Mode 0 Operation

Mode 0 operation causes the 82C55 to function either as a buffered input device or as a latched output device. These are the same as the basic input and output circuits discussed in the first section of this chapter.

Figure 10–20 shows the 82C55 connected to a set of eight 7-segment LED displays. In this circuit, both ports A and B are programmed as (mode 0) simple latched output ports. Port A provides the segment data inputs to the display and port B provides a means of selecting one display position at a time for multiplexing the displays. The 82C55 is interfaced to an 8088 microprocessor through a PAL16L8 so that it functions at I/O port numbers 0700H–0703H. The program for the PAL16L8 is listed in Example 10–7. The PAL decodes the I/O address and develops the write strobe for the WR pin of the 82C55.

EXAMPLE 10–7

```
CHIP      DECODERD PAL16L8

;pins    1  2  3  4  5  6  7  8  9   10
        A2 A3 A4 A5 A6 A7 A8 A9 A10 GND

;pins 11  12  13  14  15  16  17 18 19  20
      A11 CS  IOM A12 A13 A14 A15 NC NC VCC

EQUATIONS

/CS = /A15 * /A14 * /A13 * /A12 * /A11 *
A10 * A9 * A8 * /A6 * /A5 * /A4 * /A3 * /A2
* /IOM
```

The resistor values are chosen in Figure 10–20 so that the segment current is 80 mA. This current is required to produce an average current of 10 mA per segment as the displays are multiplexed. A 6-digit display uses a segment current of 60 mA, for an average of 10 mA per segment. In this type of display system, only one of the eight display positions is on at any given instant. The peak anode current in an 8-digit display is 560 mA (seven segments × 80 mA), but the average anode current is 80 mA. In a 6-digit display, the peak current would be 420 mA (seven segments × 60 mA). Whenever displays are multiplexed, we increase the segment current from 10 mA (for a display that uses 10 mA per segment as the nominal current) to a value equal to the number of display positions times 10 mA. This means that a 4-digit display uses 40 mA per segment, a 5-digit display uses 50 mA, and so on.

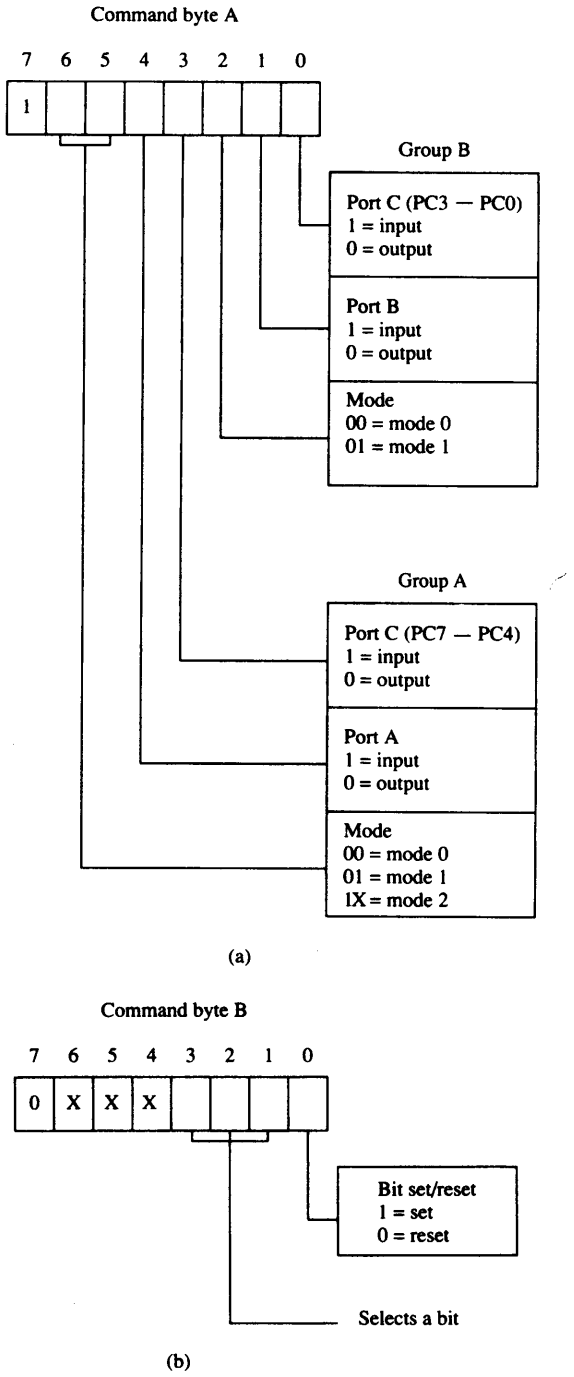


FIGURE 10–19 The command byte of the command register in the 82C55. (a) Programs ports A, B, and C (b) Sets or resets the bit indicated in the select a bit field.

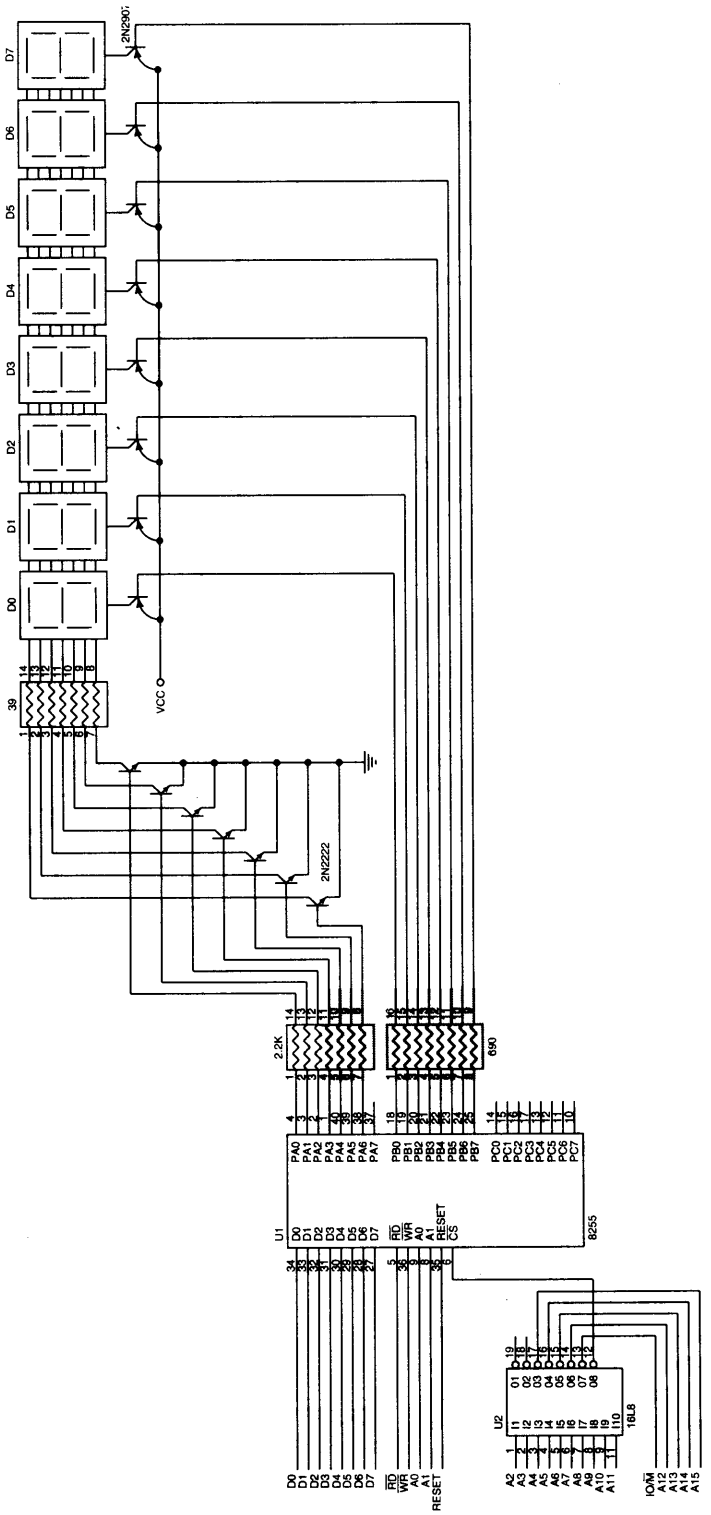


FIGURE 10–20 An 8-digit LED display interfaced to the 8088 microprocessor through an 82C55 PPI.

In this display, the segment load resistor passes 80 mA of current and has a voltage of approximately 3.0 V across it. The LED (1.65 V nominally) and a few tenths are dropped across the anode switch and the segment switch, hence a voltage of 3.0 V appears across the segment load resistor. The value of the resistor is $^{3.0\ V}/_{80\ mA}$ = 37.5 ✷. The closest standard resistor value of 39 ✷ is used in Figure 10–20 for the segment load.

The resistor in series with the base of the segment switch assumes that the minimum gain of the transistor is 100. The base current is therefore $^{80\ mA}/_{100}$ = 0.8 mA. The voltage across the base resistor is approximately 3.0 V (the minimum logic 1 voltage level of the 82C55), minus the drop across the emitter-base junction (0.7 V), or 2.3 V. The value of the base resistor is then 2.3 V/0.8 mA = 2.875 K✷. The closest standard resistor value is 2.7 K✷, but a 2.2 K✷ is chosen for this circuit.

The anode switch has a single resistor on its base. The current through the resistor is 560 mA/100 = 5.6 mA because the minimum gain of the transistor is 100. This exceeds the maximum current of 4.0 mA from the 82C55, but this is small enough so that it will work without problems. The maximum current assumes that you are using the port pin as a TTL input to another circuit. If the amount of current were over 8.0–10.0 mA, then appropriate circuitry (in the form of either a Darlington-pair or another transistor switch) would be required. Here, the voltage across the base resistor is 5.0 V, minus the drop across the emitter-base junction (0.7 V), minus the voltage at the port pin (0.4 V), for a logic 0 level. The value of the resistor is 3.9 V/5.6 mA = 696 W. The closest standard resistor value is 690 W, which is chosen for this example.

Before software to operate the display is examined, we must first program the 82C55. This is accomplished with the short sequence of instructions listed in Example 10–8. Here, port A and B are both programmed as outputs.

EXAMPLE 10–8

```
          ;programming the 82C55 PIA
          ;
0000  B0 80          MOV   AL,10000000B
0002  BA 0703        MOV   DX,703H          ;address command
0005  EE             OUT   DX,AL            ;program 82C55
```

The procedure to multiplex the displays is listed in Example 10–9. For the display system to function correctly, we must call this procedure often. Notice that the procedure calls another procedure (DELAY) that causes a 1 ms time delay. The time delay is not illustrated in this example, but it is used to allow time for each display position to turn on. It is recommended by the manufacturers of LED displays that the display flash be between 100 Hz and 1500 Hz. Using a 1 ms time delay, we light each digit for 1 ms for a total display flash rate of 1000 Hz/8 display, or a flash rate of 125 Hz.

EXAMPLE 10–9

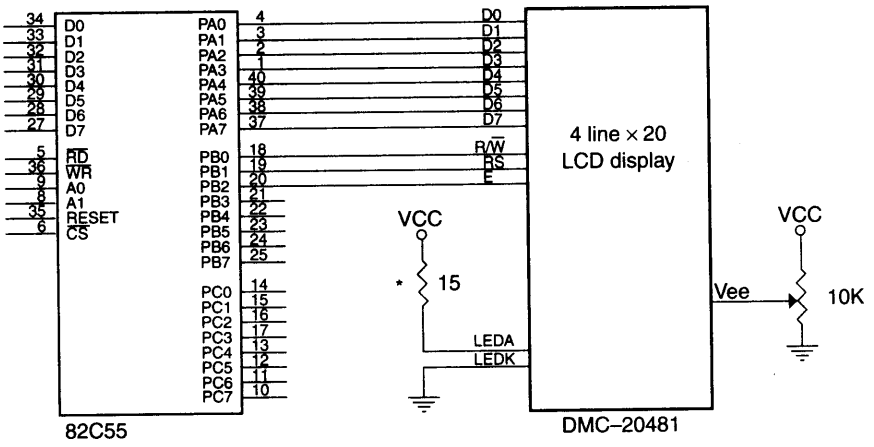
```
                    ;Procedure that multiplexes the 8-digit LED display.
                    ;This procedure must be called from a program at
                    ;whenever possible to display 7-segment
                    ;coded data from memory.
                    ;
0006                DISP  PROC  NEAR USES AX BX DX SI

000A 9C                   PUSHF                 ;save flag register

                    ;setup registers for display

000B BB 0008              MOV   BX,8            ;load count
000E B4 7F                MOV   AH,7FH          ;load selection pattern
0010 BE 00FF R            MOV   SI,OFFSET MEM-1 ;address data
0013 BA 0701              MOV   DX,701H         ;address Port B

                    ;display 8 digits
```

```
0016                    DISP1:
0016 8A C4                    MOV   AL,AH           ;select a digit
0018 EE                       OUT   DX,AL
0019 4A                       DEC   DX              ;address Port A
001A 8A 00                    MOV   AL,[BX+SI]      ;get 7-segment data
001C EE                       OUT   DX,AL
001D E8 029A R                CALL  DELAY           ;wait one millisecond
0020 D0 CC                    ROR   AH,1            ;address next digit
0022 42                       INC   DX              ;address Port B
0023 4B                       DEC   BX              ;adjust count
0024 75 F0                    JNZ   DISP1           ;repeat 8 times

0026 5E                       POPF                  ;restore registers
                              RET

002C                    DISP  ENDP
```

The display procedure (DISP) addresses an area of memory where the data, in 7-segment code, is stored for the eight display digits called MEM. The AH register is loaded with a code (7FH) that initially addresses the most-significant display position. Once this position is selected, the contents of memory location MEM +7 is addressed and sent to the most-significant digit. The selection code is then adjusted to select the next display digit, as is the address. This process repeats eight times to display the contents of location MEM through MEM +7 on the eight display digits.

An LCD Display Interfaced to the 82C55. LCDs (liquid crystal displays) are quickly replacing LED displays in many applications. The only disadvantage of the LCD display is that it is difficult to see in low-light situations in which the LED is still in limited use.

Figure 10–21 illustrates the connection of the Optrex DMC–20481 LCD display to an 82C55. The DMC–20481 is a 4-line by 20-characters-per-line display that accepts ASCII code as input data. It also accepts commands that initialize it and control its application. As you can see in Figure 10–21, the LCD display has few connections. The data connections, which are attached to the 82C55 Port A, are used to input display data and to read information from the display.



**Note:* Current max is 480 mA, nominal 260 mA

FIGURE 10–21 The DMC-20481 LCD display interfaced to the 82C55.

There are four control pins on the display. The VEE connection is used to adjust the contrast of the LED display and is normally connected to a 10 K✱ potentiometer, as illustrated. The RS (register select) input selects data (RS = 1) or instructions (RS = 0). The E (enable) input must be a logic 1 for the DMC–20481 to read or write information. Finally, the R/W pin selects a read or a write operation. Normally, the RS pin is placed at a 1 or 0, the R/W pin is set or cleared, data are placed on the data input pins, and then the E pin is pulsed to access the DMC–20481. This display also has two inputs for back-lighting LED diodes, which are not shown in the illustration.

In order to program the DMC–20481 we must first initialize it. This applies to any display that uses the HD44780 (Hitachi) display driver integrated circuit. The entire line of small display panels from Optrex is programmed in the same manner. Initialization is accomplished via the following steps:

1. Wait at least 15 ms after Vcc rises to 5.0 V.
2. Output the function set command (30H), and wait at least 4.1 ms.
3. Output the function set command (30H) a second time, and wait at least 100 ❍s.
4. Output the function set command (30H) a third time, and wait at least 40 ❍s.
5. Output the function set command (38H) a fourth time, and wait at least 40 ❍s.
6. Output a 08H to disable the display, and wait at least 40 ❍s.
7. Output a 01H to home the cursor and clear the display, and wait at least 1.64 ms.
8. Output the enable display cursor off (0CH), and wait at least 40 ❍s.
9. Output a 06H to select auto-increment, shift the cursor, and wait at least 40 ❍s.

The software to accomplish the initialization of the LCD display is listed in Example 10–10. It is long, but the display controller requires the long initialization dialog. Note that the software for the three time delays is not included in the listing. If you are interfacing to a PC, you can use the clock tick discussed in Chapter 7 for the time delay. One clock tick can be used for all timing in this software, even though the LCD display will function faster than your eye at 1/18 seconds. If you are developing the interface for another application, then you must write three separate time delays, which must provide the delay times indicated in the initialization dialog.

EXAMPLE 10–10

```
                ;procedure to initialize the LCD display
                ;
0010            INIT  PROC  NEAR

0010 BA 0303          MOV   DX,CMD8255  ;address 8255 command register
0013 B0 80            MOV   AL,80H      ;all ports are output ports
0015 EE               OUT   DX,AL
0016 B0 00            MOV   AL,0        ;clear Port B
0018 BA 0301          MOV   DX,PORTB
001B EE               OUT   DX,AL
001C E8 004A          CALL  DELAY15     ;wait 15 ms
001F B0 30            MOV   AL,30H      ;first function set command
0021 E8 002A          CALL  OUTCMD      ;send it
0024 E8 0056          CALL  DELAY41     ;wait 4.1 ms
0027 B0 30            MOV   AL,30H
0029 E8 0022          CALL  OUTCMD      ;second function set command
002C B0 30            MOV   AL,30H      ;third function set command
002E E8 001D          CALL  OUTCMD
0031 B0 38            MOV   AL,38H      ;fourth function set command
0033 E8 0018          CALL  OUTCMD
0036 E8 0058          CALL  DELAY100
0039 B0 08            MOV   AL,08H      ;display off
003B E8 0010          CALL  OUTCMD
003E B0 01            MOV   AL,01H      ;clear display
0040 E8 000B          CALL  OUTCMD
0043 B0 0C            MOV   AL,0CH      ;display on, cursor off
```

```
0045 E8 0006          CALL  OUTCMD
0048 B0 06            MOV   AL,06H       ;auto-increment, shift cursor
004A E8 0001          CALL  OUTCMD
004D C3               RET
004E            INIT  ENDP
                ;
                ;procedure to output a command
                ;
004E            OUTCMD   PROC  NEAR
004E 50               PUSH  AX           ;save command
004F BA 0303          MOV   DX,CMD8255   ;select 8255 command resgiter
0052 B0 80            MOV   AL,80H       ;all ports are outputs
0054 EE               OUT   DX,AL
0055 BA 0300          MOV   DX,PORTA
0058 58               POP   AX
0059 EE               OUT   DX,AL        ;command to port A
005A 42               INC   DX           ;address Port B
005B B0 04            MOV   AL,4         ;put 1 on E, 0 on R/W# and 0 on S
005D EE               OUT   DX,AL
005E 90               NOP                ;extra time so E = 1 longer
005F 90               NOP
0060 B0 00            MOV   AL,0
0062 EE               OUT   DX,AL        ;clear E
0063 E8 0029          CALL  DELAY100     ;wait 100 us
0066 C3               RET
0067            OUTCMD   ENDP
```

The NOP instructions are added in the OUTCMD procedure to ensure that the E bit remains a logic 1 long enough to activate the LCD display. This process should work in most systems at most clock frequencies, but additional NOP instructions may be needed to lengthen this time in some cases.

Before programming data to the display, the commands used in the initialization dialog must be explained. See Table 10–3 for a complete listing of the commands or instructions for the LCD display. Compare the commands sent to the LCD display in the initialization program to Table 10–3.

Once the LCD display is initialized, a few procedures are needed to display information and control the display. After initialization, time delays are no longer needed when sending data or many commands to the display. The clear display command still needs a time delay because the busy flag is not used with that command. Instead of a time delay, the busy flag is tested to see whether the display has completed an operation. A procedure to test the busy flag appears in Example 10–11. The BUSY procedure tests the LCD display and only returns when the display has completed a prior instruction.

EXAMPLE 10–11

```
                ;procedure to test busy and return if not busy
;0010            BUSY  PROC  NEAR
                    .REPEAT
0010 BA 0303           MOV   DX,CMD8255     ;select 8255 command register
□
0013 B0 90             MOV   AL,90H         ;port A input
0015 EE                OUT   DX,AL
0016 BA 0301           MOV   DX,PORTB       ;select port B
0019 B0 01             MOV   AL,1           ;R/W# = 1
001B EE                OUT   DX,AL
001C B0 05             MOV   AL,5           ;R/W# = 1, E = 1, RS = 0
001E EE                OUT   DX,AL
001F 90                NOP                  ;delay to allow access
0020 90                NOP
0021 BA 0300           MOV   DX,PORTA       ;select port A
0024 EC                IN    AL,DX          ;get status of busy flag
```

```
0025 50              PUSH  AX
0026 BA 0301         MOV   DX,PORTB
0029 B0 00           MOV   AL,0
002B EE              OUT   DX,AL
002C BA 0303         MOV   DX,CMD8255
002F B0 80           MOV   AL,80H
0031 EE              OUT   DX,AL
0032 58              POP   AX
0033 D0 E0           SHL   AL,1
                   .UNTIL !CARRY?          ;until not busy
0037 C3            RET
0038           BUSY  ENDP
```

Once the BUSY procedure is available, data can be sent to the display by writing another procedure called WRITE. The WRITE procedure uses BUSY to test before trying to write new data to the display. Example 10–12 shows the WRITE procedure, which transfers the ASCII character from the BL register to the current cursor position of the display. Note that the initialization dialog has sent the cursor for auto-increment, so if WRITE is called more than once, the characters written to the display will appear one next to the other, as they would on a video display.

TABLE 10–3 Instructions for most Optrex LCD displays.

Instruction	*Code*	*Description*	*Execution Time*
Clear display	0000 0001	Clears display and homes the cursor	1.64 ms
Cursor home	0000 0010	Homes the cursor	1.64 ms
Entry mode set	0000 0AS	Sets cursor movement direction (A=1 increment) and shift (S=1 shift)	40 µs
Display on/off	0000 1DCB	Sets display on/off (D=1 on) (C=1 cursor on) (B=1 cursor blink)	40 µs
Cursor/display shift	0001 SR00	Sets cursor movement and display shift (S=1 shift display) (S=0 move cursor) (R=1 right)	40 µs
Function set	001L NF00	Programs chip (L=1 8-bit, L=0 4-bits) (N=1 2 lines) (F=1 5x10, F=0 5x7)	40 µs
Set CGRAM address	01XX XXXX	Sets character generator RAM address	40 µs
Set DRAM address	10XX XXXX	Sets display RAM address	40 µs
Read busy flag	B000 0000	Reads busy flag (B=1 busy)	0
Write data	Data	Writes data to display or character generator RAM	40 µs
Read data	Data	Reads data from display or character generator RAM	40 µs

EXAMPLE 10–12

```
                ;procedure that writes the ASCII contents of the BL
                ;register to the display
                ;
003A            WRITE PROC  NEAR
003A BA 0303          MOV   DX,CMD8255
003D B0 80            MOV   AL,80H
003F EE               OUT   DX,AL
0040 BA 0300          MOV   DX,PORTA    ;data to port A
0043 8A C3            MOV   AL,BL
0045 EE               OUT   DX,AL
0046 BA 0301          MOV   DX,PORTB
0049 B0 02            MOV   AL,2        ;RS = 1, R/W# = 0, E = 0
004B EE               OUT   DX,AL
004C 90               NOP
004D 90               NOP
004E B0 06            MOV   AL,6        ;RS = 1, R/W# = 0, E = 1
0050 EE               OUT   DX,AL
0051 90               NOP
0052 90               NOP
0053 B0 00            MOV   AL,0        ;RS = 0, R/W# = 0, E = 0
0055 EE               OUT   DX,AL
0056 E8 FFB7          CALL  BUSY        ;wait for LCD
0059 C3               RET
005A            WRITE ENDP
```

The only other procedure that is needed for a basic display is the clear and home cursor procedure called CLS, shown in Example 10–13. With CLS and the procedures presented thus far, you can display any message on the display, clear it, display another message, and basically operate the display. As mentioned earlier, the clear command requires a time delay (at least 1.64 ms) instead of a call to BUSY for proper operation. In this procedure, we used the 4.1 ms time delay.

EXAMPLE 10–13

```
                ;procedure to clear the display and home the cursor
                ;
005A            CLS   PROC  NEAR
005A BA 0303          MOV   DX,CMD8255
005D B0 80            MOV   AL,80H
005F EE               OUT   DX,AL
0060 BA 0300          MOV   DX,PORTA
0063 B0 01            MOV   AL,1        ;clear instruction
0065 EE               OUT   DX,AL
0066 BA 0301          MOV   DX,PORTB
0069 B0 04            MOV   AL,4        ;RS = 0, R/W# = 0, E = 1
006B EE               OUT   DX,AL
006C 90               NOP
006D 90               NOP
006E B0 00            MOV   AL,0        ;RS = 0, R/W# = 0, E = 0
0070 EE               OUT   DX,AL
0071 E8 0044          CALL  DELAY41
0074 C3               RET
0075            CLS   ENDP
```

Additional procedures that could be developed might select a display RAM position. The display RAM address starts at 0 and progresses across the display until the last character address on the first line is location 19, location 20 is the first display position of the second line, and so forth. Once you can move the display address, you can change individual characters on the display and even read data from the display. These procedures are for you to develop if they are needed.

A word about the display RAM inside of the LCD display. The LCD contains 128 bytes of memory, addresssed from 00H to 7FH. Not all of this memory is used. For example, the one-line × 20-character display uses only the first 20 bytes of memory (00H–13H.) The first line of any of these displays always starts at address 00H. The second line of any display powered by the HD44780 always begins at address 40H. For example, a two-line × 40-character display uses addresses 00H–27H to store ASCII-coded data from the first line. The second line is stored at addresses 40H–67H for this display. In the four-line displays, the first line is at 00H, the second is at 40H, the third is at 14H, and the last line is at 54H. The largest display device that uses the HD44780 is a two-line × 40-character display. The four-line by 40-character display uses an M50530 or a pair of HD44780s. Because information on these devices can be readily found on the Internet, they are not covered in the text

A Stepper Motor Interfaced to the 82C55. Another device often interfaced to a computer system is the *stepper motor.* A stepper motor is a digital motor because it is moved in discrete steps as it traverses through 360⑤ A common stepper motor is geared to move perhaps 15⑤per step in an inexpensive stepper motor, to 1⑤per step in a more costly high-precision stepper motor. In all cases, these steps are gained through many magnetic poles and/or gearing. Notice that two coils are energized in Figure 10–22. If less power is required, one coil may be energized at a time, causing the motor to step at 45⑤ 135⑤ 225⑤ and 315⑤

Figure 10–22 shows a four-coil stepper motor that uses an armature with a single pole. Notice that the stepper motor is shown four times with the armature (permanent magnetic) rotated to four discrete places. This is

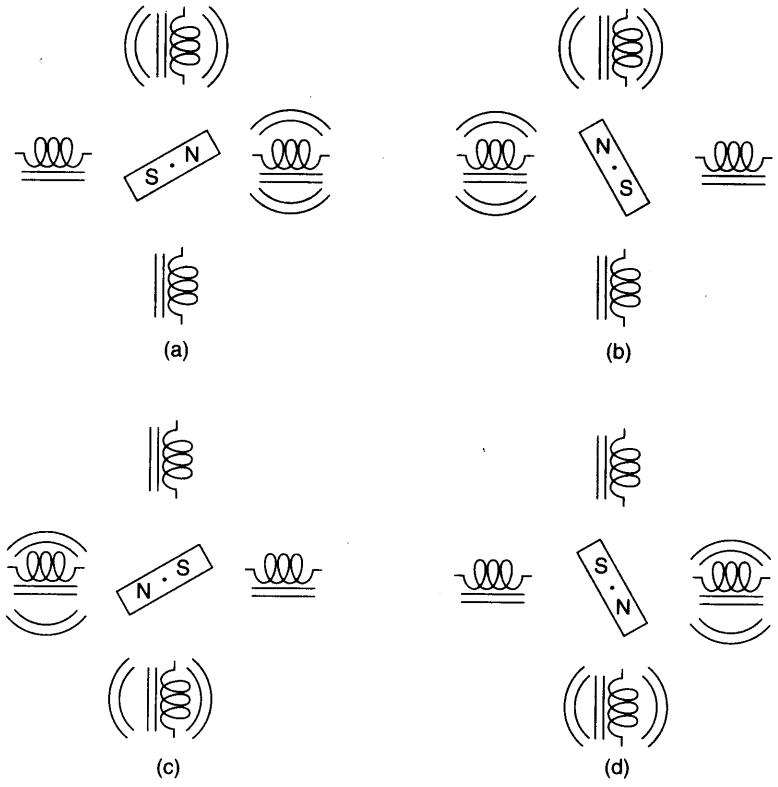


FIGURE 10–22 The stepper motor showing full-step operation. (a) 45° (b) 135° (c) 225° (d) 315°.

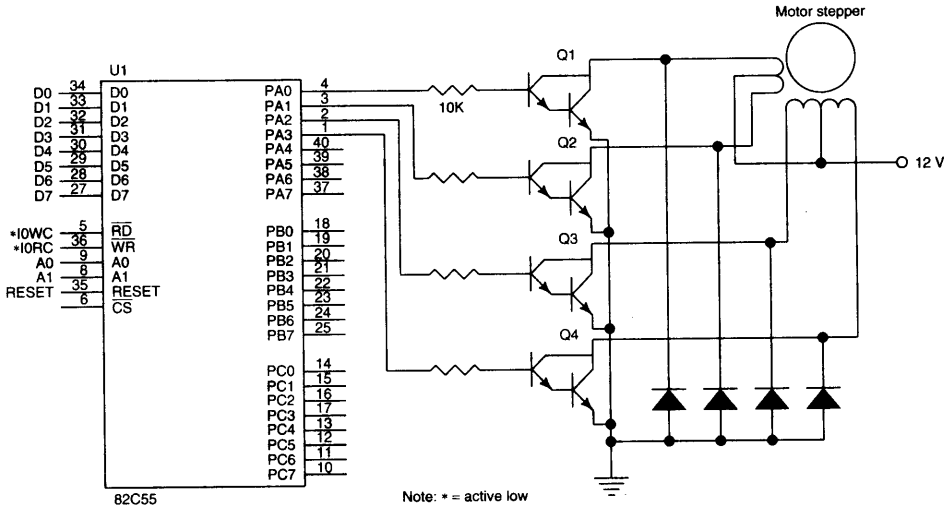


FIGURE 10–23 A stepper motor interfaced to the 82C55. This illustration does not show the decoder.

accomplished by energizing the coils, as shown. This is an illustration of full stepping. The stepper motor is driven by using NPN Darlington amplifier pairs to provide a large current to each coil.

A circuit that can drive this stepper motor is illustrated in Figure 10–23, with the four coils shown in place. This circuit uses the 82C55 to provide it with the drive signals that are used to rotate the armature of the motor in either the right-hand or left-hand direction.

A simple procedure that drives the motor (assuming that port A is programmed in mode 0 as an output device) is listed in Example 10–14. This subroutine is called, with CX holding the number of steps and direction of the rotation. If CX > 8000H, the motor spins in the right-hand direction; if CX < 8000H, it spins in the left-hand direction. The leftmost bit of CX is removed and the remaining 15 bits contain the number of steps. Notice that the procedure uses a time delay (not illustrated) that causes a 1 ms time delay. This time delay is required to allow the stepper-motor armature time to move to its next position.

EXAMPLE 10–14

```
= 0040                  PORT  EQU   40H          ;assign Port A
                        ;
                        ;A procedure to control stepper motor.
                        ;
0000                    STEP  PROC  NEAR

0000  A0 0000 R               MOV   AL,POS       ;get position
0003  81 F9 8000              CMP   CX,8000H
0007  77 10                   JA    RH           ;if right-hand direction
0009  83 F9 00                CMP   CX,0
000C  74 14                   JE    STEP_OUT     ;if no steps
000E                    STEP1:
000E  D0 C0                   ROL   AL,1         ;step left
0010  E6 40                   OUT   PORT,AL
0012  E8 0011                 CALL  DELAY        ;wait one millisecond
0015  E2 F7                   LOOP  STEP1        ;repeat until CX = 0
0017  EB 09                   JMP   STEP_OUT
```

```
0019                    RH:
0019  81 E1 7FFF              AND   CX,7FFFH      ;clear bit 15
001D                    RH1:
001D  D0 C8                   ROR   AL,1          ;step right
001F  E6 40                   OUT   PORT,AL
0021  E8 0006                 CALL  DELAY         ;wait one millisecond
0024  E2 F7                   LOOP  RH1           ;repeat until CX = 0
0026                    STEP_OUT:
0026  A2 0000                MOV   POS,AL        ;save position
0029  C3                      RET

0029                    STEP  ENDP
```

The current position is stored in memory location POS, which must be initialized with 33H, 66H, 0CCH, or 99H. This allows a simple ROR (step right) or ROL (step left) instruction to rotate the binary bit pattern for the next step.

Stepper motors can also be operated in the half-step mode, which allows eight steps per sequence. This is accomplished by using the full-step sequence described with a half step obtained by energizing one coil interspersed between the full steps. Half-stepping allows the armature to be positioned at 0.5 90 5 180 5 and 270 5 The half-step position codes are 11H, 22H, 44H, and 88H. A complete sequence of eight steps would follow as: 11H, 33H, 22H, 66H, 44H, 0CCH, 88H, and 99H. This sequence could be either output from a lookup table or generated with software.

Key Matrix Interface. Keyboards come in a vast variety of sizes, from the standard 101-key QWERTY keyboards interfaced to the microprocessor to small specialized keyboards that may contain only four to 16 keys. This section of the text concentrates on the smaller keyboards that may be purchased preassembled or may be constructed from individual key switches.

Figure 10–24 illustrates a small key-matrix that contains 16 switches interfaced to ports A and B of an 82C55. In this example, the switches are formed into a 4 × 4 matrix, but any matrix could be used such as a 2 × 8. Notice how the keys are organized into four rows (ROW0–ROW3) and four columns (COL0–COL3). Each row is connected to 5.0 V through a 10 K✱ pull-up resistor to ensure that the row is pulled high when no push-button switch is closed.

The 82C55 is decoded (PAL program is not shown) at I/O ports 50H–53H for an 8088 microprocessor. Port A is programmed as an input port to read the rows and port B is programmed as an output port to select a column. For example, if 1110 is output to port B pins PB3–PB0, column 0 has a logic 1, so the four keys in column 0 are selected. Notice that with a logic 0 on PB0, the only switches that can place a logic 0 onto port A are switches 0–3. If switches 4–F are closed, the corresponding port A pins remain a logic 1. Likewise, if a 1101 is output to port B, switches 4–7 are selected, and so forth.

A flowchart of the software required to read a key from the keyboard matrix and de-bounce the key is illustrated in Figure 10–25. Keys must be de-bounced, which is normally accomplished with a short time delay of from 10–20 ms. The flowchart contains three main sections. The first waits for the release of a key. This seems awkward, but software executes very quickly in a microprocessor and there is a possibility that the program will return to the top of this program before the key is released, so we must wait for a release first. Next, the flowchart shows that we wait for a keystroke. Once the keystroke is detected, the position of the key is calculated in the final part of the flowchart.

The software uses a procedure called SCAN to scan the keys and another called DELAY to waste 10 ms of time for de-bouncing. The main keyboard procedure is called KEY and it appears with the others in Example 10–15. Note that the SCAN procedure is generic, so it can handle any keyboard configuration from a 2 × 2 matrix to an 8 × 8 matrix. Changing the two equates at the start of the program (ROW and COL) will change the configuration of the software for any size keyboard. Also note that the steps required to initialize the 82C55 so that port A is an input port and port B is an output port are not shown.

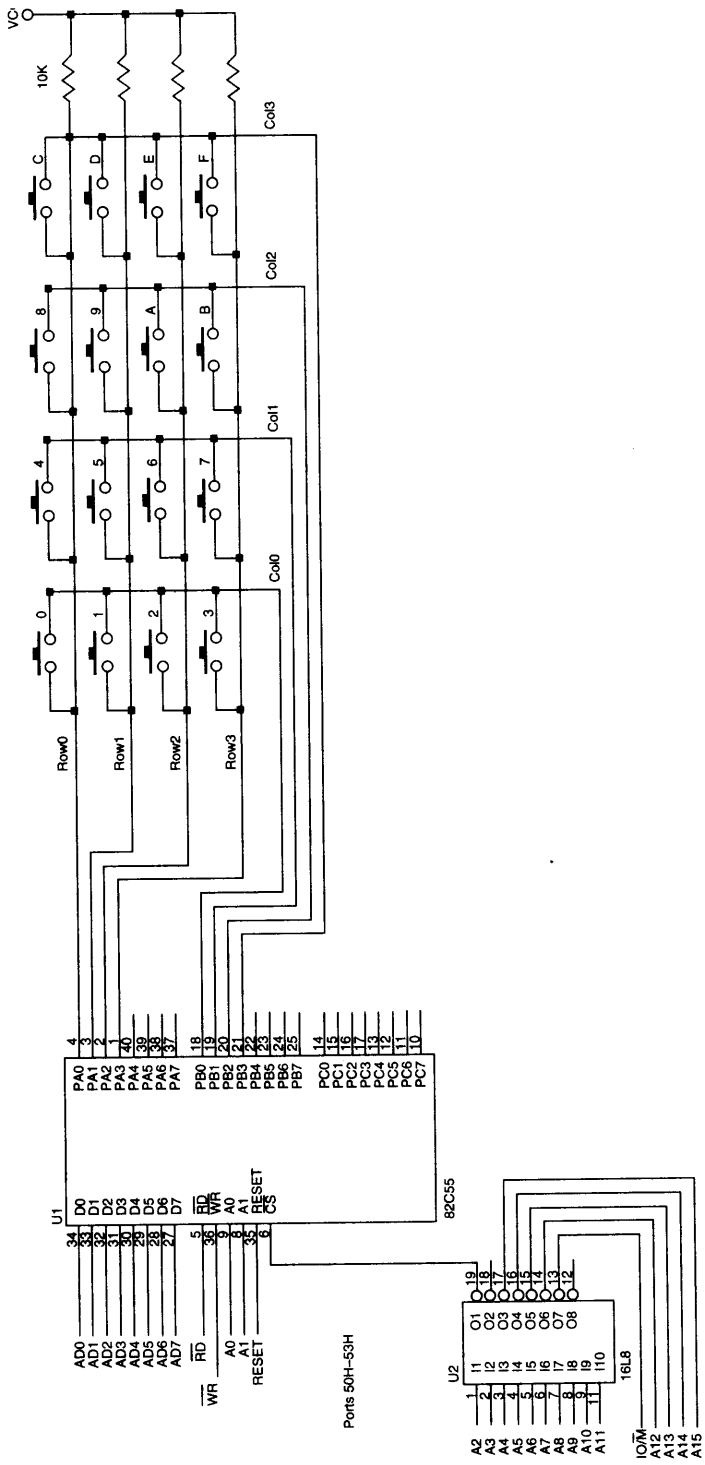


FIGURE 10–24 A 4 x 4 keyboard matrix connected to an 8088 microprocessor through the 82C55 PPI.

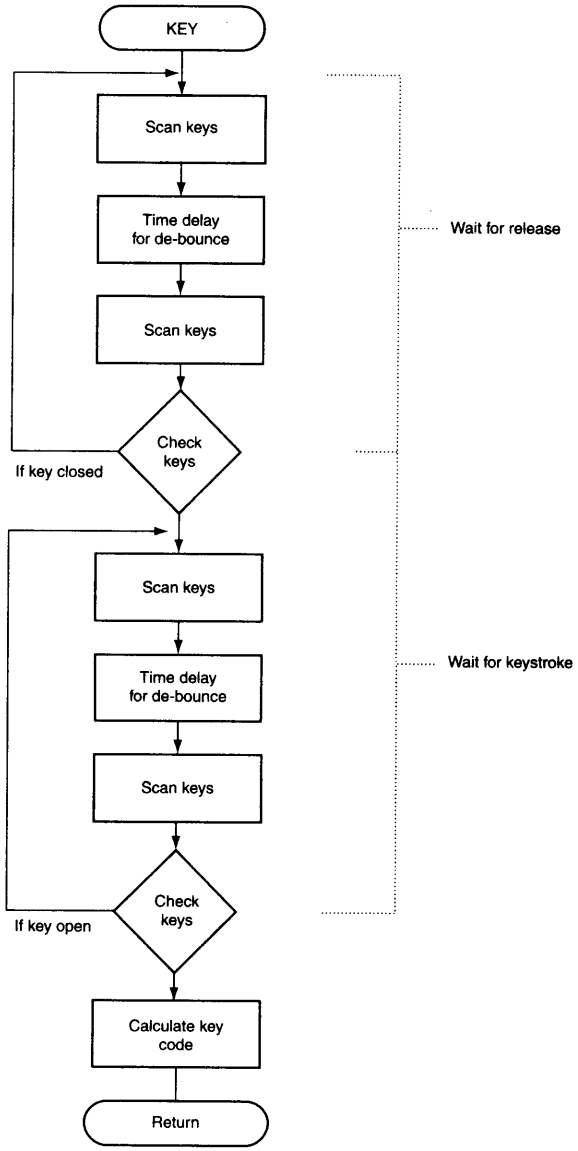


FIGURE 10–25 The flowchart of a keyboard-scanning procedure.

EXAMPLE 10–15

```
                    ;A keyboard procedure that scans the keyboard and
                    ;returns with the numeric code of the key in AL.
                    ;
= 0004              ROWS  EQU   4          ;number of rows
= 0004              COLS  EQU   4          ;number of columns
= 0050              PORTA EQU   50H        ;port A address
= 0051              PORTB EQU   51H        ;port B address

0000            KEY   PROC  NEAR USES CX
```

```
0001  E8 002F          CALL  SCAN          ;test all keys
0004  75 FA           JNZ   KEY           ;if key closed
0006  E8 0048          CALL  DELAY         ;wait for about 10 ms
0009  E8 0027          CALL  SCAN          ;test all keys
000C  75 F2           JNZ   KEY           ;if key closed
000E                KEY1:
000E  E8 0022          CALL  SCAN          ;test all keys
0011  74 FB           JZ    KEY1          ;if no key closed
0013  E8 003B          CALL  DELAY         ;wait for about 10 ms
0016  E8 001A          CALL  SCAN          ;test all keys
0019  74 F3           JZ    KEY1          ;if no key closed
001B  50              PUSH  AX            ;save row codes
001C  B0 04           MOV   AL,COLS       ;calculate starting row key
001E  2A C1           SUB   AL,CL
0020  B5 04           MOV   CH,ROWS
0022  F6 E5           MUL   CH
0024  8A C8           MOV   CL,AL
0026  FE C9           DEC   CL
0028  58              POP   AX
0029                KEY2:
0029  D0 C8           ROR   AL,1          ;find row position
002B  FE C1           INC   CL
002D  72 FA           JC    KEY2
002F  8A C1           MOV   AL,CL         ;mode code to AL
                      RET

0033                KEY   ENDP
0033                SCAN  PROC  NEAR USES BX
0034  B1 04           MOV   CL,ROWS       ;form row mask
0036  B7 FF           MOV   BH,0FFH
0038  D2 E7           SHL   BH,CL
003A  B9 0004          MOV   CX,COLS       ;load column count
003D  B3 FE           MOV   BL,0FEH       ;get selection code
003F                SCAN1:
003F  8A C3           MOV   AL,BL         ;select column
0041  E6 51           OUT   PORTB,AL
0043  D0 C3           ROL   BL,1
0045  E4 50           IN    AL,PORTA      ;read rows
0047  0A C7           OR    AL,BH
0049  3C FF           CMP   AL,0FFH       ;test for a key
004B  75 02           JNZ   SCAN2
004D  E2 F0           LOOP  SCAN1
004F                SCAN2:
                      RET

0051                SCAN  ENDP

0051                DELAY PROC  NEAR USES CX

0052 B9 1388          MOV   CX,5000       ;10ms (8MHz clock)
0055                DELAY1:
0055 E2 FE            LOOP  DELAY1
                      RET

0059                DELAY ENDP
```

A note about the SCAN procedure. The time between where the keyboard column is selected and where the rows are read is very short. In a very high-speed system, a small time delay must be placed between these two points for the data at port A to settle to its final state. In most cases, this is not needed—the SCAN procedure should not scan the display at a rate higher than 30 KHz. If it does, the Federal Communications Commission (FCC) will not approve its application in any accepted system.

Mode 1 Strobed Input

Mode 1 operation causes port A and/or port B to function as latching input devices. This allows external data to be stored into the port until the microprocessor is ready to retrieve it. Port C is also used in mode 1 operation—not for data, but for control or handshaking signals that help operate either or both port A and port B as strobed input ports. Figure 10–26 shows how both ports are structured for mode 1 strobed input operation and the timing diagram.

The strobed input port captures data from the port pins when the strobe (STB) is activated. Note that the strobe captures the port data on the 0-to-1 transition. The STB signal causes data to be captured in the port and it activates the IBF **(input buffer full)** and INTR **(interrupt request)** signals. Once the microprocessor, through software (IBF) or hardware (INTR), notices that data are strobed into the port, it executes an IN instruction to read the port (RD). The act of reading the port restores both IBF and INTR to their inactive states until the next datum is strobed into the port.

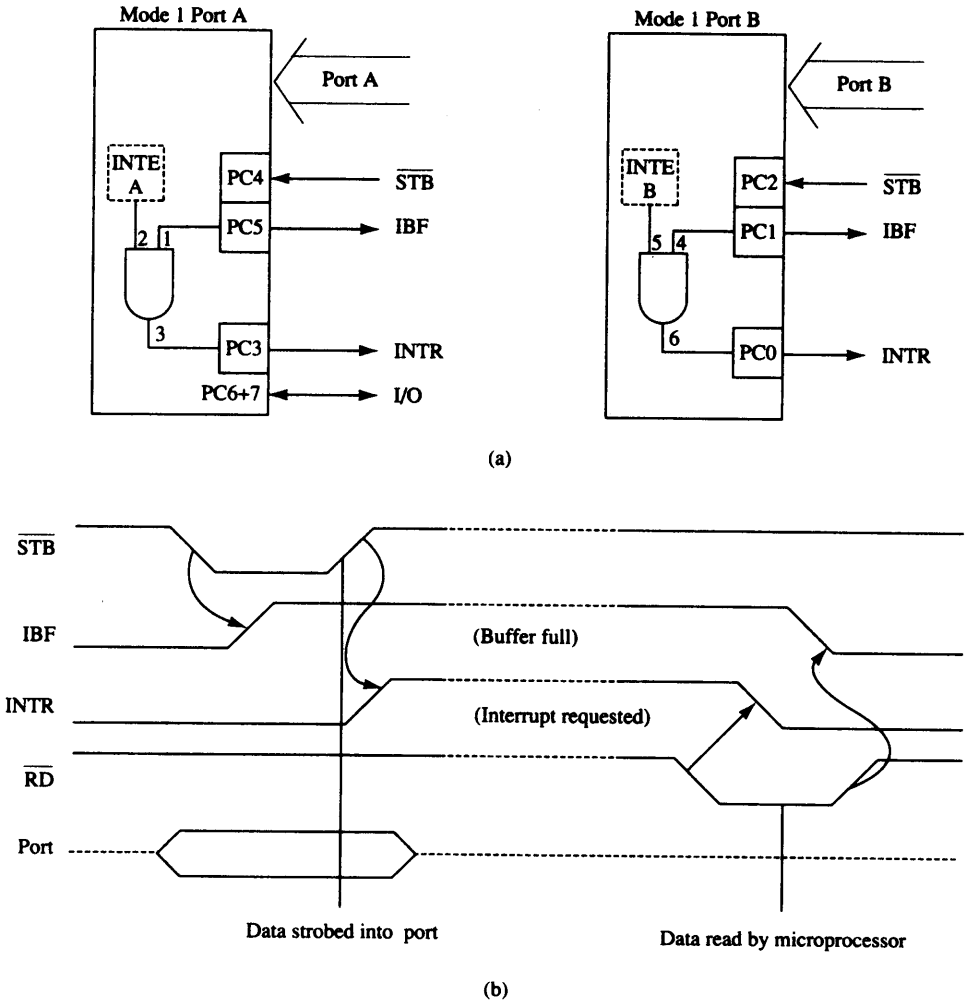


FIGURE 10–26 Strobed input operation (mode 1) of the 82C55. (a) Internal structure and (b) timing diagram.

Signal Definitions for Mode 1 Strobed Input

STB	The **strobe** input loads data into the port latch, which holds the information until it is input to the microprocessor via the IN instruction.
IBF	**Input buffer full** is an output indicating that the input latch contains information.
INTR	**Interrupt request** is an output that requests an interrupt. The INTR pin becomes a logic 1 when the STB input returns to a logic 1, and is cleared when the data are input from the port by the microprocessor.
INTE	The **interrupt enable** signal is neither an input nor an output; it is an internal bit programmed via the port PC4 (port A) or PC2 (port B) bit position.
PC7, PC6	The port C pins 7 and 6 are general-purpose I/O pins that are available for any purpose.

Strobed Input Example. An excellent example of a strobed input device is a keyboard. The keyboard encoder de-bounces the key-switches, and provides a strobe signal whenever a key is depressed and the data output contain the ASCII-coded key code. Figure 10–27 illustrates a keyboard connected to strobed input port A. Here DAV (**data available**) is activated for 1.0 μs each time that a key is typed on the keyboard. This causes data to be strobed into port A because DAV is connected to the STB input of port A. Each time a key is typed, therefore, it is stored into port A of the 82C55. The STB input also activates the IBF signal, indicating that data are in port A.

Example 10–16 shows a procedure that reads data from the keyboard each time a key is typed. This procedure reads the key from port A and returns with the ASCII code in AL. To detect a key, port C is read and the IBF bit (bit position PC5) is tested to see whether the buffer is full. If the buffer is empty (IBF = 0), then the procedure keeps testing this bit, waiting for a character to be typed on the keyboard.

EXAMPLE 10–16

```
                        ;A procedure that reads the keyboard encoder
                        ;and returns the ASCII character in AL.
                        ;
= 0020                  BIT5  EQU   20H
= 0022                  PORTC EQU   22H
```

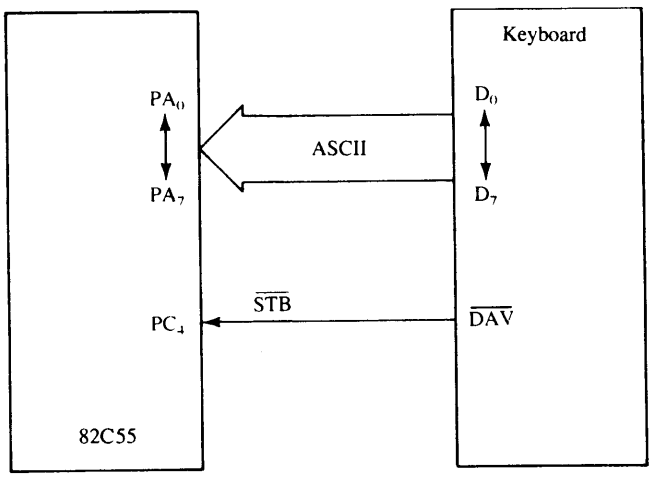


FIGURE 10–27 Using the 82C55 for strobed input operation of a keyboard.

```
= 0020                  PORTA EQU   20H

0000                    READ  PROC  NEAR

0000  E4 22                   IN    AL,PORTC      ;read Port C
0002  A8 20                   TEST  AL,BIT5       ;test IBF
0004  74 FA                   JZ    READ          ;if IBF = 0
0006  E4 20                   IN    AL,PORTA      ;read data
0008  C3                      RET

0009                    READ  ENDP
```

Mode 1 Strobed Output

Figure 10–28 illustrates the internal configuration and timing diagram of the 82C55 when it is operated as a strobed output device under mode 1. Strobed output operation is similar to mode 0 output operation, except that control signals are included to provide handshaking.

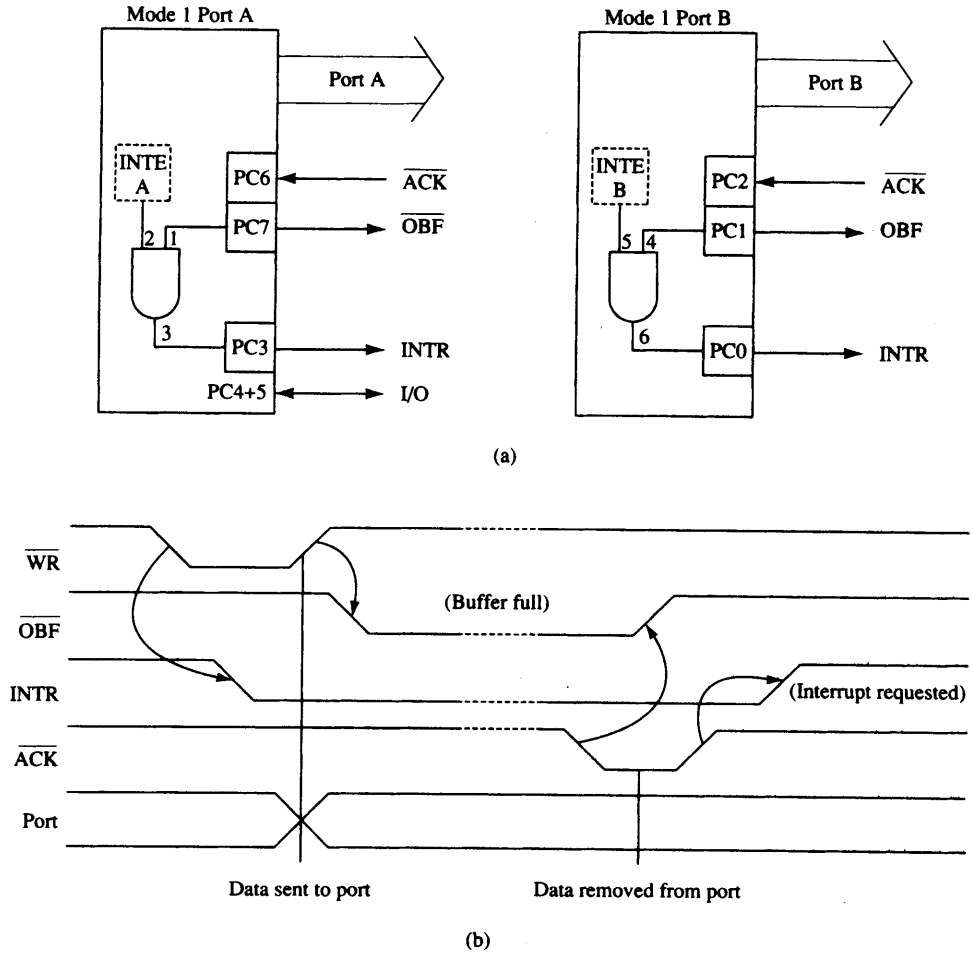


FIGURE 10–28 Strobed output operation (mode 1) of the 82C55. (a) Internal structure and (b) timing diagram.

Whenever data are written to a port programmed as a strobed output port, the OBF **(output buffer full)** signal becomes a logic 0 to indicate that data are present in the port latch. This signal indicates that data are available to an external I/O device that removes the data by strobing the $\overline{ACK}$ **(acknowledge)** input to the port. The $\overline{ACK}$ signal returns the OBF signal to a logic 1, indicating that the buffer is not full.

Signal Definitions for Mode 1 Strobed Output

OBF	**Output buffer full** is an output that goes low whenever data are output (OUT) to the port A or port B latch. This signal is set to a logic 1 whenever the ACK pulse returns from the external device.
ACK	The **acknowledge** signal causes the OBF pin to return to a logic 1 level. The ACK is a response from an external device, indicating that it has received the data from the 82C55 port.
INTR	**Interrupt request** is a signal that often interrupts the microprocessor when the external device receives the data via the ACK signal. This pin is qualified by the internal INTE **(interrupt enable)** bit.
INTE	**Interrupt enable** is neither an input nor an output; it is an internal bit programmed to enable or disable the INTR pin. The INTE A bit is programmed as PC6 and INTE B is programmed as PC2.
PC5, PC4	Port C pins 5 and 4 are general-purpose I/O pins. The bit set and reset command may be used to set or reset these two pins.

Strobed Output Example. The printer interface discussed in Section 10–1 is used here to demonstrate how to achieve strobed output synchronization between the printer and the 82C55. Figure 10–29 illustrates port B connected to a parallel printer, with eight data inputs for receiving ASCII-coded data, a DS **(data strobe)** input to strobe data into the printer, and an ACK output to acknowledge the receipt of the ASCII character.

In this circuit, there is no signal to generate the DS signal to the printer, so PC4 is used with software that generates the DS signal. The ACK signal that is returned from the printer acknowledges the receipt of the data and is connected to the ACK input of the 82C55.

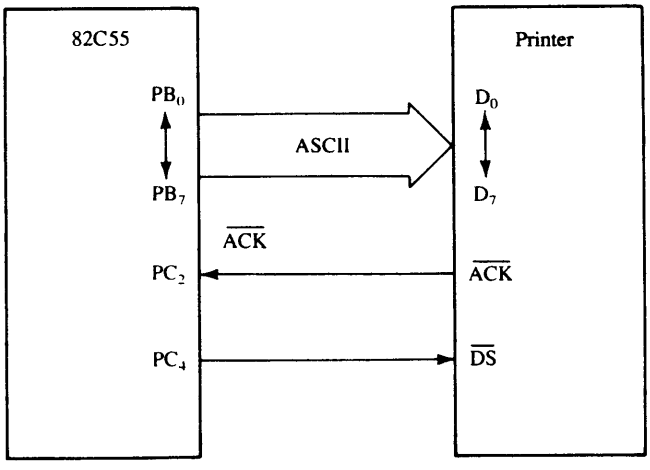


FIGURE 10–29 The 82C55 connected to a parallel printer interface that illustrates the strobed output mode of operation for the 82C55.

Example 10–17 lists the software that sends the ASCII-coded character in AH to the printer. The procedure first tests OBF to decide whether the printer has removed the data from port B. If not, the procedure waits for the ACK signal to return from the printer. If OBF = 1, then the procedure sends the contents of AH to the printer through port B and also sends the DS signal.

EXAMPLE 10–17

```
                        ;A procedure that transfers the ASCII character
                        ;from AH to the printer via port B.
                        ;
= 0002                  BIT1  EQU   2
= 0062                  PORTC EQU   62H
= 0061                  PORTB EQU   61H
= 0063                  CMD   EQU   63H

0000                    PRINT PROC  NEAR

                        ;check printer ready

0000  E4 62                   IN    AL,PORTC      ;get OBF
0002  A8 02                   TEST  AL,BIT1       ;test OBF
0004  74 FA                   JZ    PRINT         ;if OBF = 0

                        ;send character to printer

0006  8A C4                   MOV   AL,AH         ;get data
0008  E6 61                   OUT   PORTB,AL      ;print data

                        ;send data strobe to printer

000A  B0 08                   MOV   AL,8          ;clear DS
000C  E6 63                   OUT   CMD,AL
000E  B0 09                   MOV   AL,9          ;set DS
0010  E6 63                   OUT   CMD,AL
0012  C3                      RET

0013                    PRINT ENDP
```

Mode 2 Bi-directional Operation

In mode 2, which is allowed with group A only, port A becomes bi-directional, allowing data to be transmitted and received over the same eight wires. Bi-directional bused data are useful when interfacing two computers. It is also used for the IEEE–488 parallel high-speed GPIB **(general purpose instrumentation bus)** interface standard. Figure 10–30 shows the internal structure and timing diagram for mode 2 bi-directional operation.

Signal Definitions for Bi-directional Mode 2

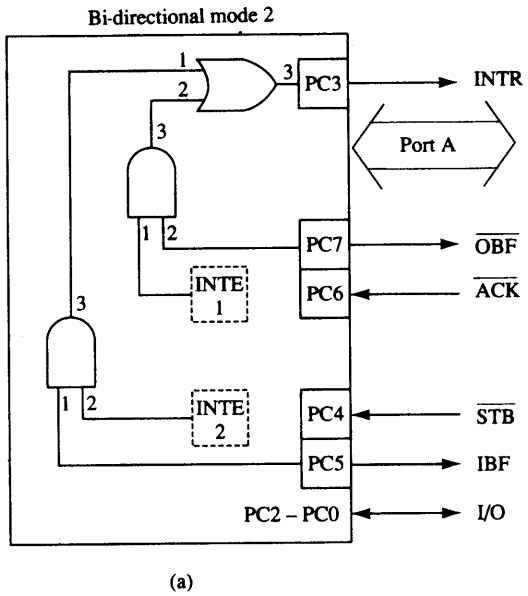
INTR — **Interrupt request** is an output used to interrupt the microprocessor for both input and output conditions.

OBF — **Output buffer full** is an output indicating that the output buffer contains data for the bi-directional bus.

ACK — **Acknowledge** is an input that enables the three-state buffers so that data can appear on port A. If ACK is a logic 1, the output buffers of port A are at their high-impedance state.

STB — The **strobe** input loads the port A input latch with external data from the bi-directional port A bus.

IBF — **Input buffer full** is an output used to signal that the input buffer contains data for the external bi-directional bus.



(a)

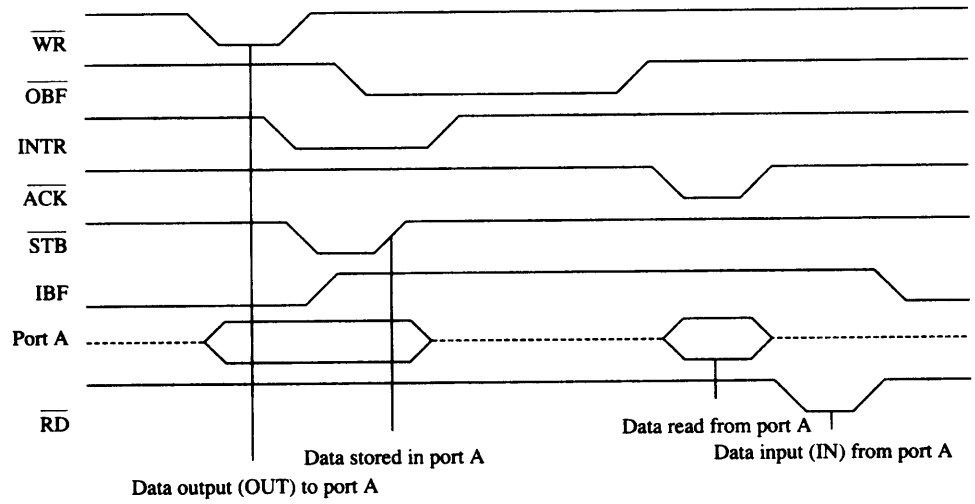


FIGURE 10–30 Mode 2 operation of the 82C55. (a) Internal structure and (b) timing diagram.

INTE — **Interrupt enable** are internal bits (INTE1 and INTE2) that enable the INTR pin. The state of the INTR pin is controlled through port C bits PC6 (INTE1) and PC4 (INTE2).

PC2, PC1, and PC0 — These pins are general-purpose I/O pins in mode 2 controlled by the bit set and reset command.

The Bi-directional Bus. The bi-directional bus is used by referencing port A with the IN and OUT instructions. To transmit data through the bi-directional bus, the program first tests the OBF signal to determine whether the output buffer is empty. If it is, then data are sent to the output buffer via the OUT instruction. The external circuitry also

monitors the OBF signal to decide whether the microprocessor has sent data to the bus. As soon as the output circuitry sees a logic 0 on OBF, it sends back the ACK signal to remove it from the output buffer. The ACK signal sets the OBF bit and enables the three-state output buffers so that data may be read. Example 10–18 lists a procedure that transmits the contents of the AH register through bi-directional port A.

EXAMPLE 10–18

```
                        ;A procedure that transmits AH through the bi-
                        ;directional bus of port A.
                        ;
= 0080                  BIT7  EQU   80H
= 0062                  PORTC EQU   62H
= 0060                  PORTA EQU   60H

0000                    TRANS PROC  NEAR

0000  E4 62                   IN    AL,PORTC     ;get OBF
0002  A8 80                   TEST  AL,BIT7      ;test OBF
0004  74 FA                   JZ    TRANS        ;if OBF = 1

0006  8A C4                   MOV   AL,AH        ;get data
0008  E6 60                   OUT   PORTA,AL     ;send data
000A  C3                      RET

000B                    TRANS ENDP
```

To receive data through the bi-directional port A bus, the IBF bit is tested with software to decide whether data have been strobed into the port. If IBF = 1, then data are input using the IN instruction. The external interface sends data into the port by using the STB signal. When STB is activated, the IBF signal becomes a logic 1 and the data at port A are held inside the port in a latch. When the IN instruction executes, the IBF bit is cleared and the data in the port are moved into AL. Example 10–19 lists a procedure that reads data from the port.

EXAMPLE 10–19

```
                        ;A procedure that reads data from the bi-
                        ;directional port A and returns it in AL.
                        ;
= 0020                  BIT5  EQU   20H
= 0062                  PORTC EQU   62H
= 0060                  PORTA EQU   60H

0000                    READ  PROC  NEAR

0000  E4 62                   IN    AL,PORTC       ;get IBF
0002  A8 20                   TEST  AL,BIT5        ;test IBF
0004  74 FA                   JZ    READ           ;if IBF = 0
0006  E4 60                   IN    AL,PORTA       ;get data
0008  C3                      RET

0009                    READ  ENDP
```

The INTR **(interrupt request)** pin can be activated from both directions of data flow through the bus. If INTR is enabled by both INTE bits, then the output and input buffers both cause interrupt requests. This occurs when data are strobed into the buffer using STB or when data are written using OUT.

82C55 Mode Summary

Figure 10–31 shows a graphical summary of the three modes of operation for the 82C55. Mode 0 provides simple I/O, mode 1 provides strobed I/O, and mode 2 provides bi-directional I/O. As mentioned, these modes are selected through the command register of the 82C55.

		Mode 0		Mode 1		Mode 2
Port A		IN	OUT	IN	OUT	I/O
Port B		IN	OUT	IN	OUT	Not used
	0			$INTR_B$	$INTR_B$	I/O
	1			IBF_B	$\overline{OBF}_B$	I/O
	2			$\overline{STB}_B$	$\overline{ACK}_B$	I/O
Port C	3	IN	OUT	$INTR_A$	$INTR_A$	INTR
	4			$\overline{STB}_A$	I/O	$\overline{STB}$
	5			IBF_A	I/O	IBF
	6			I/O	$\overline{ACK}_A$	$\overline{ACK}$
	7			I/O	$\overline{OBF}_A$	$\overline{OBF}$

FIGURE 10–31 A summary of the port connections for the 82C55 PPI.

10–4 THE 8279 PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

The 8279 is a programmable keyboard and display interfacing component that scans and encodes up to a 64-key keyboard and controls up to a 16-digit numerical display. The keyboard interface has a built-in first-in, first-out (FIFO) buffer that allows it to store up to eight keystrokes before the microprocessor must retrieve a character. The display section controls up to 16 numeric displays from an internal 16 x 8 RAM that stores the coded display information.

Basic Description of the 8279

As we shall see, the 8279 is designed to easily interface with any microprocessor. Figure 10–32 illustrates the pin-out of this device. The definition of each pin connection follows.

Pin Definitions for the 8279

A0 The A0 address input selects data or control for reads and writes between the microprocessor and the 8279. A logic 0 selects data and a logic 1 selects control or status register.

BD **Blank** is an output used to blank the displays.

CLK **Clock** is an input that generates the internal timing for the 8279. The maximum allowable frequency on the CLK pin is 3.125 MHz for the 8279-5 and 2.0 MHz for the 8279. Other timings require wait states in microprocessors executing at above 5 MHz.

CN/ST **Control/strobe** is an input normally connected to the Control key on a keyboard.

CS **Chip select** is an input that enables the 8279 for programming, reading the keyboard and status information, and writing control and display data.

DB7–DB0 The **data bus** consists of bi-directional pins that connect to the data bus on the microprocessor.

IRQ **Interrupt request** is an output that becomes a logic 1 whenever a key is pressed on the keyboard. This signal indicates that keyboard data are available for the microprocessor.

OUTA3–OUTA0 Outputs that send data to the displays (most-significant).

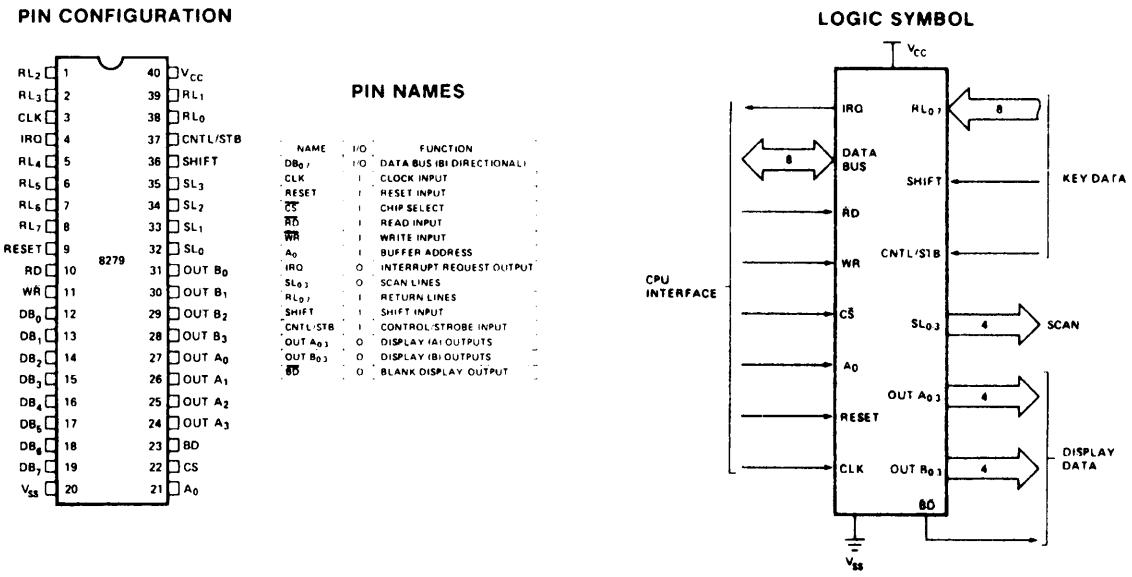


FIGURE 10–32 The pin-out and logic symbol of the 8279 programmable keyboard/display interface. (Courtesy of Intel Corporation.)

OUTB3–OUTB0	Outputs that send data to the displays (least-significant).
RD	The **read** input is directly connected to the IORC or RD signal from the system. The RD input causes, when CS is a logic 0, a read from the data registers or status register.
RESET	The **reset** input connects to the system RESET signal.
RL7–RL0	**Return lines** are inputs used to sense any key depression in the keyboard matrix.
SHIFT	The **shift** input normally connects to the Shift key on a keyboard.
SL3–SL0	The **scan line** outputs scan both the keyboard and the displays.
WR	**Write** is an input that connects to either the write strobe signal that is developed with external logic. The WR input causes data to be written to either the data registers or control registers within the 8279.
VCC	A power supply pin connected to the system +5.0 V bus.
VSS	A ground pin connected to the system ground.

Interfacing the 8279 to the Microprocessor

In Figure 10–33, the 8279 is connected to the 8088 microprocessor. The 8279 is decoded to function at 8-bit I/O address 10H and 11H, where port 10H is the data port and 11H is the control port. This circuit uses a PAL16L8 (see Example 10–20) to decode the I/O address for the 8279. Address bus bit A0 selects either the data or control port. Notice that the CS signal selects the 8279 and also provides a signal called WAIT2 that is used to cause two wait states so that this device can function with an 8 MHz 8088.

The only signal not connected to the microprocessor is the IRQ output. This is an interrupt request pin and is beyond the scope of this section of the text. Chapter 11 explains interrupts and where they operate and function in a system.

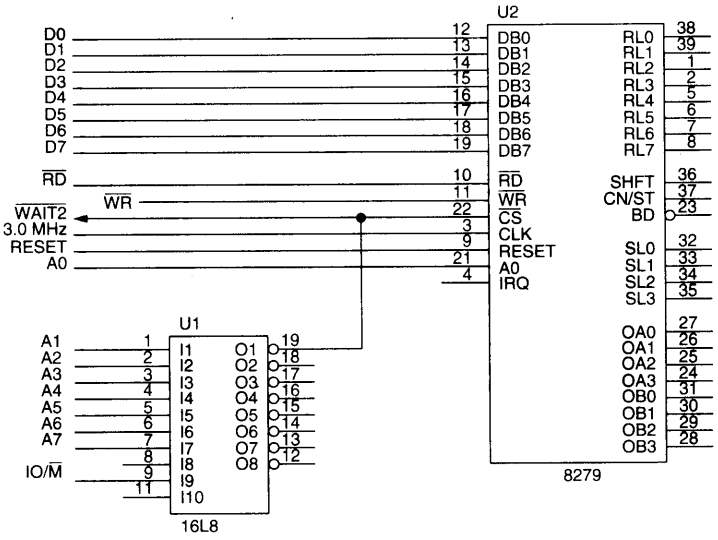


FIGURE 10–33 The 8279 interfaced to the 8088 microprocessor to function at 8-bit I/O ports 10H and 11H.

EXAMPLE 10–20

```
TITLE       Address Decoder
PATTERN     Test 14
REVISION    A
AUTHOR      Barry B. Brey
COMPANY     BreyCo
DATE        7/10/99
CHIP        DECODERE PAL16L8

;pins  1  2  3  4  5  6  7  8   9  10
      A1 A2 A3 A4 A5 A6 A7 NC IOM GND

;pins 11 12 13 14 15 16 17 18 19 20
      NC NC NC NC NC NC NC NC CS VCC

EQUATIONS

/CS = /A7 * /A6 * /A5 * A4 * /A3 * /A2 * /A1 * IOM
```

Keyboard Interface

Suppose that a 64-key keyboard (with no numeric displays) is connected through the 8279 to the 8088microprocessor. Figure 10–34 shows this connection, as well as the keyboard. With the 8279, the keyboard matrix is any size from a 2 × 2 matrix (four keys) to an 8 × 8 matrix (64 keys). (Note that each crossover point in the matrix contains a normally open push-button switch that connects one vertical column with one horizontal row when a key is pressed.)

The I/O port number decoded is the same as that decoded for Figure 10–33. The I/O port number is 10H for the data port and 11H for the control port in this circuit.

The 74ALS138 decoder generates eight active low column strobe signals for the keyboard. The selection pins SL2–SL0 sequentially scan each column of the keyboard, and the internal circuitry of the 8279 scans the RL pins, searching for a key switch closure. Pull-up resisters, normally found on input lines of a keyboard, are not required because the 8279 contains its own internal pull-ups on the RL inputs.

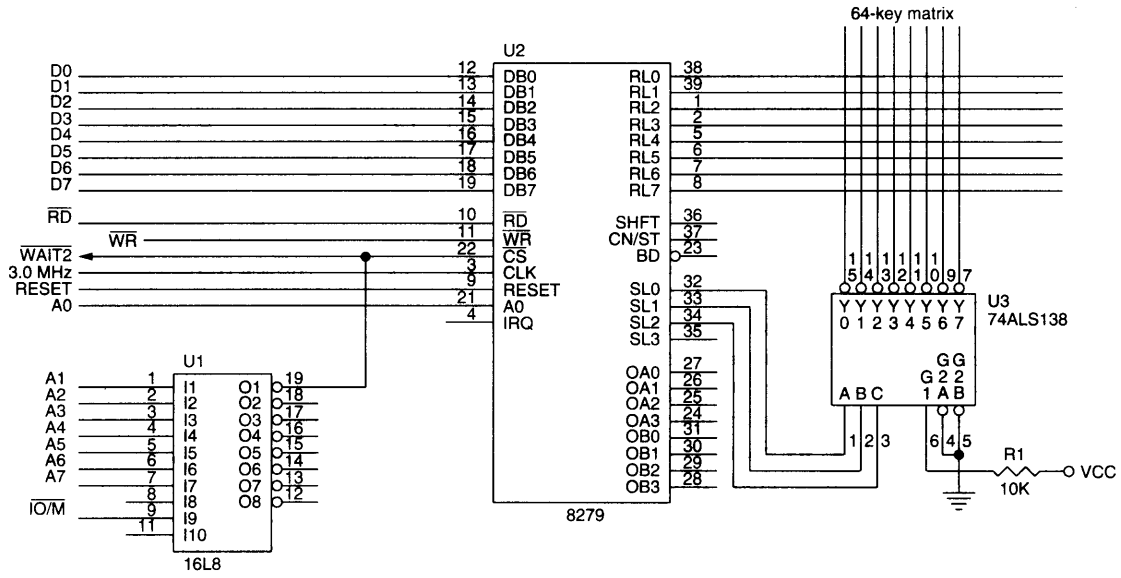


FIGURE 10–34 A 64-key keyboard interfaced to the 8088 microprocessor through the 8279.

TABLE 10–4 The 8279 control word summary.

D_7	D_6	D_5	*Function*	*Purpose*
0	0	0	Mode set	Selects the number of display positions, left or right entry, and type of keyboard scan
0	0	1	Clock	Programs the internal clock and sets the scan and debounce times
0	1	0	Read FIFO	Selects the type of FIFO read and the address of the read
0	1	1	Read display	Selects the type of display read and the address of the read
1	0	0	Write display	Selects the type of write and the address of the write
1	0	1	Display write inhibit	Allows half-bytes to be blanked
1	1	0	Clear	Clears the display or FIFO
1	1	1	End interrupt	Clears the IRQ signal to the microprocessor

Programming the Keyboard Interface. Before any keystroke is detected, the 8279 must be programmed—a more involved procedure than with the 82C55. The 8279 has eight control words to consider before it is programmed. The first three bits of the number sent to the control port (11H, in this example) select one of the eight different control words. Table 10–4 lists all eight control words and briefly describes them.

Control Word Descriptions. Following is a list of the control words that program the 8279. Note that the first three bits are the control register number from Table 10–4, which are followed by other binary bits of information as they apply to each control.

000DDMMM **Mode set** is a command with an opcode of 000 and two fields programmed to select the mode of operation for the 8279. The DD field selects the mode of operation for the displays (see Table 10–5), and the MMM field selects the mode of operation for the keyboard (see Table 10–6).

The DD field selects either an 8- or 16-digit display, and determines whether new data are entered to the rightmost or leftmost display position. The MMM field is quite a bit more complex: it provides encoded, decoded, or strobed keyboard operation.

In encoded mode, the SL outputs are active-high, and follow the binary bit pattern 0 through 7 or 0 through 15, depending whether 8- or 16-digit displays are selected. In decoded mode, the SL outputs are active-low, and only one of the four outputs is low at any given instant. The decoded outputs repeat the pattern: 1110, 1101, 1011, and 0111. In strobed mode, an active-high pulse on the CN/ST input pin strobes data from the RL pins into an internal FIFO, where they are held for the microprocessor.

It is also possible to select either 2-key lockout or N-key rollover. 2-key lockout prevents two keys from being recognized, if pressed simultaneously. N-key rollover will accept all keys pressed simultaneously, from first to last.

TABLE 10–5 Binary bit assignment for DD of the mode set control word.

DD	*Function*
00	8-digit display with left entry
01	16-digit display with left entry
10	8-digit display with right entry
11	16-digit display with right entry

TABLE 10–6 Binary bit assignment for MMM of the mode set control word.

MMM	*Function*
000	Encoded keyboard with 2-key lockout
001	Decoded keyboard with 2-key lockout
010	Encoded keyboard with N-key rollover
011	Decoded keyboard with N-key rollover
100	Encoded sensor matrix
101	Decoded sensor matrix
110	Strobed keyboard, encoded display scan
111	Strobed keyboard, decoded display scan

001PPPPP The **clock command** word programs the internal clock divider. The code PPPPP is a prescaler that divides the clock input pin (CLK) to achieve the desired operating frequency of approximately 100 KHz. An input clock of 1 MHz thus requires a prescaler of 01010_2 for PPPPP.

010Z0AAA The **read FIFO** control word selects the address of a keystroke from the internal FIFO buffer. Bit positions AAA select the desired FIFO location from 000 to 111, and Z selects auto-increment for the address. Under normal operation, this control word is used only with the sensor matrix operation of the 8279.

011ZAAAA The **display read** control word selects the read address of one of the display RAM positions for reading through the data port. AAAA is the address of the position to be read and Z selects auto-increment mode. This command is used if the information stored in the display RAM must be read.

100ZAAAA The **write display** control word selects the write address of one of the displays. AAAA addresses the position to be written to through the data port, and Z selects auto-increment so subsequent writes through the data port are to subsequent display positions.

1010WWBB The **display write inhibit** control word inhibits writing to either half of each display RAM location. The leftmost W inhibits writing to the leftmost four bits of the display RAM location, and the rightmost W inhibits the rightmost four bits. The BB field functions in a like manner, except that they blank (turn off) either half of the output pins.

1100CCFA The **clear** control word clears the display, the FIFO, or both the display and FIFO. Bit F clears the FIFO and the display RAM status, and sets the address pointer to 000. If the

CC bits are 00 or 01, all of the display RAM locations become 0000000; if CC = 10, all locations become 00100000; and if CC = 11, all locations become 11111111.

111E000 The **end of interrupt** control word is issued to clear the IRQ pin to zero in the sensor matrix mode. If E is a 1, the special error mode is used. In the special error mode, the status register indicates if multiple key closures have occurred.

The large number of control words make programming the keyboard interface appear complex. Before anything is programmed, the clock divider rate must be determined. In the circuit illustrated in Figure 10–34, we use a 3.0 MHz clock input signal. To program the prescaler to generate a 100 KHz internal rate, we program PPPPP of the clock control word with a 30 or 11110_2.

The next step involves programming the keyboard type. The example keyboard in Figure 10–34 is an encoded keyboard. Notice that the circuit includes an external decoder that converts the encoded data from the SL pins into decoded column-selection signals. We are free in this example to choose either 2-key lockout or N-key rollover, but most applications use 2-key lockout.

Finally, we program the operation of the FIFO. Once the FIFO is programmed, it never needs to be reprogrammed unless we need to read prior keyboard codes. Each time a key is typed, the data are stored in the FIFO; if they are read from the FIFO before the FIFO is full (eight characters), the data from the FIFO follows the same order as the typed data. Example 10–21 provides the software required to initialize the 8279 to control the keyboard illustrated in Figure 10–34.

EXAMPLE 10–21

```
                    ;Initialization dialog for the keyboard interface
                    ;of Figure 10-34.
                    ;
0000  B0 3E               MOV   AL,00111110B    ;program clock
0002  E6 11               OUT   11H,AL

0004  B0 00               MOV   AL,0
0006  E6 11               OUT   11H,AL          ;program mode

0008  B0 50               MOV   AL,01010000B
000A  E6 11               OUT   11H,AL          ;program FIFO
```

Once the 8279 is initialized, a procedure is required to read data from the keyboard. We determine whether a character is typed in the keyboard by looking at the FIFO status register. Whenever the control port is addressed by the IN instruction, the contents of the FIFO status word is copied into the AL register. Figure 10–35 shows the contents of the FIFO status register and defines the purpose of each status bit.

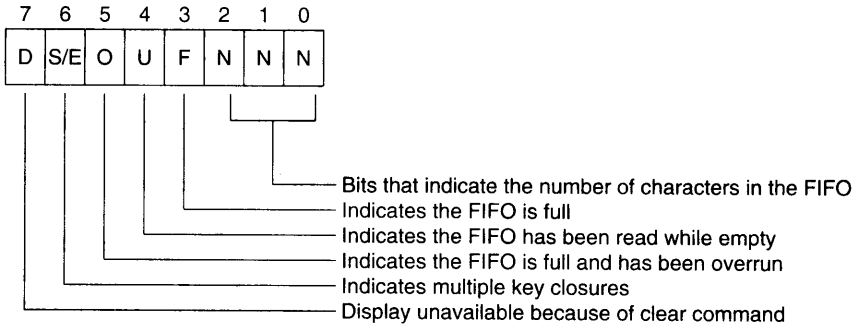


FIGURE 10–35 The 8279–5 FIFO status register.

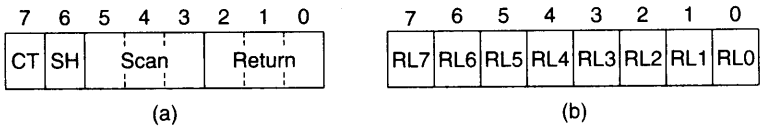


FIGURE 10–36 The (a) scanned keyboard code and (b) strobed keyboard code for the 8279–5 FIFO.

The procedure listed in Example 10–22 first tests the FIFO status register to see whether it contains any data. If NNN = 000, the FIFO is empty. Upon determining that the FIFO is not empty, the procedure inputs data to AL and returns with the keyboard code in AL.

EXAMPLE 10–22

```
                        ;A procedure that reads data from the FIFO and
                        ;returns it in AL.
                        ;
= 0007                  MASKS EQU   7

0000                    READ  PROC  NEAR

0000  E4 11                   IN    AL,11H     ;read status
0002  A8 07                   TEST  AL,MASKS   ;test NNN
0004  74 FA                   JZ    READ       ;if NNN = 0
0006  E4 10                   IN    AL,10H     ;read FIFO data
0008  C3                      RET

0009                    READ  ENDP
```

The data found in AL upon returning from the subroutine contains raw data from the keyboard. Figure 10–36 shows the format of this data for both the scanned and strobed modes of operation. The scanned code is returned from our keyboard interface and is converted to ASCII code by using the XLAT instruction with an ASCII code lookup table. The scanned code is returned with the row and column number occupying the rightmost six bits.

The SH bit shows the state of the shift pin and the CT bit shows the state of the control pin. In the strobed mode, the contents of the eight RL inputs appear as they are sampled by placing a logic 1 on the strobe input pin to the 8279.

Six-Digit Display Interface

Figure 10–37 depicts the 8279 connected to the 8088 microprocessor and a 6-digit numeric display. This interface uses a PAL16L8 (program not shown) to decode the 8279 at I/O ports 20H (data) and 21H (control/status). The segment data are supplied to the displays through the OUTA and OUTB pins of the 8279. These bits are buffered by a segment driver (ULN2003A) to drive the segment inputs to the display.

A 74ALS138 3-to-8 line decoder enables the anode switches of each display position. The SL2–SL0 pins supply the decoder with the encoded display position from the 8279. Notice that the left-hand display is at position 0101 and the right-hand display is at position 0000. These are the addresses of the display positions, as indicated in control words for the 8279.

It is necessary to choose resistor values that allow 60 mA of current flow per segment. In this circuit, we use 47 ✷ resistors. If we allow 60 mA of segment current, then the average segment current is 10 mA, or one-sixth of 60 mA because current only flows for one-sixth of the time through a segment. The anode switches must supply the current for all seven segments plus the decimal point. Here, the total anode current is 8 × 60 mA, or 480 mA.

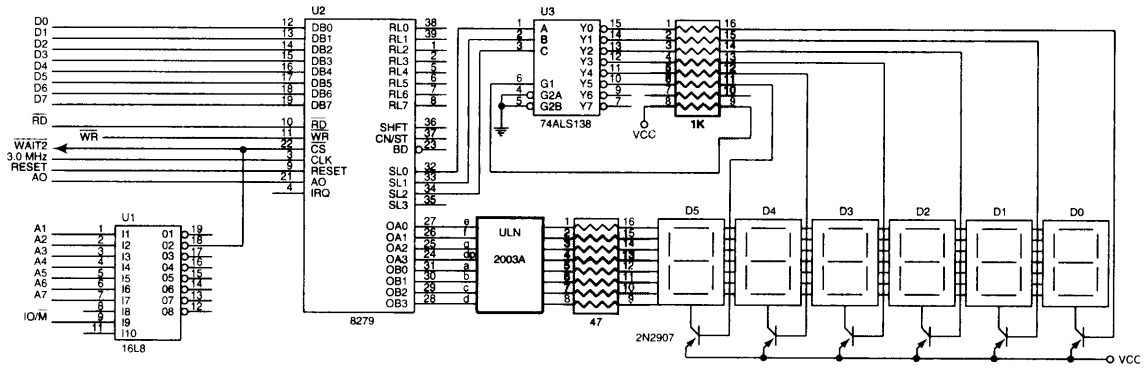


FIGURE 10–37 A 6-digit numeric display interfaced to the 8279.

Example 10–23 lists the initialization dialog for programming the 8279 to function with this 6-digit display. This software programs the display and clears the display RAM.

EXAMPLE 10–23

```
                    ;Initialization dialog for the 6-digit display of
                    ;Figure 10-37.
                    ;
0000  B0 3E               MOV   AL,00111110B    ;program clock
0002  E6 21               OUT   21H,AL

0004  B0 00               MOV   AL,0            ;program mode set
0006  E6 21               OUT   21H,AL

0008  B0 C1               MOV   AL,11000001B    ;clear display
000A  E6 21               OUT   21H,AL
```

Example 10–24 lists a procedure for displaying information on the displays. Data are transferred to the procedure through the AX register. AH contains the 7-segment display code and AL contains the address of the displayed digit.

EXAMPLE 10–24

```
                    ;A procedure that displays AH on the display
                    ;position addressed by AL.
                    ;
= 0080              MASKS EQU   80H

0000                DISP  PROC  NEAR

0000  50                  PUSH  AX          ;save data
0001  0C 80               OR    AL,MASKS    ;select digit
0003  E6 21               OUT   21H,AL
0005  8A C4               MOV   AL,AH       ;display data
0007  E6 20               OUT   20H,AL
0009  58                  POP   AX          ;restore data
000A  C3                  RET

000B                DISP   ENDP
```

10–5 8254 PROGRAMMABLE INTERVAL TIMER

The 8254 programmable interval timer consists of three independent 16-bit programmable counters **(timers).** Each counter is capable of counting in binary or binary-coded decimal (BCD). The maximum allowable input frequency to any counter is 10 MHz. This device is useful wherever the microprocessor must control real-time events. Some examples of usage include real-time clock, events counter, and motor speed and direction control.

This timer also appears in the personal computer decoded at ports 40H–43H to do the following:

1. Generate a basic timer interrupt that occurs at approximately 18.2 Hz.
2. Cause the DRAM memory system to be refreshed.
3. Provide a timing source to the internal speaker and other devices. The timer in the personal computer is an 8253 instead of an 8254.

8254 Functional Description

Figure 10–38 shows the pin-out of the 8254, which is a higher-speed version of the 8253, and a diagram of one of the three counters. Each timer contains a CLK input, a gate input, and an output (OUT) connection. The CLK input provides the basic operating frequency to the timer, the gate pin controls the timer in some modes, and the OUT pin is where we obtain the output of the timer.

The signals that connect to the microprocessor are the data bus pins (D7–D0), RD, WR, CS, and address inputs A1 and A0. The address inputs are present to select any of the four internal registers used for pro amming, reading, or writing to a counter. The personal computer contains an 8253 timer or its equivalent, decoded at I/O ports 40H–43H. Timer zero is programmed to generate an 18.2 Hz signal that interrupts the microprocessor at interrupt vector 8 for a clock tick. The tick is often used to time programs and events. Timer 1 is programmed for 15 ms, which is used on the PC/XT personal computer to request a DMA action used to refresh the dynamic RAM. Timer 2 is programmed to generate tone on the personal computer speaker.

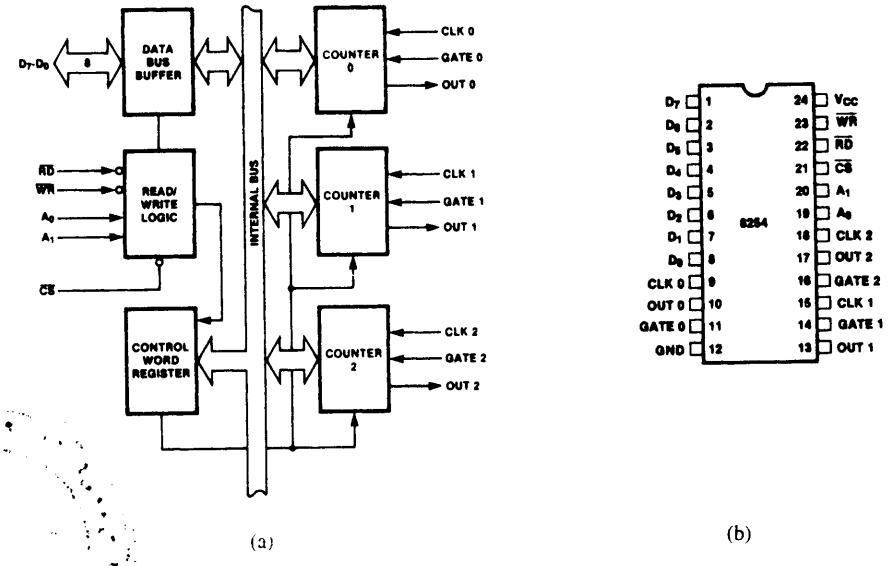


FIGURE 10–38 The 8254 programmable interval timer. (a) Internal structure and (b) pin-out. (Courtesy of Intel Corporation.)

Pin Definitions

A1, A0 The **address inputs** select one of four internal registers within the 8254. See Table 10–7 for the function of the A1 and A0 address bits.

TABLE 10–7 Address selection inputs to the 8254.

A_1	A_0	*Function*
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control word

CLK The **clock** input is the timing source for each of the internal counters. This input is often connected to the PCLK signal from the microprocessor system bus controller.

CS **Chip select** enables the 8254 for programming, and reading or writing a counter.

G The **gate** input controls the operation of the counter in some modes of operation.

GND **Ground** connects to the system ground bus.

OUT A **counter output** is where the wave-form generated by the timer is available.

RD **Read** causes data to be read from the 8254 and often connects to the IORC signal.

Vcc **Power** connects to the +5.0 V power supply.

WR **Write** causes data to be written to the 8254 and often connects to the write strobe (IOWC).

Programming the 8254

Each counter is individually programmed by writing a control word, followed by the initial count. Figure 10–39 lists the program control word structure of the 8254. The **control word** allows the programmer to select the counter, mode of operation, and type of operation (read/write). The control word also selects either a binary or BCD count. Each counter may be programmed with a count of 1 to FFFFH. A count of 0 is equal to FFFFH+1

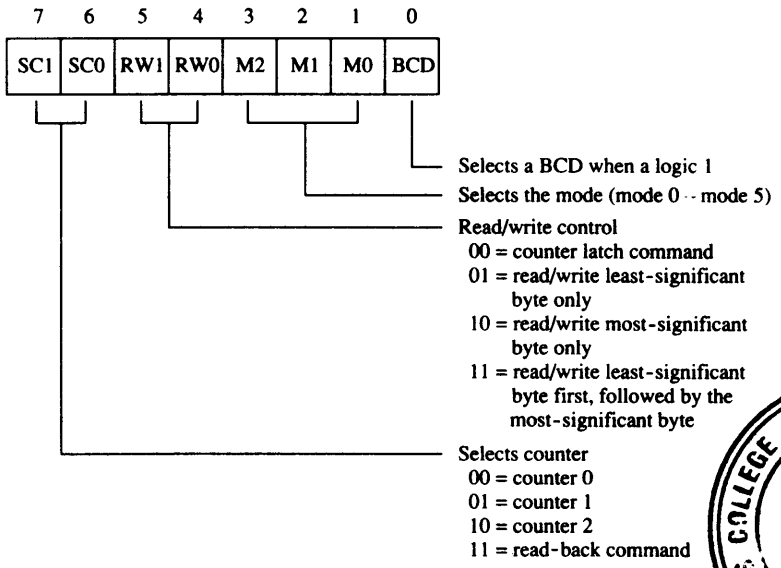


FIGURE 10–39 The control word for the 8254–2 timer.

SRINIVAS COLLEGE OF P.G. MANAGEMENT STUDIES

(65,536) or 10,000 in BCD. The minimum count of 1 applies to all modes of operation except modes 2 and 3, which have a minimum count of 2. Timer 0 is used in the personal computer with a divide by count of 64K (FFFFH) to generate the 18.2 Hz (18.196 Hz) interrupt clock tick. Timer 0 has a clock input frequency of 4.77 MHz ÷ 4 or 1.1925 MHz.

The control word uses the BCD bit to select a BCD count (BCD = 1) or a binary count (BCD = 0). The M2, M1, and M0 bits select one of the 6 different modes of operation (000–101) for the counter. The RW1 and RW0 bits determine how the data are read from or written to the counter. The SC1 and SC0 bits select a counter or the special read back mode of operation, discussed later in this section.

Each counter has a program control word used to select the way the counter operates. If two bytes are programmed into a counter, then the first byte (LSB) will stop the count, and the second byte (MSB) will start the counter with the new count. The order of programming is important for each counter, but programming of different counters may be interleaved for better control. For example, the control word may be sent to each counter before the counts for individual programming. Example 10–25 shows a few ways to program counter 1 and 2. The first method programs both control words, then the LSB of the count for each counter, which stops them from counting. Finally, the MSB portion of the count is programmed starting both counters with the new count. The second example shows one counter programmed before the other.

EXAMPLE 10–25

```
PROGRAM CONTROL WORD 1          ;setup counter 1
PROGRAM CONTROL WORD 2          ;setup counter 2
PROGRAM LSB 1                   ;stop counter 1 and program LSB
PROGRAM LSB 2                   ;stop counter 2 and program LSB
PROGRAM MSB 1                   ;program MSB of counter 1 and start it
PROGRAM MSB 2                   ;program MSB of counter 2 and start it

      or

PROGRAM CONTROL WORD 1          ;setup counter 1
PROGRAM LSB 1                   ;stop counter 1 and program LSB
PROGRAM MSB 1                   ;program MSB of counter 1 and start it
PROGRAM CONTROL WORD 2          ;setup counter 2
PROGRAM LSB 2                   ;stop counter 2 and program LSB
PROGRAM MSB 2                   ;program MSB of counter 2 and start it
```

Modes of Operation. Six modes (mode 0–mode 5) of operation are available to each of the 8254 counters. Figure 10–40 shows how each of these modes functions with the CLK input, the gate (G) control signal, and OUT signal. A description of each mode follows:

Mode 0 Allows the 8254 counter to be used as an events counter. In this mode, the output becomes a logic 0 when the control word is written and remains there until N plus the number of programmed counts. For example, if a count of 5 is programmed, the output will remain a logic 0 for 6 counts beginning with N. Note that the gate (G) input must be a logic 1 to allow the counter to count. If G becomes a logic 0 in the middle of the count, the counter will stop until G again becomes a logic 1.

Mode 1 Causes the counter to function as a retriggerable monostable multivibrator (one-shot). In this mode the G input triggers the counter so that it develops a pulse at the OUT connection that becomes a logic 0 for the duration of the count. If the count is 10, then the OUT connection goes low for 10 clocking periods when triggered. If the G input occurs within the duration of the output pulse, the counter is again reloaded with the count and the OUT connection continues for the total length of the count.

Mode 2 Allows the counter to generate a series of continuous pulses that are one clock pulse wide. The separation between pulses is determined by the count. For example, for a

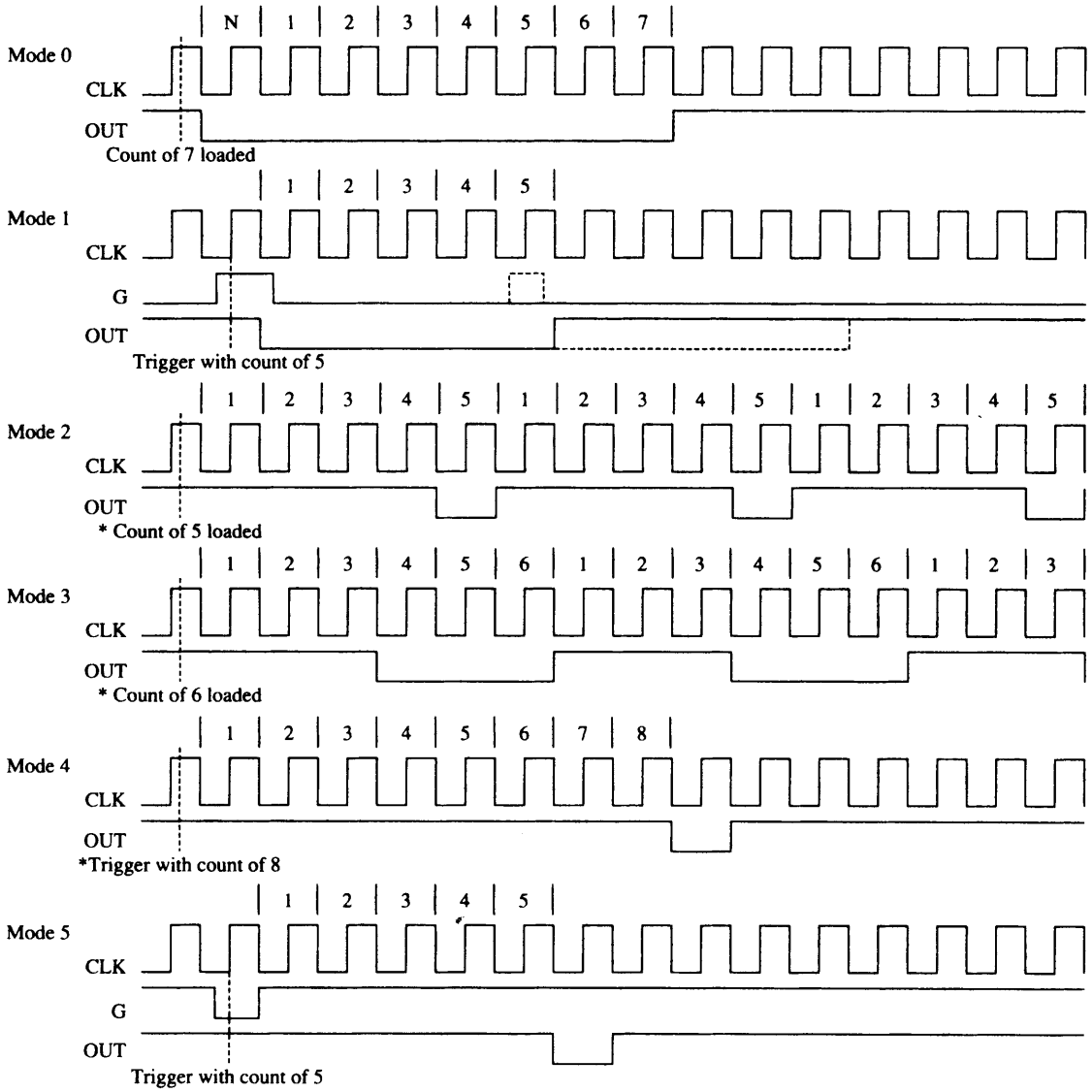


FIGURE 10–40 The six modes of operation for the 8254-2 programmable interval timer. **Note:* The G input stops the count when 0 in modes 2, 3, and 4.

count of 10, the output is a logic 1 for nine clock periods and low for one clock period. This cycle is repeated until the counter is programmed with a new count or until the G pin is placed at a logic 0 level. The G input must be a logic 1 for this mode to generate a continuous series of pulses.

Mode 3 Generates a continuous square-wave at the OUT connection, provided that the G pin is a logic 1. If the count is even, the output is high for one-half of the count and low for one-half of the count. If the count is odd, the output is high for one clocking period longer than it is low. For example, if the counter is programmed for a count of 5, the output is high for three clocks and low for two clocks.

Mode 4 Allows the counter to produce a single pulse at the output. If the count is programmed as a 10, the output is high for 10 clocking periods and low for one clocking period. The cycle does not begin until the counter is loaded with its complete count. This mode operates as a software triggered one-shot. As with modes 2 and 3, this mode also uses the G input to enable the counter. The G input must be a logic 1 for the counter to operate for these three modes.

Mode 5 A hardware triggered one-shot that functions as mode 4, except that it is started by a trigger pulse on the G pin instead of by software. This mode is also similar to mode 1 because it is retriggerable.

Generating a Wave-form with the 8254. Figure 10–41 shows an 8254 connected to function at I/O ports 0700H, 0702H, 0704H, and 0706H of an 80386SX microprocessor. The addresses are decoded by using a PAL16L8 that also generates a write strobe signal for the 8254, which is connected to the low order data bus connections. The PAL also generates a wait signal for the microprocessor that causes two wait states when the 8254 is accessed. The wait state generator connected to the microprocessor actually controls the number of wait states inserted into the timing. The program for the PAL is not illustrated here because it is the same as many of the prior examples.

Example 10–26 lists the program that generates a 100 KHz square-wave at OUT0 and a 200 KHz continuous pulse at OUT1. We use mode 3 for counter 0 and mode 2 for counter 1. The count programmed into counter 0 is 80 and the count for counter 1 is 40. These counts generate the desired output frequencies with an 8 MHz input clock.

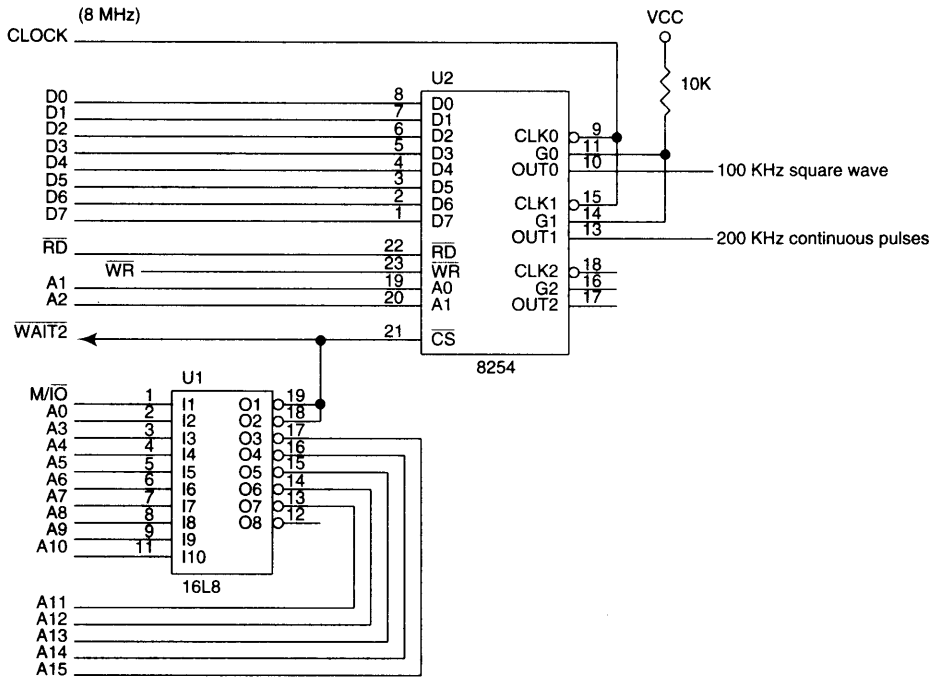


FIGURE 10–41 The 8254 interfaced to an 8 MHz 8086 so that it generates a 100 KHz square wave at OUT0 and a 200 KHz continuous pulse at OUT1.

EXAMPLE 10–26

```
                    ;A procedure that programs the 8254 timer to function
                    ;as illustrated in Figure 10-41.
                    ;
0000                TIME  PROC  NEAR

0000  50                  PUSH  AX              ;save registers
0001  52                  PUSH  DX

0002  BA 0706             MOV   DX,706H         ;address control word
0005  B0 36               MOV   AL,00110110B    ;program counter 0
0007  EE                  OUT   DX,AL           ;for mode 3

0008  B0 74               MOV   AL,01110100B    ;program counter 1
000A  EE                  OUT   DX,AL           ;for mode 2
000B  BA 0700             MOV   DX,700H         ;address counter 0
000E  B0 50               MOV   AL,80           ;load count of 80
0010  EE                  OUT   DX,AL
0011  32 C0              XOR   AL,AL
0013  EE                  OUT   DX,AL

0014 BA 0702              MOV   DX,702H         ;address counter 1
0017   B0 28              MOV   AL,40           ;load count of 40
0019  EE                  OUT   DX,AL
001A  32 C0               XOR   AL,AL
001C  EE                  OUT   DX,AL

001D  5A                  POP   DX              ;restore registers
001E  58                  POP   AX
001F  C3                  RET

0020                TIME  ENDP
```

Reading a Counter. Each counter has an internal latch that is read with the read counter port operation. These latches will normally follow the count. If the contents of the counter are needed, then the latch can remember the count by programming the counter latch control word (see Figure 10–42), which causes the contents of the counter to be held in a latch until it is read. Whenever a read from the latch or the counter is programmed, the latch tracks the contents of the counter.

When it is necessary for the contents of more than one counter to be read at the same time, we use the read-back control word, illustrated in Figure 10–43. With the read-back control word, the CNT bit is a logic 0 to cause the counters selected by CNT0, CNT1, and CNT2 to be latched. If the status register is to be latched, then the ST

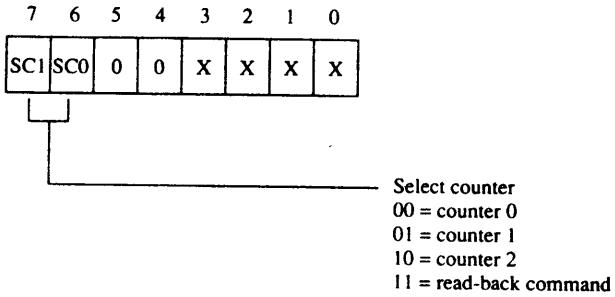


FIGURE 10–42 The 8254-2 counter latch control word.

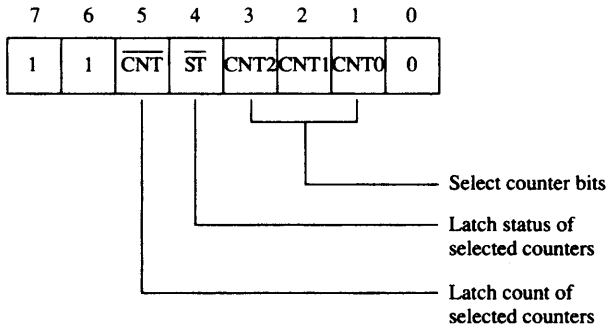


FIGURE 10–43 The 8254-2 read-back control word.

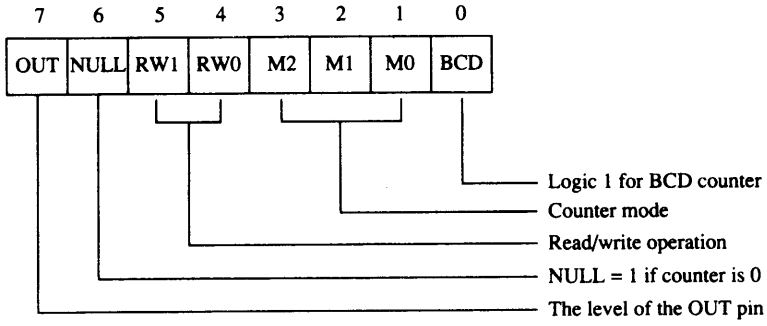


FIGURE 10–44 The 8254-2 status register.

bit is placed at a logic 0. Figure 10–44 shows the status register, which shows the state of the output pin, whether the counter is at its null state (0), and how the counter is programmed.

DC Motor Speed and Direction Control

One application of the 8254 timer is as a motor speed controller for a DC motor. Figure 10–45 shows the schematic diagram of the motor and its associated driver circuitry. It also illustrates the interconnection of the 8254, a flip-flop, and the motor and its driver.

The operation of the motor driver circuitry is straightforward. If the Q output of the 74ALS112 is a logic 1, the base Q2 is pulled up to +12 V through the base pull-up resistor, and the base of Q2 is open circuited. This means that Q1 is off and Q2 is on, with ground applied to the positive lead of the motor. The bases of both Q3 and Q4 are pulled low to ground through the inverters. This causes Q3 to conduction or turn on and Q4 to turn off, applying ground to the negative lead of the motor. The logic 1 at the Q output of the flip-flop therefore connects +12 V to the positive lead of the motor and ground to the negative lead. This connection causes the motor to spin in its forward direction. If the state of the Q output of the flip-flop becomes a logic 0, then the conditions of the transistors are reversed and +12 V is attached to the negative lead of the motor, with ground attached to the positive lead. This causes the motor to spin in the reverse direction.

If the output of the flip-flop is alternated between a logic 1 and 0, the motor spins in either direction at various speeds. If the duty cycle of the Q output is 50 percent, the motor will not spin at all and exhibits some holding torque because current flows through it. Figure 10–46 shows some timing diagrams and their effects on the speed and direction of the motor. Notice how each counter generates pulses at different positions to vary the duty cycle at the Q output of the flip-flop. This output is also called *pulse width modulation.*

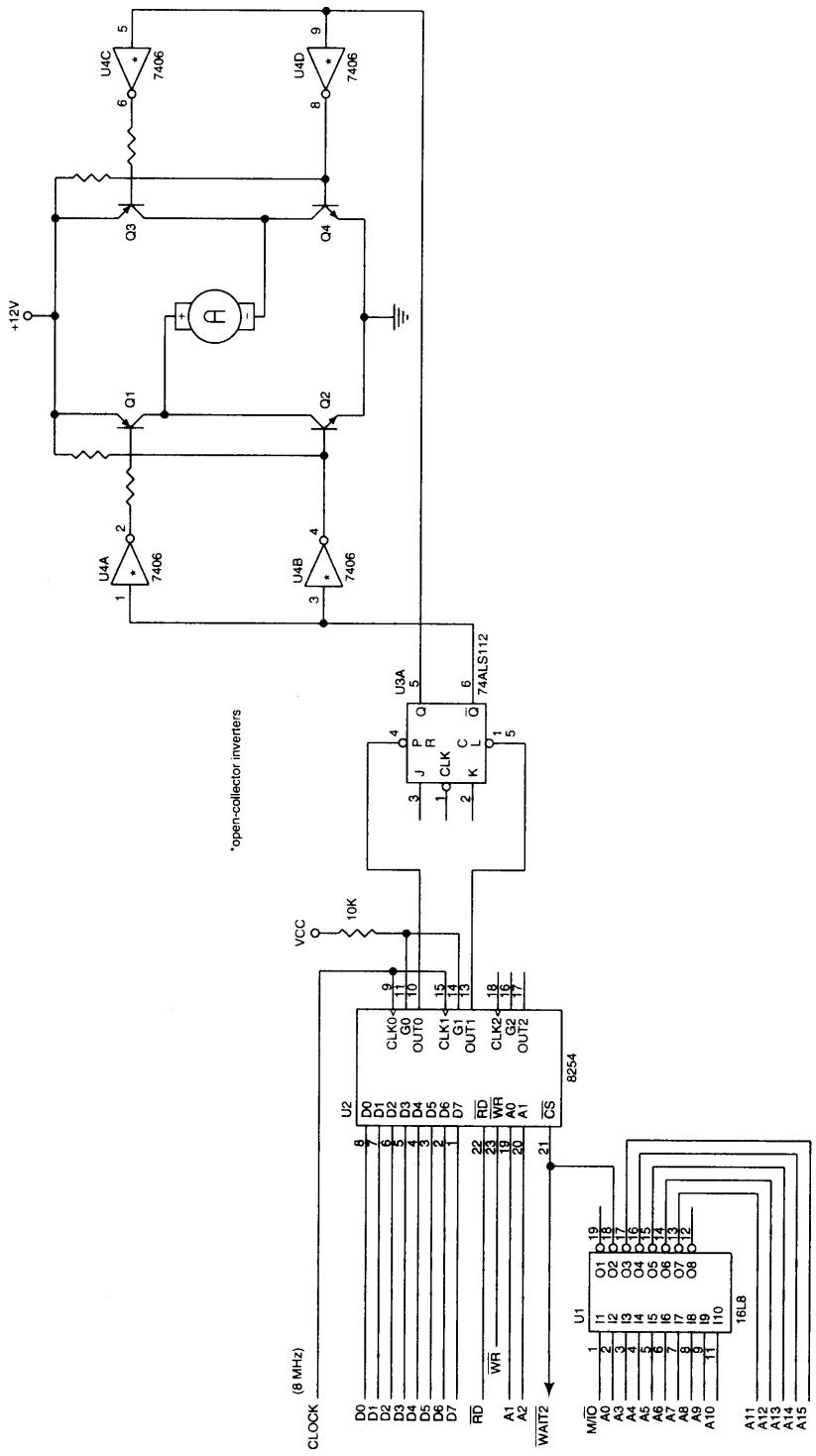


FIGURE 10–45 Motor speed and direction control using the 8254 timer.

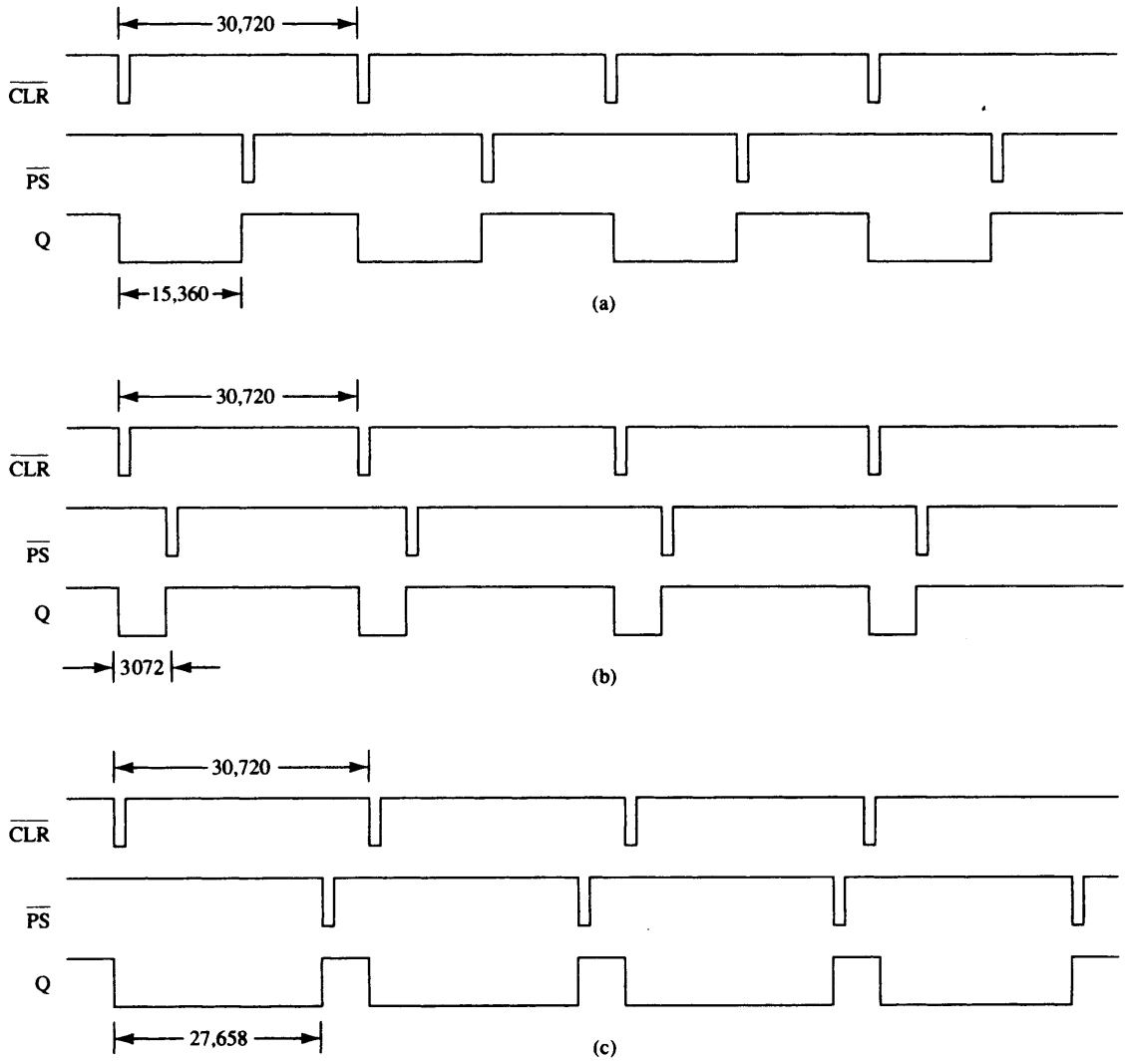


FIGURE 10–46 Timing for the motor speed and direction control circuit of Figure 10–45. (a) No rotation, (b) high-speed rotation in the reverse direction, and (c) high-speed rotation in the forward direction.

To generate these wave forms, counters 0 and 1 are both programmed to divide the input clock (PCLK) by 30,720. We change the duty cycle of Q by changing the point at which counter 1 is started in relationship to counter 0. This changes the direction and speed of the motor. But why divide the 8 MHz clock by 30,720? The divide rate of 30,720 is divisible by 256, so we can develop a short program that allows 256 different speeds. This also produces a basic operating frequency for the motor of about 260 Hz, which is low enough in frequency to power the motor. It is important to keep this operating frequency below 1000 Hz, but above 60 Hz.

Example 10–27 lists a procedure that controls the speed and direction of the motor. The speed is controlled by the value of AH when this procedure is called. Because we have an 8-bit number to represent speed, a 50 percent duty cycle, for a stopped motor, is a count of 128. By changing the value in AH when the procedure is called, we can adjust the motor speed. The speed of the motor will increase in either direction by changing the number in

AH when this procedure is called. As the value in AH approaches 00H, the motor begins to increase its speed in the reverse direction. As the value of AH approaches FFH, the motor increases its speed in the forward direction.

EXAMPLE 10–27

```
                        ;A procedure that controls the speed and direction
                        ;of the motor in Figure 10-45.
                        ;
                        ;When this procedure is called, the contents of
                        ;AH determine the speed and direction of the
                        ;motor where AH is between 00H and FFH.
   ;
= 0706                  CNTR  EQU   706H
= 0700                  CNT0  EQU   700H
= 0702                  CNT0  EQU   702H
= 7800                  COUNT EQU   30720

0000                    SPEED PROC  NEAR

0000  50                      PUSH  AX            ;save registers
0001  51                      PUSH  DX
0002  53                      PUSH  BX

0003  8A CD                  MOV   BL,AL         ;calculate count
0005  B8 0078                MOV   AX,120
0008  F6 E3                  MUL   BL
000A  8B D8                  MOV   BX,AX
000C  B8 7800                MOV   AX,COUNT
000F  2B C3                  SUB   AX,BX
0011  8B D8                  MOV   BX,AX

0013  BA 0706                MOV   DX,CNTR       ;program control words
0016  B0 34                  MOV   AL,00110100B
0018  EE                     OUT   DX,AL
0019  B0 74                  MOV   AL,01110100B
001B  EE                     OUT   DX,AL

001C  BA 0702                MOV   DX,CNT1       ;program counter 1
001F  B8 7800                MOV   AX,COUNT      ;to generate a clear
0022  EE                     OUT   DX,AL
0023  8A C4                  MOV   AL,AH
0025  EE                     OUT   DX,AL
0026                    SPE:
0026  EC                     IN    AL,DX         ;wait for counter 1
0027  86 C4                  XCHG  AL,AH         ;to reach calculated
0029  EC                     IN    AL,DX         ;count
002A  86 C4                  XCHG  AL,AH
002C  3B C3                  CMP   AX,BX
002E  72 F6                  JB    SPE

0030  BA 0700                MOV   DX,CNT0       ;program counter 0
0033  B8 7800                MOV   AX,COUNT      ;to generate a set
0036  EE                     OUT   DX,AL
0037  8A C4                  MOV   AL,AH
0039  EE                     OUT   DX,AL

003A  5B                      POP   BX            ;restore registers
003B  5A                      POP   DX
003C  58                      POP   AX
003D  C3                      RET

003E                    SPEED ENDP
```

The procedure adjusts the wave form at Q by first calculating the count that counter 0 is to start in relationship to counter 1. This is accomplished by multiplying AH by 120 and then subtracting it from 30,720. This is required because the counters are down-counters that count from the programmed count to 0 before restarting. Next, counter 1 is programmed with a count of 30,720 and started to generate the clear wave form for the flip-flop. After counter 1 is started, it is read and compared with the calculated count. Once it reaches this count, counter 0 is started with a count of 30,720. From this point forward, both counters continue generating the clear and set wave forms until the procedure is again called to adjust the speed and direction of the motor.

10–6 16550 PROGRAMMABLE COMMUNICATIONS INTERFACE

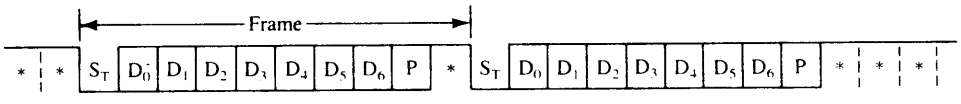
The National Semiconductor Corporation's PC16550D is a programmable communications interface designed to connect to virtually any type of serial interface. The 16550 is a universal asynchronous receiver/transmitter (UART) that is fully compatible with the Intel microprocessors. The 16550 is capable of operating at 0–1.5 M Baud. Baud rate is the number of bits transferred per second, including start, stop, data, and parity. The 16550 also includes a programmable Baud rate generator and separate FIFOs for input and output data to ease the load on the microprocessor. Each FIFO contains 16 bytes of storage. This is the most common communications interface found in modern microprocessor-based equipment, including the personal computer and many modems.

Asynchronous Serial Data

Asynchronous serial data are transmitted and received without a clock or timing signal. Figure 10–47 illustrates two frames of asynchronous serial data. Each frame contains a start bit, seven data bits, parity, and one stop bit. The figure shows a frame that contains one ASCII character and 10 bits. Most dial-up communications systems, such as CompuServe, Prodigy, and America Online, use 10 bits for asynchronous serial data with even parity. Most Internet and bulletin board services also use 10 bits, but they normally do not use parity. Instead, eight data bits are transferred, replacing parity with a data bit. This makes byte transfers of non-ASCII data much easier to accomplish.

16550 Functional Description

Figure 10–48 illustrates the pin-out of the 16550 UART. This device is available as a 40-pin DIP **(dual in-line package)** or as a 44-pin PLCC **(plastic lead-less chip carrier).** Two completely separate sections are responsible for data communications: the receiver and the transmitter. Because each of these sections is independent of each other, the 16550 is able to function in simplex, half-duplex, or full-duplex modes. One of the main features of the 16550 is its internal receiver and transmitter FIFO (first-in, first-out) memories. Because each is 16 bytes deep, the UART requires attention only from the microprocessor after receiving 16 bytes of data. It also holds 16 bytes before the microprocessor must wait for the transmitter. The FIFO makes this UART ideal when interfacing to high-speed systems because less time is required to service it.



* = STOP
ST = START

FIGURE 10–47 Asynchronous serial data.

An example **simplex** system is one in which the transmitter or receiver is used by itself such as in an FM **(frequency modulation)** radio station. An example **half-duplex** system is a CB **(citizens band)** radio, where we transmit and receive, but not both at the same time. The **full-duplex** system allows transmission and reception in both directions simultaneously. An example full-duplex system is the telephone.

The 16550 can control a **modem** (modulator/demodulator), which is a device that converts TTL levels of serial data into audio tones that can pass through the telephone system. Six pins on the 16650 are devoted to modem control: DSR (data set ready), DTR (data terminal ready), CTS (clear-to-send), RTS (request-to-send), RI (ring indicator), and DCD (data carrier detect). The modem is referred to as the **data set** and the 16550 is referred to as the **data terminal.**

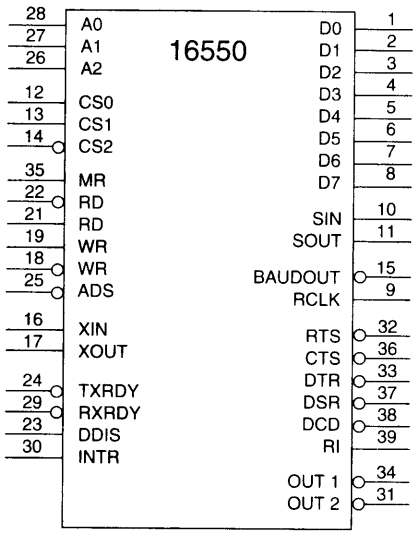


FIGURE 10–48 The pin-out of the 16550 UART.

16550 Pin Functions

A0, A1, A2 The address inputs are used to select an internal register for programming and also data transfer. See Table 10–8 for a list of each combination of the address inputs and the registers selected.

ADS The **address strobe** input is used to latch the address lines and chip select lines. If not needed (as in the Intel system), connect this pin to ground. The ADS pin is designed for use with Motorola microprocessors.

BAUDOUT The **Baud out** pin is where the clock signal generated by the Baud rate generator from the transmitter section is made available. It is most often connected to the RCLK input to generate a receiver clock that is equal to the transmitter clock.

CS0, CS1, CS2 The **chip select** inputs must all be active to enable the 16550 UART.

CTS The **clear to send** (if low) indicates that the modem or data set is ready to exchange information. This pin is often used in a half-duplex system to turn the line around.

D7–D0 The **data bus** pins are connected to the microprocessor data bus.

TABLE 10–8 The registers selected by A0, A1, and A2.

A2	*A1*	*A0*	*Register*
0	0	0	Receiver buffer (read) and transmitter holding (write)
0	0	1	Interrupt enable
0	1	0	Interrupt identification (read) and FIFO control (write)
0	1	1	Line control
1	0	0	Modem control
1	0	1	Line status
1	1	0	Modem status
1	1	1	Scratch

DCD	The **data carrier detect** input is used by the modem to signal the 16550 that a carrier is present.
DDIS	The **disable driver** output becomes a logic 0 to indicate that the microprocessor is reading data from the UART. DDIS can be used to change the direction of data flow through a buffer.
DSR	**Data set ready** is an input to the 16550, indicating that the modem or data set is ready to operate.
DTR	**Data terminal ready** is an output that indicates that the data terminal (16550) is ready to function.
INTR	**Interrupt request** is an output to the microprocessor used to request an interrupt (INTR = 1) whenever the 16550 has a receiver error, it has received data, and if the transmitter is empty.
MR	**Master reset** initializes the 16550 and should be connected to the system RESET signal.
OUT1, OUT2	User-defined output pins that can provide signals to a modem or any other device, as needed in a system.
RCLK	**Receiver clock** is the clock input to the receiver section of the UART. This input is always 16 × the desired receiver Baud rate.
RD, RD	**Read** inputs (either may be used) cause data to be read from the register specified by the address inputs to the UART.
RI	The **ring indicator** input is placed at the logic 0 level by the modem to indicate that the telephone is ringing.
RTS	**Request-to-send** is a signal to the modem, indicating that the UART wishes to send data.
SIN, SOUT	These are the serial data pins. SIN accepts serial data and SOUT transmits serial data.
RXRDY	**Receiver ready** is a signal used to transfer received data via DMA techniques (see text).
TXRDY	**Transmitter ready** is a signal used to transfer transmitter data via DMA techniques (see text).
WR, WR	**Write** (either may be used) connects to the microprocessor write signal to transfer commands and data to the 16550.
XIN, XOUT	These are the **main clock connections.** A crystal is connected across these pins to form a crystal oscillator, or XIN is connected to an external timing source.

Programming the 16550

Programming the 16550 is simple, although may be slightly more involved when compared to some of the other programmable interfaces described in this chapter. Programming is a two-part process that includes the initialization dialog and operational dialog.

In the personal computer, which uses the 16550 or its programming equivalent, the I/O port addresses are decoded at 3F8H through 3FFH for COM port 0 and 2F8H through 2FFH for COM port 2. Although the examples in this section of the chapter are not written specifically for the personal computer, they can be adapted by changing the port numbers to control the COM ports on the PC.

Initializing the 16550. Initialization dialog, which occurs after a hardware or software reset, consists of two parts: programming the line control register and the Baud rate generator. The line control register selects the number of data bits, number of stop bits, and parity (whether it's even or odd, or if parity is sent as a 1 or a 0). The Baud rate generator is programmed with a divisor that determines the Baud rate of the transmitter section.

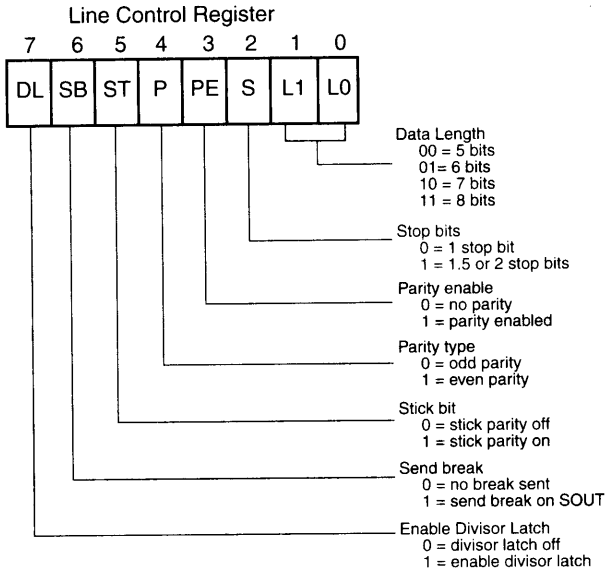


FIGURE 10–49 The contents of the 16550 line control register.

Figure 10–49 illustrates the line control register. The line control register is programmed by outputting information to I/O port 011 (A2, A1, A0). The rightmost two bits of the line control register select the number of transmitted data bits (5, 6, 7, or 8). The number of stop bits is selected by S in the line control register. If S = 0, one stop bit is used; if S = 1, 1.5 stop bits are used for five data bits, and two stop bits are used with six, seven, or eight data bits.

The next three bits are used together to send even or odd parity, to send no parity, or to send a 1 or a 0 in the parity bit position. To send even or odd parity, the ST **(stick)** bit must be placed at a logic 0 level, and parity enable must be a logic 1. The value of the parity bit then determines even or odd parity. To send no parity (common in Internet connections), ST = 0 as well as the parity enable bit. This sends and receives data without parity. Finally, if a 1 or a 0 must be sent and received in the parity bit position for all data, ST = 1 with a 1 in parity enable. To send a 1 in the parity bit position, place a 0 in the parity bit; to send a 0, place a 1 in the parity bit. (See Table 10–9 for the operation of the parity and stick bits.)

TABLE 10–9 The operation of the ST and parity bits.

ST	*P*	*PE*	*Function*
0	0	0	No parity
0	0	1	Odd parity
0	1	0	No parity
0	1	1	Even parity
1	0	0	Undefined
1	0	1	Send/receive 1
1	1	0	Undefined
1	1	1	Send/receive 0

The remaining bits in the line control register are used to send a break and to select programming for the Baud rate divisor. If bit position 6 of the line control register is a logic 1, a break is transmitted. As long as this bit is a 1, the break is sent from the SOUT pin. A break, by definition, is at least two frames of logic 0 data. The software in the system is responsible for timing the transmission of the break. To end the break, bit position 6 or the line control register is returned to a logic 0 level. The Baud rate divisor is only programmable when bit position 7 of the line control register is a logic 1.

Programming the Baud Rate. The Baud rate generator is programmed at I/O addresses 000 and 001 (A2, A1, A0). Port 000 is used to hold the least-significant part of the 16-bit divisor, and port 001 is used to hold the most-

significant part. The value used for the divisor depends on the external clock or crystal frequency. Table 10–10 illustrates common Baud rates obtainable if a 18.432 MHz crystal is used as a timing source. It also shows the divisor values programmed into the Baud rate generator to obtain these Baud rates. The actual number programmed into the Baud rate generator causes it to produce a clock that is 16 times the desired Baud rate. For example, if 240 is programmed into the Baud rate divisor, the Baud rate is $^{18.432\ MHz}/_{16\times240}$ = 4800 Baud.

TABLE 10–10 The divisor used with the Baud rate generator for an 18.432 MHz crystal illustrating common Baud rates.

Baud rate	*Divisor value*
110	10,473
300	3840
1200	920
2400	480
4800	240
9600	120
19,200	60
38,400	30
57,600	20
115,200	10

Sample Initialization. Suppose that an asynchronous system requires seven data bits, odd parity, a Baud rate of 9600, and one stop bit. Example 10–28 lists a procedure that initializes the 16550 to function in this manner. Figure 10–50 shows the interface to the 8088 microprocessor, using a PAL16L8 to decode the 8-bit port addresses F0H and F7H. (The PAL program is not shown.) Here, port F3H accesses the line control register and F0H and F1H access the Baud rate divisor registers. The last part of Example 10–28 is described with the function of the FIFO control register in the next few paragraphs.

EXAMPLE 10–28

```
                    ;Initialization Dialog for Figure 10-50.
                    ;Baud rate 9600, 7 data, odd parity, one stop
                    ;
= 00F3              LINE   EQU   0F3H
= 00F0              LSB    EQU   0F0H
= 00F1              MSB    EQU   0F1H
= 00F2              FIFO   EQU   0F2H

0000                START  PROC  NEAR

0000  B0 8A               MOV   AL,10001010B     ;enable Baud divisor
0002  E6 F3               OUT   LINE,AL

0004  B0 78               MOV   AL,120           ;program Baud rate
0006  E6 F0               OUT   LSB,AL
0008  B0 00               MOV   AL,0
000A  E6 F1               OUT   MSB,AL

000C  B0 0A               MOV   AL,00001010B     ;program 7-data, odd
000E  E6 F3               OUT   LINE,AL          ;parity, one stop

0010  B0 07               MOV   AL,00000111B     ;enable transmitter and
0012  E6 F2               OUT   FIFO,AL          ;receiver

0014  C3                  RET

0015                START  ENDP
```

After the line control register and Baud rate divisor are programmed into the 16550, it is still not ready to function. After programming the line control register and Baud rate, we still must program the FIFO control register, which is at port F2H in the circuit of Figure 10–50. Figure 10–51 illustrates the FIFO control register for the 16550. This register enables the transmitter and receiver (bit 0 = 1), and clears the transmitter and receiver FIFOs. It also provides control for the 16550 interrupts, which are discussed in Chapter 12. Notice that the last section of Example 10–28 places a 7 into the FIFO control register. This enables the transmitter and receiver, and clears both FIFOs. The 16550 is now ready to operate, but without interrupts. Interrupts are automatically disabled when the MR (master reset) input is placed at a logic 1 by the system RESET signal.

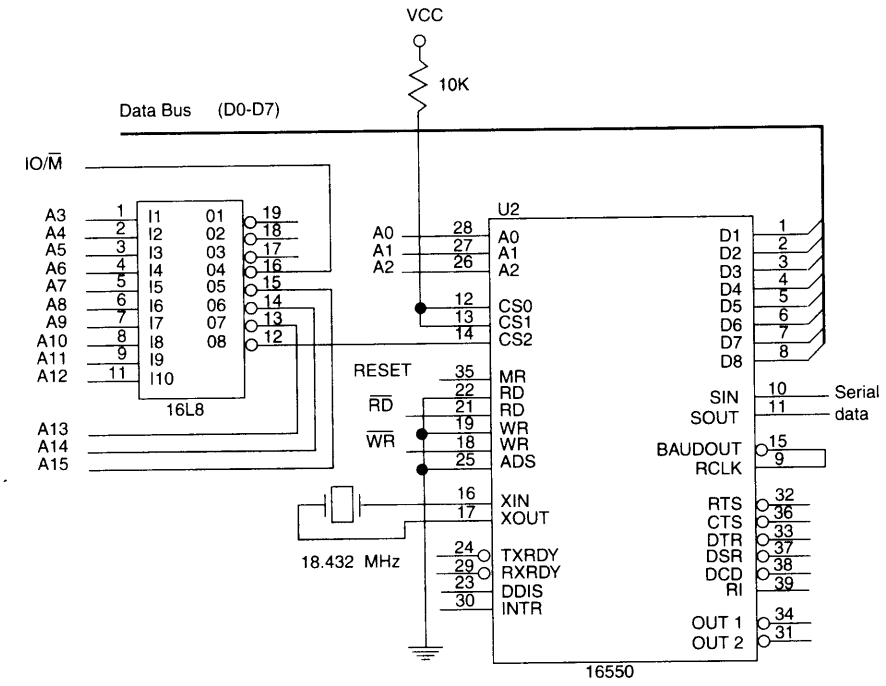


FIGURE 10–50 The 16550 interfaced to the 8088 microprocessor at ports 00F0H–00F7H.

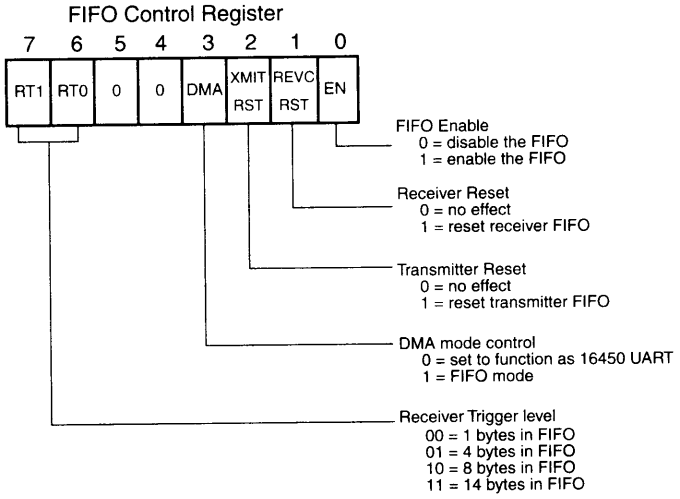


FIGURE 10–51 The FIFO control register of the 16550 UART.

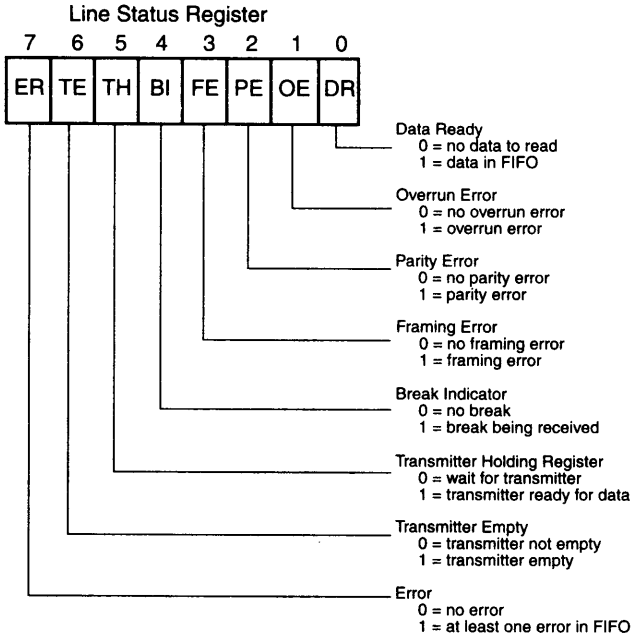


FIGURE 10–52 The contents of the line status register of the 16550 UART.

Sending Serial Data. Before serial data can be sent or received through the 16550, we need to know the function of the line status register (see Figure 10–52). The line status register contains information about error conditions and the state of the transmitter and receiver. This register is tested before a byte is transmitted or can be received.

Suppose that a procedure (see Example 10–29) is written to transmit the contents of AH to the 16550 and out through its serial data pin (SOUT). The TH bit is polled by software to determine whether the transmitter is ready to receive data. This procedure uses the circuit of Figure 10–50.

EXAMPLE 10–29

```
                        ;A Procedure that transmits AH via the 16550 UART
                        ;
= 00F5                  LSTAT EQU   0F5H         ;line status port
= 00F0                  DATA  EQU   0F0H         ;data port

0000                    SEND  PROC  NEAR

0000  50                      PUSH  AX           ;save AX
0001  E4 F5                   IN    AL,LSTAT     ;get line status register
0003  A8 20                   TEST  AL,20H       ;test TH bit
0005  74 FA                   JZ    SEND         ;if transmitter not ready

0007  8A C4                   MOV   AL,AH        ;get data
0009  E6 F0                   OUT   DATA,AL      ;transmit data
000B  58                      POP   AX           ;restore AX
000C  C3                      RET

000D                    SEND  ENDP
```

Receiving Serial Data. To read received information from the 16550, we test the DR bit of the line status register. Example 10–30 lists a procedure that tests the DR bit to decide whether the 16550 has received any data. Upon the reception of data, the procedure tests for errors. If an error is detected, the procedure returns with AL equal to an ASCII '?'. If no error has occurred, then the procedure returns with AL equal to the received character.

EXAMPLE 10–30

```
                        ;A procedure that receives data from the 16550 UART
                        ;and returns it in AL.
                        ;
= 00F5                  LSTAT EQU   0F5H          ;line status port
= 00F0                  DATA  EQU   0F0H          ;data port

0000                    RECV  PROC  NEAR

0000  E4 F5                   IN    AL,LSTAT      ;get line status register
0002  A8 01                   TEST  AL,1          ;test DR bit
0004  74 FA                   JZ    RECV          ;if no data in receiver

0006  A8 0E                   TEST  AL,0EH        ;test all 3 error bits
0008  75 03                   JNZ   ERR           ;for an error

000A  E4 F0                   IN    AL,DATA       ;read data from 16550
000C  C3                      RET
000D                    ERR:
000D  B0 3F                   MOV   AL,'?'        ;get question mark
000F  C3                      RET

0010                    REVC  ENDP
```

UART Errors. The types of errors detected by the 16550 are parity error, framing error, and overrun error. A **parity error** indicates that the received data contain the wrong parity. A **framing error** indicates that the start and stop bits are not in their proper places. An **overrun error** indicates that data have overrun the internal receiver FIFO buffer. These errors should not occur during normal operation. If a parity error occurs, it indicates that noise was encountered during reception. A framing error occurs if the receiver is receiving data at an incorrect Baud rate. An overrun error occurs only if the software fails to read the data from the UART before the receiver FIFO is full. This example does not test the BI (break indicator bit) for a break condition. Note that a break is two consecutive frames of logic 0s on the SIN pin of the UART. The remaining registers, which are used for interrupt control, are developed in Chapter 11.

10–7 ANALOG-TO-DIGITAL (ADC) AND DIGITAL-TO-ANALOG (DAC) CONVERTERS

Analog-to digital (ADC) and digital-to-analog (DAC) converters are used to interface the microprocessor to the analog world. Many events that are monitored and controlled by the microprocessor are analog events. These often include monitoring all forms of events, even speech, to controlling motors and like devices. In order to interface the microprocessor to these events, we must have an understanding of the interface and control of the ADC and DAC, which convert between analog and digital data.

The DAC0830 Digital-to-Analog Converter

A fairly common and low-cost digital-to-analog converter is the DAC0830 (a product of National Semiconductor Corporation). This device is an 8-bit converter that transforms an 8-bit binary number into an analog voltage. Other converters are available that convert from 10-, 12-, or 16-bit binary numbers into analog voltages. The number of voltage steps generated by the converter is equal to the number of binary input combinations. Therefore, an 8-bit converter

generates 256 different voltage levels, a 10-bit converter generates 1024 levels, and so forth. The DAC0830 is a medium speed converter that transforms a digital input to an analog output in approximately 1.0 μs.

Figure 10–53 illustrates the pin-out of the DAC0830. This device has a set of eight data bus connections for the application of the digital input code, and a pair of analog outputs labeled Iout1 and Iout2 that are designed as inputs to an external operational amplifier. Because this is an 8-bit converter, its output step voltage is defined as $-V_{REF}$ (reference voltage), divided by 255. For example, if the reference voltage is –5.0 V, its output step voltage is +.0196 V. Note that the output voltage is the opposite polarity of the reference voltage. If an input of 1001 0010_2 is applied to the device, the output voltage will be the step voltage times 1001 0010_2, or, in this case +2.862 V. By changing the reference voltage to –5.1 V, the step voltage becomes +.02 V. The step voltage is also often called the **resolution** of the converter

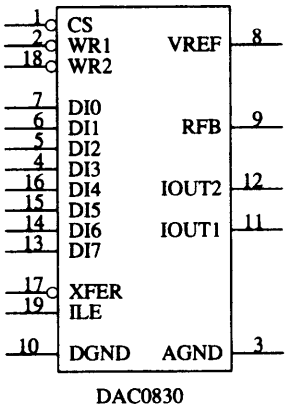


FIGURE 10–53 The pin-out of the DAC0830 digital-to-analog converter.

Internal Structure of the DAC0830. Figure 10–54 illustrates the internal structure of the DAC0830. Notice that this device contains two internal registers. The first is a holding register, while the second connects to the R–2R internal ladder converter. The two latches allow one byte to be held while another is converted. In many cases, we disable the first latch and only use the second for entering data into the converter. This is accomplished by connecting a logic 1 to ILE and a logic 0 to CS (**chip select**).

Both latches within the DAC0830 are transparent latches. That is, when the G input to the latch is a logic 1, data pass through the latch, but when the G input becomes a logic 0, data are latched or held. The converter has a reference input pin (V_{REF}) that establishes the full scale output voltage. If –10 V is placed on V_{REF}, the full scale (11111111_2) output voltage is +10 V. The output of the R–2R ladder within the converter appears at Iout1 and Iout2. These outputs are designed to be applied to an operational amplifier such as a 741 or similar device.

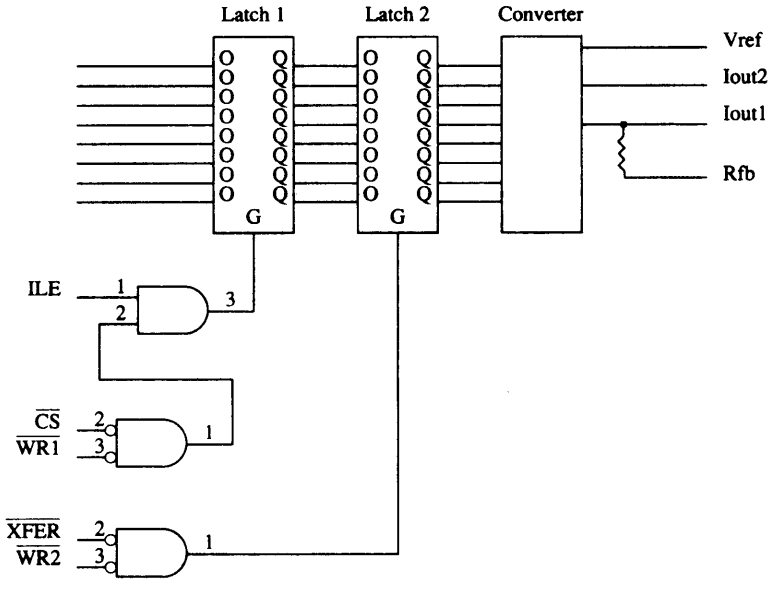


FIGURE 10–54 The internal structure of the DAC0830.

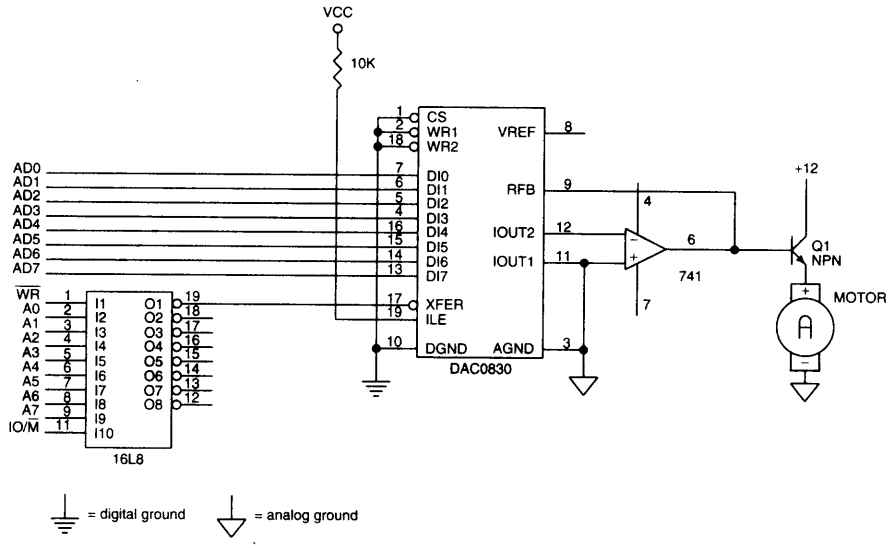


FIGURE 10–55 A DAC0830 interfaced to the 8086 microprocessor at 8-bit I/O location 20H.

Connecting the DAC0830 to the Microprocessor. The DAC0830 is connected to the microprocessor, as illustrated in Figure 10–55. Here, a PAL16L8 is used to decode the DAC0830 at 8-bit I/O port address 20H. Whenever an OUT 20H,AL instruction is executed, the contents of data bus connections AD0–AD7 are passed to the converter within the DAC0830. The 741 operational amplifier, along with the –12 V zener reference voltage, causes the full scale output voltage to equal +12 V. The output of the operational amplifier feeds a driver that powers a 12 V DC motor. This driver is a Darlington amplifier for large motors. This example shows the converter driving a motor, but other devices could be used as outputs.

The ADC080X Analog-To-Digital Converter

A common, low-cost ADC is the ADC0804, which belongs to a family of converters that are all identical, except for accuracy. This device is compatible with a wide range of microprocessors such as the Intel family. Although there are faster ADCs available and some have more resolution than eight bits, this device is ideal for many applications that do not require a high degree of accuracy. The ADC0804 requires up to 100 μs to convert an analog input voltage into a digital output code.

Figure 10–56 shows the pin-out of the ADC0804 converter (a product of National Semiconductor Corporation). To operate the converter, the WR pin is pulsed with CS grounded to start the conversion process. Because this converter requires a considerable amount of time for the conversion, a pin labeled INTR signals the end of the conversion. Refer to Figure 10–57 for a timing diagram that shows the interaction of the control signals. As can be seen, we start the converter with the WR pulse, we wait for INTR to return to a logic 0 level, and then we read the data from the converter. If a time delay is used that allows at least

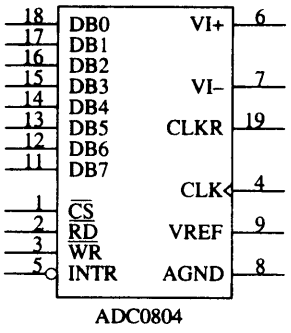


FIGURE 10–56 The pin-out of the ADC0804 analog-to-digital converter.

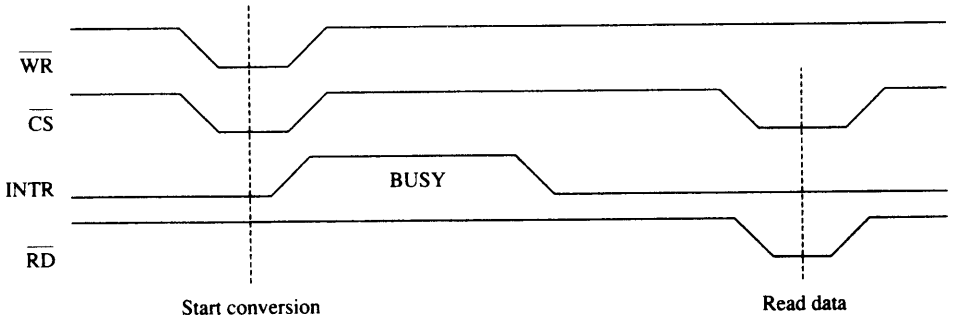


FIGURE 10–57 The timing for the ADC0804 analog-to-digital converter.

100 μs of time, then we don't need to test the INTR pin. Another option is to connect the INTR pin to an interrupt input, so that when the conversion is complete, an interrupt occurs.

The Analog Input Signal. Before the ADC0804 can be connected to the microprocessor, its analog inputs must be understood. There are two analog inputs to the ADC0804: VIN (+) and VIN (–). These inputs are connected to an internal operational amplifier and are differential inputs, as shown in Figure 10–58. The differential inputs are summed by the operational amplifier to produce a signal for the internal analog-to-digital converter. Figure 10–58 shows a few ways to use these differential inputs. The first way (see Figure 10–58a) uses a single input that can vary between 0 V and +5.0 V. The second way (see Figure 10–58b) shows a variable voltage applied to the VIN (–) pin, so the zero reference for VIN (+) can be adjusted.

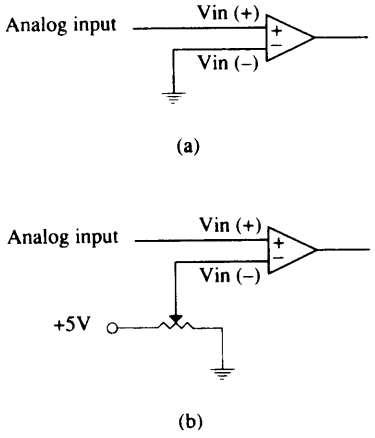


FIGURE 10–58 The analog inputs to the ADC0804 converter. (a) To sense a 0- to +5.0-V input. (b) To sense an input offset from ground.

Generating the Clock Signal. The ADC0804 requires a clock source for operation. The clock can be an external clock applied to the CLK IN pin or it can be generated with an RC circuit. The permissible range of clock frequencies is between 100 KHz and 1460 KHz. It is desirable to use a frequency that is as close as possible to 1460 KHz, so conversion time is kept to a minimum.

If the clock is generated with an RC circuit, we use the CLK IN and CLK R pins connected to an RC circuit, as illustrated in Figure 10–59. When this connection is in use, the clock frequency is calculated by the following equation:

$$Fclk = \frac{1}{1.1RC}$$

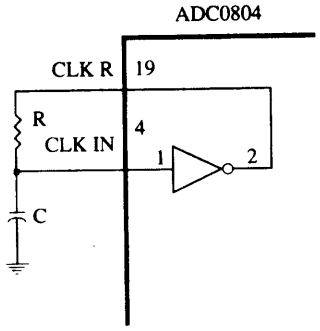


FIGURE 10–59 Connecting the RC circuit to the CLK IN and CLK R pins on the ADC0804.

Connecting the ADC0804 to the Microprocessor. The ADC0804 is interfaced to the 8086 microprocessor, as illustrated in Figure 10–60. Note that the V_{REF} signal is not attached to anything, which is normal. Suppose that the ADC0804 is decoded at 8-bit I/O port address 40H for the data and port address 42H for the INTR signal, and a procedure is required to start and read the data from the ADC. This procedure is listed in Example 10–31. Notice