a negative input voltage, the integrator output will ramp positive, as shown in the right-hand side of Figure 10-19*b*. When the integrator output crosses 0 V, the comparator output will drop low and shut off the clock signal to the counters. The number of counts required for the integrator output to

converter. In many industrial applications where the sensor is a long distance from the A/D converter, however, the signals from the sensors or transducers are converted to currents instead of voltages. Sending a signal as a current has the advantages that the signal amplitude is not affected by resistance, induced-voltage noise, or voltage drops in a long connecting line. A common range of currents used to represent analog signals in industrial environments is 4 to 20 mA. A current of 4 mA represents a zero output, and a current of 20 mA represents the full-scale value. The reason the current range is offset from zero is so that a current of zero is left to represent an open circuit. At the receiving end of the line, a resistor or a simple op-amp circuit is used to convert the current to a proportional voltage which can be applied to the input of the A/D converter.

D/A CONVERTER OPERATION, INTERFACING, AND APPLICATIONS

In the previous sections of this chapter we have discussed how we use sensors to get electrical signals proportional to pressure, temperature, etc, and how we use op amps to amplify and filter these electrical signals. The next logical step would be to show you how to use an A/D converter to get these signals into digital form that a microcomputer can work with. However, since D/A converters are simpler and since several types of A/D converters have D/As as part of their circuitry, we will discuss D/As first.

D/A Converter Operation and Specifications

OPERATION

The purpose of a digital-to-analog converter is to convert a binary word to a proportional current or voltage. To see how this is done, let's look at the simple 4-input adder circuit in Figure 10-14.

Since the noninverting input of the op amp is grounded, the op amp will work day and night to hold the inverting input also at 0 V. Remember that the inverting input in this circuit is referred to as the summing point. When one of the switches is closed, a current will flow from -5 V (V_{REF}) through that resistor to the summing point. The op amp will pull the current on through the feedback resistor to produce a proportional output voltage. If you close switch D0, for example, a current of 0.05 mA will flow into the summing point. In order to pull this current through the feedback resistor, the op amp must put a voltage of 0.05 mA × 10 kΩ or 0.5 V on its output. If you also close switch D1, it will send another 0.1 mA into the summing point. In order to pull the sum of the currents through the feedback resistor, the op amp has to output a voltage of 0.15 mA × 10 kΩ or 1.5 V.

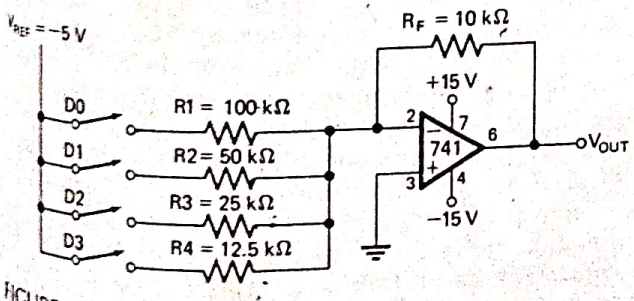


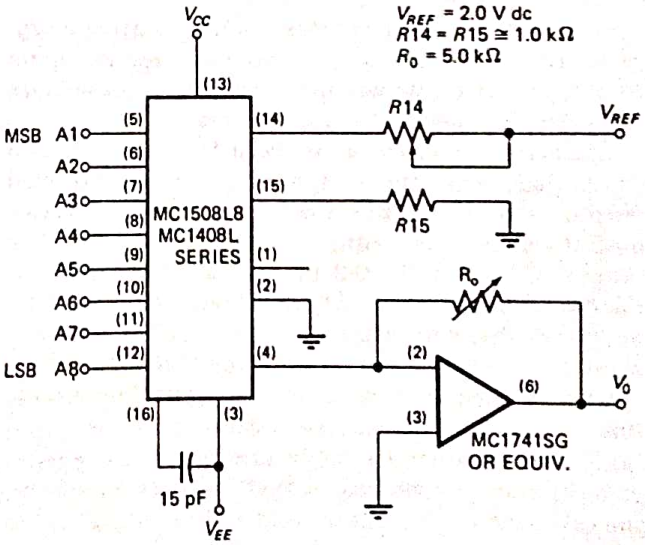
FIGURE 10-14 Simple 4-bit D/A converter.

The point here is that the binary-weighted resistors produce binary-weighted currents which are summed by the op amp to produce a proportional output voltage. The binary word applied to the switches produces a proportional output voltage. Technically the output voltage is "digital" because it can only have certain fixed values, just as the display on a digital voltmeter can. However, the output simulates an analog signal, so we refer to it as analog. Switch D3 in Figure 10-14 represents the most significant bit because closing it produces the largest current. Note that since V_{REF} is negative, the output will go positive as switches are closed.

As you see here, the heart of a D/A converter is a set of binary-weighted current sources which can be switched on or off according to a binary word applied to its inputs. Since these current sources are usually inside an IC, we don't need to discuss the different ways the binary-weighted currents can be produced. The op-amp circuit in Figure 10-14 converts the sum of the currents to a proportional voltage.

D/A CHARACTERISTICS AND SPECIFICATIONS

Figure 10-15 shows the circuit for an inexpensive IC D/A converter with an op-amp circuit as a current-to-



Theoretical V_0

$$V_0 = \frac{V_{REF}}{R14}(R_0)\left\{\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \frac{A4}{16} + \frac{A5}{32} + \frac{A6}{64} + \frac{A7}{128} + \frac{A8}{256}\right\}$$

ADJUST V_{REF}, $R14$ OR R_0 SO THAT V_0 WITH ALL DIGITAL INPUTS AT HIGH LEVEL IS EQUAL TO 9.961 V

$$V_0 = \frac{2\text{ V}}{1\text{ k}\Omega}(5\text{ k}\Omega)\left\{\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256}\right\}$$

$$= 10\text{ V}\left\{\frac{255}{256}\right\} = 9.961\text{ V}$$

FIGURE 10-15 Motorola MC1408 8-bit D/A with current-to-voltage converter.