Unit 6: Input/ Output Interfaces

Syllabus

- **Parallel Communication – Introduction and Applications**
- **Serial Communication**
  - Introduction and Applications
  - Introduction to Programmable Communication Interface 8251
  - Basic Concept of Synchronous and Asynchronous Modes
- **Simple I/O, Strobe I/O, Single handshake I/O, Double handshake I/O**
- **8255A and it's Working**
  - Block Diagram
  - Modes of Operation
  - Control Word
- **RS-232 – Introduction, Pin Configuration (9 pin and 25 pin) and function of each pin, Interconnection between DTE-DTE and DTE-DCE**

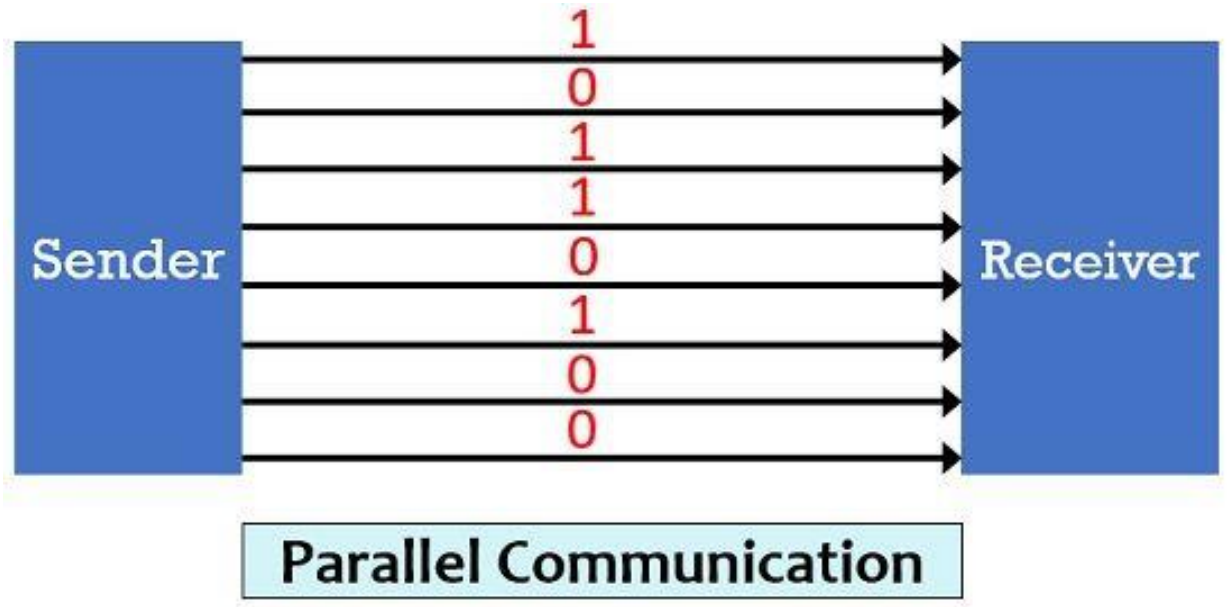
Communication Interface

❖There are two ways in which a microprocessor can connect with outside world or other memory systems.

1. Serial Communication Interface:
   - ❑In serial communication interface, the interface gets a single byte of data from the microprocessor and sends it bit by bit to other system serially (or) the interface receives data bit by bit serially from the external systems and converts the data into a single byte and transfers it to the microprocessor.
2. Parallel Communication Interface:
   - ❑This interface gets a byte of data from microprocessor and sends it bit by bit to the other systems in simultaneous (or) parallel fashion. The interface also receives data bit by bit simultaneously from the external system and converts the data into a single byte and transfers it to microprocessor.

Parallel Interface/Communication

- In parallel communication the various data bits are simultaneously transmitted using multiple communication links between sender and receiver. Here, despite using a single channel between sender and receiver, various links are used and each bit of data is transmitted separately over all the communication link.
- The figure below shows the transmission of 8 byte data using parallel communication technique:

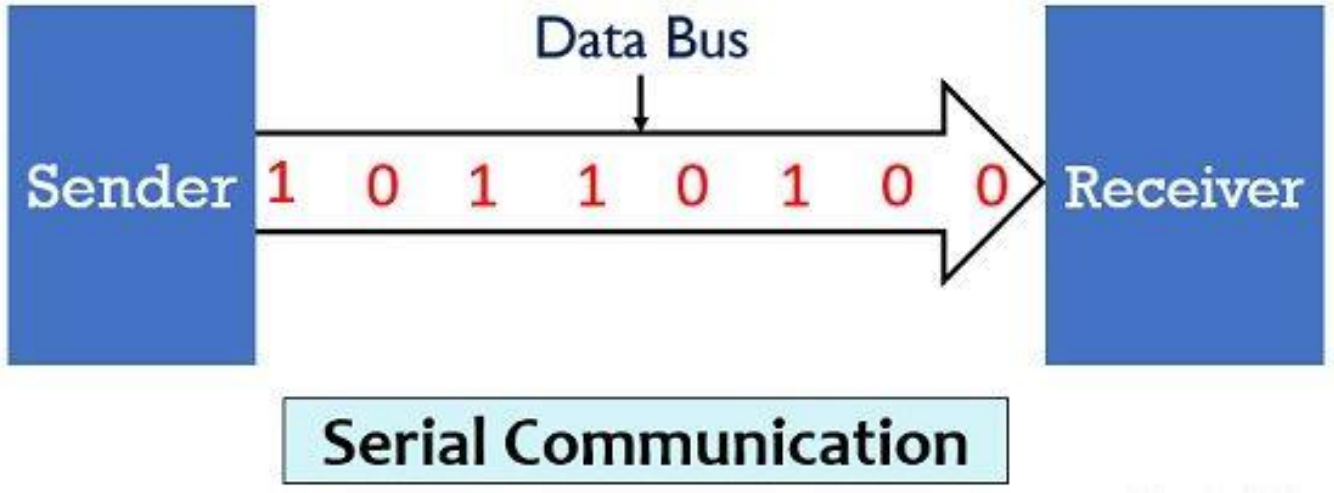


Parallel Communication

- Here, as we can see that for the transmission of 8-bit of data, 8 separate communication links are utilized. And so rather following a sequential data transmission, simultaneous transmission of data is allowed. This leads to a faster communication between the sender and receiver.
- But for connecting multiple lines between sender and receiver multiple connecting unit are to be present between a pair of sender and receiver. And this is the reason why parallel communication is not suitable for long distance transmission, because connecting multiple lines to large distances is very difficult and expensive.

Serial Interface/Communication

- In serial communication the data bits are transmitted serially over a common communication link one after the other. Basically it does not allow simultaneous transmission of data because only a single channel is utilized. Thereby allowing sequential transfer rather than simultaneous transfer.
- The figure below shows the serial data transmission:



Serial Communication

Serial Communication

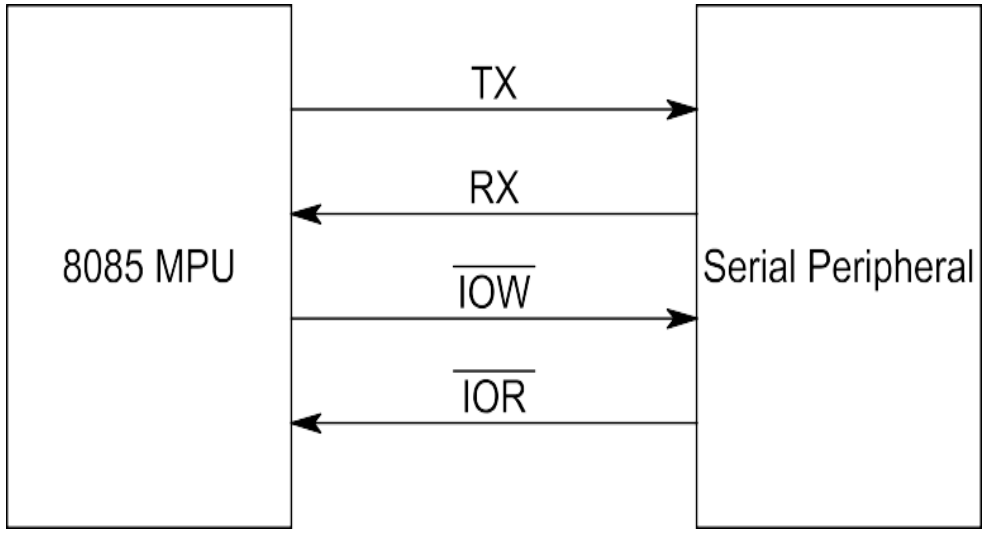
- It is highly suitable for long distance signal transmission as only a single wire or bus is used. So, it can be connected between two points that are separated at a large distance with respect to each other. But as only a **single data bit is transmitted per clock pulse** thus the transmission of data is a quiet time taking process.

Comparison Chart

Basis for Comparison	Serial Communication	Parallel Communication
Data transmission speed	Slow	Comparatively fast
Number of communication link used	Single	Multiple
Number of transmitted bit/clock cycle	only one bit.	n number of link will carry n bits.
Cost	Low	High
Crosstalk	Not present	Present
System Up-gradation	Easy	Quite difficult
Mode of transmission	Full duplex	Half duplex
Suitable for	Long distance	Short distance
High frequency operation	More efficient	Less efficient

Serial Interface/Communication

- It is a method of conveying a single bit at a time i.e. only one bit of a word is transmitted at a time.
- It is slower.
- Hardware requirement is simple.
- E.g. RS-232C



Modes of Serial Data Transfer

I. Simplex Mode

- In simplex transmission, data are transmitted in only one direction. A typical example is transmission from a microcomputer to a printer.
- Data travel in only one direction. E.g. from computer to printer

II. Duplex Mode

- In duplex transmission, data flow in both directions. However, if the transmission goes one way at a time, it is called **half duplex**; if it goes both ways simultaneously, it is called **full duplex**.
- Generally, transmission between two computers or between a computer and a terminal is full duplex.

Types of Serial Data Transfer

I. Synchronous

- Both transmitter and receiver are synchronized by same clock pulse.
- It is also called clock-oriented data transmission.
- Speed: >20 Kbps
- Always implemented with hardware.

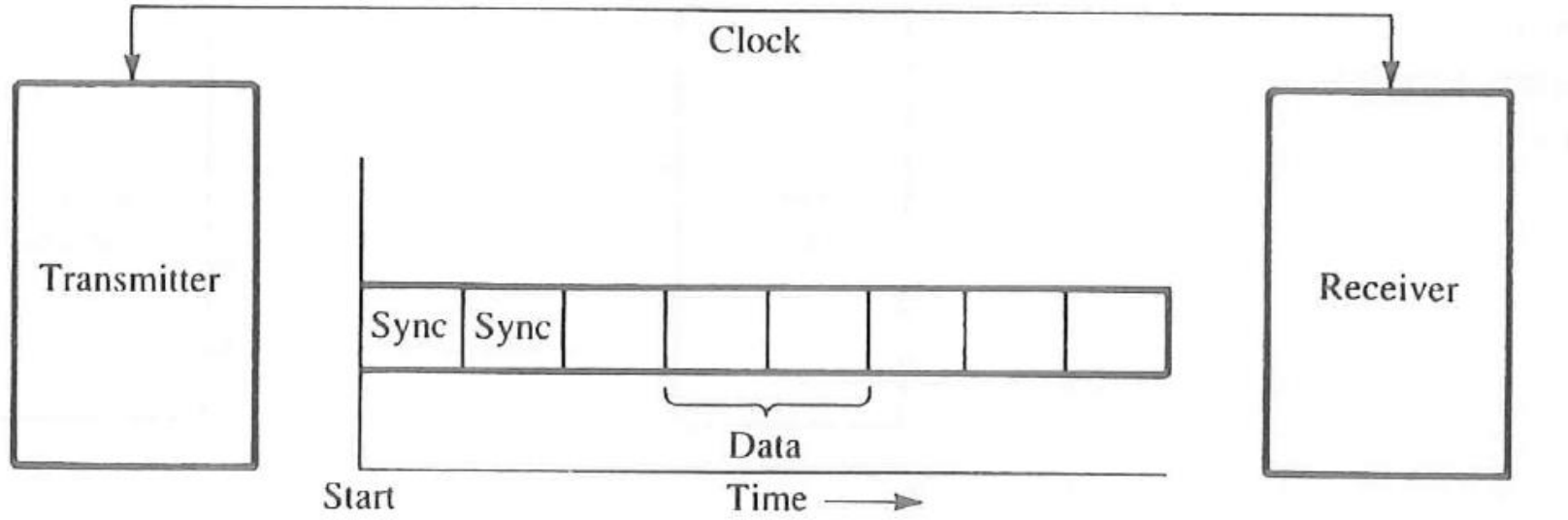


Fig a: Synchronous Transmission Format

Types of Serial Data Transfer

II. Asynchronous

- ❑ Both transmitter and receiver are synchronized by separate clock pulse.
- ❑ It is also called character-oriented data transmission.
- ❑ Speed: <20 Kbps.
- ❑ Always implemented with hardware and software.

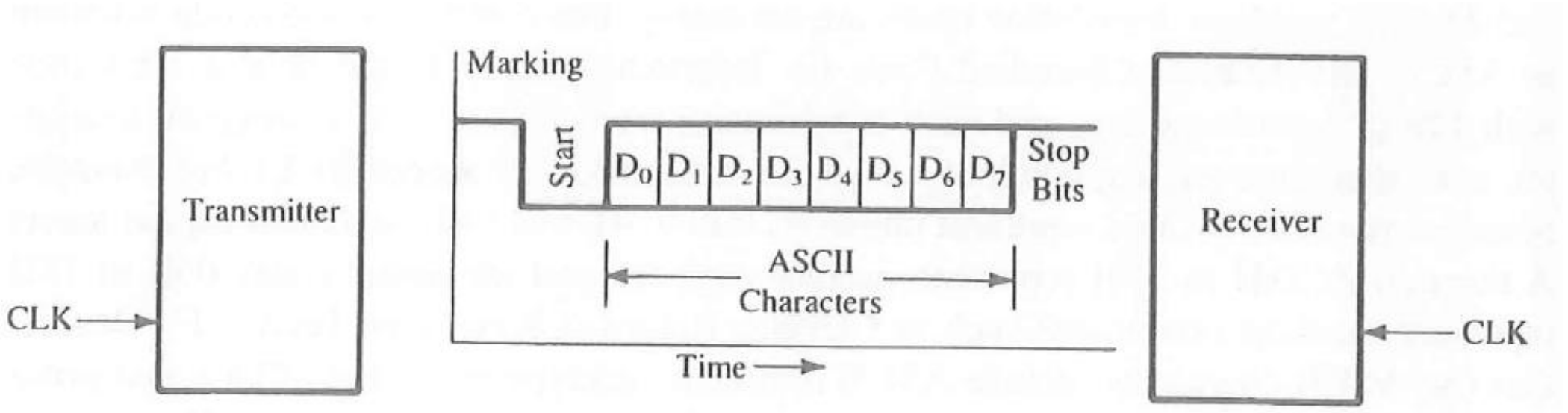


Fig b: Asynchronous Transmission Format

SYNCHRONOUS VS. ASYNCHRONOUS TRANSMISSION

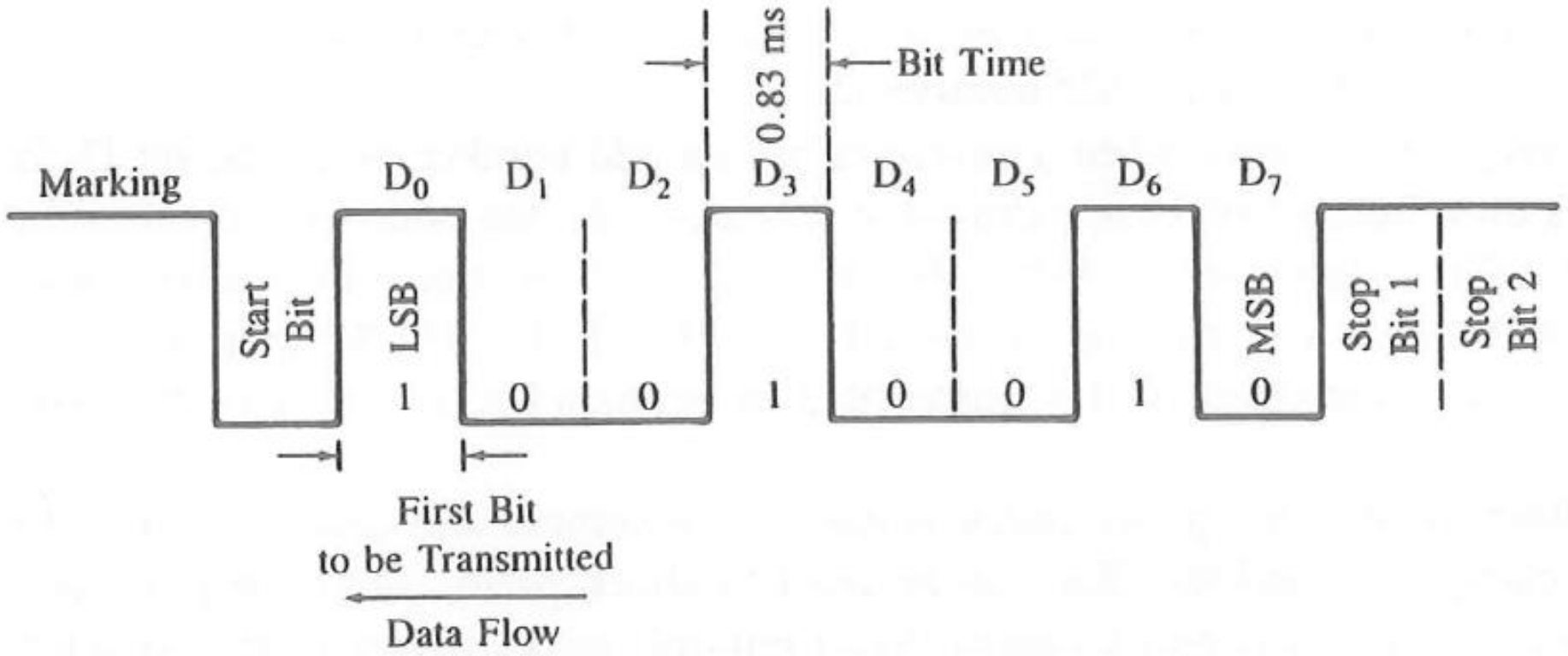
- Serial communication occurs in either synchronous or asynchronous format.
- In the **synchronous format**, a receiver and a transmitter are synchronized; a block of characters is transmitted along with the synchronization information, as in **Figure a**. This format is generally used for high-speed transmission (more than 20 k bits/second).
- The **asynchronous format** is character-oriented. Each character carries the information of the Start and the Stop bits, shown in **Figure b**. When no data are being transmitted, a receiver stays high at logic 1, called **Mark**; logic 0 is called **Space**. Transmission begins with one **Start bit** (low), followed by a character and **one or two Stop bits** (high). This is also known as **framing**. **Figure b** shows the transmission of 11 bits for an ASCII character in the asynchronous format: one Start bit, eight character bits, and two Stop bits.
- The format shown in **Figure b** is similar to Morse code, but the dots and dashes are replaced by logic 0s and 1s. The asynchronous format is generally used in low speed transmission (less than 20 k bits/second).

SYNCHRONOUS DATA TRANSFER	ASYNCHRONOUS DATA TRANSFER
A data transfer method that sends a continuous stream of data to the receiver using regular timing signals that ensures both transmitter and receiver are synchronized with each other	A data transfer method that sends data from transmitter to receiver with parity bits (start and stop bits) in uneven intervals
Sender and receiver operate on the same clock frequencies	Sender and receiver operate on different clock frequencies
Faster	Slower
There is no overhead of extra start and stop bits	Uses start and stop bits
There are no gaps between data - data flows as a continuous stream	There can be gaps between data
Uses constant time intervals	Uses random or irregular time intervals
Used in chat rooms and video conferencing	Used in emails

Rate of Transmission (BAUD)

- In parallel 1/0, data bits are transferred when a control signal enables the interfacing device; the transfer takes place in less than three T-states. However, in serial I/O, one bit is sent out at a time; therefore, how long the bit stays on or off is determined by the speed at which the bits are transmitted. Furthermore, the receiver should be set up to receive the bits at the same rate as the transmission; otherwise, the receiver may not be able to differentiate between two consecutive 0s or 1s.
- The rate at which the bits are transmitted-bits/second-is called a **baud**; technically, however, it is defined as the number of signal changes/second. Each piece of equipment has its own baud requirement.
- For example, **a teletype** (TTY) generally runs at 110 baud. However, in most terminals and printers, the baud is adjustable, typically, in the range of 50 to 9600 baud.
- Figure shows how the ASCII character I (49H) will be transmitted with 1200 baud with the framing information of one Start and two Stop bits.
- The transmission begins with an active low Start bit, followed by the LSB-bit D0 .

Rate of Transmission (BAUD)



Rate of Transmission (BAUD)

- The bit time-the delay between any two successive bits-is 0.83 ms; this is determined by the baud as follows:

  1200 bits= 1 second

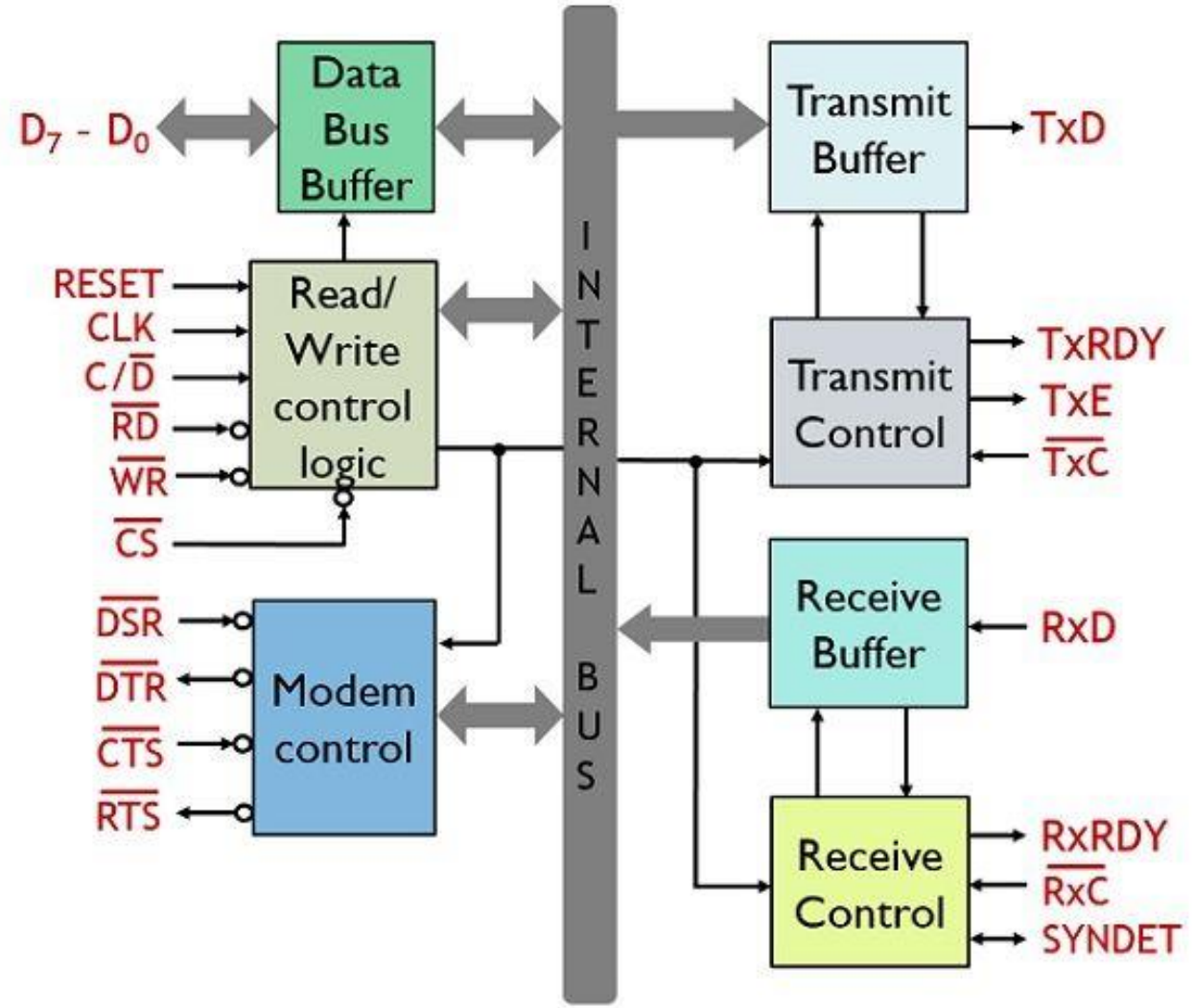
  For 1 bit= 1/1200 = 0.83 ms

- Therefore, to transmit one character, a parallel byte (49H) should be converted into a stream of 11 bits by adding framing bits (one Start and two Stop bits), and each bit must be transmitted at the interval of 0.83 ms. This can be implemented either through software or through programmable hardware chips. To receive a character in the serial mode, the process is reversed-one bit at a time is received and the bits are converted into a parallel word.

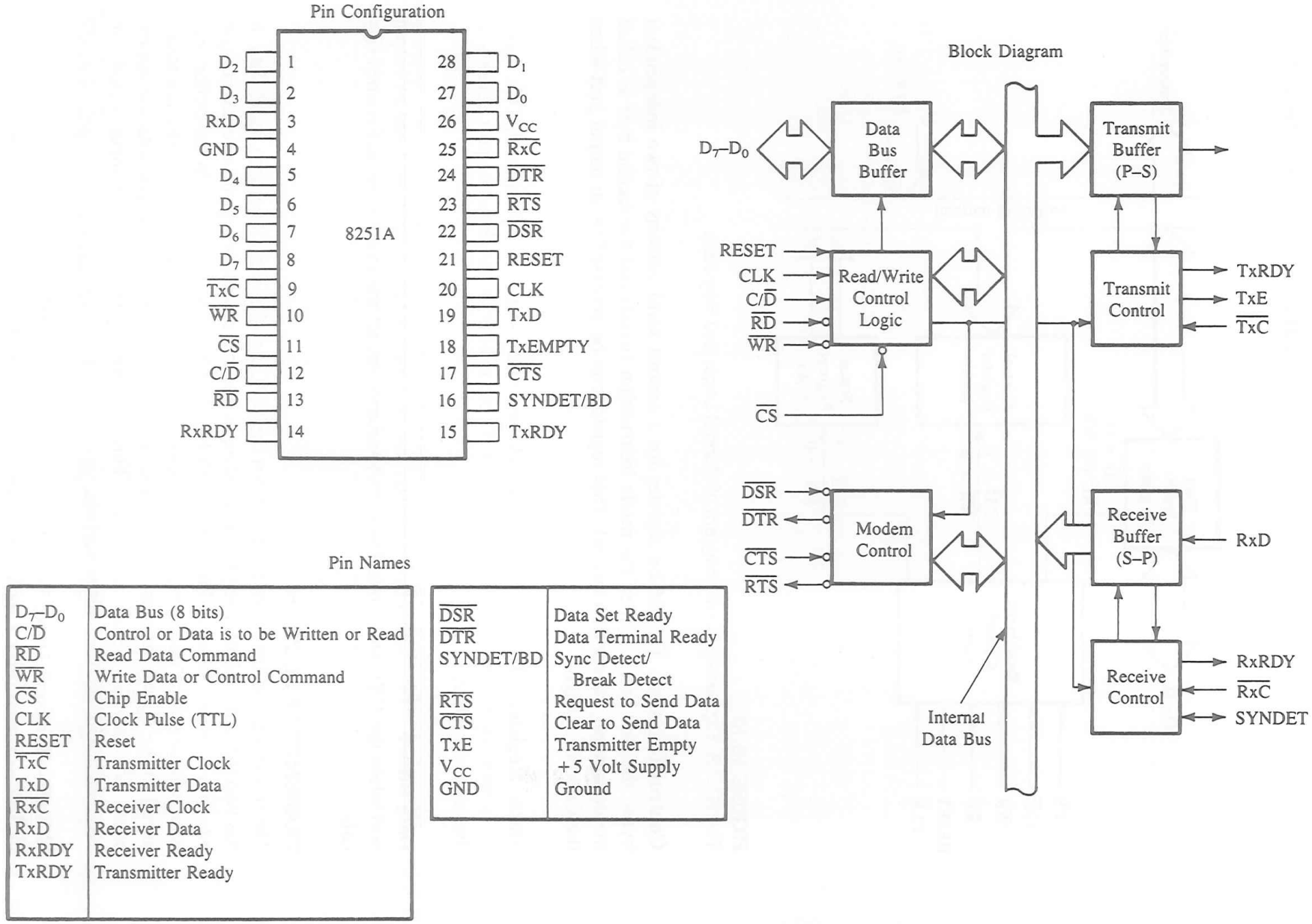
Introduction to 8251/8251A PCI (Programmable Communication Interface)

- The 8251A is a programmable serial communication interface chip designed for synchronous and asynchronous serial data communication.
- It supports the serial transmission of data.
- It is packed in a 28 pin DIP.
- It is also called USART (Universal Synchronous Asynchronous Receiver Transmitter).

Block Diagram of 8251A



Architecture of 8251 USART



Pin Names

D_7–D_0	Data Bus (8 bits)
C/$\overline{D}$	Control or Data is to be Written or Read
$\overline{RD}$	Read Data Command
$\overline{WR}$	Write Data or Control Command
$\overline{CS}$	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
$\overline{TxC}$	Transmitter Clock
TxD	Transmitter Data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready
TxRDY	Transmitter Ready

$\overline{DSR}$	Data Set Ready
$\overline{DTR}$	Data Terminal Ready
SYNDET/BD	Sync Detect/ Break Detect
$\overline{RTS}$	Request to Send Data
$\overline{CTS}$	Clear to Send Data
TxE	Transmitter Empty
V_{CC}	+5 Volt Supply
GND	Ground

FIGURE 16.12
The 8251A: Block Diagram, Pin Configuration, and Description

Block Diagram of 8251A

❖The functional block diagram of 8251A consists five sections. They are:

1. Read/Write control logic
2. Transmitter
3. Receiver
4. Data bus buffer
5. Modem control.

Components of 8251A

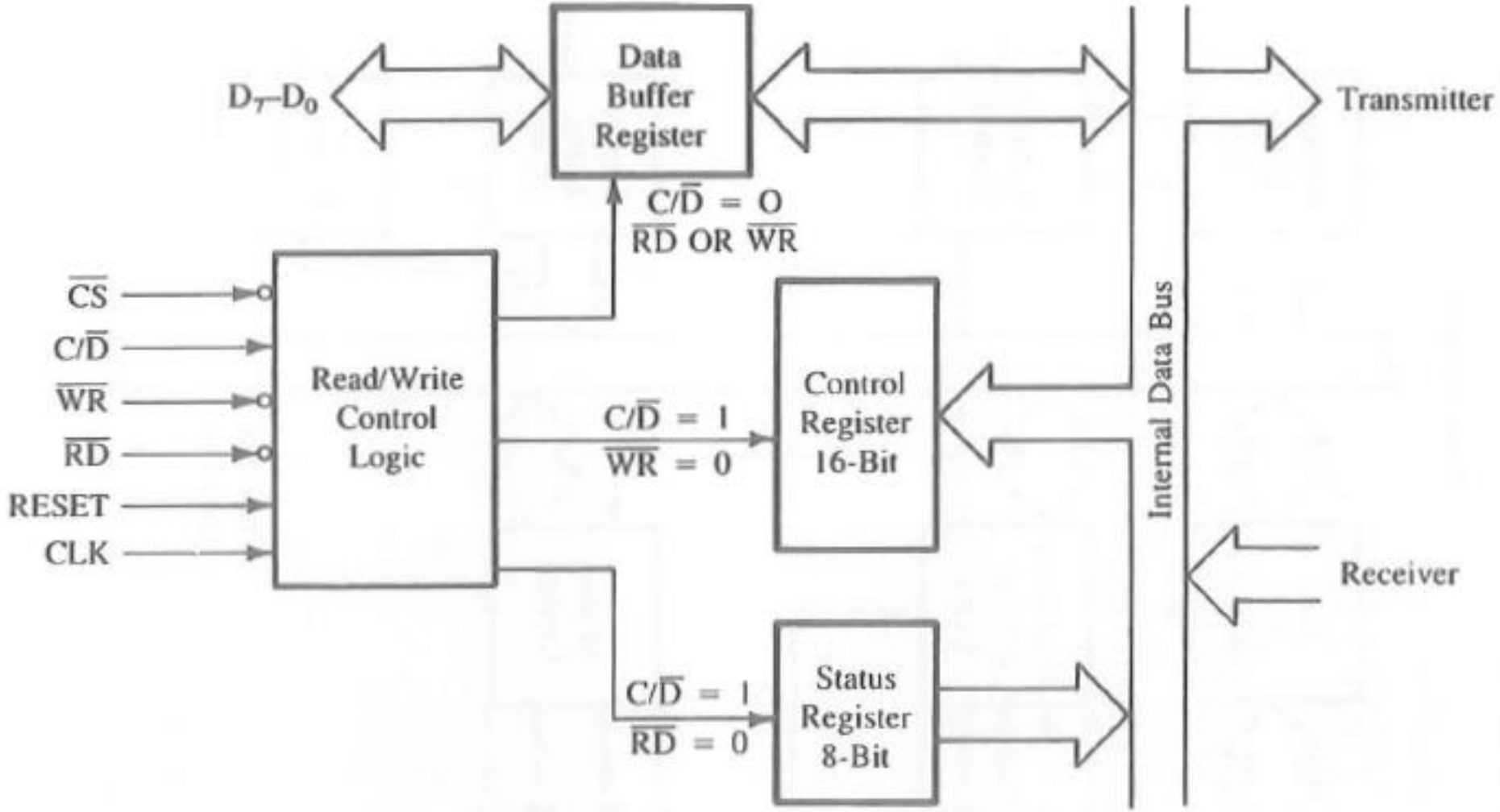


FIGURE 16.13
The 8251A: Expanded Block Diagram of Control Logic and Registers

Components of 8251A

- **Read/Write control logic:**
  - The Read/Write Control logic interfaces the 8251A with CPU, determines the functions of the 8251A according to the control word written into its control register.
  - It monitors the data flow.
  - This section has three registers and they are **control register**, **status register** and **data buffer**.
  - The active low signals **$\overline{\text{RD}}$**, **$\overline{\text{WR}}$**, **$\overline{\text{CS}}$** and **C/$\bar{\text{D}}$** are used for read/write operations with these three registers.
  - **$\overline{\text{CS}}$-Chip Select**: When this signal goes low, the 8251A is selected by the MPU for communication. This is usually connected to a decoded address bus.
  - **C/$\bar{\text{D}}$ - Control/Data:** When **C/$\bar{\text{D}}$** is high, the control register is selected for writing control word or reading status word.
  - When **C/$\bar{\text{D}}$** is low, the data buffer is selected for read/write operation.
  - When the **RESET (Reset)** is high, it forces 8251A into the idle mode.
  - The **CLK-clock** input is necessary for 8251A for communication with the microprocessor and this clock does not control either the serial transmission or the reception rate.

Components of 8251A

Summary of Control Signals for the 8251A

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	Function
0	1	1	0	MPU writes instructions in the control register
0	1	0	1	MPU reads status from the status register
0	0	1	0	MPU outputs data to the Data Buffer
0	0	0	1	MPU accepts data from the Data Buffer
1	X	X	X	USART is not selected

Components of 8251A

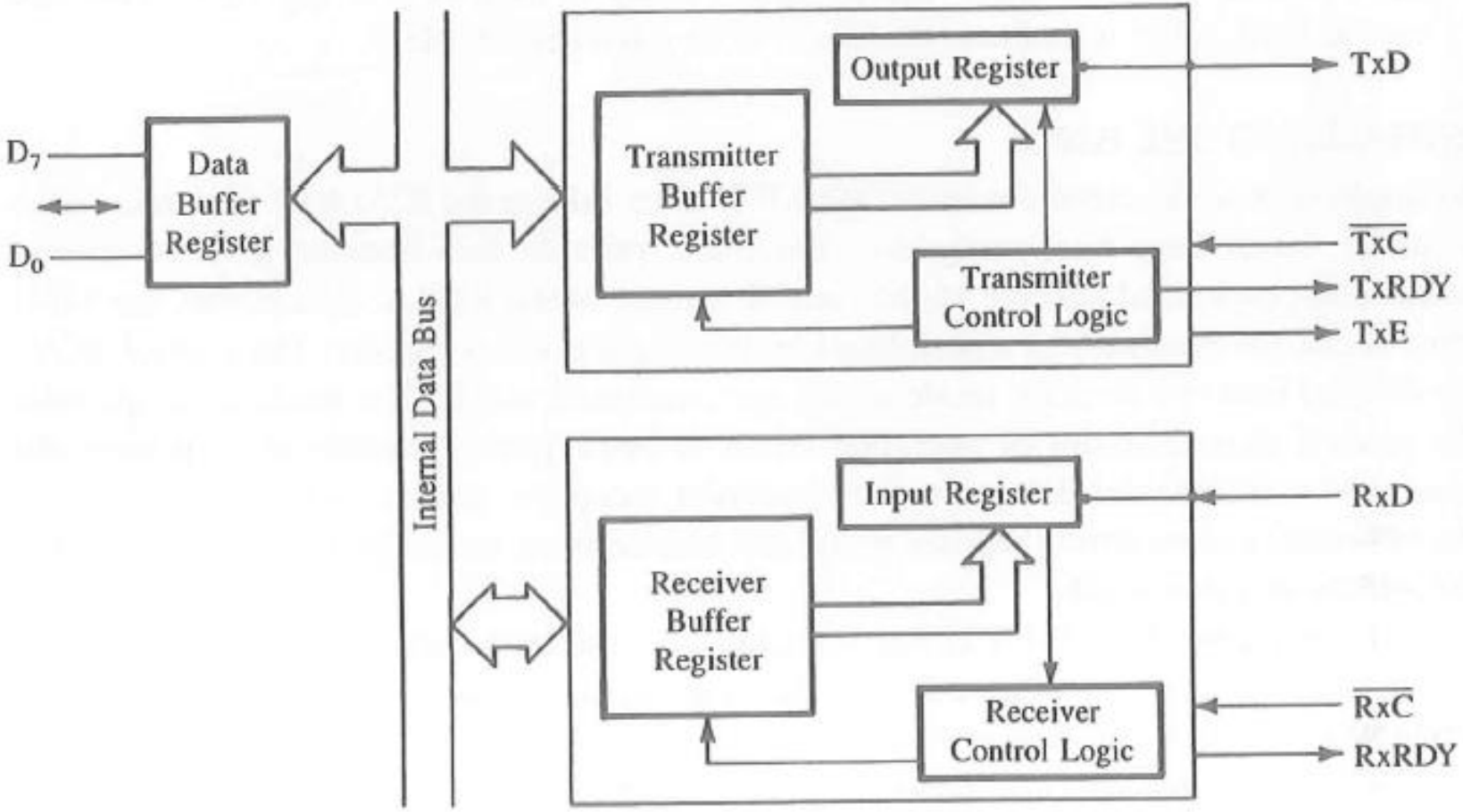


FIGURE 16.14
The 8251A: Expanded Block Diagram of Transmitter and Receiver Sections

Components of 8251A

- **Transmitter section:**
  - The transmitter section accepts parallel data from CPU and converts them into serial data.
  - The transmitter section is double buffered, i.e., it has **a buffer register** to hold an **8-bit parallel data** and another register called **output register** to convert the **parallel data** into **serial bits**.
  - This section transmits data on the **TxD** pin with the appropriate framing bits (Start and Stop).
  - Three output signals and one input signal are associated with the transmitter section.
  - When output register is empty, the data is transferred from buffer to output register. Now the processor can again load another data in buffer register.
  - If buffer register is empty, then TxRDY is goes to high. When it is high, it indicates that the buffer register is empty and the USART is ready to accept a byte. It can be used either to interrupt the MPU or to indicate the status. This signal is reset when a data byte is loaded into the buffer.
  - If output register is empty then TxEMPTY goes to high. Logic 1 on this line indicates that the output register is empty. This signal is reset when a byte is transferred from the buffer to the output registers.
  - The clock signal, TxC (low) controls the rate at which the bits are transmitted by the USART.
  - The clock frequency can be 1,16 or 64 times the baud rate.

Components of 8251A

❖ **Receiver Section:**

- ❑ The receiver accepts serial data on the RxD line from a peripheral and converts them into parallel data.
- ❑ The receiver section is **double buffered**, i.e., it has an **input register** to receive **serial data** and convert to parallel, and a **buffer register** to hold the **parallel data**.
- ❑ When the RxD line goes low, the control logic assumes it as a START bit, waits for half a bit time and samples the line again.
- ❑ If the line is still low, then the input register accepts the following bits, forms a character and loads it into the buffer register.
- ❑ The CPU reads the parallel data from the buffer register.
- ❑ When the input register loads a parallel data to buffer register, the RxRDY line goes high. It goes high when the USART has a character in the buffer register and is ready to transfer it to the MPU. This line can be used either to indicate the status or to interrupt the MPU.
- ❑ The clock signal RxC (low) controls the rate at which bits are received by the USART.
- ❑ During asynchronous mode, the signal SYNDET/BRKDET will indicate the break in the data transmission.
- ❑ During synchronous mode, the signal SYNDET/BRKDET will indicate the reception of synchronous character.

Components of 8251A

- **MODEM Control:**
  - The MODEM control unit allows to interface a MODEM to 8251A and to establish data communication through MODEM over telephone lines.
  - This unit takes care of handshake signals for MODEM interface.
  - It includes DTS, RTS, DTR and CTS.
  - These are all active low signals.

    DSR: Stands for data set ready and the signal is used to check whether the data set is ready or not when the processor is in the urge of communication.

    DTR: Implies data terminal ready. An active-low signal at this pin shows that the 8251 is now ready to accept the data from the processor.

    RTS: It stands for the request to send. A low signal shows an assertion for data transmission.

    CTS: Clear to send. When 8251 receives a low signal at this pin then it clears all the data present in the modem in order to allow further communication.

Example of MODEM

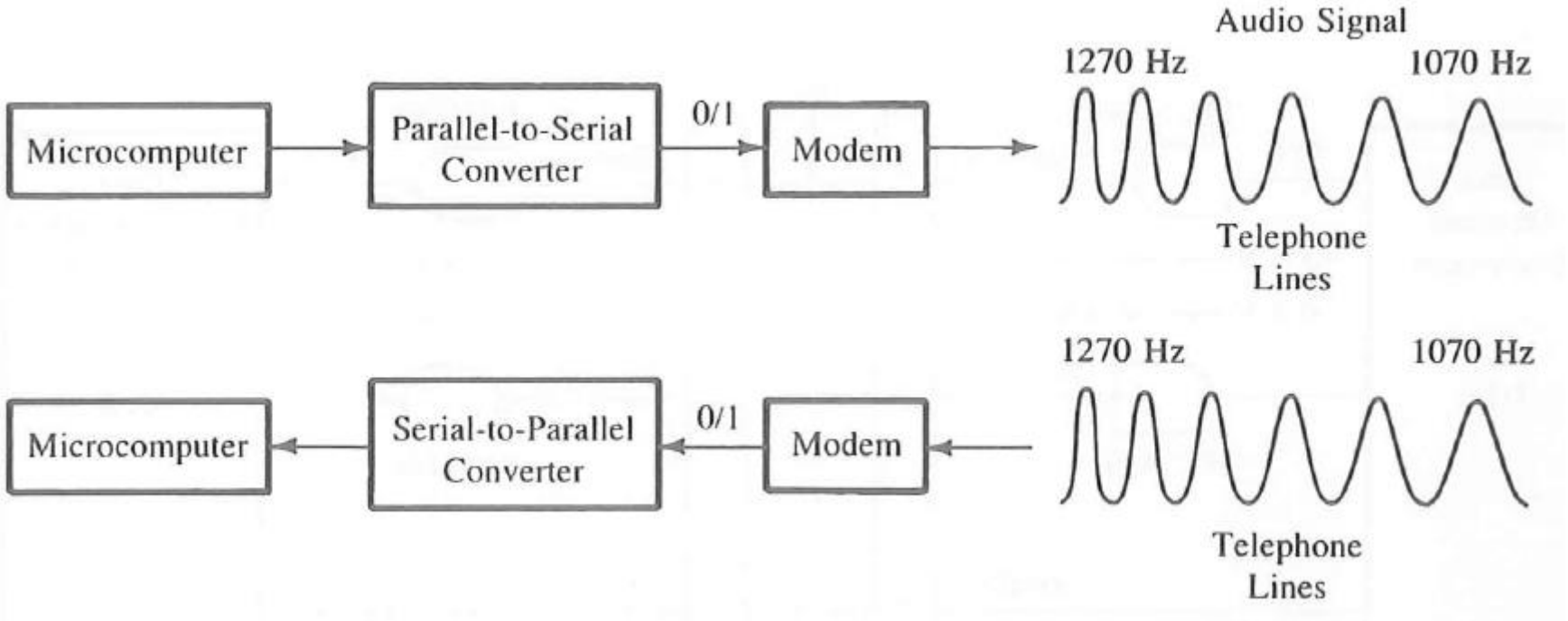


FIGURE: Communication over Telephone Lines Using Modems

Parallel Interface/Communication

- It is a method of conveying multiple binary digits (bits) simultaneously i.e. all bits of word are transferred at a time.
- The device which can handle data at higher speed cannot support with serial interface.
- N bits of data are handled simultaneously by the bus and the links to the device directly.
- Achieves faster communication but becomes expensive due to need of multiple wires.
- Hardware requirement is complex.
- E.g. 8255A PPI

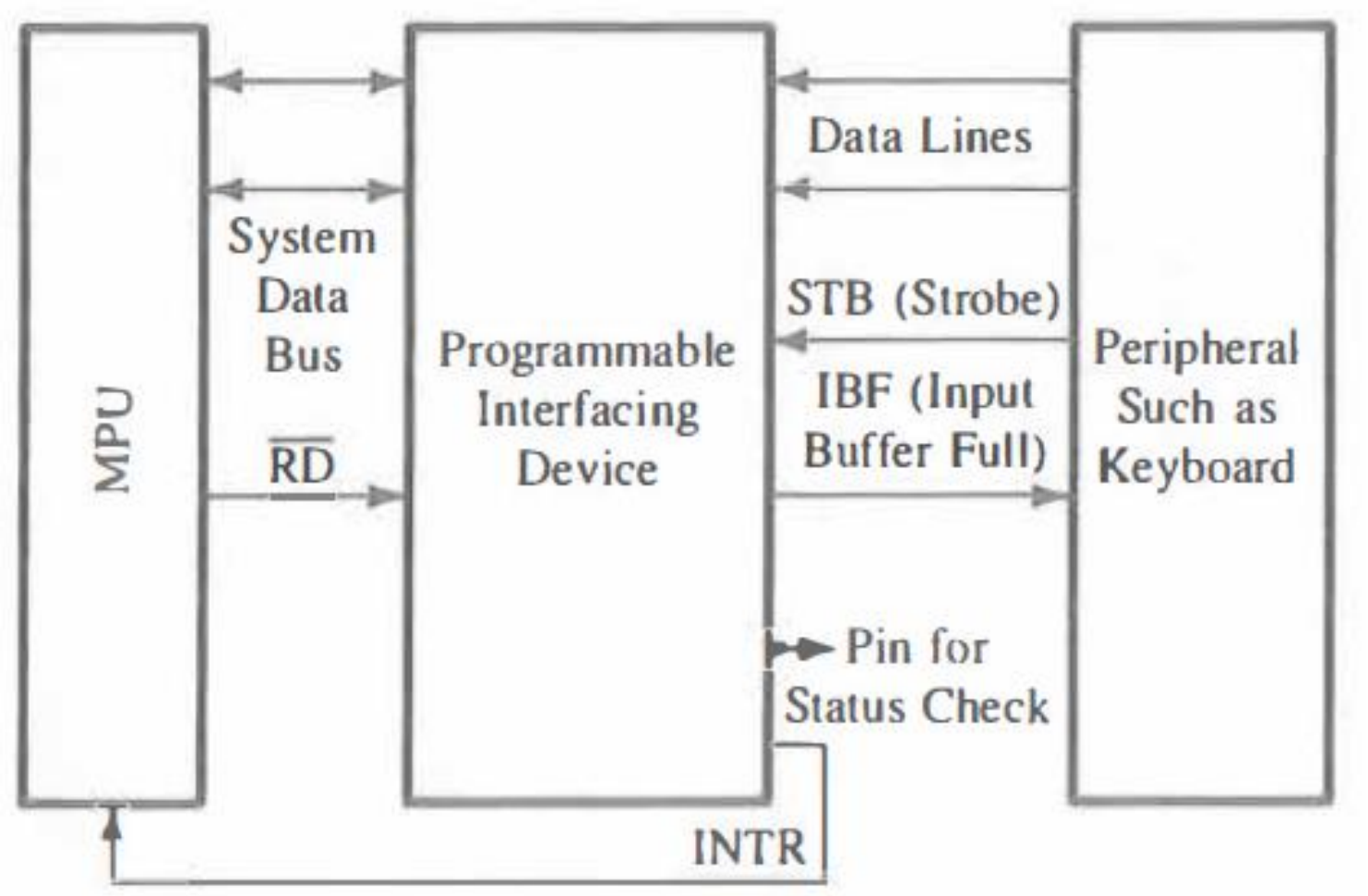
Programmable Devices with Handshake Signals

- The MPU and peripherals operate at different speeds; therefore, signals are exchanged prior to data transfer between the fast-responding MPU and slow-responding peripherals such as printers and data converters. These signals are called **handshake signals**.
- A handshake signal refers to a set of control signals used to coordinate and manage data communication between different devices or components. Handshake signals ensure that data transfers occur accurately and reliably by signaling when data can be sent or received, acknowledging the receipt of data, and indicating readiness for data exchange.
- The exchange of handshake signals prevents the MPU from writing over the previous data before a peripheral has had a chance to accept it or from reading the same data before a peripheral has had time to send the next data byte.
- These signals are generally provided by programmable devices. Figure 14.4(a) shows a programmable device in the input mode, with two handshake signals (STB and IBF) and one interrupt signal (INTR).
- Now the MPU has two ways of finding out whether a peripheral is ready: either by checking the status of a handshake signal or through the interrupt technique.

Handshake Signals

- Handshake signals ACK and STB are input signals to the device and perform similar functions, although they are called by different names.
- Handshake signals OBF and IBF are output signals from the device and perform similar functions (Buffer Full).

Data Input With Handshake

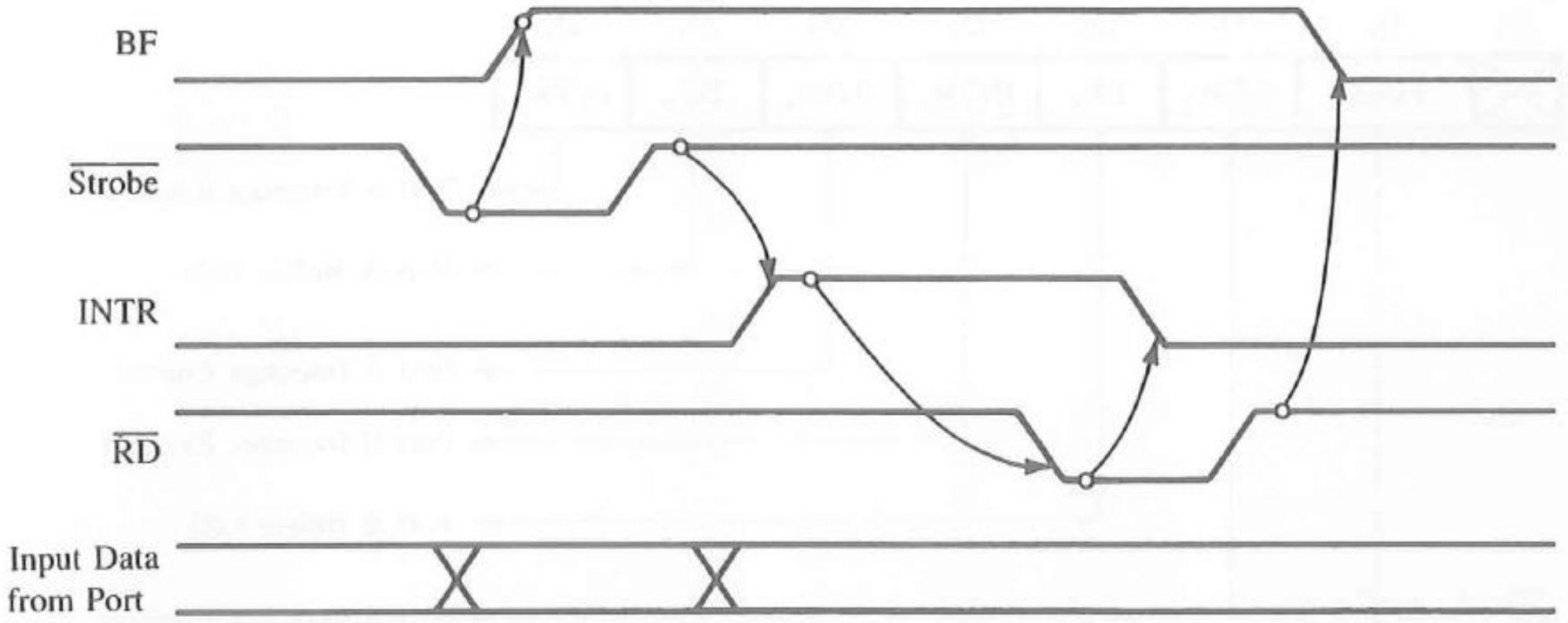


Data Input With Handshake

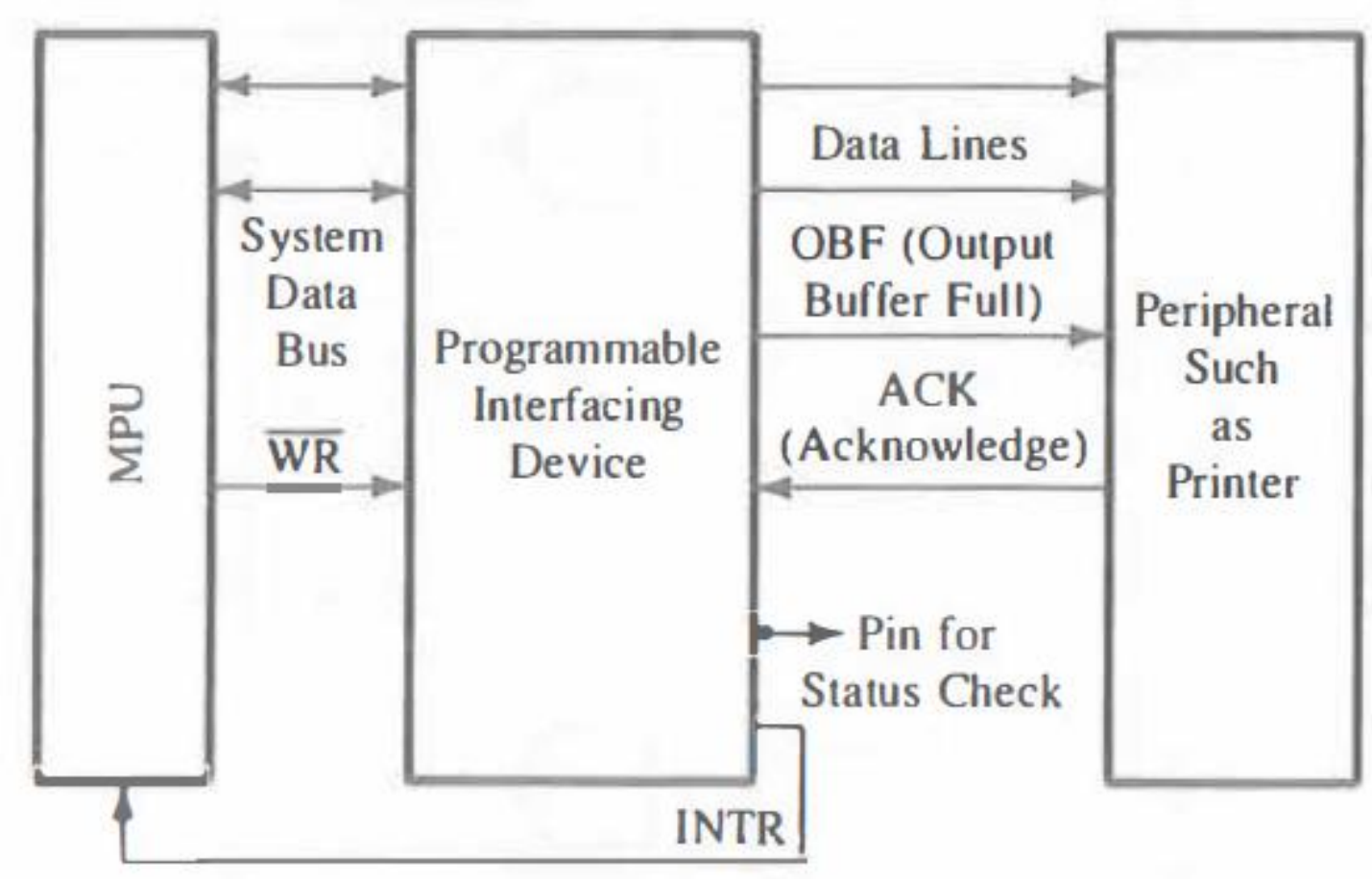
❖The steps in data input from a peripheral such as a keyboard are as follows:

1) A peripheral strobes or places a data byte in the input port and informs the interfacing device by sending handshake signal STB (Strobe).
2) The device informs the peripheral that its input port is full-do not send the next byte until this one has been read. This message is conveyed to the peripheral by sending handshake signal IBF (Input Buffer Full).
3) The MPU keeps checking the status until a byte is available. Or the interfacing device informs the MPU, by sending an interrupt, that it has a byte to be read.
4) The MPU reads the byte by sending control signal $\overline{RD}$.

Input Mode Handshake Timing Waveform



Data Output With Handshake

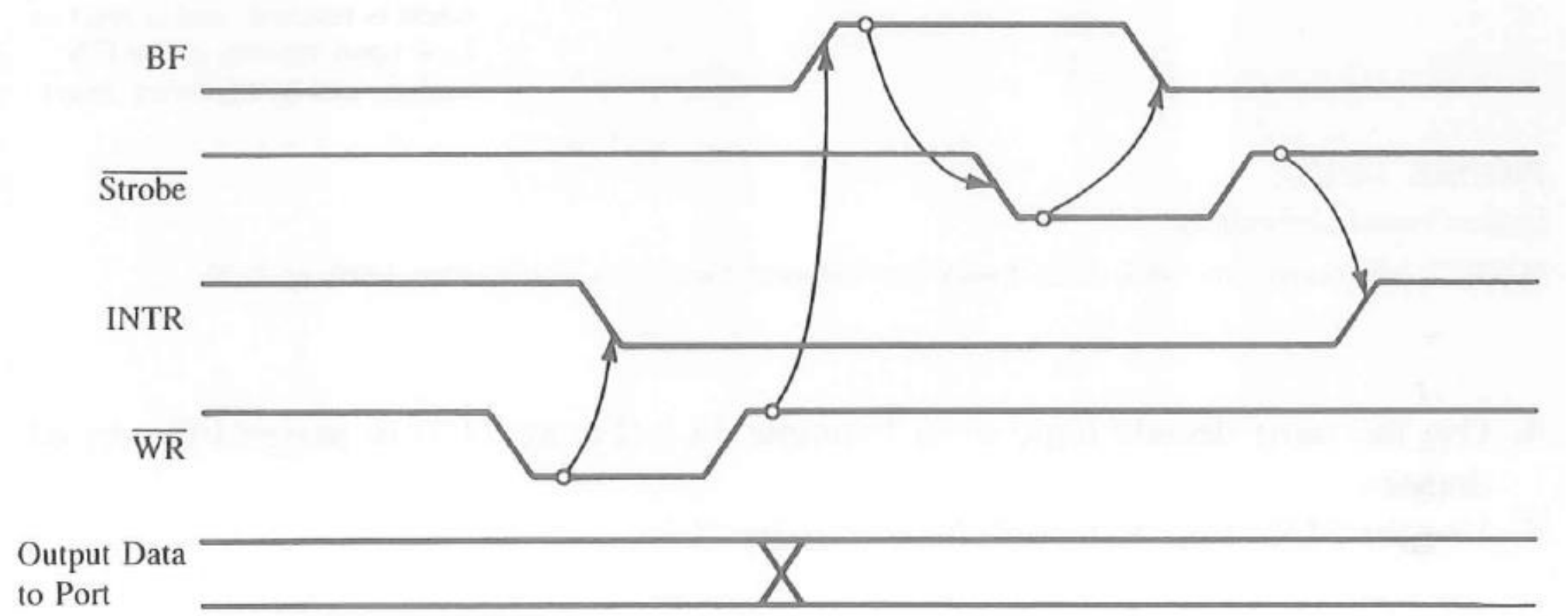


Data Output With Handshake

❖Figure 14.4(b) shows the programmable device in the output mode using the same two handshake signals, except that they are labelled differently. The steps in data output to a peripheral such as a printer are as follows:

1) The MPU writes a byte into the output port of the programmable device by sending control signal $\overline{WR}$.
2) The device informs the peripheral, by sending handshake signal OBF (Output Buffer Full), that a byte is on the way.
3) The peripheral acknowledges the byte by sending back the ACK (Acknowledge) signal to the device.
4) The device interrupts the MPU to ask for the next byte, or the MPU finds out that the byte has been acknowledged through the status check.

Output Mode Handshake Timing Waveform



Data Transfer Modes of Parallel Interfacing

- Simple I/O
- Strobe I/O,
- Single handshake I/O
- Double handshake I/O

Simple I/O

- When you need to get digital data from simple switch, such as thermostat, into microprocessor, all you have to do is connect the switch to an I/O port line and read the port.
- Likewise, when you need to output data to simple display device, such as LED, all you have to do is connect the input of the LED buffer on an output port pin and output the logical level required to turn on the light.

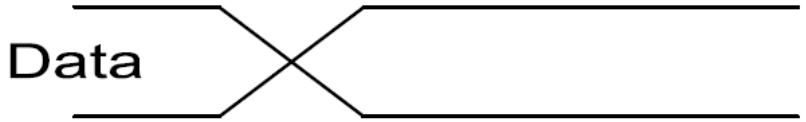


Fig: Simple I /O

- The timing waveform represents the situation.
- The crossed lines on the waveform represent the time at which a new data byte becomes valid on the output lines of the port.
- The absence of other waveforms indicates that this output operation is not directly dependant on any other signal.

Strobe I/O

- In many applications, valid data is present on an external device only at a certain time, so it must be read in at that time.
- E.g. the ASCII-encoded keyboard. When a key is pressed, circuitry on the keyboard sends out the ASCII code for the pressed key on eight parallel data lines, and then sends out a strobe signal on another line to indicate that valid data is present on the eight data lines.

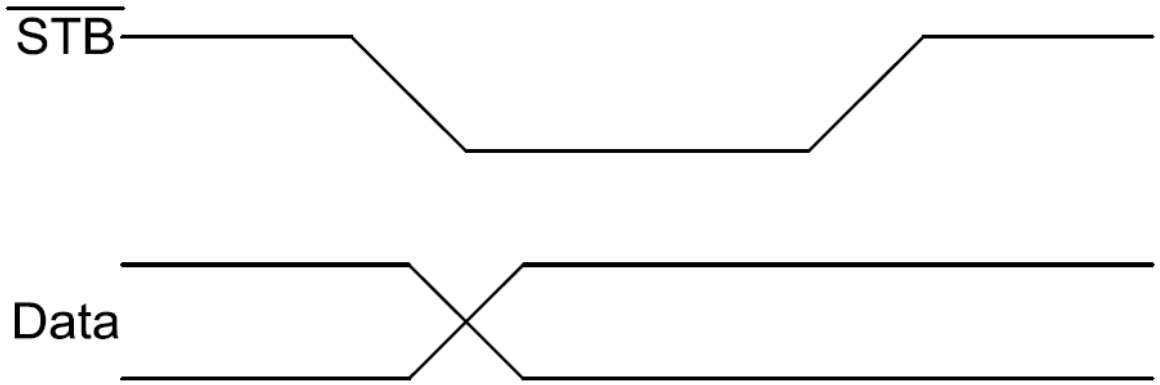


Fig: Simple Strobe I/O

Strobe I/O

- This timing waveform represents strobe I/O.
- The sending device, such as a keyboard, outputs a parallel data on the data lines, and then outputs an $\overline{STB}$ signal to let you know that valid data is present.
- For low rates of data transfer, such as from a keyboard to a MP, a simple strobe transfer works well.
- However, for higher speed data transfer, this method does not work because there is no signal which tells the sending device when it is safe to send the next data byte.
- In other words, the sending system might send data bytes faster than the receiving system could read them.
- To prevent this problem, a handshake data transfer scheme is used.

Single handshake I/O

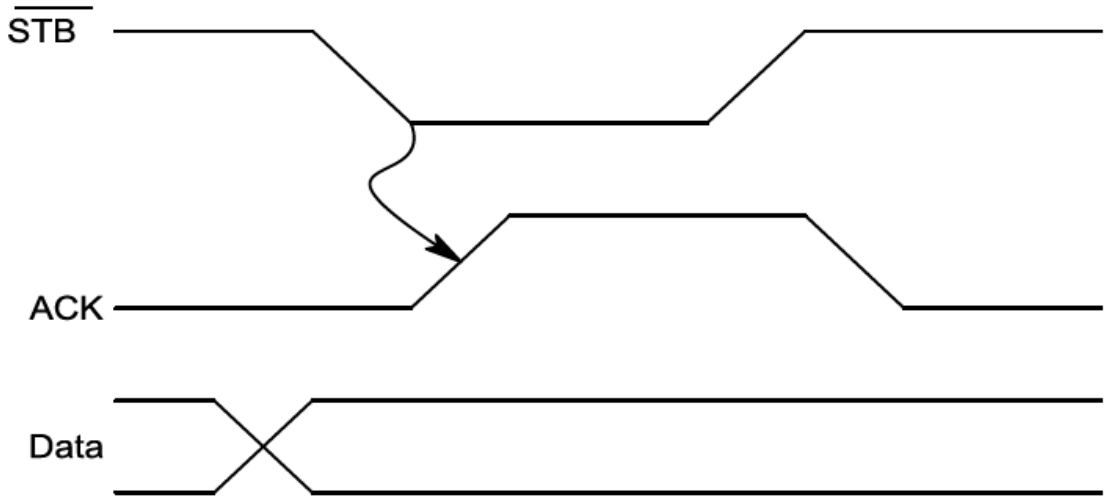


Fig: Single Handshaking

- It shows the timing waveform for a handshake data transfer from a peripheral device to a MP.
- The peripheral outputs some parallel data and sends an $\overline{STB}$ signal to the MP.
- The MP detects the asserted $\overline{STB}$ signal on a polled or interrupts basis and reads in the bytes of data.
- Then, the MP sends ACK (acknowledge) signal to the peripheral to indicate that the data has been read and that the peripheral can send next byte of data.
- The point of this method is that the sending device or system is designed so that it does not send the next byte until the receiving device or system indicates with an ACK signal that it is ready to receive the next byte.

Double handshake I/O

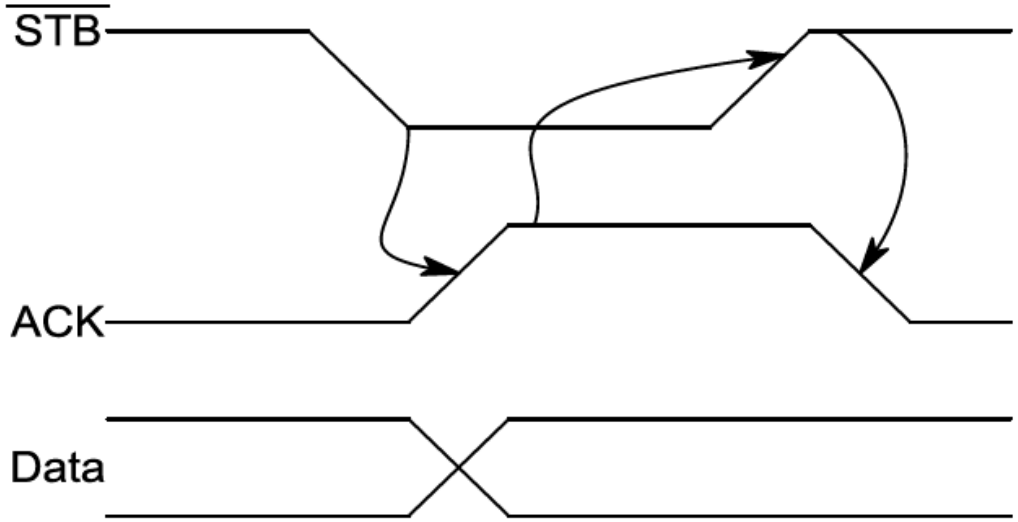


Fig: Double Handshaking

- For data transfer where even more coordination is required between the sending system and the receiving system, a double handshake is used.
- The sending (peripheral) device asserts its $\overline{STB}$ line low to ask the receiving device whether it is ready or not for data reception.
- The receiving system raises its ACK line high to indicate that it is ready.
- The peripheral device then sends the byte of data and raises its $\overline{STB}$ line high to assure that the valid data is available for the receiving device (MP).
- When MP reads the data, it drops its ACK line low to indicate that it has received the data and requests the sending system to send next byte of data.
- The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt.

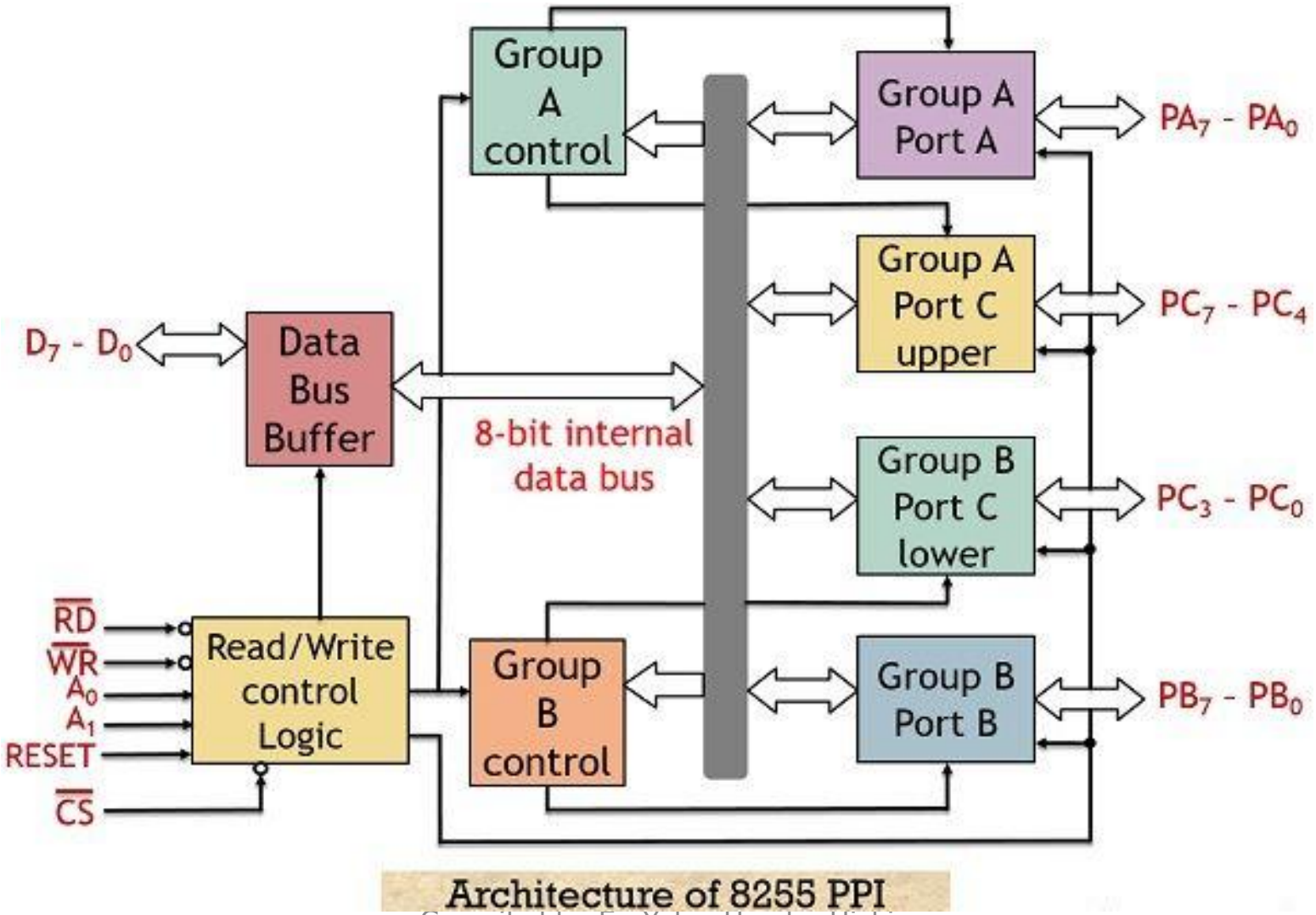
Introduction to 8255A PPI (Programmable Peripheral Interface)

- The INTEL 8255 is a device used to parallel data transfer between processor and slow peripheral devices like ADC, DAC, keyboard, 7-segment display, LCD, etc.
- The 8255 has three ports: **Port-A, Port-B and Port-C**.
- **Port-A** can be programmed to work in any one of the three operating modes mode-0, mode-1 and mode-2 as input or output port.
- **Port-B** can be programmed to work either in mode-0 or mode-1 as input or output port.
- **Port-C** (8-pins) has different assignments depending on the mode of port-A and port-B.
- If port-A and B are programmed in mode-0, then the port-C can perform any one of the following functions.
  - As 8-bit parallel port in mode-0 for input or output.
  - As two numbers of 4-bit parallel ports in mode-0 for input or output.
  - The individual pins of port-C can be set or reset for various control applications.

Introduction to 8255A PPI (Programmable Peripheral Interface)

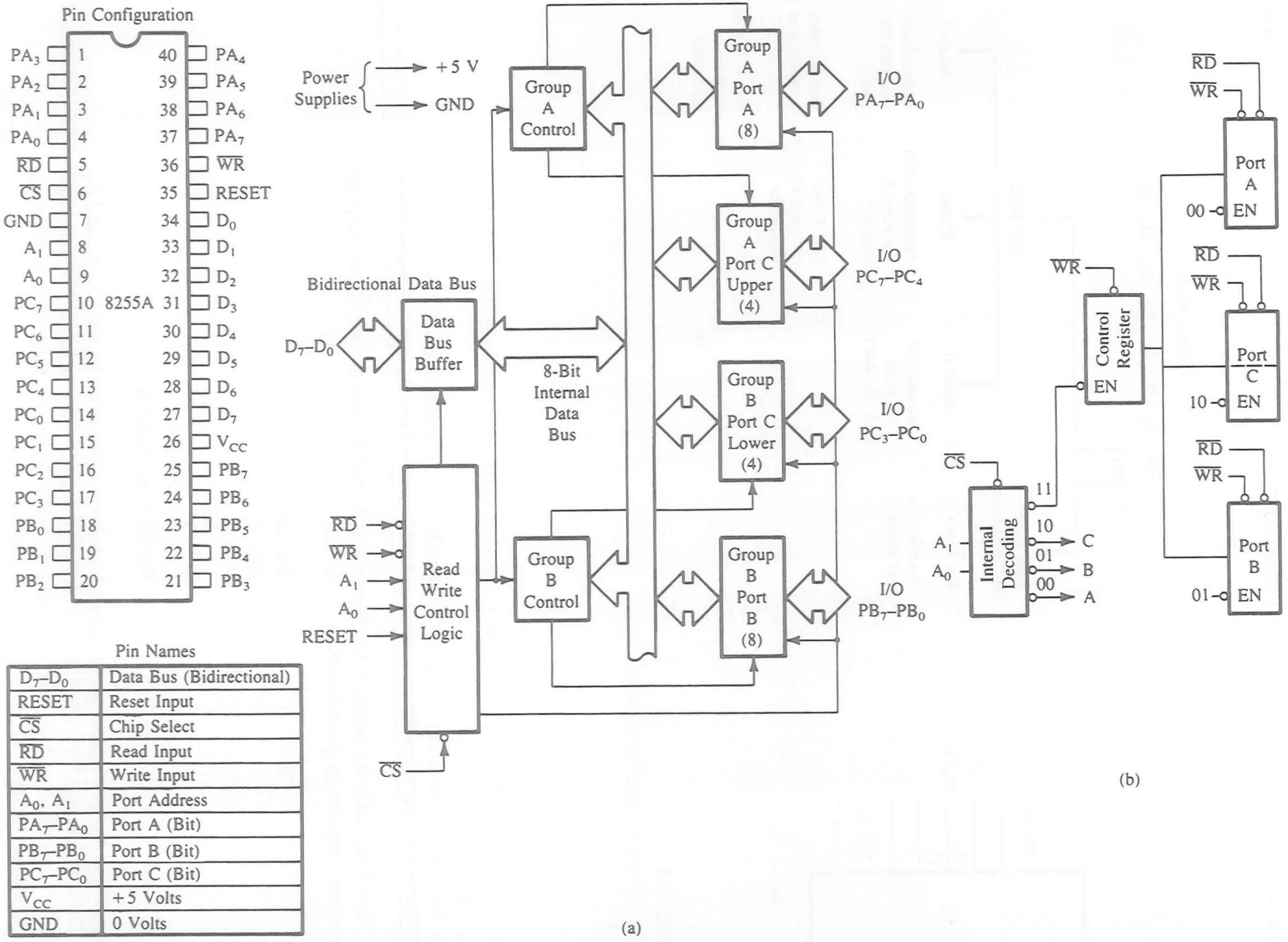
- If port-A is programmed in mode- 1/mode-2 and port-B is programmed in mode-1 then some of the pins of port-C are used for handshake signals and the remaining pins can be used as input/ output lines or individually set/reset for control application.
- 8255A is widely used programmable parallel I/O device. It can be programmed to transfer data under various conditions, from simple I/O to interrupt I/O.
- It is flexible, versatile and economical (when multiple I/O ports are required), but somewhat complex. It is an important general purpose I/O device that can be used with almost any microprocessor.
- The 8255A has **24 input output** pins that can be grouped primarily in **two** 8 bits parallel ports: **A** and **B**, with the remaining 8 bits as **port C**. The 8 bits of port C can be used as individual bits or be grouped in two four bits ports: C_{upper} (C_U) and C_{lower} (C_L) as shown in block diagram.
- The functions of these ports are defined by writing a control word in the control registers.

Block Diagram of 8255A



Architecture of 8255 PPI

Block Diagram of 8255A



Pin Names

D_7–D_0	Data Bus (Bidirectional)
RESET	Reset Input
$\overline{CS}$	Chip Select
$\overline{RD}$	Read Input
$\overline{WR}$	Write Input
A_0, A_1	Port Address
PA_7–PA_0	Port A (Bit)
PB_7–PB_0	Port B (Bit)
PC_7–PC_0	Port C (Bit)
V_{CC}	+5 Volts
GND	0 Volts

FIGURE 15.2
8255A Block Diagram (a) and an Expanded Version of the Control Logic and I/O Ports (b)

Components of 8255A

- **Data Bus Buffer**
  - It is a tri-state 8-bit buffer, which is used to interface the microprocessor to the system data bus. Data is transmitted or received by the buffer as per the instructions by the CPU. Control words and status information is also transferred using this bus.
- **Read/Write Control Logic**
  - This block is responsible for controlling the internal/external transfer of data/control/status word. It accepts the input from the CPU address and control buses, and in turn issues command to both the control groups.
  - **$\overline{WR}$**

    It stands for write. This control signal enables the write operation. When this signal goes low, the microprocessor writes into a selected I/O port or control register.
  - **$\overline{RD}$**

    It stands for Read. This control signal enables the Read operation. When the signal is low, the microprocessor reads the data from the selected I/O port of the 8255.
  - **RESET**

    This is an active high signal. It clears the control register and sets all ports in the input mode.

Components of 8255A

- **Read/Write Control Logic**
  - **$\overline{CS}$**

    It stands for Chip Select. A LOW on this input selects the chip and enables the communication between the 8255A and the CPU. It is connected to the decoded address, and A0 & A1 are connected to the microprocessor address lines.

    Their result depends on the following conditions:

$\overline{CS}$	A_1	A_0	Result
0	0	0	PORT A
0	0	1	PORT B
0	1	0	PORT C
0	1	1	Control Register
1	X	X	No Selection

Components of 8255A

- **Read/Write Control Logic**
  - **A0 and A1**

    These input signals work with **$\overline{RD}$**, **$\overline{WR}$**, and one of the control signal. Following is the table showing their various signals with their result.

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Result
0	0	0	1	0	**Input Operation** PORT A → Data Bus
0	1	0	1	0	PORT B → Data Bus
1	0	0	1	0	PORT C → Data Bus
0	0	1	0	0	**Output Operation** Data Bus → PORT A
0	1	1	0	0	Data Bus → PORT B
1	0	1	0	0	Data Bus → PORT C
1	1	1	0	0	Data Bus → Control Register

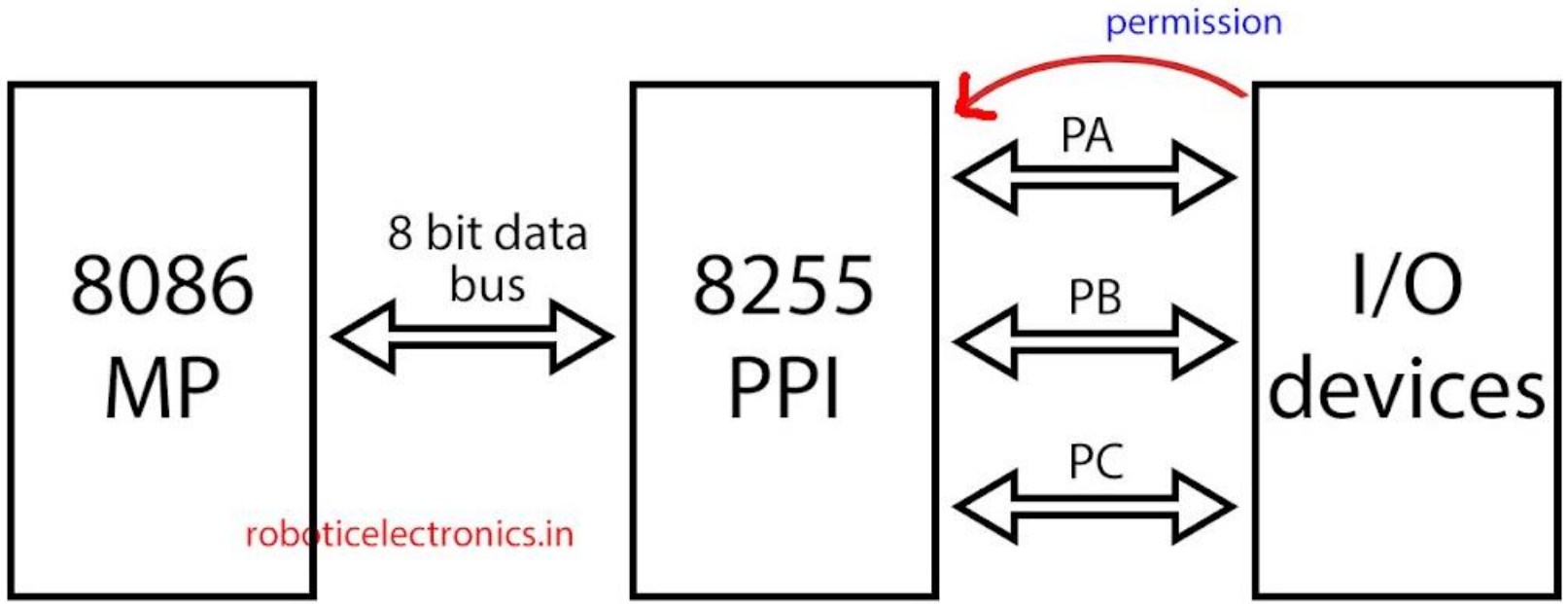
Components of 8255A

- **Group A and Group B Controls**
  - The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255. Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.
- **Ports A, B, and C**
  - The 8255 contains three 8-bit ports (A, B, and C). All can be configured to a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255.
- **Port A** One 8-bit data output latch/buffer and one 8-bit data input latch. Both "pull-up" and "pull-down" bus-hold devices are present on Port A.
- **Port B** One 8-bit data input/output latch/buffer and one 8-bit data input buffer.
- **Port C** One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch

8255 Working

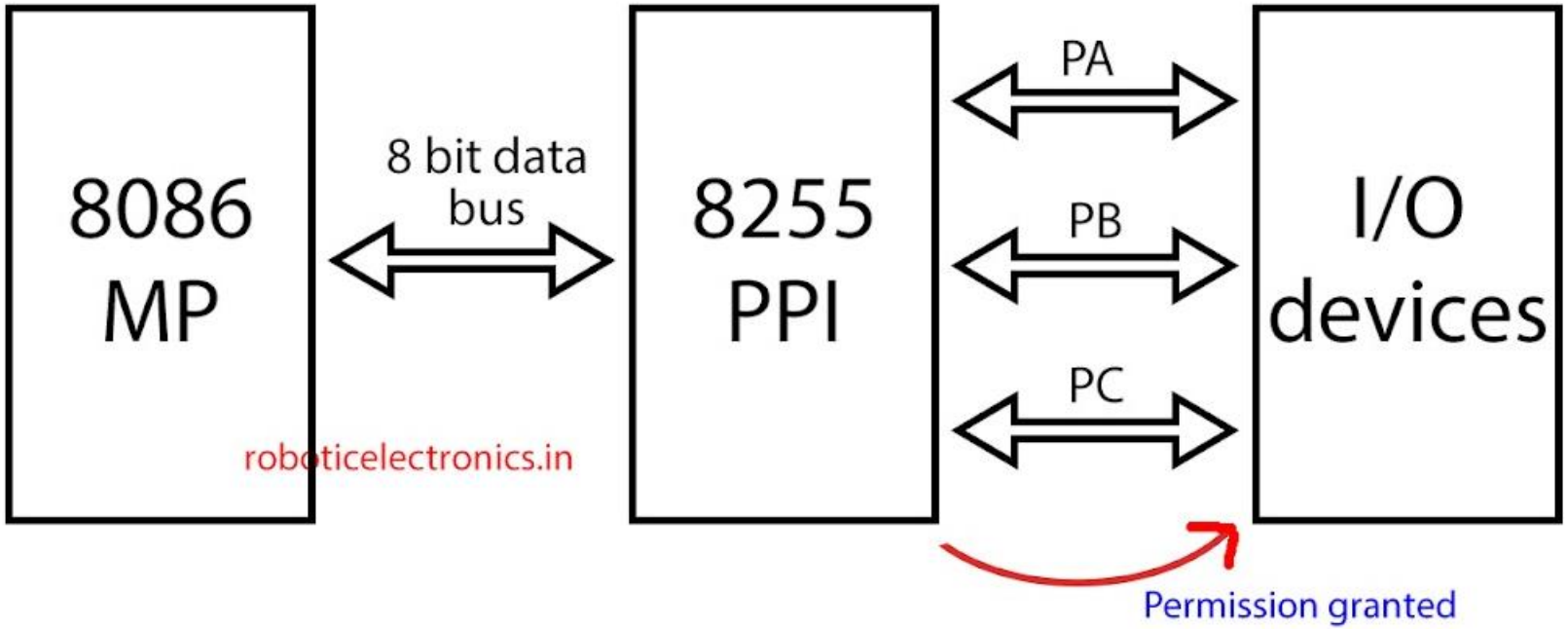
❖ As already mentioned 8255 has 3 data ports namely Port – A (PA) and PB, PC.

1. Assuming the connected device to be an input device. Initially, the Input device seeks for permission from PPI so that it can send data.



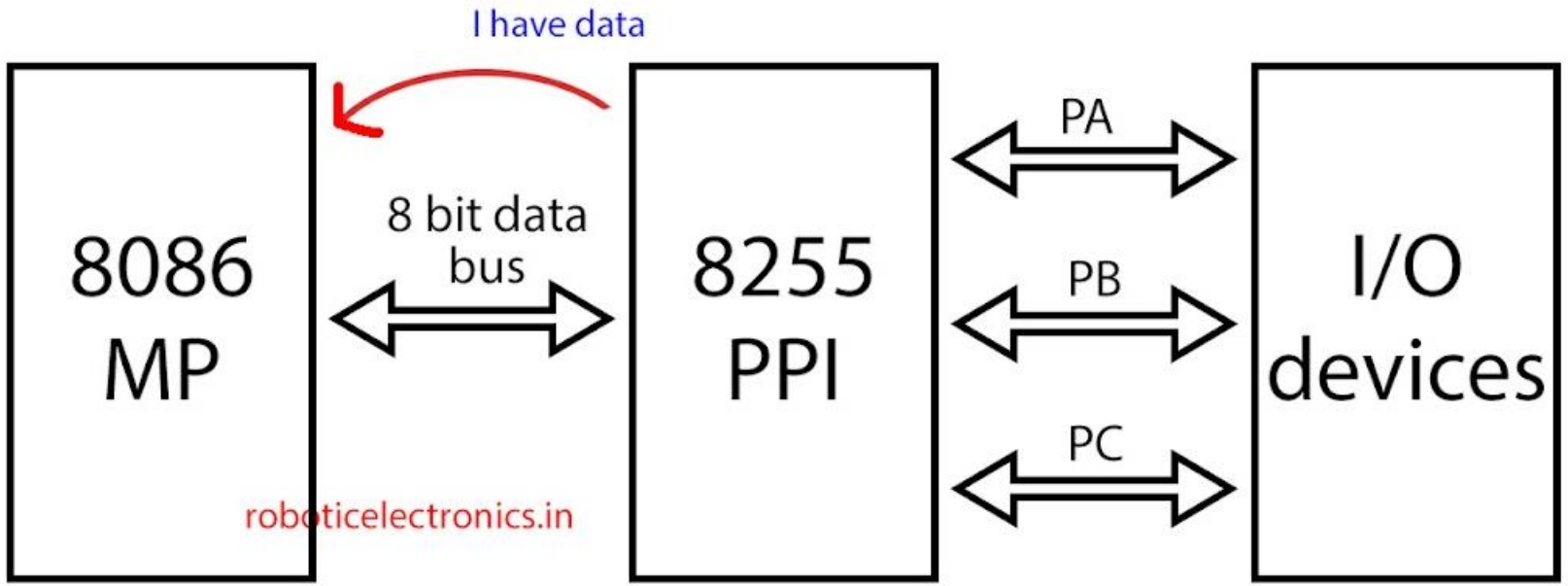
8255 Working

2. PPI permits Input devices to send data, only when there is no left data in 8255 which should be sent to the 8086 processor. If there is some previous data left in 8255, which is still not sent to 8086, then it doesn't permit Input device.



8255 Working

3. Once 8255 permits input device, data is received and stored in temporary registers in 8255. Once 8255 holds some data, which should be sent to 8086, then it send signal to 8086.



8255 Working

4. Whenever 8086 is free to receive the data, then 8086 sends back a signal , after that data transmission happens between 8255 and 8086. If 8086 do not becomes free up to long time, which means 8255 has some value in it which is still not sent to 8086, so 8255 does not permit the Input device to send any data because the existing data will be overwritten.

❖ All the signal in the above diagrams represented using red curved arrow are known as **handshake signals.** This process of data transmission is known as **handshaking.**

Control Word of 8255

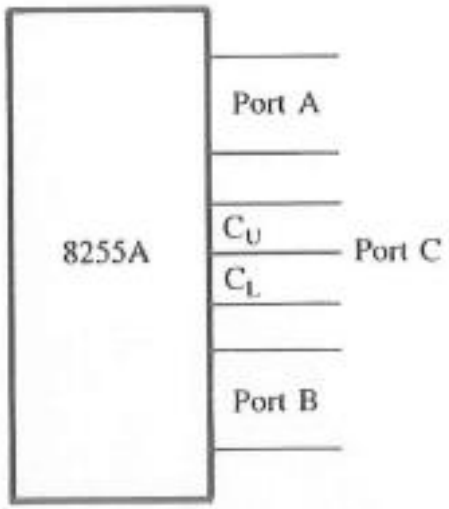
- The contents of **Control register**, called the **control word**, specify an I/O function for each port. This register can be accessed to write a control word when A_0 and A_1 are at logic 1. The register is not accessible for a Read operation.
- Bit D_7 of the control register specifies either the **I/O function** or the **Bit Set/Reset function**.
- If bit $\mathbf{D_7 = 1}$, bits D_6-D_0 determine **I/O functions** in various modes, as shown in figure.
- If bit $\mathbf{D_7 = 0}$, **Port C** operates in the **Bit Set/Reset (BSR) mode**. The BSR control word does not affect the functions of ports A and B.
- To communicate with peripherals through the 8255A, three steps are necessary:
  1) Determine the addresses of ports A, B, and C and of the control register according to the Chip Select logic and address lines A_0 and A_1.
  2) Write a control word in the control register.
  3) Write I/O instructions to communicate with peripherals through ports A, B, and C.

8255A Modes of Operation

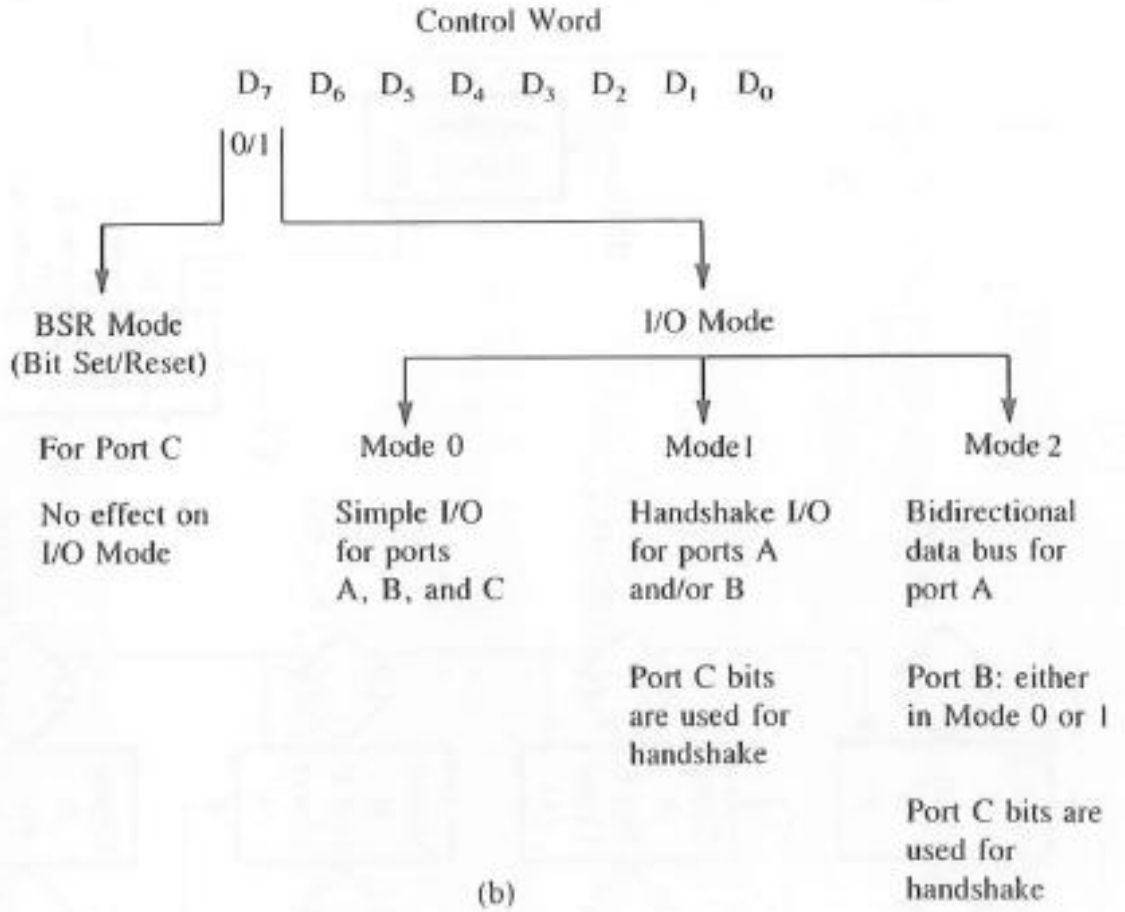
❖There are two different modes of 8255. These modes are:

1) **Bit Set Reset (BSR) Mode**
2) **Input/ Output Mode**

8255A Modes of Operation



(a)

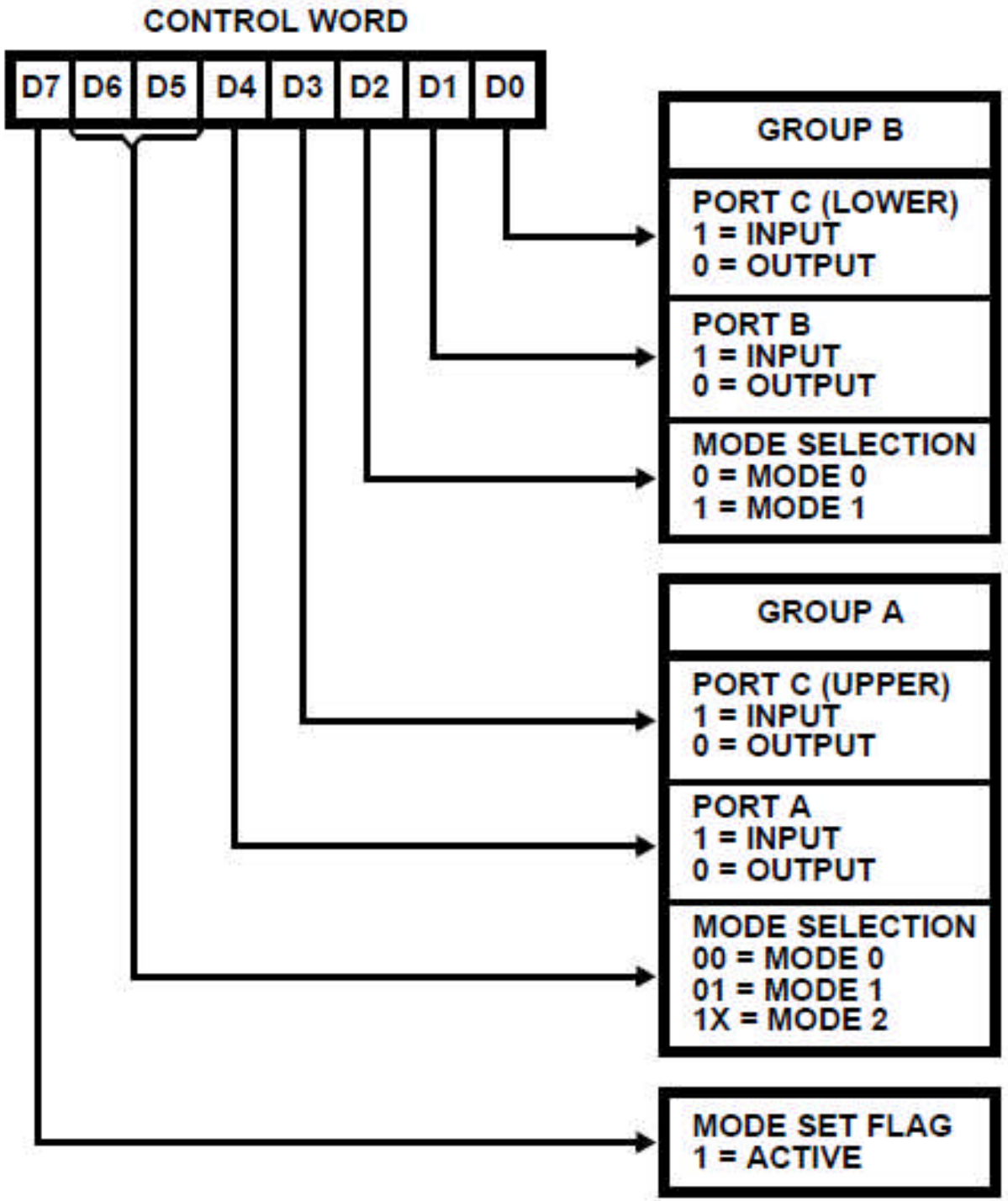


(b)

FIGURE 15.1
8255A I/O Ports (a) and Their Modes (b)

Control Word of 8255 - I/O Mode (I/O Command)

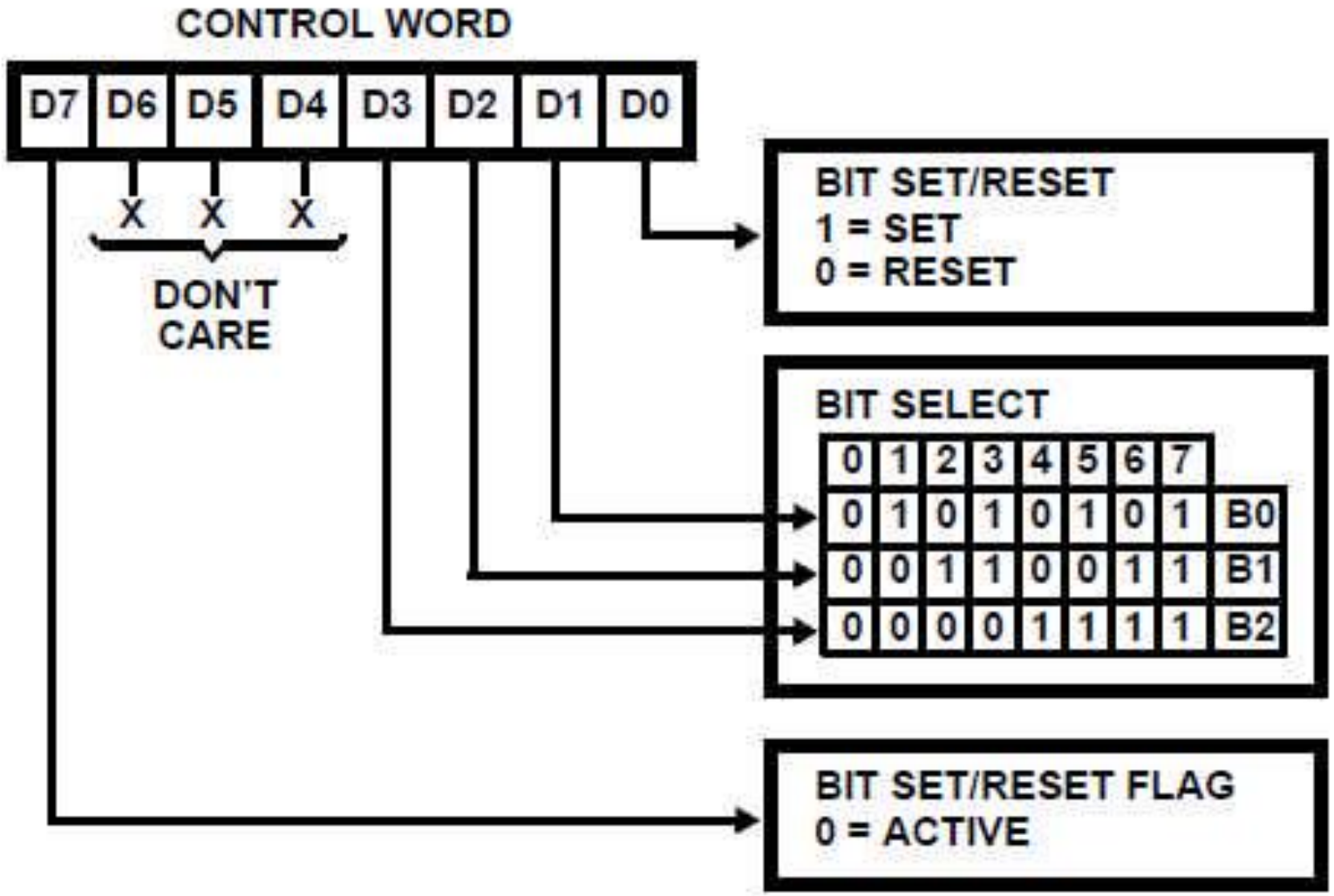
- To do 8-bit data transfer using the Ports A, B or C, 8255 needs to be in the IO mode. The bit pattern for the control word in the IO mode is as follows:


CONTROL WORD
D7
D6
D5
D4
D3
D2
D1
D0
GROUP B
PORT C (LOWER)
1 = INPUT
0 = OUTPUT
PORT B
1 = INPUT
0 = OUTPUT
MODE SELECTION
0 = MODE 0
1 = MODE 1
GROUP A
PORT C (UPPER)
1 = INPUT
0 = OUTPUT
PORT A
1 = INPUT
0 = OUTPUT
MODE SELECTION
00 = MODE 0
01 = MODE 1
1X = MODE 2
MODE SET FLAG
1 = ACTIVE

Control Word of 8255 – BSR Mode (BSR Command)

- The BSR Mode is used **ONLY for Port C.**
- In this Mode the **individual bits** of Port C can be **set or reset**.
- This is very useful as it provides **8 individually controllable lines** which can be used while interfacing with devices like an **A to D Converter** or a 7-segment display etc.
- The individual bit is **selected** and Set/reset through the **control word**.
- Since the D7 bit of the Control Word is 0, the BSR operation **will not affect the I/O operations** of 8255.

Control Word of 8255 - BSR Mode (BSR Command)



Data Transfer Modes Of 8255

- Simple Bi-Directional I/O mode
- Handshake I/O mode
- Bi-Directional Handshake I/O mode

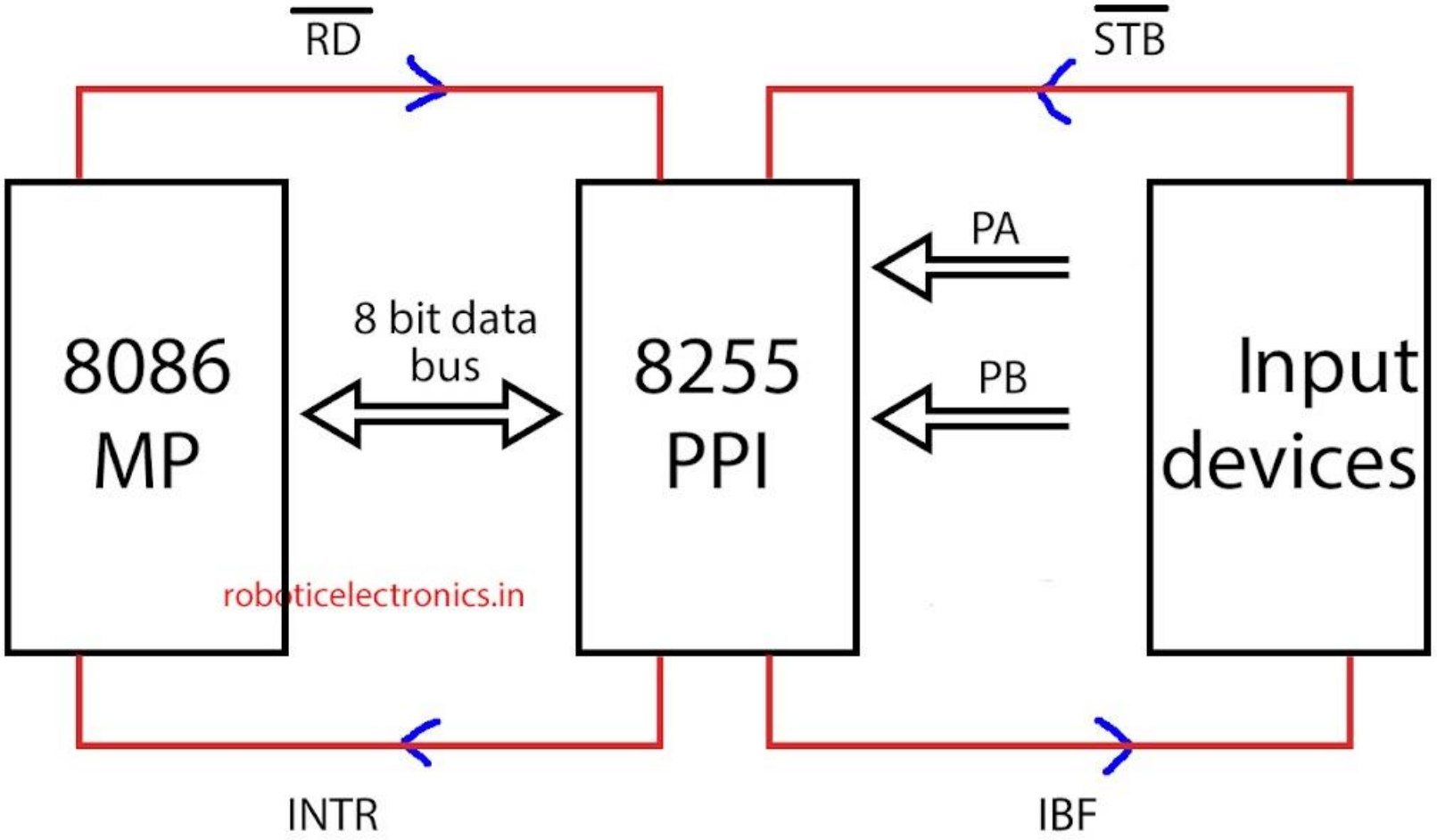
Mode 0 (Simple Bi-directional I/O)

- Port A and Port B used as 2 Simple 8-bit I/O Ports.
- Port C is used as 2 simple 4-bit I/O Ports.
- Each port can be programmed as input or output individually.
- Ports do not have handshake or interrupting capability.
- Hence, **slower** devices cannot be interfaced.

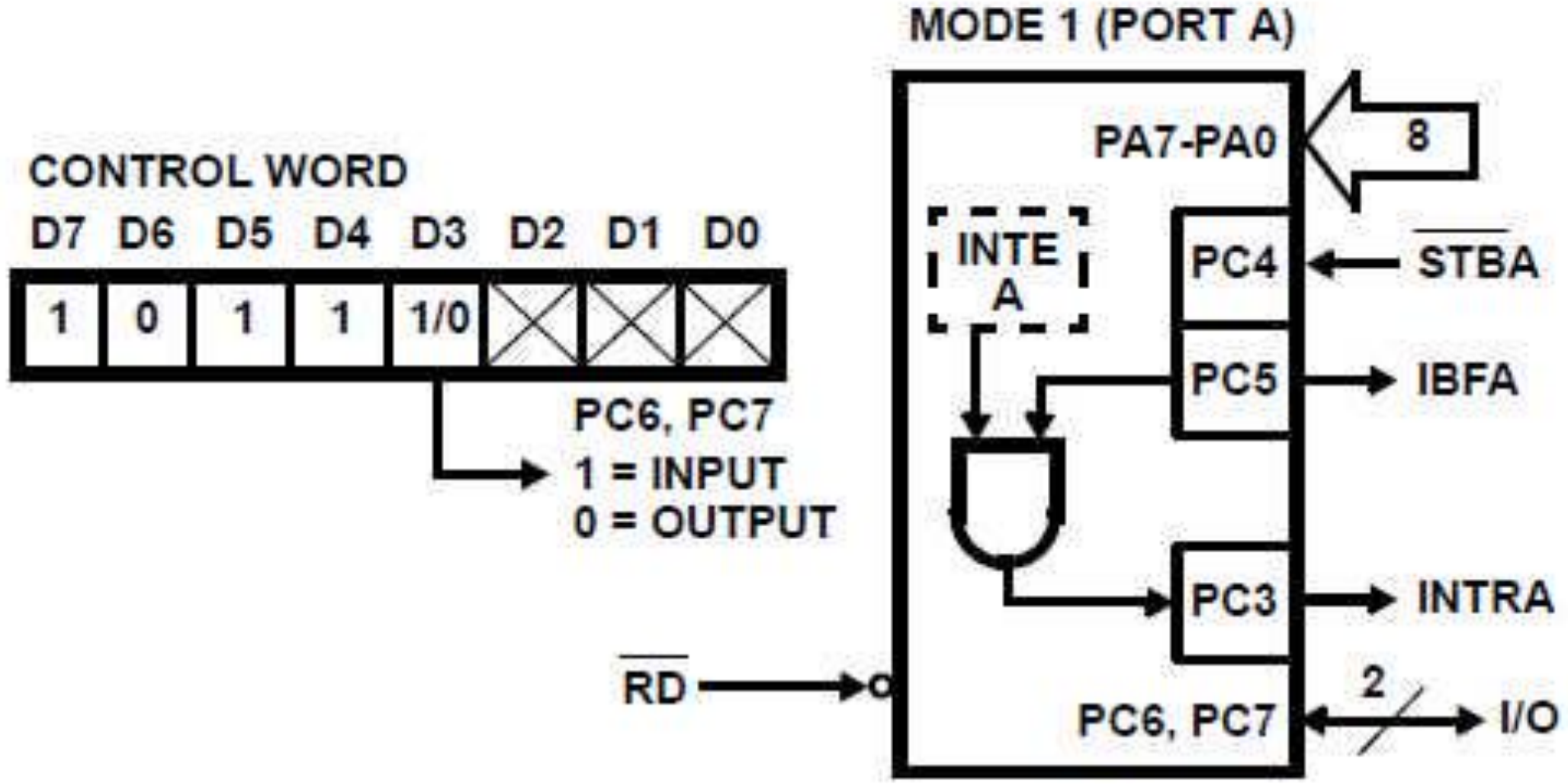
Mode 1 (Handshake I/O)

- In Mode 1, handshake signals are exchanged between the devices before the data transfer takeplace.
- Port A and Port B used as 2 8-bit I/O Ports that can programmed in Input OR in output mode.
- Each Port uses 3 lines from Port C for handshake. The remaining lines of Port C can be used for simple
- IO.
- **Interrupt driven** data transfer and **Status driven** data transfer possible.
- Hence, **slower** devices can be interfaced.
- The handshake signals are different for input and output modes.

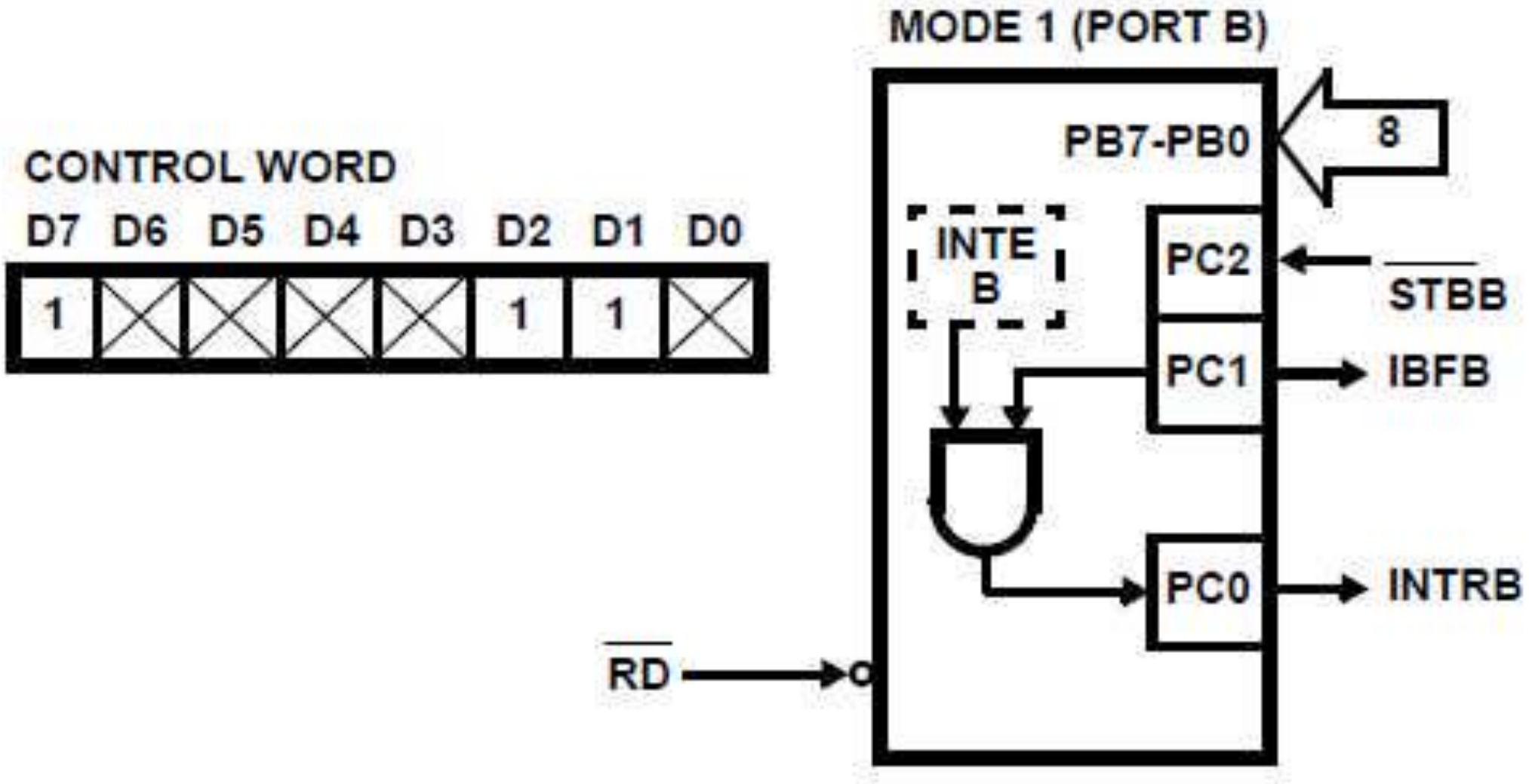
Mode 1 (Input Handshaking)



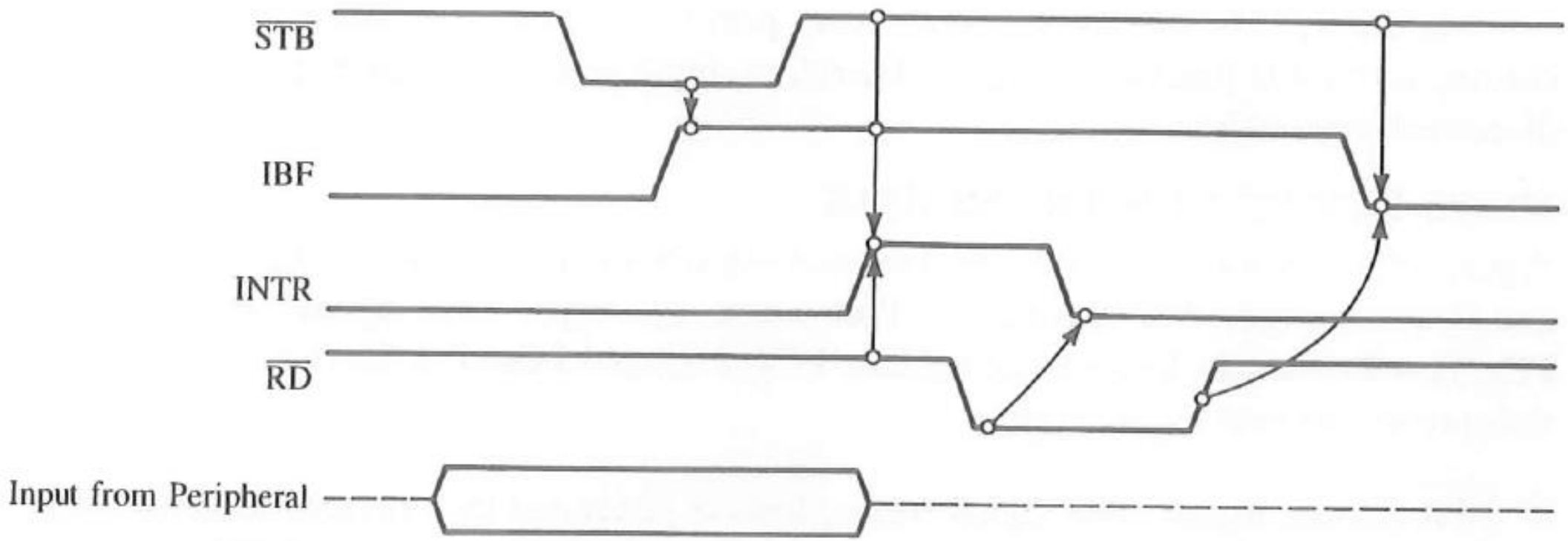
Mode 1 (Input Handshaking)



Mode 1 (Input Handshaking)



Mode 1 (Input Handshaking)



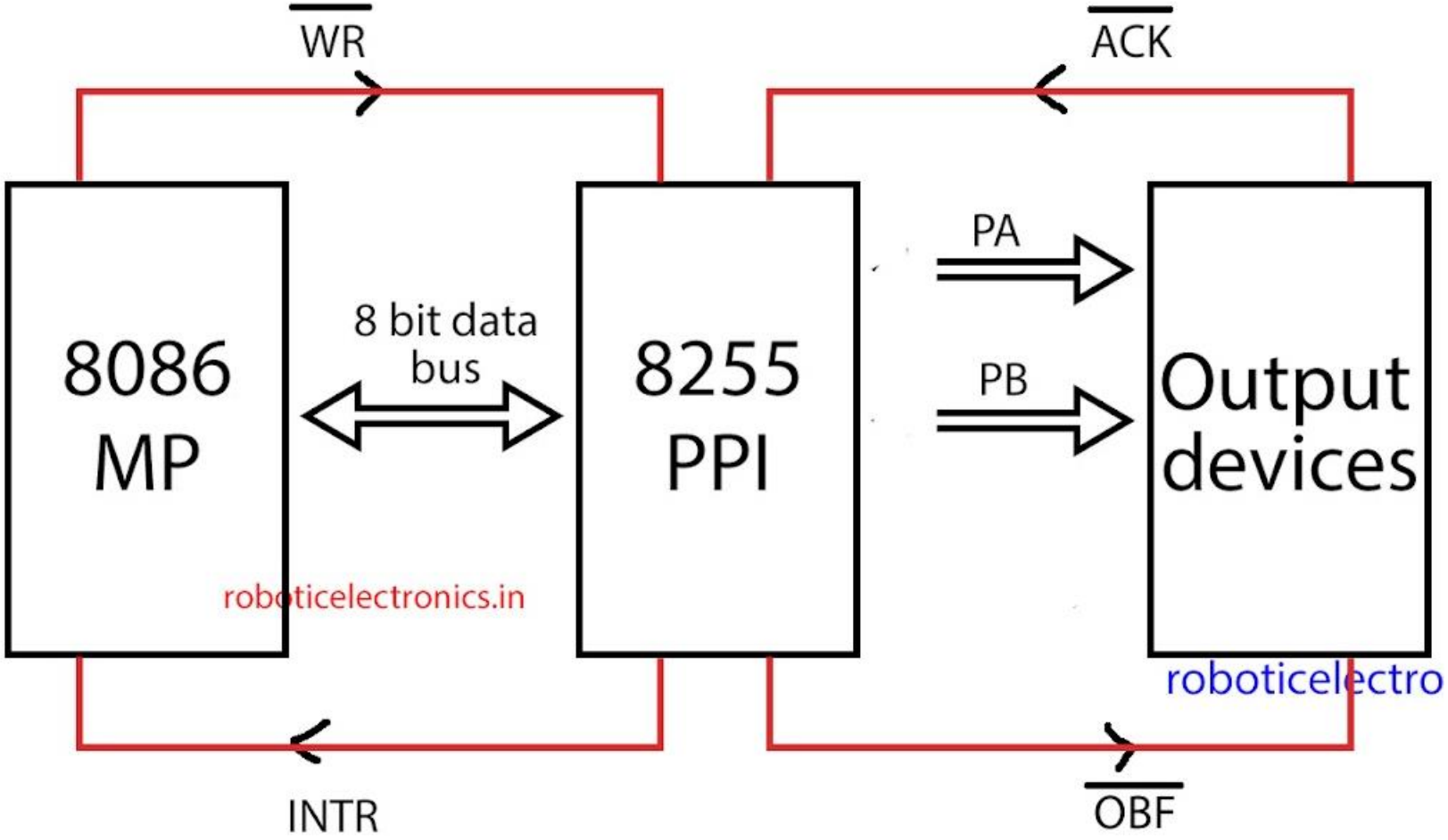
Mode 1 (Input Handshaking)

Working:

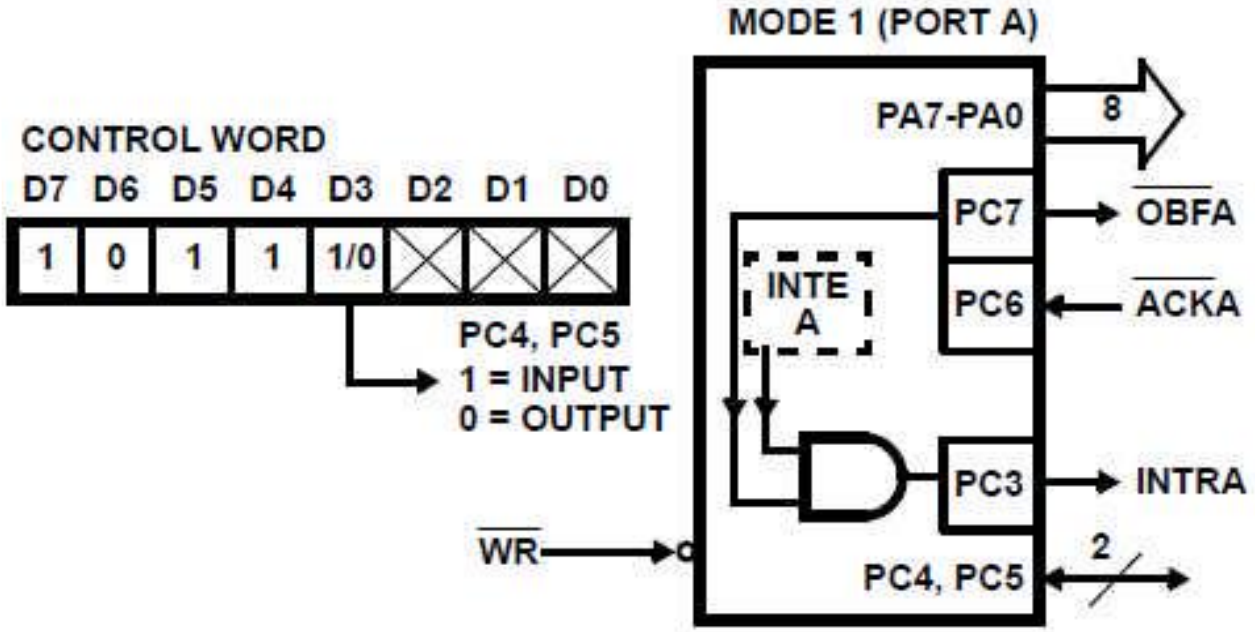
- **Each port** uses **3 lines** of **Port C** for the following signals:
- **STB** (Strobe), **IBF** (Input Buffer Full) ➔ Handhsake signals
- **INTR** (interrupt) ➔ Interrupt signal
- Additionally the $\overline{RD}$ signal of 8255 is also used.
- **Handshaking** takes place in the following manner:

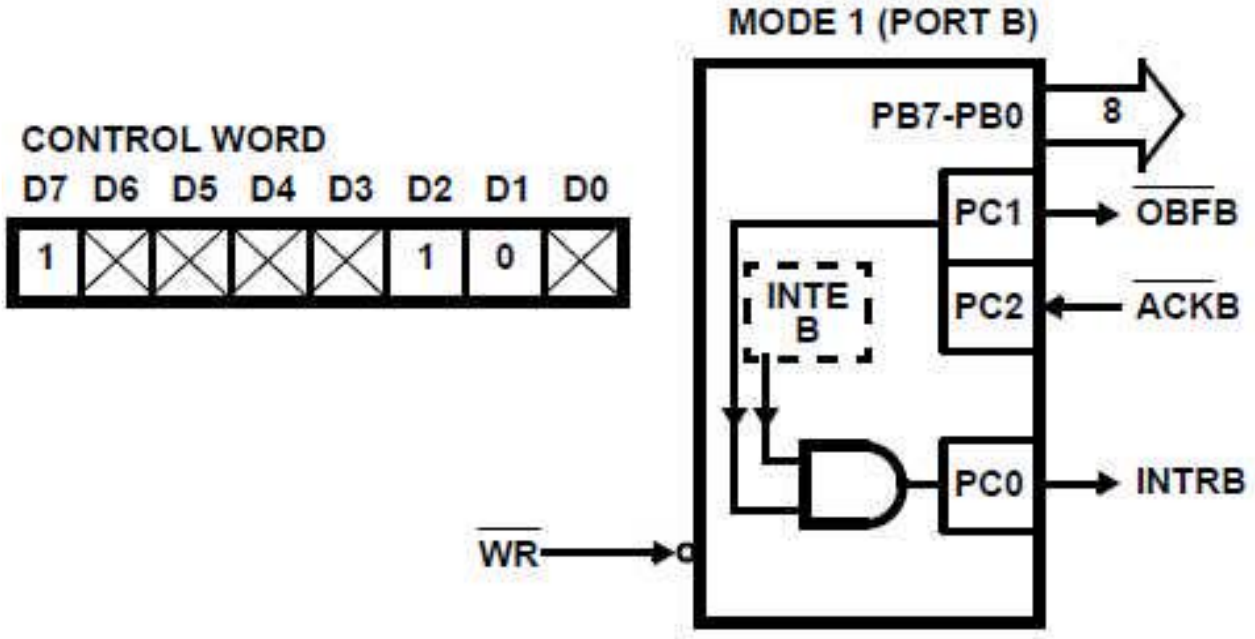
1. The **peripheral** device **places data** on the Port **bus** and informs the Port by **making STB low**.
2. The **input Port accepts** the **data** and informs the peripheral to wait by making **IBF high**.
3. This **prevents** the peripheral from **sending more data** to the 8255 and **hence data loss** is prevented.
4. **8255 interrupts** the **µP** through the **INTR** line provided the INTE flip-flop is set.
5. **In response** to the Interrupt, the **µP issues** the $\overline{RD}$ signal and **reads** the **data**. The **data byte** is **thus transferred** to the **µP**.
6. Now, the **IBF** signal **goes low** and the peripheral can **send more data** in the above sequence.

Mode 1 (Output Handshaking)

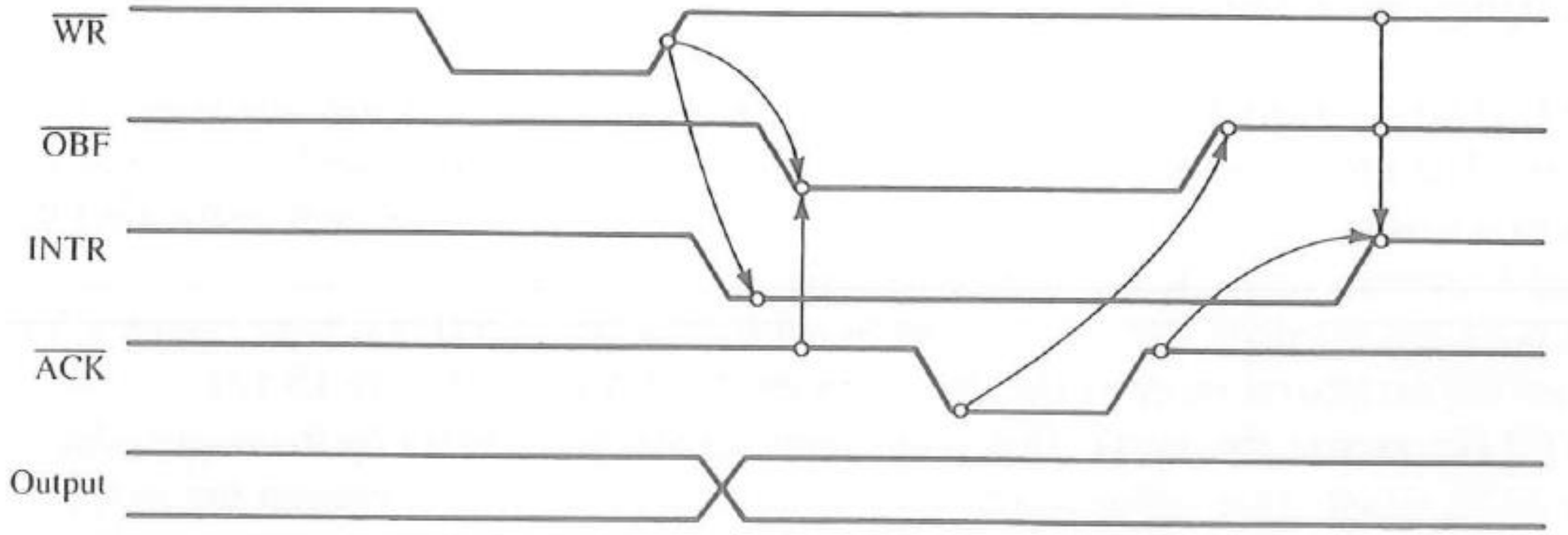


Mode 1 (Output Handshaking)





Mode 1 (Output Handshaking)



Mode 1 (Output Handshaking)

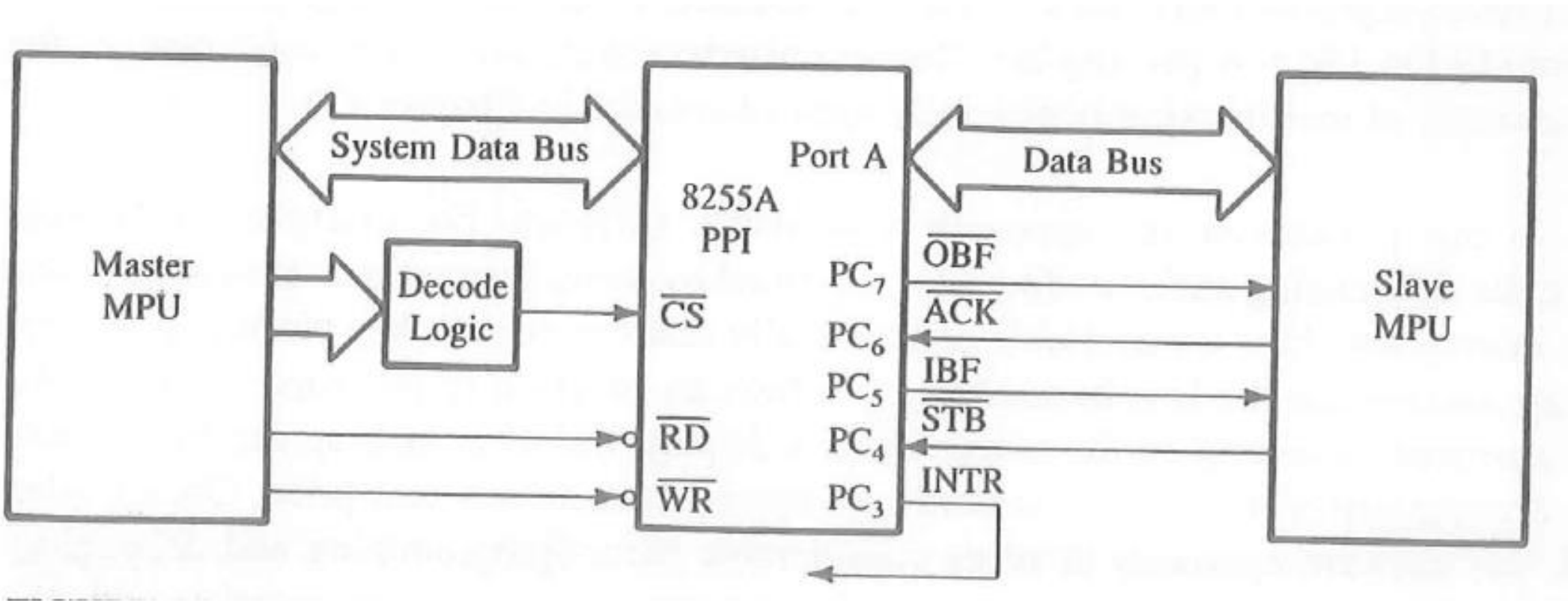
Working

- **Each port** uses **3 lines** of **Port C** for the following signals:
- **OBF** (Output Buffer Full), **ACK** (Acknowledgement) ➔ Handshake signals
- **INTR** (interrupt) ➔ Interrupt signal. Additionally the **WR** signal of 8255 is also used. **Handshaking**
- takes place in the following manner:

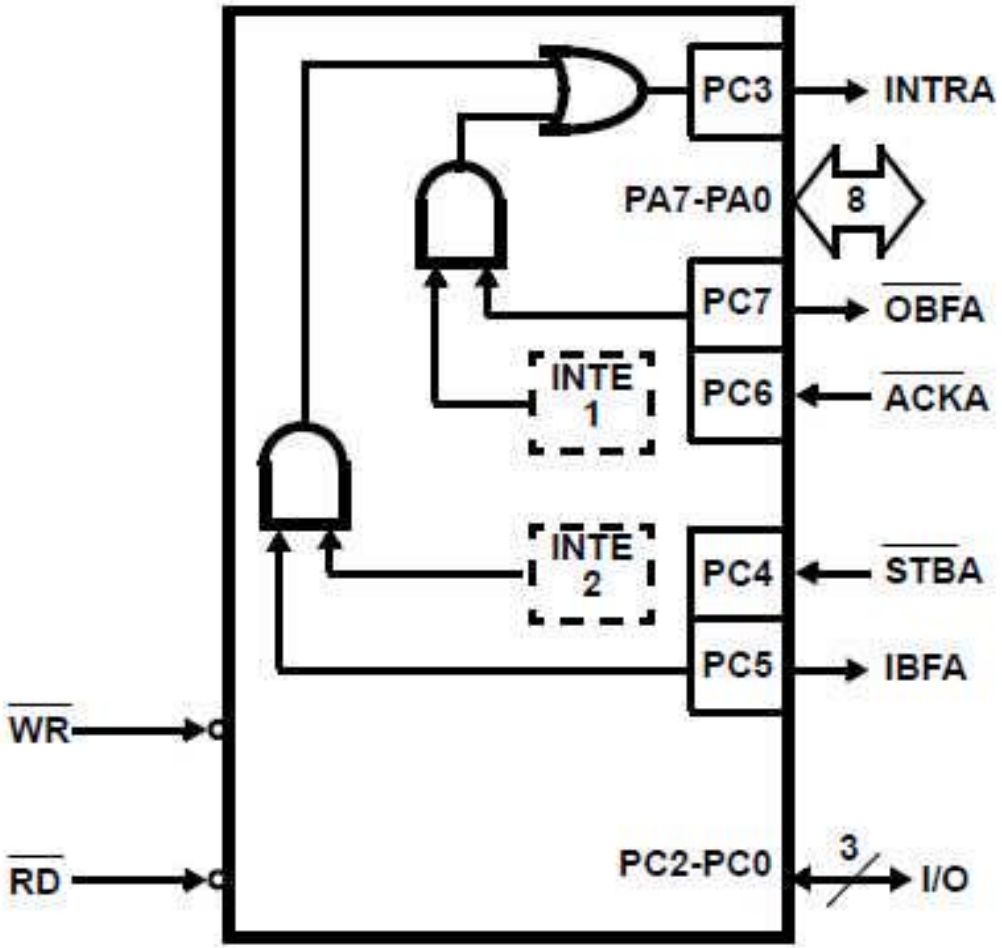
1) When the output port is empty (indicated by a high on the INTR line), the **µP writes data** on the output port by giving the **WR** signal.
2) As soon as the **WR** operation is complete, the **8255 makes the INTR low**, indicating that the µP should **wait**. This **prevents** the µP from **sending more data** to the 8255 and **hence data loss** is prevented.
3) **8255** also **makes** the **OBF low** to indicate to the output peripheral that **data** is **available** on the data bus.
4) The **peripheral accepts** the **data** and sends an acknowledgement by making the **ACK low**. The **data byte** is **thus transferred** to the peripheral.
5) Now, the **OBF** and **ACK** lines **go high**.
6) The **INTR** line **becomes high** to **inform** the **µP** that **another byte** can be **sent.** i.e. the output port is empty.

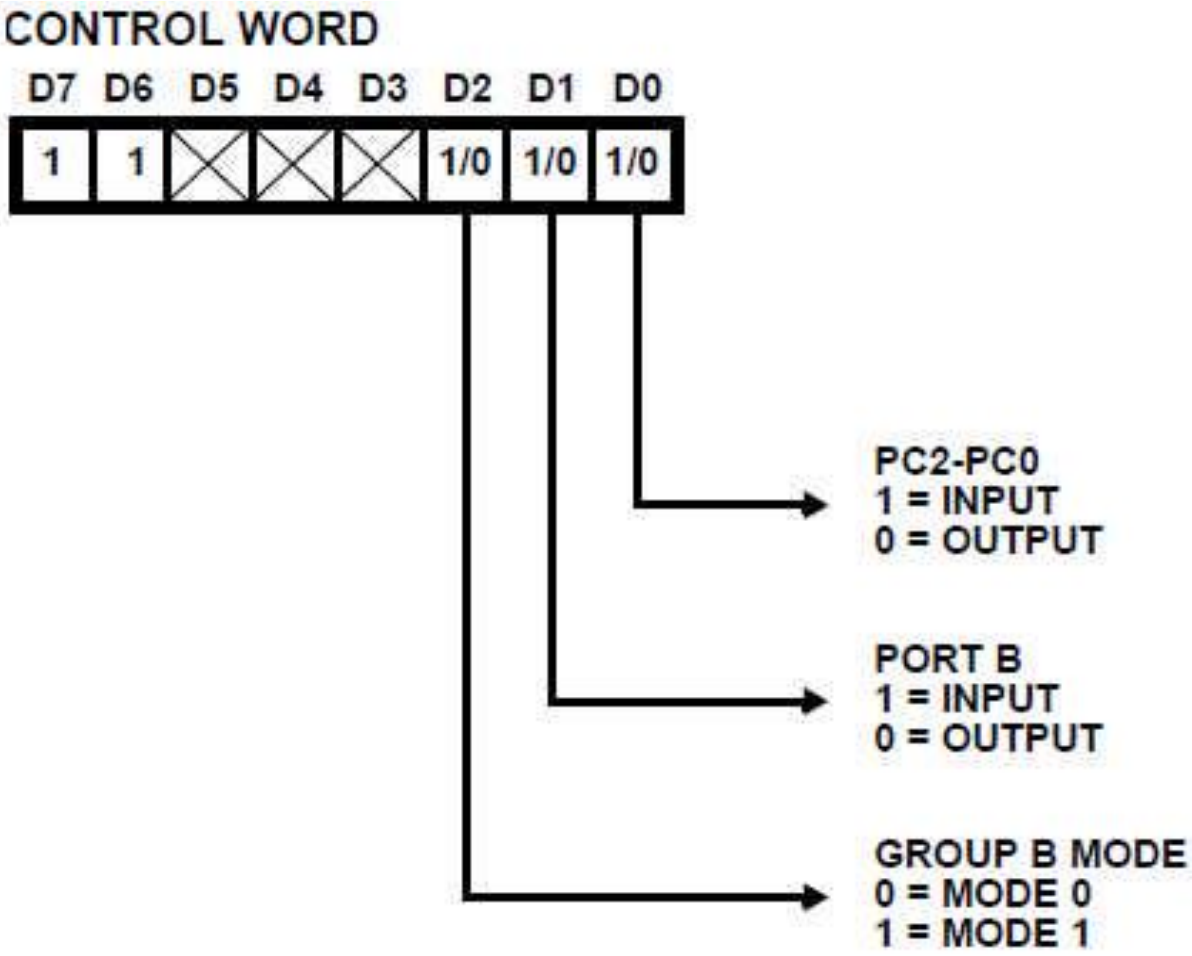
This process is repeated for further bytes.

Mode 2 (Bi-directional Handshake I/O)

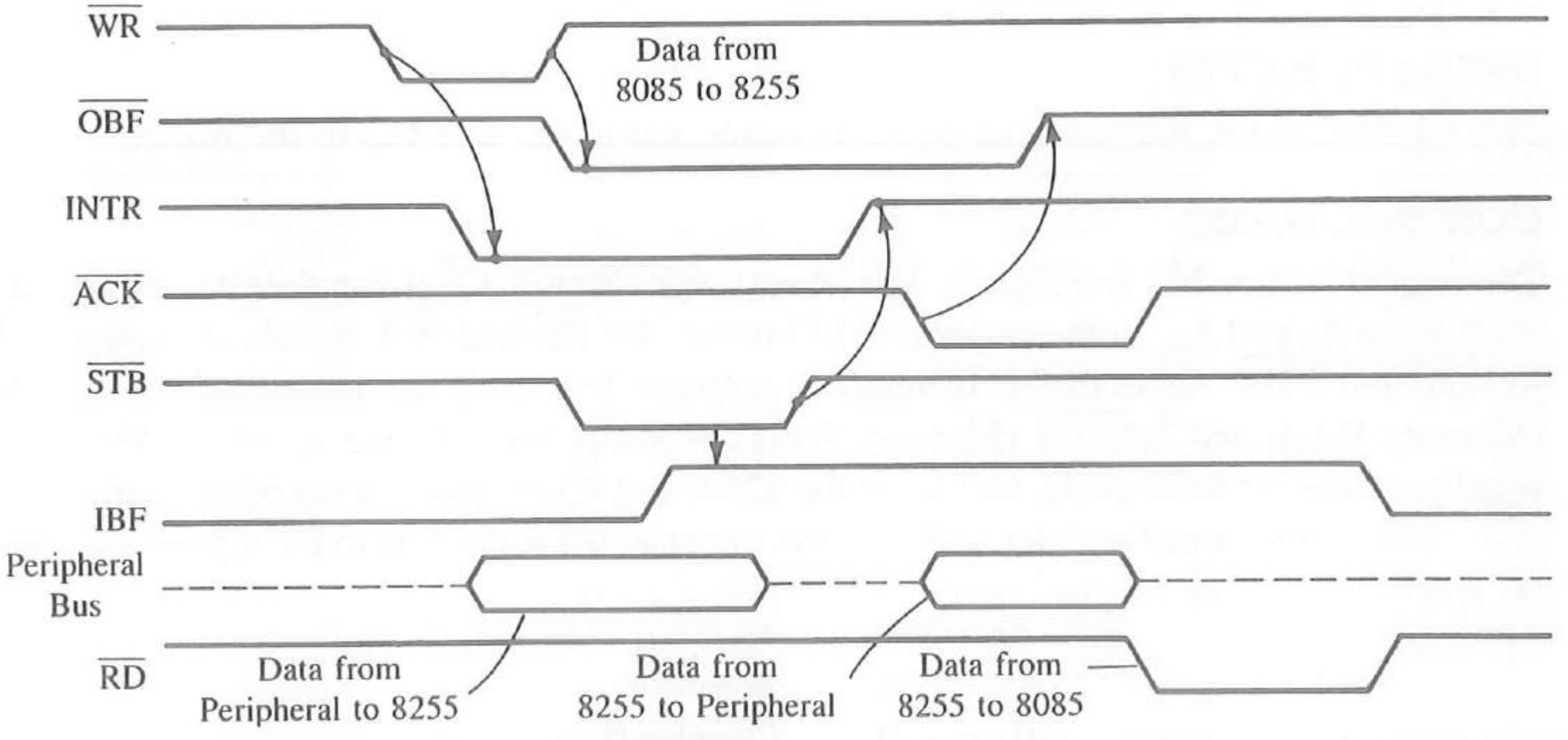


Mode 2 (Bi-directional Handshake I/O)





Mode 2 (Bi-directional Handshake I/O)



Mode 2 (Bi-directional Handshake I/O)

Working:

- In this mode, **Port A** is used as an **8-bit bi-directional Handshake I/O Port**.
- **Port A** requires **5 signals from Port C** for doing Bi-directional handshake.
- **Port B** has the following **options**:

1) **Use** the **remaining 3 lines** of **Port C** for handshaking so that **Port B is in Mode 1**. Here **Port C** lines will be **completely used for handshaking** (5 by Port A and 3 by Port B).

OR

2) **Port B** works in **Mode 0** as simple I/O.

   In this case the **remaining 3 lines** of **Port C** can be used for **data transfer**.

- Port A can be used for data transfer between two computers as shown.
- The high-speed computer is known as the master and the dedicated computer is known as the slave.
- Handshaking process is similar to Mode 1.

Mode 2 (Bi-directional Handshake I/O)

Working:

- For **Input:**
  STB and **IBF** → handshaking signals, **INTR** → Interrupt signal.
- For **Output:**
  $\overline{\textbf{OBF}}$ and $\overline{\textbf{ACK}}$ → handshaking signals, **INTR** → Interrupt signal.
- Thus the 5 signals used from Port C are:
  STB, IBF, INTR, $\overline{\textbf{OBF}}$ and $\overline{\textbf{ACK}}$.

8255 Modes Summary

Modes		PA	PB	PCL	PCU
Mode 0	Simple or Basic I/O	Works as port	Works as port	Works as port	Works as port
Mode 1	Handshake or Strobed I/O	Works as port	Works as port	Used for handshaking in PB	Used for handshaking in PA
Mode 2	Bidirectional Handshake I/O	Works as port	Works in mode – 0 or mode – 1	Used for handshaking in PB	Used for handshaking in PA