Weeks 9-10
Input/Output Interface Circuits and LSI
Peripheral Devices
Core and Special Purpose I/O Interfaces

- **Special purpose I/O interfaces** are implemented as add-on cards on the PC:
  - display
  - parallel printer interface
  - serial communication interface
  - local area network interface
  - not all microcomputer systems employ each of these types

- **Core input/output interfaces** are considered to be the part of the I/O subsystem such as:
  - parallel I/O to read the settings of the DIP switches on the processor board
  - interval timers used in DRAM refresh process

- We will study both
Isolated I/O

- I/O devices are treated separately from memory
- Address 0000 to 00FF: referred to page 0. Special instructions exist for this address range
  - Byte wide ports
  - Word wide ports
Memory Mapped I/O

Diagram showing memory mapped I/O addresses and corresponding I/O ports.
Maximum Mode I/O interface - 8088/8086
### I/O Bus Cycle Status Codes

<table>
<thead>
<tr>
<th>Status inputs</th>
<th>CPU cycle</th>
<th>8288 command</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bar{s}_2 ) ( \bar{s}_1 ) ( \bar{s}_0 )</td>
<td>Interrupt acknowledge</td>
<td>( \text{INTA} )</td>
</tr>
<tr>
<td>0 0 0</td>
<td>Read I/O port</td>
<td>( \text{IORC} )</td>
</tr>
<tr>
<td>0 0 1</td>
<td>Write I/O port</td>
<td>( \text{IOWC, AIOWC} )</td>
</tr>
<tr>
<td>0 1 0</td>
<td>Halt</td>
<td>None</td>
</tr>
<tr>
<td>1 0 0</td>
<td>Instruction fetch</td>
<td>( \text{MRDC} )</td>
</tr>
<tr>
<td>1 0 1</td>
<td>Read memory</td>
<td>( \text{MRDC} )</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Write memory</td>
<td>( \text{MWTC, AMWC} )</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Passive</td>
<td>None</td>
</tr>
</tbody>
</table>
## I/O Instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Meaning</th>
<th>Format</th>
<th>Operation</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>Input direct</td>
<td>IN Acc,Port</td>
<td>(Acc) ← (Port)</td>
<td>Acc = AL or AX</td>
</tr>
<tr>
<td></td>
<td>Input indirect (variable)</td>
<td>IN Acc,DX</td>
<td>(Acc) ← ((DX))</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Output direct</td>
<td>OUT Port,Acc</td>
<td>(Port) ← (Acc)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Output indirect (variable)</td>
<td>OUT DX,Acc</td>
<td>((DX)) ← (Acc)</td>
<td></td>
</tr>
</tbody>
</table>

**Example.** Write a sequence of instructions that will output the data FFh to a byte wide output at address ABh of the I/O address space

```
MOV AL,0FFh
OUT 0ABh, AL
```

**Example.** Data is to be read from two byte wide input ports at addresses AAh and A9h and then this data will then be output to a word wide output port at address B000h

```
IN AL, 0AAh
MOV AH,AL
IN AL, 0A9h
MOV DX,0B00h
OUT DX,AX
```
Input Bus Cycle of the 8088

Figure 8-32  Input bus cycle of the 8088.
Output Bus Cycle of the 8088

Figure 8-53  Output bus cycle of the 8088.
I/O Example

• Assume that AX = 76A9h.: Analyze the data transfer for a) 8088 b) 8086 when MOV DX, 648h
  OUT DX, AX

• 8088 case
  – 1st bus cycle
    • T1: Address 0648h is put on pins AD0-AD7, A8-15 and latched when ALE is activated
    • T2: The low byte A9h is put on the data bus pins AD0-AD7 and IOWC is activated
    • T3: Setup time
    • T4: Byte is written to the port assuming zero wait states
  – 2nd Bus Cycle (Similar to 1st Bus Cycle)
    • T1: Address 0649h is put on pins AD0-AD7, A8-15 and latched when ALE is activated
    • T2: The high byte 76h is put on the data bus pins and IOWC is activated
    • T3: Setup time
    • T4: Byte is written to the port assuming zero wait states
Example continued

• 8086 case
  – T1: Address 0648h is put on pins AD0-AD15 plus BHE=low is latched by the 74LS373 when ALE is activated
  – T2: 76A9h, the contents of AX, is put on AD0-AD15 (A9h on AD0-AD7, 76h on AD8-AD15) and IOWC is activated
  – T3: Setup time
  – T4: During this interval, with the help of the signals A0=0 and BHE=0, the low and high bytes are written to the appropriate ports.

  ➢ It must be noted that since the operand is a 16 bit word and the port address is an even address, the 8086 CPU does not generate address 0649h
  ➢ Port address 648h is connected to the D0-D7 data bus and port address 649h is connected to the D8-D15 data bus
I/O Design in the 8088/86

• In every computer, when data is sent out by the CPU, the data on the data bus must be latched by the receiving device.
• While memories have an internal latch to grab the data on the data bus, a latching system must be designed for ports.
• Since the data provided by the CPU to the port is on the system data bus for a limited amount of time (50 - 1000ns) it must be latched before it is lost.
• Likewise, when data is coming in by way of a data bus (either from port or memory) it must come in through a three-state buffer.
I/O Design

- Design for OUT 9CH, AL
Example - 64 line parallel output circuit - 8088
Examples

• To which output port in the previous figure are data written when the address put on the bus during an output bus cycle is 8002h?
  – A15 .. A0 = 1000 0000 0000 0010b
  – A15L = 1
  – A0L = 0
  – A3L A2L A1L = 001
  – P1 = 0

• Write a sequence of instructions that output the byte contents of the memory address DATA to output port 0 in the previous figure

MOV DX, 8000h
MOV AL, DATA
OUT DX, AL
MOV DX, 8000h ; initialize address of port 0
MOV AL, 00h ; load data with bit 7 as logic 0
ON_OFF: OUT DX,AL ; turned on
MOV CX,0FFFFh ; load delay count of FFFFh
HERE: LOOP HERE
XOR AL,80h ; complement bit 7
JMP ON_OFF

Aprox. 17 T states * 64K * Frequency
IN port design using the 74LS244

- In order to prevent any unwanted data (garbage) to come into the system (global) data bus, all input devices must be isolated through the tri-state buffer. The 74LS244 not only plays this role but also provides the incoming signals sufficient strength (driving capability) to travel all the way to the CPU.
- It must be emphasized that every device (memory, peripheral) connected to the global data bus must have a latch or a tri-state buffer. In some devices such as memory, they are internal but must be present.
Read from input port 7 to the memory location DATA:
MOV DX, 800Eh
IN AL,DX
MOV DATA, AL

Example - 64 line parallel input circuit
Example

- In practical applications, it is sometimes necessary within an I/O service routine to repeatedly read the value at an input line and test this value for a specific logic level.

```assembly
74F244 Port 0

Poll the switch waiting for it to close
MOV DX,8000h
POLL: IN AL,DX
SHL AL,1
JC POLL
```
Input Output Handshaking

- The I/O ports of a computer typically operate at different data rates.
- A hard disk drive, for example, might require the computer to input data at 10Mbps → 100Mbps.
- CD-ROM drives operate at 300-600 Kbps.
- However, when inputting keystrokes from the operator, the data rate may fall to only one or two characters per sec.
- If the processor is to operate efficiently, one needs to develop a strategy to control or synchronize the flow of data between the processor and the widely varying rates of its I/O devices.
- This type of synchronization is achieved by implementing what is known as handshaking as part of the input/output interface.
- Printers typically have buffers that can be filled by the computer at high speed.
- Once full, the computer must wait while the data in the buffer is printed.
- Most printer manufacturers have settled on a standard set of data and control signals, Centronics Parallel Printer Interface.
**Parallel Printer Interface**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Strobe</td>
</tr>
<tr>
<td>2</td>
<td>Data 0</td>
</tr>
<tr>
<td>3</td>
<td>Data 1</td>
</tr>
<tr>
<td>4</td>
<td>Data 2</td>
</tr>
<tr>
<td>5</td>
<td>Data 3</td>
</tr>
<tr>
<td>6</td>
<td>Data 4</td>
</tr>
<tr>
<td>7</td>
<td>Data 5</td>
</tr>
<tr>
<td>8</td>
<td>Data 6</td>
</tr>
<tr>
<td>9</td>
<td>Data 7</td>
</tr>
<tr>
<td>10</td>
<td>Ack</td>
</tr>
<tr>
<td>11</td>
<td>Busy</td>
</tr>
<tr>
<td>12</td>
<td>Paper Empty</td>
</tr>
<tr>
<td>13</td>
<td>Select</td>
</tr>
<tr>
<td>14</td>
<td>Auto Foxt</td>
</tr>
<tr>
<td>15</td>
<td>Error</td>
</tr>
<tr>
<td>16</td>
<td>Initialize</td>
</tr>
<tr>
<td>17</td>
<td>Select</td>
</tr>
<tr>
<td>18</td>
<td>Ground</td>
</tr>
<tr>
<td>19</td>
<td>Ground</td>
</tr>
<tr>
<td>20</td>
<td>Ground</td>
</tr>
<tr>
<td>21</td>
<td>Ground</td>
</tr>
<tr>
<td>22</td>
<td>Ground</td>
</tr>
<tr>
<td>23</td>
<td>Ground</td>
</tr>
<tr>
<td>24</td>
<td>Ground</td>
</tr>
<tr>
<td>25</td>
<td>Ground</td>
</tr>
</tbody>
</table>

**Data:**  
Data0, Data1, ..........., Data7

**Control:**  
Strobe  
Auto Foxt  
Initialize  
Select  
Error

**Status:**  
Ack  
Busy  
Paper Empty  
Select  
Error

ACK is used by printer to acknowledge receipt of data and can accept a new character.

BUSY high if printer is not ready to accept a new character

SELECT when printer is turned on

ERROR goes low when there are conditions such as paper jam, out of paper, offline

STROBE when PC presents a character

INITIALIZE Clear Printer Buffer and reset control
Operational Principle - Parallel Printer Port

• The computer checks the BUSY signal from the printer, if not BUSY then
• When the PC presents a character to the data pins of the printer, it activates the STROBE pin, telling it that there is a byte sitting at the data pins. Prior to asserting STROBE pin, the data must be at at the printer’s data pins for at least 0.5 microsec. (data setup time)
• The STROBE must stay for 0.5 microsec

• The printer asserts BUSY pin indicating the computer to wait
• When the printer picks up the data, it sends back the ACK signal, keeps ACK low for 5 microsec.
• As the ACK signal is going high, the printer makes the BUSY pin low to indicate that it is ready to accept the next byte
• The CPU can use ACK or BUSY signals from the printer to initiate the process of sending another byte
Handshaking
Printer Interface Circuit
Example

- Write a program that implements the flowchart. Character data is held in memory starting at address PRNT_BUFF, the number of characters held in the buffer is identified by the count address CHAR_COUNT.

```
MOV CL, CHAR_COUNT
MOV SI, OFFSET PRNT_BUFF

POLL_BUSY:
    MOV DX,8004h
    IN AL,DX
    AND AL,01h
    JNZ POLL_BUSY
    BUSY input checked

    MOV AL, [SI]
    MOV DX,8000h
    OUT DX,AL
    Character is output

    MOV AL, 00h ;STB = 0
    MOV DX,8002h
    OUT DX,AL

    MOV BX,0Fh ; delay for STB = 0
    STROBE: DEC BX
    JNZ STROBE
    MOV AL,01h
    OUT DX,AL ; STB bar = 1

    INC SI
    DEC CL
    JNZ POLL_BUSY
```
IBM PC Printer Interfacing

• Base I/O port address of each printer is written into the BIOS data area 0040:0008 to 0040:000Fh
• 0040:0008 and 0040:0009 LPT1 (e.g., 0378h)
  – 0378h: data port
  – 0379h: status port read only
  – 037Ah: control port
• 0040:000A and 0040:000B LPT2
• BIOS INT 17h provides three services: printing a character, initializing the printer port and getting the printer status port
• AH = 0; print a character in AL and have the LPT number in DX (0 for LPT1, 1 for LPT2) also returns status
• AH = 01; initialize the printer port by setting the printer to the top of the page
• AH = 2; get the printer status after calling AH = status as:
  – bit 7: 1 ready 0 busy
  – bit 3: 1 I/O error
  – bit 5: 1 out of paper
Printer Time Out

Timeout occurs due to: printer offline / not able to print for any reason although it is properly connected.

BIOS repeatedly tries for a period of 20 seconds to see if the printer is ready to accept data.

It is possible to alter the entered by BIOS at boot time.

0040:0078 → 0040:007B holds the wait times.

```
MOV AH,0 ; print character option
MOV DX,0 ; select LPT1
MOV AL,41h ; ASCII code for letter A
INT 17h
```

<table>
<thead>
<tr>
<th>ASCII Table</th>
<th>Hex Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS</td>
<td>08</td>
<td>Backspace</td>
</tr>
<tr>
<td>HT</td>
<td>09</td>
<td>Horizontal Tab</td>
</tr>
<tr>
<td>LF</td>
<td>0A</td>
<td>Line Feed</td>
</tr>
<tr>
<td>VT</td>
<td>0B</td>
<td>Vertical Tab</td>
</tr>
<tr>
<td>FF</td>
<td>0C</td>
<td>Form Feed</td>
</tr>
<tr>
<td>CR</td>
<td>0D</td>
<td>Carriage Return</td>
</tr>
</tbody>
</table>
The 8255 Programmable Peripheral Interface

- Intel has developed several peripheral controller chips designed to support the 80x86 processor family. The intent is to provide a complete I/O interface in one chip.
- 8255 PPI provides three 8 bit input ports in one 40 pin package making it more economical than 74LS373 and 74LS244.
- The chip interfaces directly to the data bus of the processor, allowing its functions to be programmed; that is in one application a port may appear as an output, but in another, by reprogramming it as an input. This is in contrast with the 74LS373 and 74LS244 which are hardwired and fixed.
- Other peripheral controller chips include the 8259 Programmable Interrupt Controller (PIC), the 8253/54 Programmable Interval Timer (PIT) and the 8237 DMA controller.
8255A internal
8255 Pins

- PA0 - PA7: input, output, or bi-directional port
- PB0 - PB7: input or output
- PC0 - PC7: This 8 bit port can be all input or output. It can also be split into two parts, CU (PC4 - PC7) and CL (PC0 - PC3). Each can be used for input and output.
- RD or WR
  - IOR and IOW of the system are connected
- RESET
- A0, A1, and CS
  - CS selects the entire chip whereas A0 and A1 select the specific port (A, B, or C)

<table>
<thead>
<tr>
<th>CSBAR</th>
<th>A1</th>
<th>A0</th>
<th>SELECTS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Port A</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Port B</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Port C</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Control Register</td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>8255 not selected</td>
</tr>
</tbody>
</table>
Addressing an 8255

<table>
<thead>
<tr>
<th>To Select</th>
<th>A1A0CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port A</td>
<td>00</td>
</tr>
<tr>
<td>Port B</td>
<td>01</td>
</tr>
<tr>
<td>Port C</td>
<td>10</td>
</tr>
<tr>
<td>Control Reg.</td>
<td>11</td>
</tr>
</tbody>
</table>
8255 Control Word Format

FIGURE 15.4
8255A Control Word Format for I/O Mode
SOURCE: Adapted from Intel Corporation, *Peripheral Components* (Santa Clara, Calif.: Author, 1993), p. 3-
Mode 0 - Simple input/output

- Simple I/O mode: any of the ports A, B, CL, and CU can be programmed as input or output.
- Example: Configure port A as input, B as output, and all the bits of port C as output assuming a base address of 50h
- Control word should be 1001 0000b = 90h

```assembly
PORTA EQU 50h
PORTB EQU 51h
PORTC EQU 52h
CNTREG EQU 53h
MOV AL, 90h
OUT CNTREG, AL
IN AL, PORTA
OUT PORTB, AL
OUT PORTC, AL
```
Mod 0 Simple I/O

Initialize this device with the appropriate Control Word. Read from PORT C_L and display at PORT A.

PORT A 8000h
PORT B 8001H
PORT C 8002H
CONTROL 8003H

Be careful Memory I/O!

MOV AL,83H
MOV BX,8003H
MOV [BX],AL
MOV BX,8002H
MOV AL,[BX]
AND AL,0FH
DCX BX
DCX BX
MOV [BX],AL
BSR Mode

- Concerned with the eight bits of port C only which can be set or reset by writing appropriate control word with D7=1
- It does not alter the previously transmitted control word with D7=0

Ex: Write a BSR word subroutine to set PC7 and PC3
To Set PC7 → OFH ; To set PC3 → 07H

MOV AL,0FH
OUT 83H,AL
MOV AL,07H
OUT 83h,AL
Example 11.4 of Textbook

• Find the control word
  – PA = out
  – PB = in
  – PC0 – PC3 = in
  – PC4 – PC7 = out

• Program the 8255 to get data from port B and send it to port A; in addition data from PCL is sent out to the PCU

• Use port addresses 300h – 303h for the 8255 chip

Control Word:
The control word should be
1000 0011b = 83h
B8255 EQU 300h
CNTL EQU 83h

MOV DX,B8255+3
MOV AL,CNTL
OUT DX,AL
MOV DX,B8255+1
IN AL,DX
MOV DX,B8255
OUT DX,AL
MOV DX,B8255+2
IN AL,DX
AND AL,0Fh
MOV CL,4
ROL AL,CL
OUT DX,AL
Example 11-6 Textbook

• Assume 8255 has a base address 300h
• Write a program to toggle all bits of port A continuously with a \( \frac{1}{4} \) sec. Delay
• Use int 16h to exit if there is a key press

```
AGAIN:
    MOV DX,303h
    MOV AL,80h
    OUT DX,AL
    CALL QSDELAY
    MOV AL,0AAh
    OUT DX,AL
```
CALL QSDELAY
MOV AH,01
INT 16h
JZ AGAIN
MOV AH,4Ch
INT 21h

; to create a processor independent delay IBM made PB4 of port 61h to toggle every 15.085 microsec. (for 286 and higher processors)

QSDELAY PROC NEAR
MOV CX,16572 ;16572*15.085 microsec = ¼ s
PUSH AX
W1:        IN AL,61h
            AND AL,00010000b
            CMP AL,AHh
            JE  W1
            MOV AH,AL
            LOOP W1
POP AX
RET
QSDELAY ENDP
Mode 0 Design Example - Interfacing a matrix keyboard
Key Debounce

✓ When a mechanical push button key is pressed or released the metal contacts of the key momentarily bounce before giving a steady-state reading. Therefore it is necessary that the bouncing of the key not be read as an input.

✓ Key debounce can be eliminated using either software or hardware.
Program

; Function: Scan the keyboard shown in Fig. 8.14
; and return with the encoded key
; value in register AL.
; Inputs: none
; Outputs: hex key value in AL.
; Calls: 10 ms delay procedure for debouncing
; Destroys: AX and flags

;********
; Set up segment to store key values
;********

0000 00 01 02 03 04
0005 05 06 07
0008 08 09 0A 0B 0C
000D 0D 0E 0F
0010

KEY_CODE SEGMENT RVTR
COL1 DB 0,1,2,3,4
   DB 5,6,7
COL2 DB 8,9,0AH,0BH,0CH
   DB 0DH,0EH,0FH
KEY_CODE ENDS

CODE SEGMENT BYTE
ASSUME CS:CODE, DS:KEY_CODE

;********
; Program equates
;********

= 00F0 PORT_A EQU 00H          ; PPI port A address (see Fig. 8.12)
= 00F2 PORT_C EQU 08H          ; PPI port C address
= 00BF COL_1_LOW EQU 10111111B ; PC6 low
= 007F COL_2_LOW EQU 01111111B ; PC7 low
= 003F BOTH_COL_LOW EQU 00111111B ; PC6 and PC7 low
= 00FF KEY_UP EQU 0FFH         ; Input 0FFH when no keys are down
= 16FA T1 EQU 8B82H            ; ~ 10 ms time delay assuming 25 MHz 80
;********
; 10 ms time delay for debouncing
;********

DELAY PROC NEAR
  MOV CX, T1
COUNT:
  LOOP COUNT
  RET
DELAY ENDP

;********
; Main program begins here
;********

KEYBOARD PROC NEAR
  PUSH DS
  PUSH CX
  PUSH SI
  MOV AX, KEY_CODE
  MOV DS, AX

; Wait for previous key to be released

MOV AL, BOTH_COL_LOW
OUT PORT_C, AL
IN AL, PORT_A
CMP AL, KEY_UP
JNE POLL1
CALL DELAY

; Wait for a new key to be pressed

POLL1:
IN AL, PORT_A
CMP AL, KEY_UP
JE POLL2
CALL DELAY

; Save registers about to be used
; Point DS to the key codes
; Scan both columns
; Column lines on PC6 and PC7
; Read keyboard
; All keys up?
; No - so wait
; Yes - wait for bounce on release

; Read keyboard
; Any keys down?
; No - so wait
; Yes - wait for bounce
Program Cont.

;See if the key is in column 1

0024 B0 BF MOV AL, COL_1_LOW ;Test for column 1
0026 B6 08 OUT PORT_C, AL ;PC6 low
0028 E4 00 IN AL, PORT_A ;Read column 1 keys
002A 3C FF CMP AL, KEY_UP ;Any key down?
002C 74 07 JE CHECK_COL_2 ;No - check for column 2
002E 8D 36 0000 R LEA SI, COL1 ;Yes point SI at the key values 0-7
0032 EB 0F 90 JMP LOOKUP ;Now lookup code

If not column 1 then column 2

0035 B0 7F CHECK_COL_2: MOV AL, COL_2_LOW ;Test for column 2
0037 B6 08 OUT PORT_C, AL ;PC7 low
0039 E4 00 IN AL, PORT_A ;Read column 2 keys
003B 3C FF CMP AL, KEY_UP ;Any key down?
003D 74 1C JE PO1.0 ;No - false input so repeat
003F 8D 36 0008 R LEA SI, COL2 ;Yes - point SI at key values 8-F

;Now lookup the key's value and store in AL

0043 D0 D8 LOOKUP: RCH AL, 1 ;Rotate keyboard input code right
0045 73 03 JNC MATCH ;If 0 key is found - so retrieve it
0047 46 INC SI ;No advance pointer to next value
0049 EB F9 JMP LOOKUP ;Repeat the loop

004A 8A 04 MATCH: MOV AL, [SI] ;Get the key code
004C 5E POP SI ;Restore all registers
004D 59 POP CX (except AX and flags)
004E 1F POP DS
004F C3 RET

0050 KEYBOARD ENDP
0050 CODE ENDS
0050 END KEYBOARD
8 Digit LED

FIGURE 11-20  An 8-digit LED display interfaced to the 8088 microprocessor through an 82C55 PIA.
8 Digit LED

; INIT 8255
MOV AL,10000000B
MOV DX,703H
OUT DX,AL

; SETUP REGISTERS TO DISPLAY
MOV BX,8
MOV AH,7FH
MOV SI,OFFSET MEM-1
MOV DX.701H

; DISPLAY 8 DIGITS
DISP1: MOV AL,AH  ; select digit
OUT DX,AL
DEC DX  ; address PORT A
MOV AL,[BX+SI]
OUT DX,AL
CALL DELAY  ; wait 1 ms
ROR AH,1
INC DX  ; address PORT B
DEC BX  ; adjust count
JNZ DISP1

RET
FIGURE 11–22 The stepper motor showing full-step operation. (a) 45° (b) 135° (c) 225° (d) 315°.
Motor Application

; assign Port A to control the stepper motor.

; get position

; if right-hand direction

; if no steps

; step left

; A procedure to control the stepper motor.

FIGURE 11–23 A stepper motor interfaced to the 82C55. This illustration does not show the decoder.
Mode 1: I/O with Handshaking Capability

- Handshaking refers to the process of communicating back and forth between two intelligent devices
- **Example:** Process of communicating with a printer
  - a byte of data is presented to the data bus of the printer
  - the printer is informed of the presence of a byte of data to be printed by activating its strobe signal
  - whenever the printer receives the data it informs the sender by activating an output signal called ACK
  - the ACK signal initiates the process of providing another byte of data to the printer
- 8255 in mode 1 is equipped with resources to handle handshaking signals
Mode 1 Strobed Output

Enabled through PC6 using the BSR mode

Enabled through PC2 using the BSR mode

Figure 12-10. 8255A Mode 1 Output Diagram
(Reprinted by permission of Intel Corporation, Copyright Intel Corp. 1983)

Figure 12-11. Timing Diagram for Mode 1 Output
(Reprinted by permission of Intel Corporation, Copyright Intel Corp. 1983)
Mode 1 Strobed Output Signals

- **OBFa (output buffer full for port A)**
  - indicates that the CPU has written a byte of data into port A
  - must be connected to the STROBE of the receiving equipment
  - Goes back high again after ACK’ed by the peripheral.

- **ACKa (acknowledge for port A)**
  - through ACK, 8255 knows that data at port A has been picked up by the receiving device
  - 8255 then makes OBFa high to indicate that the data is old now. OBFa will not go low until the CPU writes a new byte of data to port A.

- **INTRa (interrupt request for port A)**
  - it is the rising edge of ACK that activates INTRa by making it high. INTRa is used to get the attention of the microprocessor.
  - it is important that INTRa is high only if INTEa, OBFa, ACKa are all high
  - it is reset to zero when the CPU writes a byte to port A

- **The 8255 enables the monitoring the status signals INTR, OBF, and INTE for both ports A and B. This is done by reading port C into the accumulator and testing the bits. This feature allows the implementation of polling**
Interrupts vs Polling

CPU services devices in two ways:

Interrupts and Polling

- In the interrupt method, whenever the device needs the service of the CPU, the device informs the CPU by sending an interrupt signal.
  - The CPU interrupts whatever it is doing and serves the request
  - The advantage of interrupts is that the CPU can serve many devices
  - Each receives a service based on its priority
  - Disadvantage of interrupts is that they require more hardware and software

- In polling, CPU monitors continuously a status condition and when the conditions are met, it will perform the service.
  - In contrast, polling is cheap and requires minimal software
  - But it ties down the CPU
Example

From the data segment:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MY DATA</td>
<td>DB</td>
<td>&quot;Hi. How are you?&quot;,CR,LF</td>
</tr>
<tr>
<td></td>
<td>DB</td>
<td>&quot;I am fine. How are you?&quot;,CR,LF,&quot;$&quot;</td>
</tr>
<tr>
<td>PA</td>
<td>EQU 300H</td>
<td>port A address</td>
</tr>
<tr>
<td>PB</td>
<td>EQU 301H</td>
<td>port B address</td>
</tr>
<tr>
<td>PC</td>
<td>EQU 302H</td>
<td>port C address</td>
</tr>
<tr>
<td>CWP</td>
<td>EQU 303H</td>
<td>control word port</td>
</tr>
<tr>
<td>LF</td>
<td>EQU 0AH</td>
<td>line feed</td>
</tr>
<tr>
<td>CR</td>
<td>EQU 0DH</td>
<td>carriage return</td>
</tr>
</tbody>
</table>

From the code segment:

```
MOV SI,OFFSET MY_DATA
AGAIN: MOV AH,[SI]
CMP AH,'$
JZ OVER
BACK: MOV DX,PC
IN AL,DX
AND AL, 80
JZ BACK
MOV DX,PA
MOV AL,AH
OUT DX,AL
INC SI
JMP AGAIN
OVER:
...
JMP AGAIN
```

OBF signal can also be checked (easier) instead of enabling INTA
Mode 1 Strobed Input

Port A with Handshake Signals

Port B with Handshake Signals

Control Word -- Mode 1 Input

D7 D6 D5 D4 D3 D2 D1 D0
1 0 1 1 1 1 x

I/O Mode
Port A Mode 1
Port A Input

Port B Input
Port B Mode 1
PC 6,7
1 = Input
0 = Output

Status Word -- Mode 1 Input

D7 D6 D5 D4 D3 D2 D1 D0
I/O I/O IBFA INTEA INTRA INTEB IBFB INTRB

INTEA is controlled by PC4 in BSR mode.
INTEB is controlled by PC2 in BSR mode.

2-12. 8255A Mode 1 Input Diagram

Permission of Intel Corporation, Copyright Intel Corp. 1983)
Mode 1 Input Ports with Handshaking Signals

- **STB**
  - When an external peripheral device provides a byte of data to an input port, it informs the 8255 through the STB pin. STB is of limited duration.

- **IBF (Input Buffer Full)**
  - In response to STB, the 8255 latches into its internal register the data present at PA0-PA7 or PB0-PB7.
  - Through IBF it indicates that it has latched the data but it has not been read by the CPU yet.
  - To get the attention of the CPU, IBF activates INTR.

- **INTR**
  - Falling edge of RD makes INTR low.
  - The RD signal from the CPU is of limited duration and when it goes high the 8255 in turn makes IBF inactive by setting it low.
  - IBF in this way lets the peripheral know that the byte of data was latched by the 8255 and read into the CPU as well.

- The two flip flops INTEA and INTB are set/reset using the BSR mode. The INTEA is enabled or disabled through PC4 and INTEB is enabled or disabled through PC2.
Write a procedure that reads data from the keyboard each time a key is typed.

Mode 1 Strobed Input Example

8255

PA0 - PA7

PC4

ASCII

D0 - D7

DAV

Keyboard

BIT5 EQU 20h
PORTC EQU 22h
PORTA EQU 20h
READ PROC NEAR
IN AL, PORTC
TEST AL, BIT5
JZ READ
IN AL, PORTA
RET
READ ENDP
Mode 1 Other Configurations

Combinations of Mode 1: Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

FIGURE 10. COMBINATIONS OF MODE 1
8255 MODE 2

- Bidirectional Operation is performed
- Used when interfacing two computers
- Also used in IEEE 488 parallel high speed general purpose instrumentation bus.
- Usually one CPU is the Master and the other is assigned as the Slave
Mode 2 Master-Slave Procedure

MASTERW PROC NEAR
trans: IN AL, PORTC
TEST AL, 80h
JZ TRANS
MOV AL, AH
OUT PORTA, AL
RET
;Master Write Process

Master Write to Slave MPU

1. The master MPU reads the status of OBF to verify whether the previous byte has been read by the slave MPU.

2. The master writes the data into port A and the 8255 informs the slave by causing the signal OBF to go low.

3. The Slave checks the OBF signal from the master for data availability.

4. The slave MPU reads data from port A and acknowledges the reading at the same time by making the ACK signal low.
MASTERW PROC NEAR
trans: IN AL, PORTC
TEST AL, 80h
JZ TRANS
MOV AL, AH
OUT PORTA, AL
RET
; Master Write Process

MASTERR PROC NEAR
IN AL, PORTC
TEST AL, 20h
JZ READ
IN AL, PORTA
RET
; Master Read Process
Interfacing DAC to a PC

• The digital to analog converter (DAV) is a device widely used to convert digital pulses to analog signals.
• The first criterion to judge a DAC is its resolution which is the a function of the number of binary inputs.
• The number of analog input levels is $2^n$ where n is the number of data bit inputs.
• Therefore the 8 input DAC such as the MC 1408 (DAC 808) provides 256 discrete voltage (or current) levels of output.
• $I_{out} = I_{ref} \left( \frac{D7}{2} + \frac{D6}{4} + \frac{D5}{8} + \frac{D4}{16} + \frac{D3}{32} + \frac{D2}{64} + \frac{D1}{128} + \frac{D0}{256} \right)$
Calculate the values of the LSB, MSB and full scale output for an 8 bit DAC for the 0-10v range?

1. **LSB** = \(1/2^8 = 1/256\)
   
   for 10 v, **LSB** = 10/256

2. **MSB** = \(1/2 \) full scale = 5v

3. **Full Scale Output** = (Full Scale Value – 1 LSB)
   
   = 10 v – 0.039v = 9.961v
Interfacing DAC via 8255

For ex:

If \( R_{14} = 2.5 \)

\[
I_0 = \frac{5 \text{ V}}{2.5 \text{ k}} \left( \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128} + \frac{1}{256} \right)
\]

\[
= 2 \text{ mA (255/256)}
\]

\[
= 1.992 \text{ mA}
\]
Example - Generate a Stair-Step Ramp

A1:

```
MOV AL,80H
MOV DX,303H
OUT DX,AL
MOV AH,01
INT 16H
JNZ STOP
SUB AL,AL
MOV DX,300H
OUT DX,AL
INC AL
CMP AL,0
JZ A1
MOV CX,02FFH
LOOP WT
JMP A2
MOV AH,4CH
INT 21H
```
Interfacing ADC to a PC

- A physical quantity (temperature, pressure, humidity, velocity) is converted to electrical (voltage, current) signals using a device called a transducer (sensor)
- We need an analog-to-digital converter (ADC) to translate the analog signals to digital numbers so that the PC can read them
- The techniques for ADC are
  - Successive Approximation Technique
    - Conversion Speed is important (Critical Conversion Time)
  - Integrating Type Converters
    - Conversion Accuracy is important
Using A/D with status check
Successive Approximation Technique

Analog voltage input to be converted

Digital test input

DA

Turn on bit 7

Compare Hi?

Y

Turn Off Bit

N

Keep Bit

Last Bit?

Y

End Of Conversion

N

Turn On Next Lower Bit
<table>
<thead>
<tr>
<th>Test Bit</th>
<th>DA Binary Value</th>
<th>Decimal</th>
<th>(5 * DAvalue)/256</th>
<th>Comparison Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000000</td>
<td>10000000</td>
<td>128</td>
<td>2.5</td>
<td>low - keep bit</td>
</tr>
<tr>
<td>01000000</td>
<td>11000000</td>
<td>192</td>
<td>3.75</td>
<td>high - drop bit</td>
</tr>
<tr>
<td>00100000</td>
<td>10100000</td>
<td>160</td>
<td>3.125</td>
<td>low - keep bit</td>
</tr>
<tr>
<td>00010000</td>
<td>10110000</td>
<td>176</td>
<td>3.4375</td>
<td>high - drop bit</td>
</tr>
<tr>
<td>00001000</td>
<td>10101000</td>
<td>168</td>
<td>3.28125</td>
<td>high - drop bit</td>
</tr>
<tr>
<td>00000100</td>
<td>10100100</td>
<td>164</td>
<td>3.203125</td>
<td>low - keep bit</td>
</tr>
<tr>
<td>00000010</td>
<td>10100110</td>
<td>166</td>
<td>3.2421875</td>
<td>high - drop bit</td>
</tr>
<tr>
<td>00000001</td>
<td>10100101</td>
<td>165</td>
<td>3.22265625</td>
<td>high - drop bit</td>
</tr>
</tbody>
</table>
ADC0848

- conversion time less than 110 µs
- RD active low: ADC converts the analog input to its binary equivalent and holds it in an internal register; with RD from high to low the data in the register shows at the D0-D7 pins
- WR start conversion: low to high input to inform the 804 to start the conversion process
- when the data conversion is complete, the INTR pin is forced low by the 804
- after INTR goes low, we make CS = 0 and send a high to low pulse to the RD pin to get the data out of the 804

MA0 – MA4 (multiplexed address)

Table 12-11: ADC0848 Analog Channel Selection (Single-Ended Mode)

<table>
<thead>
<tr>
<th>Selected Analog Channel</th>
<th>MA4</th>
<th>MA3</th>
<th>MA2</th>
<th>MA1</th>
<th>MA0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CH2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CH3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CH4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CH5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CH6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CH7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CH8</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: Channel is selected when CS = 0, RD = 1, and an L-to-H pulse is applied to WR.
ADC operation

1. Make CS = 0 and send a low to high pulse to the WR pin
2. Keep monitoring the INTR pin. If INTR is low, the conversion is finished and we can go to the next step
3. We then make CS = 0 again and send a high to low pulse to the RD pin to get the data output of the 804 chip

Note: CS is set to low for both RD and WR pulses.
ADC 804 Pins

• Vin (+) and Vin (-) are the differential analog inputs
• Vin = Vin (+) - Vin (-) ; to set this mode use MA4 and MA3 low.
• Vref is an input voltage used as the reference voltage

<table>
<thead>
<tr>
<th>Vref (volts)</th>
<th>Vin (volts)</th>
<th>Step Size (mv)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not connected</td>
<td>0 to 5</td>
<td>5/256 = 19.53</td>
</tr>
<tr>
<td>4</td>
<td>0 to 4</td>
<td>4/256 = 15.62</td>
</tr>
<tr>
<td>3</td>
<td>0 to 3</td>
<td>3/256 = 11.71</td>
</tr>
<tr>
<td>2</td>
<td>0 to 2.56</td>
<td>2/2.56 = 7.81</td>
</tr>
<tr>
<td>1</td>
<td>0 to 1</td>
<td>1/256 = 3.90</td>
</tr>
</tbody>
</table>

For given ADC, Vref = 2.56. Calculate the D0-D7 output if the analog input is

a) 1.7 v
b) 2.1 v

Answer: Step size is 2.56/256 = 10 mv
Dout = 1.7/10 mv = 170
170 is 10101011

Dout = Vin / (step size)
Connecting the 804

<table>
<thead>
<tr>
<th>PA0-PA7 to DO-D7 of ADC</th>
<th>CHANNEL SELECTION(out), DATA READ (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB0 TO INTR</td>
<td>PORT B AS INPUT</td>
</tr>
<tr>
<td>PC0 TO WR</td>
<td>PORT C AS OUTPUT</td>
</tr>
<tr>
<td>PC1 TO RD</td>
<td>PORT C AS OUTPUT</td>
</tr>
</tbody>
</table>
Required Steps

• While CS=0 RD=1
  – Provide the address of the selected channel on DB0 – DB7
  – Apply a L to H WR pulse
  – Channel 2 address is 09h, Channel 1 address is 08h, etc.
  – Not only we select the channel but conversion starts

• While WR=1 RD=1
  – Keep monitoring INTR asserted low
  – When INTR goes low, conversion finished

• After INTR becomes low
  – CS=0 and WR=1 and apply a low pulse to the RD pin to get the data out of the 848 IC chip
Example

MOV AL,82h  ;PA=out PB=in PC=out
MOV DX,CNT_PORT
OUT DX,AL
MOV AL,09  ;channel 2 address
MOV DX,PORT_A
OUT DX,AL
MOV AL,02  ;WR=0 RD=1
MOV DX,PORT_C ; not only selects channel but also starts ;
; conversion
OUT DX,AL
CALL DELAY ;few microsecs
MOV AL,03 ; WR=1 RD=1
OUT DX,AL
CALL DELAY
MOV AL,92h; PA=in PB=in PC=out
MOV DX,CNT_PORT
OUT DX,AL
Example

MOV DX,PORT_B
B1: IN AL,DX
AND AL,01
CMP AL,01
JNE B1
MOV AL,01 ;WR=1 RD=0
MOV DX,PORT_C
OUT DX,AL
MOV DX,PORT_A
IN AL,DX ;get the converted data
8254 Programmable Interval Timer

✓ Programmable Interval Timer generates accurate time delays

✓ Can be used in applications such as real-time clock, an event counter, a square wave generator or complex waveform generator

✓ Includes three identical 16 bit counters that can operate independently in any one of the six modes.

✓ To operate a counter, a 16 bit count is loaded in its register and begins to decrement the count until it reaches 0. At the end it generates a pulse.

✓ The counter can count either in BCD or Binary.

✓ Superset of its predecessor 8253
8254 Programmable Interval Timer

Pin Configuration

<table>
<thead>
<tr>
<th>A₁</th>
<th>A₀</th>
<th>Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Counter 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Counter 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Counter 2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Control Register</td>
</tr>
</tbody>
</table>

Pin Names

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D₇–D₀</td>
<td>Data Bus (8 Bit)</td>
</tr>
<tr>
<td>CLK</td>
<td>Counter Clock Inputs</td>
</tr>
<tr>
<td>GATE</td>
<td>Counter Gate Inputs</td>
</tr>
<tr>
<td>OUT</td>
<td>Counter Outputs</td>
</tr>
<tr>
<td>RD</td>
<td>Read Counter</td>
</tr>
<tr>
<td>WR</td>
<td>Write Command or Data</td>
</tr>
<tr>
<td>CS</td>
<td>Chip Select</td>
</tr>
<tr>
<td>A₀–A₁</td>
<td>Counter Select</td>
</tr>
<tr>
<td>VCC</td>
<td>+5 Volts</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
</tr>
</tbody>
</table>
## Accessing the registers

<table>
<thead>
<tr>
<th>CS</th>
<th>RD</th>
<th>WR</th>
<th>A1</th>
<th>A0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Write to counter 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Write to counter 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Write to counter 2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Write control word</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Read counter 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Read counter 2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Read counter 3</td>
</tr>
</tbody>
</table>
1) Counting is stopped by controlling the Gate input or the Clock input of the selected counter and two I/O read operations are performed by the CPU. The first I/O operation reads the low order byte and the second I/O operation reads the high order byte.

2) Second Method: An appropriate control word is written into the control register to latch the count in the output latch and again two read operations are performed.
8254 example

Problem:
1. Identify the port addresses of control register and counter 2
2. Write a subroutine to init counter2 in mode 0 with count of (50000) decimal. The subroutine should also include reading count on the fly.

1. Port Addresses
   The chip select is enabled when A7=1.
   Control Reg = 83H
   Counter 2 = 82H
8254 example contd.

COUNTER:

```assembly
MOV AL, B0H ; control word to init counter 2
OUT 83h, AL ; write in the control register
MOV AL, LOBYTE ; low order byte of the count 50000
OUT 82h, AL ; load
MOV AL, HIBYTE ; high order byte of the count 50000
OUT 82h, AL ; load
```

READ:

```assembly
MOV AL, 80H ; latch the current position in the count
OUT 83h, AL ; send it to control word
IN AL, 82h ; read the control word
MOV DL, AL
IN AL, 82h
MOV DH, AL
CMP DX, 00 ; check if the count is zero?
JNZ READ
```
8254 Modes

Mode 0: Interrupt on Terminal Count
- Count reaches zero the output goes high can be used as an interrupt, counting can be temporarily interrupted.

Mode 1: Programmable One-Shot
- Triggers by the gate, will generate a single pulse, length of count time.

Mode 2: Rate Generator Clock
- Similar to Mode 1 except the count is reloaded at the end. Goes on until a new count loaded.

Mode 3: Square Wave Generator
- Count decremented twice at each clock. Count reloaded when finished.

Mode 4: Software Triggered Strobe
- Output goes low for one clock at the end of count. COUNT must be reloaded.

Mode 5: Hardware Triggered Strobe
- Triggered by the rising pulse at the gate. COUNT begins/continues at Gate1.