products, make the other 3 bits of the control word 0's. The result, 00000111B, is shown with proper documentation in Fig. 9.6*b*.

To send this control word to the 8255A, simply load it into AL with the MOV AL,00000111B instruction, point DX at the control register address with the MOV DX,0FFFEH instruction if DX is not already pointing there, and send the control word with the OUT DX,AL instruction. As part of the application examples in the following sections, we will show you how you know which bit in port C to set to enable the interrupt output signal for handshake data transfer.

8255A Handshake Application Examples

INTERFACING TO A MICROCOMPUTER-CONTROLLED LATHE

All the machines in the machine shop of our computer-controlled electronics factory ~~operate under~~ microcomputer control. One example of these machines is a lathe which makes bolts from long rods of stainless steel. The cutting instructions for each type of bolt that we need to make are stored on a 3/4-in.-wide teletype-like metal tape. Each instruction is represented by a series of holes in the tape. A tape reader pulls the tape through an optical or mechanical sensor to detect the hole patterns and converts these to an 8-bit parallel code. The microcomputer reads the instruction codes from the tape reader on a handshake basis and sends the appropriate control instructions to the lathe. The microcomputer must also monitor various conditions around the lathe. It must, for example, make sure the lathe has cutting lubricant oil, is not out of material to work on, and is not jammed up in some way. Machines that operate in this way are often referred to as *computer numerical control*, or *CNC, machines*.

Figure 9.7 shows in diagram form how you might use an 8255A to interface a microcomputer to the tape reader and lathe. Later in the chapter, we will show you some of the actual circuitry needed to interface the port pins of the 8255A to the sensors and the high-power motors of the lathe. For now, we want to talk about initializing the 8255A for this application and analyze the timing waveforms for the handshake input of data from the tape reader.

Your first task is to make up the control word which will initialize the 8255A in the correct modes for this application. To do this, start by making a list showing how you want each port pin or group of pins to function. Then put in the control word bits that implement those pin functions. For our example here,

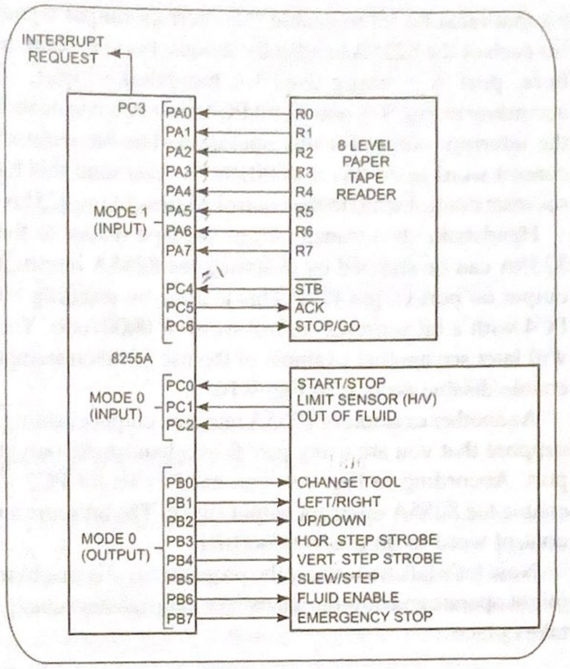

Fig. 9.7 *Interfacing a microprocessor to a tape reader and lathe.*

Port A needs to be initialized for handshake input (mode 1) because instruction codes have to be read in from the tape reader on a handshake basis.

Port B needs to be initialized for simple output (mode 0). No handshaking is needed here because this port is being used to output simple on or off control signals to the lathe.

Port C, bits PC0, PC1, and PC2 are used for simple input of sensor signals from the lathe.

Port C, bits PC3, PC4, and PC5 function as the handshake signals for the data transfer from the tape reader connected to port A.

Port C, bit PC6 is used for output of the STOP/GO signal to the tape reader.

Port C, bit PC7 is not used for this example.

Figure 9.8 shows the control word to initialize the 8255A for these pin functions. You send this word to the control register address of the 8255A as described above.

Before we go on, there is one more point we have to make about initializing the 8255A for this microcomputer-controlled lathe application. In order for the handshake input data transfer from the tape reader to work correctly, the interrupt request signal from bit PC3 has to be enabled. This is done by sending a bit set/reset control word for the appropriate bit of port C. Figure 9.9 shows the port C bit that must be set to enable the interrupt output signal for each of the 8255A handshake modes. For the example here, port A is being used for handshake input, so according to Fig. 9.9, port C, bit PC4 must be set to enable the interrupt output for this operation. The bit set/reset control word to do this is 00001001B. You send this bit set/reset control word to the control address of the 8255A.

Handshake data transfer from the tape reader to the 8255A can be stopped by disabling the 8255A interrupt output on port C, pin PC3. This is done by resetting bit PC4 with a bit set/reset control word of 00001000. You will later see another example of the use of this interrupt enable/disable process in Fig. 9.16.

As another example of 8255A interrupt output enabling, suppose that you are using port B as a handshake output port. According to Fig. 9.9, you need to set bit PC2 to enable the 8255A interrupt output signal. The bit set/reset control word to do this is 00000101.

Now let's talk about how the program for this machine might operate and how the handshake data transfer actually takes place.

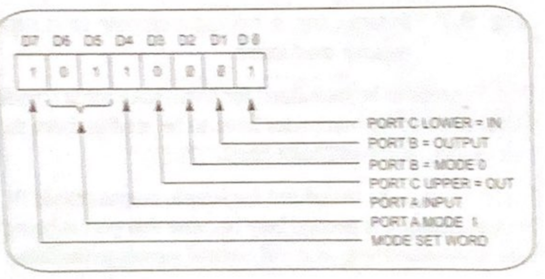

Fig. 9.8 *Control word to initialize 8255A for interface with tape reader and lathe.*

	Port C Interrupt Signal Pin Number	To Enable Interrupt Request Set Port C bit
MODE 1		
Port A IN	PC3	PC4
Port B IN	PC0	PC2
Port A OUT	PC3	PC6
Port B OUT	PC0	PC2
MODE 2		
Port A IN	PC3	PC4
Port A OUT	PC3	PC6

Fig. 9.9 *Port C bits to set to enable interrupt request outputs for handshake modes.*

After initializing everything, you would probably read port C, bits PC0, PC1, and PC2 to check if the lathe was ready to operate. For any 8255A mode, you read port C by simply doing an input from the port C address. Then you output a start command to the tape reader on bit PC6. This is done with a bit set/reset command. Assuming that you want to reset bit PC6 to start the tape reader, the bit set/reset control word for this is 00001100. When the tape reader receives the Go command, it will start the handshake data transfer to the 8255A. Let's work our way through the timing waveforms in Fig. 9.10, to see how the data transfer takes place.

The tape reader starts the process by sending out a byte of data to port A on its eight data lines. The tape reader then asserts its $\overline{\text{STB}}$ line low to tell the 8255A that a new byte of data has been sent. In response, the 8255A raises its Input Buffer Full (IBF) signal on PC5 high to tell the tape reader that it is ready for the data. When the tape reader detects the IBF signal at a high level, it raises its $\overline{\text{STB}}$ signal high again. The rising edge of the $\overline{\text{STB}}$ signal has two effects on the 8255A. It first latches the data byte in the input latches of the 8255A. Once the data is latched, the tape reader can remove the data byte in preparation for sending the next data byte. This is shown by the dashed section on the right side of the data waveform in Fig. 9.10. Second, if the interrupt signal output has been enabled, the rising edge of the $\overline{\text{STB}}$ signal will cause the 8255A to output an Interrupt Request signal to the microprocessor on bit PC3.

The processor's response to the interrupt request will be to go to an interrupt service procedure which reads in the byte of data latched in port A. When the $\overline{\text{RD}}$ signal from the microprocessor goes low for this read of port A, the 8255A will automatically reset its Interrupt Request signal on PC3. This is done so that a second interrupt cannot be caused by the same data byte transfer. When the processor raises its $\overline{\text{RD}}$ signal high again at the end of the read operation, the 8255A automatically drops its IBF signal on PC5 low again. IBF going low again is the signal to the tape reader that the data transfer is complete and that it can send the next byte of data. The time between when the 8255A sends the Interrupt Request signal and when the processor reads the data byte from port A depends on when the processor gets around to servicing that interrupt. The point here is that this time doesn't matter. The tape reader will not send the next byte of data until it detects that the IBF signal has gone low again. The transfer cycle will then repeat for the next data byte.

After the processor reads in the lathe control instruction byte from the tape reader, it will decode this instruction and output the appropriate control byte to the lathe on port B of the 8255A. The tape reader then sends the next instruction byte. If the instruction tape is made into a continuous loop, the lathe will keep making the specified

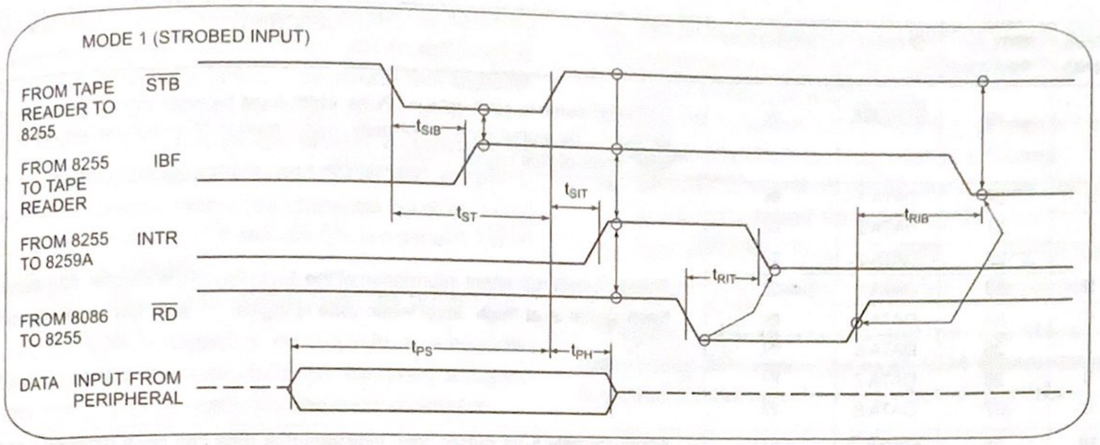

Fig. 9.10 *Timing waveforms for 8255 handshake data input from a tape reader.*

parts until it runs out of material. The unused bit of port C, PC7, could be connected to a mechanism which loads in more material so the lathe can continue.

The microcomputer-controlled lathe we have described here is a small example of automated manufacturing. The advantage of this approach is that it relieves humans of the drudgery of standing in front of a machine continually making the same part, day after day. We hope society can find more productive use for the human time made available.

PARALLEL PRINTER INTERFACE—HANDSHAKE OUTPUT EXAMPLE

At the end of Chapter 8, we showed you how to send a string of text characters to a printer by calling a BIOS procedure with a software interrupt. In this section we show you the hardware connections and software required to interface with a parallel printer in a system which does not have a BIOS procedure you can call to do the job.

For most common printers, such as the IBM PC printers, the Epson dot-matrix printers, and the Panasonic dot-matrix printers, data to be printed is sent to the printer as ASCII characters on eight parallel lines. The printer receives the characters to be printed and stores them in an internal RAM buffer. When the printer detects a carriage return character (ODH), it prints out the first row of characters from the print buffer. When the printer detects a second carriage return, it prints out the second row of characters, etc. The process continues until all the desired characters have been printed.

Transfer of the ASCII codes from a microcomputer to a printer must be done on a handshake basis because the microcomputer can send characters much faster than the printer can print them. The printer must in some way let the microcomputer know that its buffer is full and that it cannot accept any more characters until it prints some out. A common standard for interfacing with parallel printers is the *Centronics Parallel Interface Standard*, named after the company that developed it. In the following sections, we show you how a Centronics parallel interface works and how to implement it with an 8255A.

Centronics Interface Pin Descriptions and Circuit Connections

Centronics-type printers usually have a 36-pin interface connector. Figure 9.11 shows the pin assignments and descriptions for this connector as it is used in the IBM PC printer and the Epson printers. Some manufacturers use one or two pins differently, so consult the manual for your specific printer before connecting it up as we show here.

Thirty-six pins may seem like a lot of pins just to send ASCII characters to a printer. The reason for the large number of lines is that each data and signal line has its own individual ground return line. For example, as shown in Fig. 9.11, pin 2 is the LSB of the data character sent to the printer, and pin 20 is the ground return for this signal. Individual ground returns reduce the chance of picking up electrical noise in the lines. If you are making an interface cable for a parallel printer, these ground return lines should only be connected together and to ground at the microcomputer end of the cable, as shown in Fig. 9.12.

While we are talking about grounds, note that pin 16 is listed as logic ground and pin 17 is listed as chassis ground. In order to prevent large noise currents from flowing in the logic ground wires, these wires should only be connected together in the microcomputer. (This precaution is necessary whenever you connect any external device or system to a microcomputer.)