Dec Hex Bin
18 12 00010010

ORG ; EIGHTEEN

Keyboard and Printer Interfacing

The x86 PC
assembly language, design, and interfacing
fifth edition
Prentice Hall

The x86 PC
assembly language, design, and interfacing
fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

OBJECTIVES

this chapter enables the student to:

- Diagram how a keyboard matrix is connected to the I/O ports of a PC.
- Describe the processes of key press detection and key identification performed by a microprocessor with pseudocode or flowchart.
- Describe the respective functions of the keyboard microcontroller, INT 09, and the motherboard in keyboard input.
- Code Assembly language instructions using INT 16H to get and check the keyboard input buffer and status bytes.

OBJECTIVES (*cont*)

this chapter enables the student to:

- Diagram how data from the keyboard is stored in the keyboard buffering.
- State the differences between hard contact and capacitance keyboards.
- List the 36-pin assignments of the Centronics printer interface.
- Describe the BIOS programming for the four parallel printer ports LPT1–LPT4.
- Diagram the I/O port assignment for printer data, status, and control.

OBJECTIVES (*cont*)

this chapter enables the student to:

- Code Assembly language instructions using INT 17H to check printer status, initialize printers, and send data to the printer.
- Describe the printer time-out problem and how it can be alleviated.
- Discuss the evolution of the PC's parallel port.
- Contrast and compare parallel port types, including SPP, PS/2, EPP, and ECP.

18.1: INTERFACING KEYBOARD TO CPU

- Keyboards are organized in a matrix of rows and columns, and the CPU accesses rows & columns through ports.
  - With two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor.
- On a keypress, a row & column make contact.
  - Otherwise, there is no connection between rows/columns.
- In PC keyboards, a single microcontroller takes care of hardware & software interfacing of the keyboard.
  - It scans keys continuously, identifies which one has been activated & presents it to the motherboard CPU.

18.1: INTERFACING KEYBOARD TO CPU

If *no* key has been pressed, the input port will yield 1s for all columns, since they are all connected to high. (Vcc)

If all the rows are grounded and a key is pressed, one of the columns will have 0.

The key pressed provides the path to ground.

4 x 4 matrix connected to two ports.

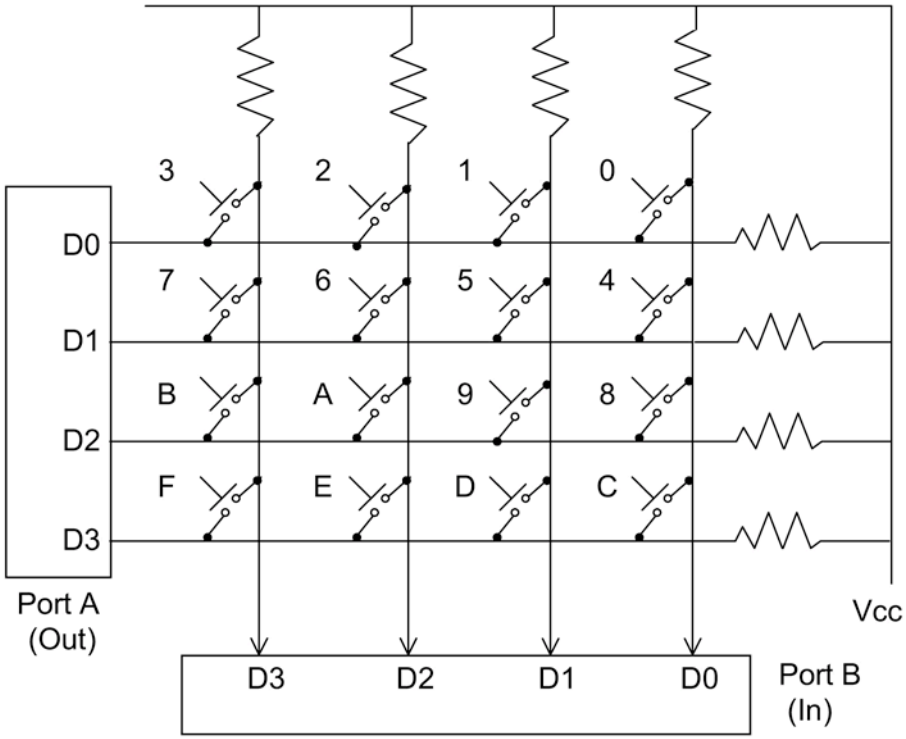


Fig. 18-1 Matrix Keyboard Connection to Ports

18.1: INTERFACING KEYBOARD TO CPU
grounding rows and reading the columns

- To detect the key pressed, the processor grounds all rows by providing 0 to the output latch, and then reads the columns.
  - If data read from the columns is D3–D0 = 1111, no key has been pressed
    - The process continues until a key press is detected.
  - If one of the column bits has a zero, a key press *has* occurred.

18.1: INTERFACING KEYBOARD TO CPU
grounding rows and reading the columns

- To identify the key, the microprocessor, starting with the top row, grounds it by providing a *low* to row D0 only, then it reads the columns.
  - If the data read is all 1s, no key in that row is activated.
    - The process is moved to the next row.
  - This process continues until the row is identified.

18.1: INTERFACING KEYBOARD TO CPU
grounding rows and reading the columns

- After identification of the row, the next task is to find out which column the pressed key belongs to.
  - Easy, since the CPU knows at any time which row and column are being accessed.

Example 18-1

From Figure 18-1, identify the row and column of the pressed key for each of the following.
(a) D3–D0 = 1110 for the row, D3–D0 = 1011 for the column
(b) D3–D0 = 1101 for the row, D3–D0 = 0111 for the column

Solution:

From Figure 18-1 the row and column can be used to identify the key.
(a) The row belongs to D0 and the column belongs to D2; therefore, the key number 2 was pressed.
(b) The row belongs to D1 and the column belongs to D3; therefore, the key number 7 was pressed.

18.1: INTERFACING KEYBOARD TO CPU
grounding rows and reading the columns

Program 18-1 is the Assembly language program for detection and identification of the key activation.

In this program, it is assumed that PORT_A and PORT_B are initialized as output and input, respectively.

```
;the following look-up scan codes are in the data segment
KCOD_0 DB 0,1,2,3          ;key codes for row zero
KCOD_1 DB 4,5,6,7          ;key codes for row one
KCOD_2 DB 8,9,0AH,0BH          ;key codes for row two
KCOD_3 DB 0CH,0DH,0EH,0FH      ;key codes for row three

       from the code segment
                           ;save BX
       AL=0 to ground all rows rows at once
       AL;to ensure all keys are open (no contact)
       _B                  ;read the columns
       111B                ;mask the unused bits (D7-D4)
       CMP AL,00001111B    ;are all keys released
       JNE K1              ;keep checking for all keys released
       CALL DELAY          ;wait for 20 ms
K2:    IN  AL,PORT_B       ;read columns
```

See the flowchart and entire program listing on page 466 - 468 of your textbook.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING

- The keyboard microcontroller used widely in the PC and compatibles is Intel's 8042. (or some variation)
  - A scan code is assigned to each key, and the controller provides the code for the pressed key to the motherboard.
- IBM PC/AT keyboards use the following data frame to send scan code serially to the motherboard.
  - For each scan code, a total of 11 bits are transferred.
    - One start bit (always 0)
    - 8 bits for scan code
    - Odd parity bit
    - One stop bit (always 1)

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
make and break

- In the IBM PC, the key press and release are represented by two different scan codes.
  - The key press is referred to as a *make*, for which the keyboard sends a scan code.
  - Release of the same key is called a *break*, for which another scan code is sent.
    - The scan code for the *break* is always 127 decimal (80H) larger than the *make* scan code.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
IBM PC scan codes

- The original IBM PC keyboard had 83 keys in three major groupings:
  - 1. The standard typewriter keys
  - 2. Ten function keys, F1 to F10
  - 3. 15-key keypad

See all the tables of keyboard scan codes on pages 470 - 471 of your textbook.

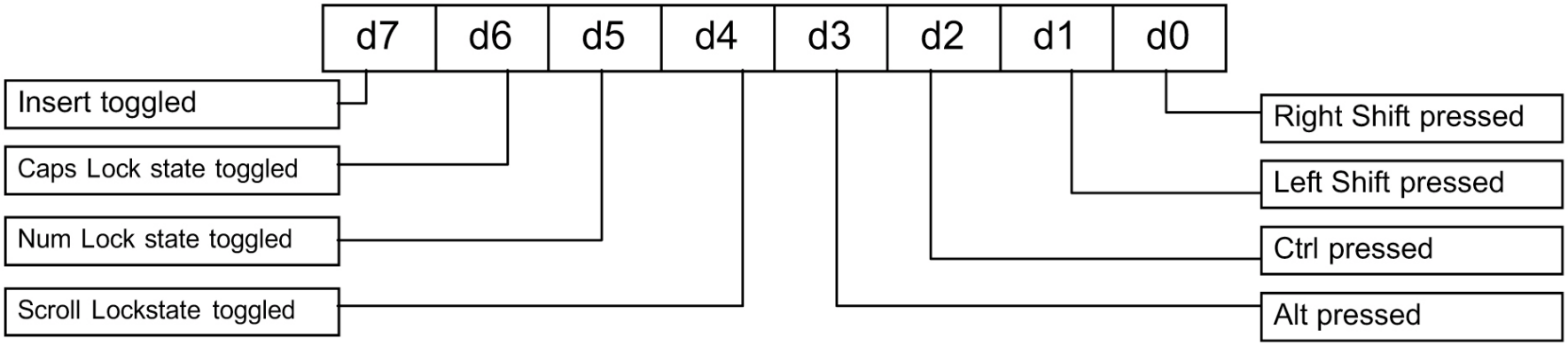
IBM later introduced the the enhanced keyboard, with the number of keys increased to 101.

Table 18-1: PC Scan Codes for 83 PC Keys

Hex	Key	Hex	Key	Hex	Key	Hex	Key
01	Esc	15	Y and y	29	~ and `	3D	F3
02	! and 1	16	U and u	2A	Left Shift	3E	F4
03	@ and 2	17	I and i	2B	| and \	3F	F5
04	# and 3	18	O and o	2C	Z and z	40	F6
05	$ and 4	19	P and p	2D	X and x	41	F7
06	% and 5	1A	{ and [	2E	C and c	42	F8
07	^ and 6	1B	} and]	2F	V and v	43	F9
08	& and 7	1C	enter	30	B and b	44	F10
09	* and 8	1D	ctrl			45	Num Lock

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
IBM PC scan codes

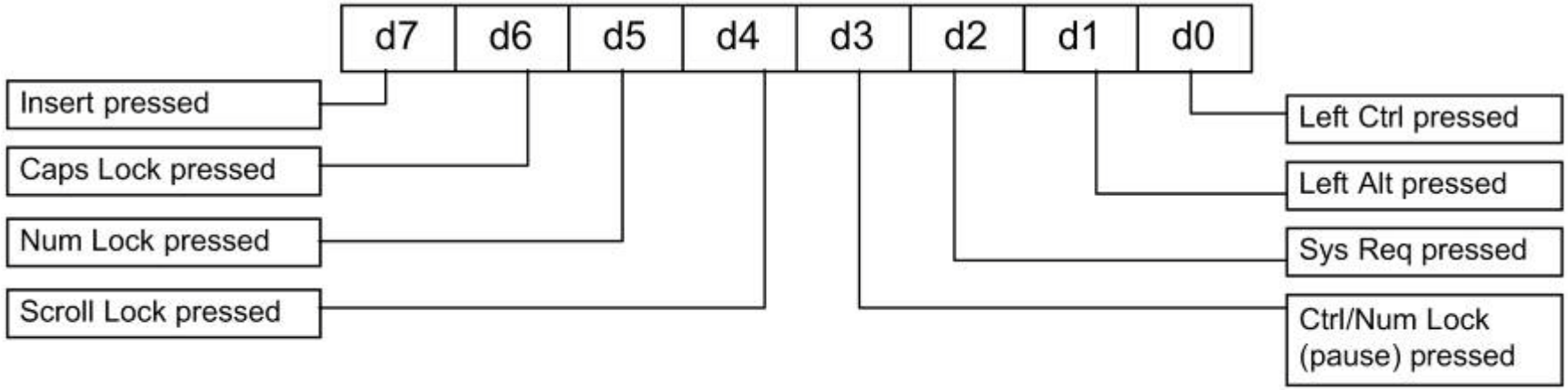
- The same scan code is used for a given lowercase letter & capital, as for all the keys with dual labels.
  - Data location 0040:0017H holds the *shift status byte*.



When a key is pressed, the interrupt service routine of INT 9 receives the scan code and stores it in a memory location called a keyboard buffer, located in the BIOS data area.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
IBM PC scan codes

- The same scan code is used for a given lowercase letter & capital, as for all the keys with dual labels.
  - BIOS location 0040:0018H holds the second status byte.



Some of the bits are used for the 101-key enhanced keyboards.

To relieve programmers from details of keyboard and motherboard interaction, IBM provides INT 16H.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
IBM PC scan codes

- **INT 16H, AH = 0** (read a character) - checks the keyboard buffer for a character.
  - If a character is available, it returns its scan code in AH and its ASCII code in AL.
  - If no character is available in the buffer, it waits for a key press and returns it.
  - For characters for which there is no ASCII code, it provides the scan code in AH and AL = 0.
    - Such as F1–F10.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
IBM PC scan codes

- **INT 16H, AH = 0** (read a character) - checks the keyboard buffer for a character.

Example 18-2

Run the following program in DEBUG. Interpret the result after typing each of the following. Run the program for each separately. (a) Z (b) F1 (c) ALT

```
MOV    AH,0
INT    16H
INT    3
```

Solution:

(a) AX = 2C7A AH = 2C, the scan code, and AL = 7A, ASCII for 'Z'

(b) AX = 3B00 AH = 3B, the scan code for F1, and AL = 00, because F1 is not an ASCII key

(c) Nothing happens because there is no scan code for the Alt key. The status of keys such as Alt is found in the keyboard status byte stored in BIOS at 40:17 and 40:18.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
IBM PC scan codes

- **INT 16H, AH = 01** (find if a character is available) - checks the keyboard buffer for a character.
  - If a character is available, it returns its scan code in AH, its ASCII code in AL, and sets ZF = 0.
  - If no character is available in the buffer, it does not wait for a key press, and simply makes ZF = 1.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
IBM PC scan codes

- **INT 16H, AH = 02** (return current keyboard status byte) - provides keyboard status in register AL.
  - The keyboard status byte (also referred to as the keyboard flag byte) is located in the BIOS data area memory location 0040:0017H.

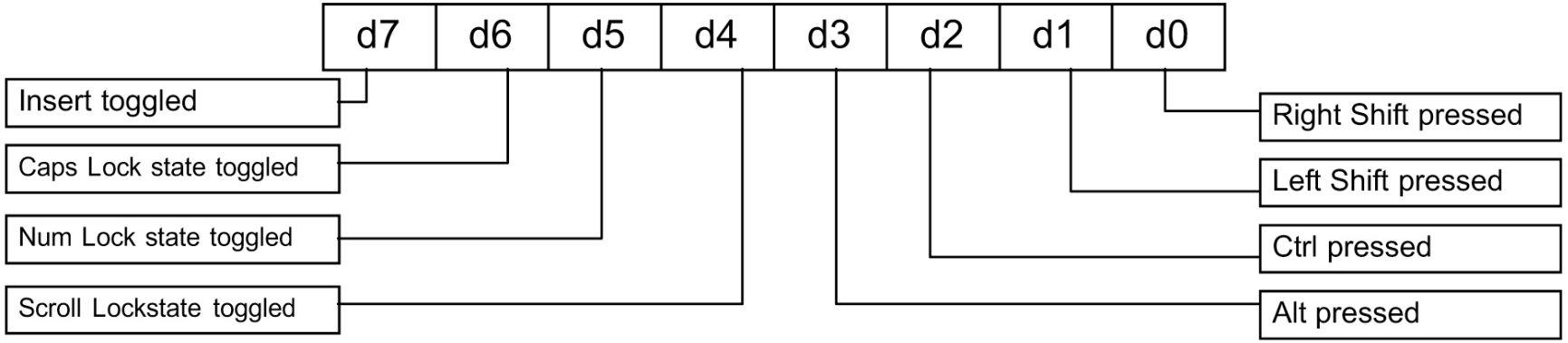


Fig. 18-3 First Keyboard Status Byte

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
IBM PC scan codes

- **INT 16H, AH = 02** (return current keyboard status byte) - provides keyboard status in register AL.

Example 18-3

Run the following program in DEBUG while the right shift key is held down and the Caps Lock key light is on. Verify it also by dumping the 0040:0017 location using DEBUG.

```
MOV   AH,02
INT   16H
INT   3
```

Solution:

Running the program while the Right Shift and Caps Lock keys are activated gives AH = 41H = 0100 0001 in binary, which can be checked against Figure 18-3. In DEBUG,

```
-d 0:417 418
```

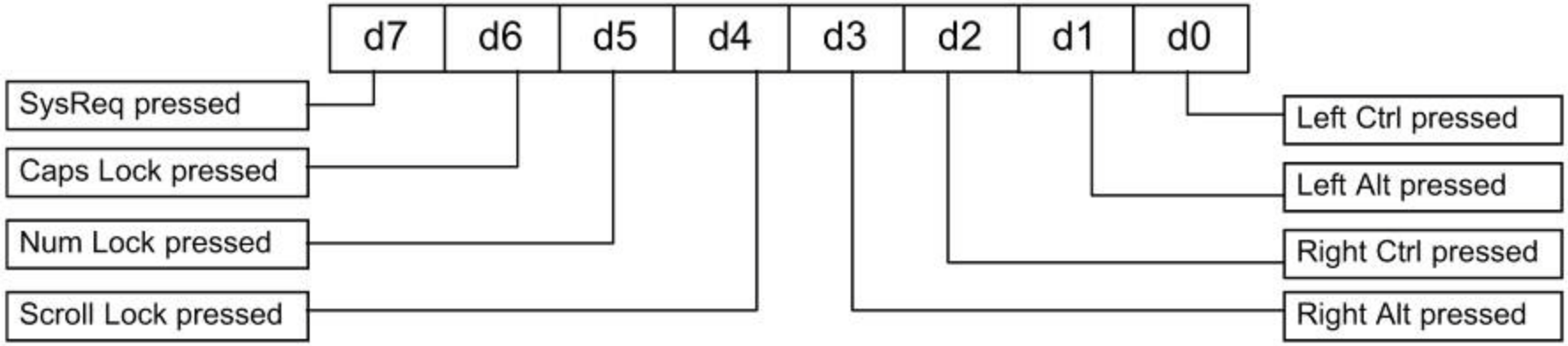
will provide the keyboard status byte 41-00.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
IBM PC scan codes

- **INT 16H, AH = 10H** (read a character) - same as AH = 0 except it also accepts the additional keys on the IBM extended (enhanced) keyboard.
- **INT 16H, AH = 11H** (find if a character is available) - same as AH = 1 except it accepts the additional keys on the IBM extended (enhanced) keyboard.
- **INT 16H, AH = 12H** (return the current status byte) - same as AH = 2 except it also provides the shift status byte of the IBM extended (enhanced) keyboard to AH.

See Fig. 18-5 and Example 18-4 listing on page 473 of your textbook.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
IBM PC scan codes



Example 18-4

Run the following program in DEBUG. Interpret the result after typing each of the following. Run the program for each separately. (a) F11 (b) ALT F11 (c) ALT TAB

```
MOV   AH,10H
INT   16H
INT   3
```

Solution: After running the program above in DEBUG for each case, we have the following:

(a) AX = 8500, where 85H is the scan code for F11

(b) AX = 8B00, where 8BH is the scan code for Alt-F11

(c) AX = A500, where A5H is the scan code for Alt-Tab

All of the cases above have AL = 00 since there is no ASCII code for these keys.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING INT 09 interrupt service

- To understand fully principles in the PC keyboard, it is necessary to know how INT 09 works.
  - The PC keyboard communicates with the motherboard through hardware interrupt IRQ1 of the 8259.

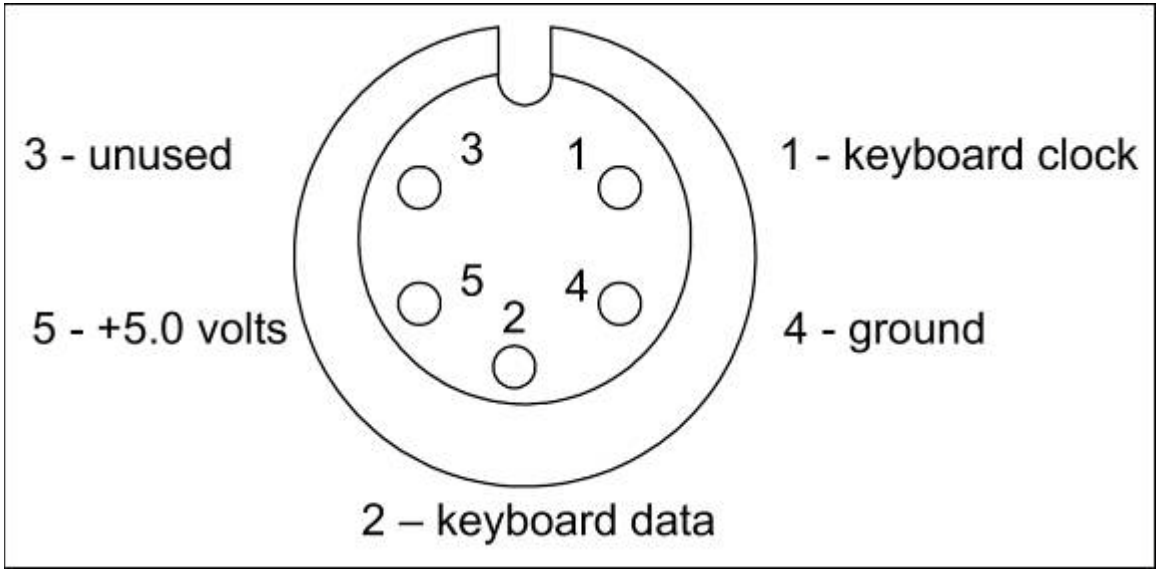


Fig. 18-6 Keyboard Cable Jack for the PC

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
INT 09 interrupt service sequence

1. The keyboard microcontroller scans the keyboard matrix continuously.
   - When a key is pressed (a *make*)…
     - It is identified and its scan code is sent serially to the motherboard through the keyboard cable.
   - The circuitry on the motherboard…
     - Receives the serial bits.
     - Gets rid of the frame bits
     - Makes one byte (scan code) with its serial-in-parallel-out shift register
     - Presents this 8-bit scan code to port A of 8255 at I/O address 60H.
     - Activates IRQ1.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
INT 09 interrupt service sequence

2. Since IRQ1 is set to INT 09, its interrupt service routine (ISR) residing in BIOS ROM is invoked.
3. ISR of INT 09 reads the scan code from port 60H.
4. ISR of INT 09 tests the scan code to see if it is a the Right or Left Shift, Alt, Ctrl keys, etc.
   - If so, the appropriate bit of the keyboard status bytes in BIOS 0040:0017H and 0018H are set.
     - It will not write the scan code to the keyboard buffer.
5. Before returning from INT 09, ISR will issue EOI to unmask IRQ1, followed by the IRET instruction.
   - This allows IRQ1 activation to be responded to again.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
INT 09 interrupt service sequence

6. When the key is released (a break), the keyboard generates the second scan code by adding 80H to it and sends it to the motherboard.
7. ISR of INT 09 checks the scan code to see if there is 80H difference between this code and the one.
   - If D7 is *high*, it is interpreted as meaning the key has been released & the system ignores the 2nd scan code.
   - If the key is held down more than 0.5 seconds, it is interpreted as a new key and INT 09 will write it into the keyboard buffer next to the preceding one.
     - Commonly referred to as *typematic* in IBM literature, which means repeating the same key.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
keyboard overrun

- On the keyboard side, the 8042 must serialize the scan code & send it by cable to the motherboard.
- On the motherboard side, circuits get the serial data & make a single byte of scan code out of the bit streams, and holds it for the CPU to read.
- The CPU can fall behind, and fail to keep up with the number of keystrokes, called *keyboard overrun*.
  - The motherboard beeps the speaker when overrun occurs.
- BIOS ROM on the motherboard is responsible for beeping the speaker in the event of keyboard overrun.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
BIOS keyboard buffer

- INT 09 gets the scan code from the keyboard and stores it in memory locations in the BIOS data area.
  - Referred to as the BIOS keyboard buffer, it should *not* be confused with the buffer in the keyboard itself.
- 32 bytes (16 words) of BIOS data memory is set aside, at addresses 40:001EH - 40:003DH.
  - Physical addresses 0041EH and 0043DH.
- Each two consecutive locations are used for a single character.
  - One for the scan code, the other for the ASCII code (if any) of the character.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
buffer pointers – tail pointer

- There are two keyboard buffer pointers:
  - The head pointer and the tail pointer.

Table 18-4: BIOS Data Area Used by Keyboard Buffer

Address of Head Pointer	Address of Tail Pointer	Keyboard Buffer
41A and 41B	41C and 41D	41E to 43D

  - It is the job of INT 09 to put the character in the keyboard buffer and advance the tail by incrementing the word contents of location 0041C, where the tail pointer is held.
  - Memory locations 0040:001CH & 0040:001DH (physical addresses **0041CH** & **0041DH**) hold the address for the tail.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
buffer pointers – head pointer

- As INT 16H reads a character from the keyboard buffer, it advances the head pointer, which is held by memory locations 41AH and 41BH.
  - As INT 09 *inserts* the character *into* the keyboard buffer, it *advances* the ***tail***.
  - As INT 16H *reads* the character *from* the keyboard buffer it *advances* the ***head***.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
buffer pointers

- As INT 16H reads a character from the keyboard buffer, it advances the head pointer, which is held by memory locations 41AH and 41BH.
  - When they come to the end of the keyboard buffer, they both wrap around, creating a ring of 16 words where the head is continuously chasing the tail.
  - If the buffer is empty, head address equals tail address.

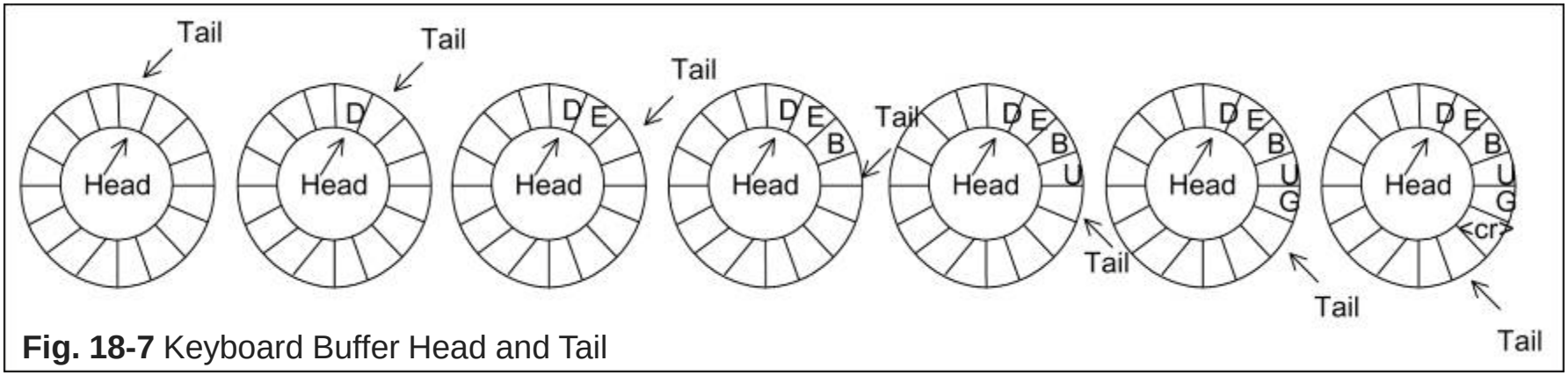


Fig. 18-7 Keyboard Buffer Head and Tail

18.2: PC KEYBOARD INTERFACING/PROGRAMMING buffer pointers

- As INT 09 inserts characters into the buffer, the tail is moved.
  - If the buffer is not read by INT 16H, it becomes full, which causes the tail to be right behind the head.

Example 18-6

Using DEBUG, dump the location where the head and tail pointers are held. Compare them. Is the buffer full or empty?

Solution:

```
C>DEBUG
-D 0:410 41F
0000:0410 63 44 F0 80 02 00 01 40-00 00 3C 00 3C 00 20 39 cD.....@..<.<. 9
-
```

In this case, the head and tail pointers point to the same location; therefore, the buffer is empty.

18.2: PC KEYBOARD INTERFACING/PROGRAMMING
PC keyboard technology

- The keyboard shown is a *hard contact* keyboard.
  - When a key is pressed, a physical contact in the row & column causes the column to be pulled to ground.
  - The alternative to hard contact keyboards are *capacitive keyboards*.
    - No physical contact.
    - There is a capacitor for each point of the matrix.

4 x 4 matrix connected to two ports.

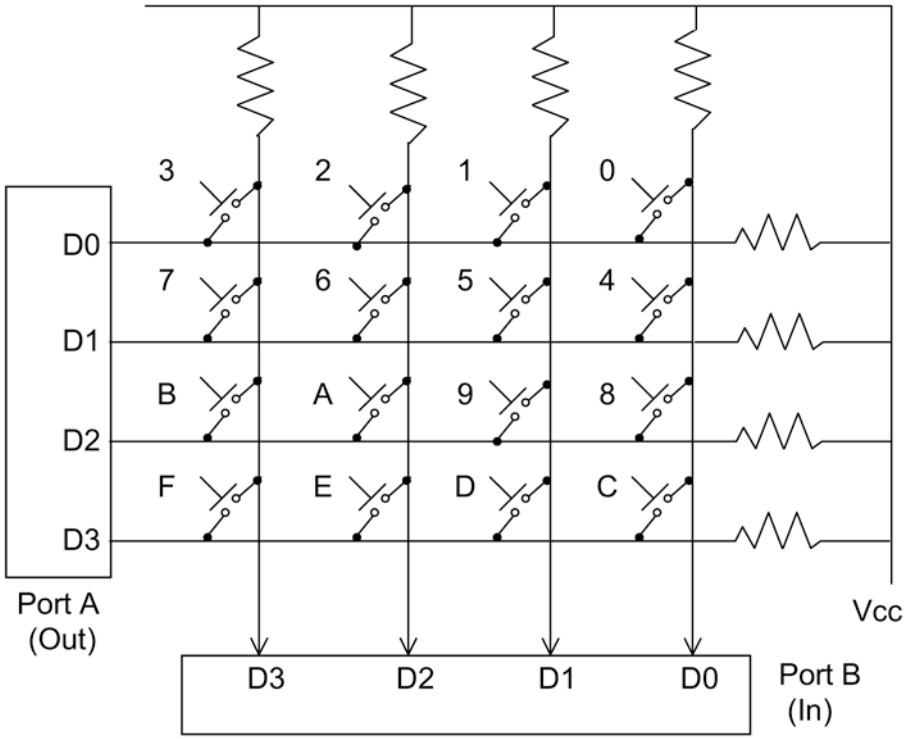


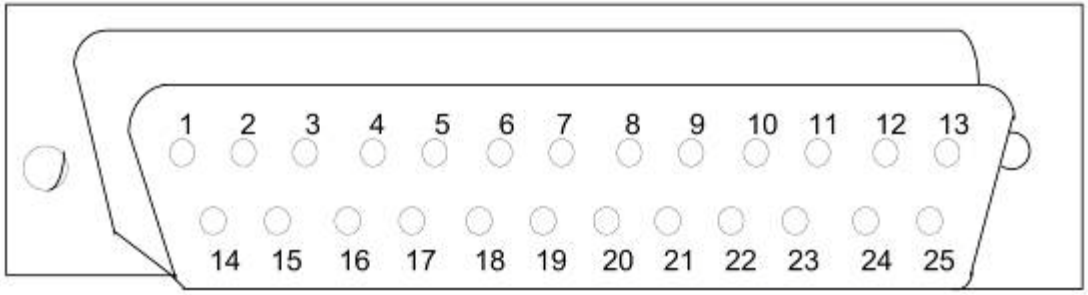
Fig. 18-1 Matrix Keyboard Connection to Ports

18.3: PRINTER & INTERFACING IN THE PC Centronics printer interface pins

- The Centronics type parallel printer interface is the printer interface standard in the x86 PC.
  - Referred to as Epson FX 100 standard, it is a 36-pin interface connector with pins labeled as 1 to 36.
    - Many are used for ground, to reduce electrical noise.
- The 36 Centronics pins can be grouped as follows.
  - 1. The **data lines** carry data, sent by the PC to the printer.
  - 2. The **printer status signals** indicate the printer status at any given time.
  - 3. **Printer control signals** tell the printer what to do.
  - 4. **Ground signals** provide an individual ground return line for each data line, and certain control & status lines.

18.3: PRINTER & INTERFACING IN THE PC
data lines and grounds

- Input pins DATA 1 to DATA 8 provide a parallel pathway for 8-bit data sent by the PC to the printer.



These are all output pins from the printer to the PC used by the printer to indicate its own status.

Table 18-5: DB-25 Printer Pins

Pin	Description	Pin	Description
1	$\overline{\text{Strobe}}$	13	Select
2	Data bit 0	14	$\overline{\text{Auto feed}}$
3	Data bit 1	15	$\overline{\text{Error}}$
4	Data bit 2	16	Initialize printer
5	Data bit 3	17	$\overline{\text{Select input}}$
6	Data bit 4	18	Ground
7	Data bit 5	19	Ground
8	Data bit 6	20	Ground
9	Data bit 7	21	Ground
10	$\overline{\text{Acknowledge}}$	22	Ground
11	Busy	23	Ground
12	Out of paper	24	Ground
		25	Ground

Fig. 18-8 DB25 Printer Pin

18.3: PRINTER & INTERFACING IN THE PC
data lines and grounds

- **PE** (pin 12) - used by the printer to indicate it is out of paper.
- **BUSY** (pin 11) - is *high* if the printer is not ready to accept a new character.
  - This pin is *high* when the printer is off line or when it is printing & can't accept any data.
  - As long as this pin is high, the PC will not transfer data to the printer.

Table 18-5: DB-25 Printer Pins

Pin	Description	Pin	Description
1	$\overline{\text{Strobe}}$	13	Select
2	Data bit 0	14	$\overline{\text{Auto feed}}$
3	Data bit 1	15	$\overline{\text{Error}}$
4	Data bit 2	16	Initialize printer
5	Data bit 3	17	$\overline{\text{Select input}}$
6	Data bit 4	18	Ground
7	Data bit 5	19	Ground
8	Data bit 6	20	Ground
9	Data bit 7	21	Ground
10	$\overline{\text{Acknowledge}}$	22	Ground
11	Busy	23	Ground
12	Out of paper	24	Ground
		25	Ground

18.3: PRINTER & INTERFACING IN THE PC
data lines and grounds

- **ACKNLG** (pin 10) - used to acknowledge printer receipt of data & that it can accept a new character.
- **ERROR** (pin 32) - normally high output, activated (*goes low*) in conditions such as out of paper, off line state, or jammed printhead, in which the printer cannot print.
- **SLCT** (pin 13) is *active-high*, from printer to PC when the printer is on & online.

Table 18-5: DB-25 Printer Pins

Pin	Description	Pin	Description
1	$\overline{\text{Strobe}}$	13	Select
2	Data bit 0	14	$\overline{\text{Auto feed}}$
3	Data bit 1	15	$\overline{\text{Error}}$
4	Data bit 2	16	Initialize printer
5	Data bit 3	17	$\overline{\text{Select input}}$
6	Data bit 4	18	Ground
7	Data bit 5	19	Ground
8	Data bit 6	20	Ground
9	Data bit 7	21	Ground
10	$\overline{\text{Acknowledge}}$	22	Ground
11	Busy	23	Ground
12	Out of paper	24	Ground
		25	Ground

18.3: PRINTER & INTERFACING IN THE PC
printer control signals

- **STROBE** (pin 1) and **ACKNLG** are the most widely used signals among control and status pins.
  - When the PC presents a character to the data pins of the printer, it activates the STROBE pin of the printer.
    - Telling it that there is a byte sitting at the data pins.
- **INIT** (pin 31) - an input to the printer, normally *high*.
  - When it is activated (*active-low*) it resets the printer.

18.3: PRINTER & INTERFACING IN THE PC
printer control signals

- **AUTO FEED XT** & **SLCT IN** are two other control signals in the printer.

Table 18-6: Centronics Printer Specification

Serial #	Return #	Signal	Direction	Description
1	19	$\overline{\text{STROBE}}$	IN	STROBE pulse to read data in . Pulse width must be more than 0.5 µs at receiving terminal . The signal level is normally "high"; read -in of data is performed at the "low" level of this signal
2	20	DATA 1	IN	These signals represent information of the 1st to 8th bits of parallel data, respectively. Each signal is at "high" level when data is logical "1" , and "low" when logical "0".
3	21	DATA 2	IN	
4	22	DATA 3	IN	
5	23	DATA 4	IN	
6	24	DATA 5	IN	
7	25	DATA 6	IN	
8	26	DATA 7	IN	
9	27	DATA 8	IN	
10	28	$\overline{\text{ACKNLG}}$	OUT	Approximately 0.5 µs pulse; "low" indicates data has been received and printer is ready for data
11	29	BUSY	OUT	A "high" signal indicates that the printer cannot receive ... becomes "high" in the following cases : (1) ... "off"

See the entire table on page 479 of your textbook.

18.3: PRINTER & INTERFACING IN THE PC
steps in computer & printer communication

1. The computer checks to see if a BUSY signal from the printer indicates the printer is ready (not busy).
2. The computer puts 8-bit data on the data line connected to the printer data pins.
3. The computer activates STROBE by making it *low*.
   - Prior to asserting the printer input STROBE pin, the data must be at the printer's data pins at least for 0.5 µs.
4. The STROBE must stay *low* for at least 0.5 µs before the computer brings it back to *high*.
   - The data must stay at the printer's data pins at least 0.5 µs after the STROBE pin is deasserted (brought *high*).

18.3: PRINTER & INTERFACING IN THE PC
steps in computer & printer communication

5. The activation of STROBE causes the printer to assert its BUSY output pin high character.
   - Indicating to the computer to wait until it finishes taking care of the last byte.

18.3: PRINTER & INTERFACING IN THE PC
steps in computer & printer communication

6. When the printer is ready to accept another byte, it sends the ACKNLG signal back to the computer by making it low.
   - The printer keeps the ACKNLG signal low only for 5 µs.
   - At the rising edge of ACKNLG, the printer makes the BUSY (not BUSY = ready) pin low to indicate that it is ready to accept the next byte.
   - The CPU can use either the ACKNLDG or BUSY signals from the printer to initiate the process of sending another byte to printer.
   - Some systems use BUSY and some use ACKNLG.

18.3: PRINTER & INTERFACING IN THE PC
IBM PC printer interfacing

- As printers connected to parallel ports are identified by the In the IBM PC, the POST (power-on self-test), the base I/O port address of each is written into the BIOS data area.
  - 0040:0008 to 0040:000FH—just like the COM port.

18.3: PRINTER & INTERFACING IN THE PC
IBM PC printer interfacing

- Memory locations 0040:0008H and 0040:0009H hold the base I/O address of LPT1, etc.
  - Physical locations 00408H and 00409H.
    - If no printer port is available, 0s are found.

Base I/O port addresses assigned to LPTs can vary from system to system.

Table 18-7: BIOS I/O Base Addresses for LPT

I/O Base Address	LPT
0040:0008 – 0040:0009	LPT1
0040:000A – 0040:000B	LPT2
0040:000C – 0040:000D	LPT3
0040:000E – 0040:000F	LPT4

18.3: PRINTER & INTERFACING IN THE PC
IBM PC printer interfacing

- 8 bytes in the BIOS data area can store the base I/O address of four printers, each taking 2 bytes.
  - 00408 to 0040FH can be checked to see which LPT (line printer) port is available.

Table 18-8: IBM PC Printer Ports and Their Functions

Line Printer	Data Port (R/W)	Status Port (Read Only)	Control Port (R/W)
LPT 1	03BCH	03BDH	03BEH
LPT 2	0378H	0379H	037AH
LPT 3	0278H	0279H	027AH

18.3: PRINTER & INTERFACING IN THE PC
what is printer time-out?

- The printer time-out message means the printer port is installed, but the printer is not ready to print.
  - On detecting the printer port installed, BIOS tries for 20 seconds to see if it is ready to accept data.
    - If the printer is not ready, the PC gives up. (time-out)
  - The amount of time BIOS tries to get a response is stored in BIOS data area 0040:0078 to 0040:007B.
    - At boot time, these locations are initialized to 20 seconds.

18.3: PRINTER & INTERFACING IN THE PC
ASCII control characters

- Certain characters in ASCII control the printer.

Table 18-9:
ASCII Printer Control Characters

ASCII Symbol	Hex Code	Function
BS	08	Backspace
HT	09	Horizontal tab
LF	0A	Line feed (advances one line)
VT	0B	Vertical tab
FF	0C	Form feed (advances to next Page)
CR	0D	Carriage return (return to left margin)

18.3: PRINTER & INTERFACING IN THE PC
Programming the printer with BIOS INT 17H

- BIOS INT 17H provides three services:
  - Printing a character.
  - Initializing the printer port.
  - Getting the printer status port.
    - Options are selected according to the value set in the AH register.

18.3: PRINTER & INTERFACING IN THE PC
Programming the printer with BIOS INT 17H

- **INT 17H, AH = 0** (print a character) - expects to have the LPT number in register DX (0 for LPT1, 1 for LPT2, and 2 for LPT3) and the ASCII character to be printed in the AL register.
  - On return, INT 17H provides the status of the selected printer port .

Bit No.	Function
71 =	Not BUSY(ready), 0 = BUSY
61 =	Acknowledge
51 =	Out of paper
41 =	Printer selected
31 =	I/O error 2,1 unused
01 =	Printer time-out

18.3: PRINTER & INTERFACING IN THE PC
Programming the printer with BIOS INT 17H

- **INT 17H, AH = 02** (get the printer port status) - allows a programmer to check status of the printer.
  - Before calling the function, AH is set to 2 and DX holds the printer number (0 = LPT1, 1 = LPT2, and 2 = LPT3).
  - After calling, AH = status, the situation is the same as shown under option 0.

18.3: PRINTER & INTERFACING IN THE PC
SPP

- SPP stands for *standard parallel port*.
  - The parallel port of the first IBM PC introduced in 1981.
  - Unidirectional,designed to send data from PC to printer.
- Internal logic circuitry is set for data output only.
  - Attempts to use the data bus for input can damage the LPT port.
- For further information, refer to web page *http://www.lvr.com.*

18.3: PRINTER & INTERFACING IN THE PC
PS/2 and bidirectional data bus

- 1987 saw the first change in the data bus portion of LPT, with introduction of PS/2 computers by IBM.
  - Internal circuitry of the data section of the LPT port in the x86 PC was changed to make it bidirectional.
- However, upon bootup, BIOS configured the LPT port as SPP, to be used only for data output.
  - By making C5 = 1, we make the data port an input port.
- By making the data port bidirectional in the PS/2, IBM allowed many devices such as tape backup, scanners, and data acquisition instruments to use the LPT port instead of a PC expansion slot.

18.3: PRINTER & INTERFACING IN THE PC
EPP

- EPP stands for *enhanced parallel port*, in which a higher speed was achieved by delegating hand-shaking signals to circuitry on the LPT port itself.
  - The standard also added new registers to the I/O port address space beyond base address +2.
    - In EPP, the I/O space goes from base to base +7.

18.3: PRINTER & INTERFACING IN THE PC
ECP

- ECP stands for *extended capability port* - with all the features of EPP, plus DMA (direct memory address) capability.
  - Allowing it to transfer data via the DMA channel.
    - It also has data compression capability.
- An ideal port for high-speed peripherals such as laser printers and scanners.
- While an ECP type LPT port is supposed to support SPP, PS/2, and EPP, not all can emulate EPP.
  - To see if a given ECP supports EPP, examine the PC documentation or check the CMOS setup on your PC.

Dec Hex Bin
18 12 00010010

ENDS ; EIGHTEEN

The x86 PC

assembly language, design, and interfacing

fifth edition

MUHAMMAD ALI MAZIDI
JANICE GILLISPIE MAZIDI
DANNY CAUSEY

The x86 PC
assembly language, design, and interfacing
fifth edition

Prentice Hall