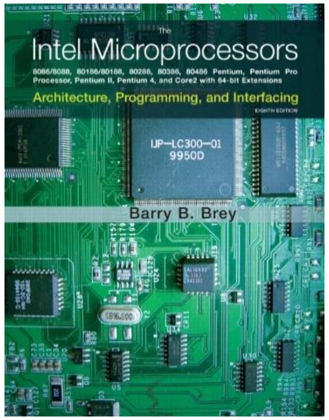
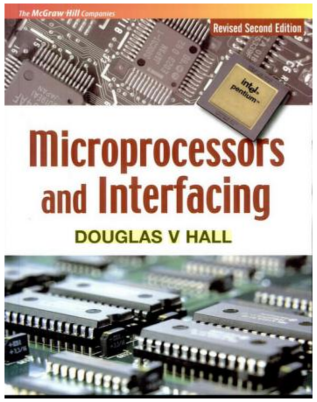
REFERENCE BOOKS


The Intel Microprocessors
8086/8088, 80186/80188, 80286, 80386, 80486 Pentium, Pentium Pro Processor, Pentium II, Pentium 4, and Core2 with 64-bit Extensions
Architecture, Programming, and Interfacing
EIGHTH EDITION
Barry B. Brey

The Intel Microprocessors
Author: Barry B. Brey
Edition: Eighth Edition



Microprocessors and Interfacing: Programming and Hardware
Author: Douglas V Hall
Edition: Revised Second Edition

INTERFACES AND INTERFACING

Definitions of “interface” from Webster’s Dictionary:

- noun: the place at which independent systems meet and act or communicate with each other

  examples:

  human - machine interface (analogue-machine interface)

Informal Definition

i) The physical, electrical and logical means of exchanging Information with a functional module

ii) The process of enabling a computer to communicate with the external world through Software, Hardware and Protocols

INTERFACES AND INTERFACING

- An interface is a device and/or set of rules to match the output of one device to send information to the input of another device
    - physical connection
    - the hardware
    - rules and procedures
    - the software

- Interfacing is the process of connecting devices together so that they can exchange information

IMPORTANCE OF INTERFACING

- ✓ The human - machine interface determines the ultimate success or failure of many computer- based systems
- ✓ Digital systems exist within and must successfully interact with an analogue natural environment (Digital – analogue interfaces are unavoidable)

I/O SYSTEM

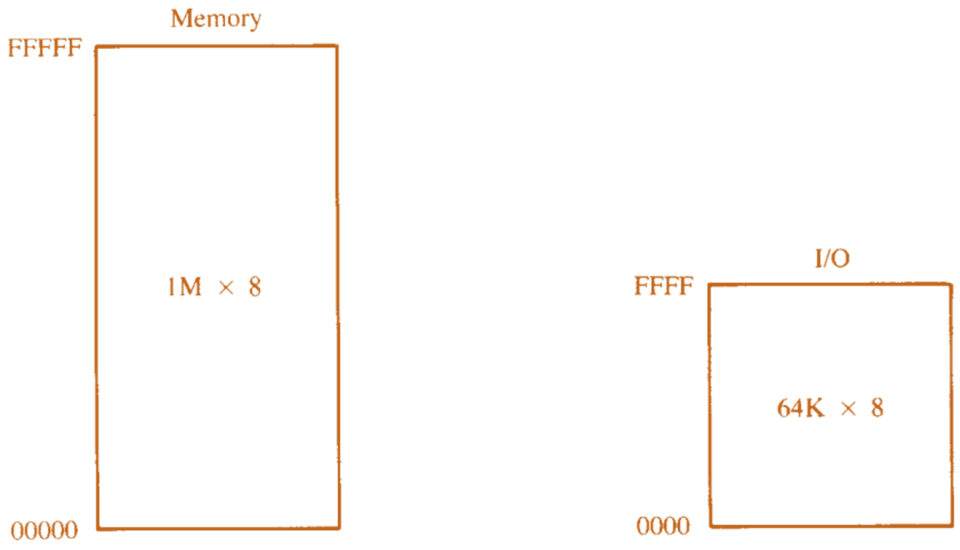
- What about I/O?
  - Without I/O, computers are useless (disembodied brains?)
  - But… thousands of devices, each slightly different
    - » How can we standardize the interfaces to these devices?
  - Devices unreliable: media failures and transmission errors
    - » How can we make them reliable???
  - Devices unpredictable and/or slow
    - » How can we manage them if we don't know what they will do or how they will perform?

METHODS OF COMMUNICATION BETWEEN CPU & I/O SYSTEM

- Isolated I/O or port-mapped I/O (PMIO) or I/O mapped I/O
- Memory-mapped I/O (MMIO)

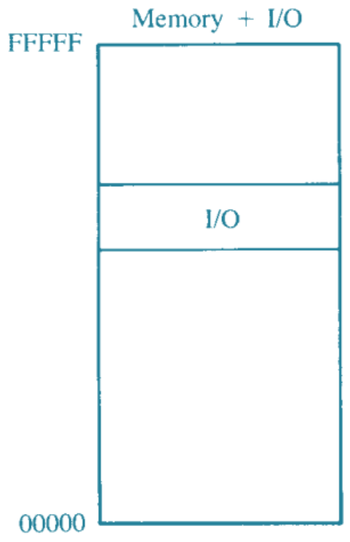
ISOLATED I/O

- ✓ I/O locations are isolated from the memory system
- ✓ The desired I/O port is selected by an I/O address
- ✓ Data transferred between I/O and the microprocessor must be accessed by
  - ✓ IN
  - ✓ OUT



MEMORY-MAPPED I/O

- ✓ Uses the same address bus to address both memory and I/O devices
  - ✓ Memory and registers of the I/O devices are mapped to (associated with) address values.
- ✓ So, when an address is accessed by the CPU, it may refer to a portion of physical RAM, but it can also refer to memory of the I/O device.
- ✓ Thus, the CPU instructions used to access the memory can also be used for accessing devices.



METHODS OF PARALLEL DATA TRANSFER

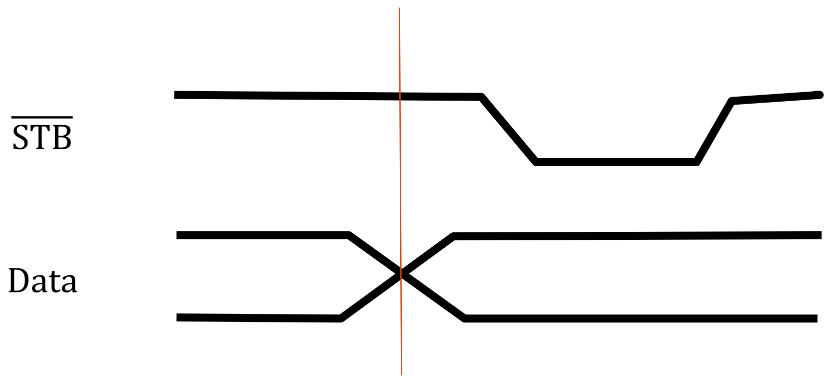
- Simple I/O
- Simple strobe I/O
- Single-handshake I/O
- Double-handshake I/O

SIMPLE I/O

- I/O device is always ready
- Assumption
  - CPU Receiver: Data is always present
    - Can be read/write at any time
  - CPU Sender: Receiver is always ready
- Data transfer is not time dependent
- Example:
  - Output to LED
  - Reading from a switch
- Limitations ?

SIMPLE STROBE I/O

- Valid data present on an external device only at a certain time
  - Must be read at that time
- Data transfer is time dependent
- Sender send Strobe signal after sending valid data
  - To notify reeiver
- Receiver can read data only after getting strobe signal
  - Polling
  - Interrupt
- Example
  - Reading from Keyboard
- Limitations ?

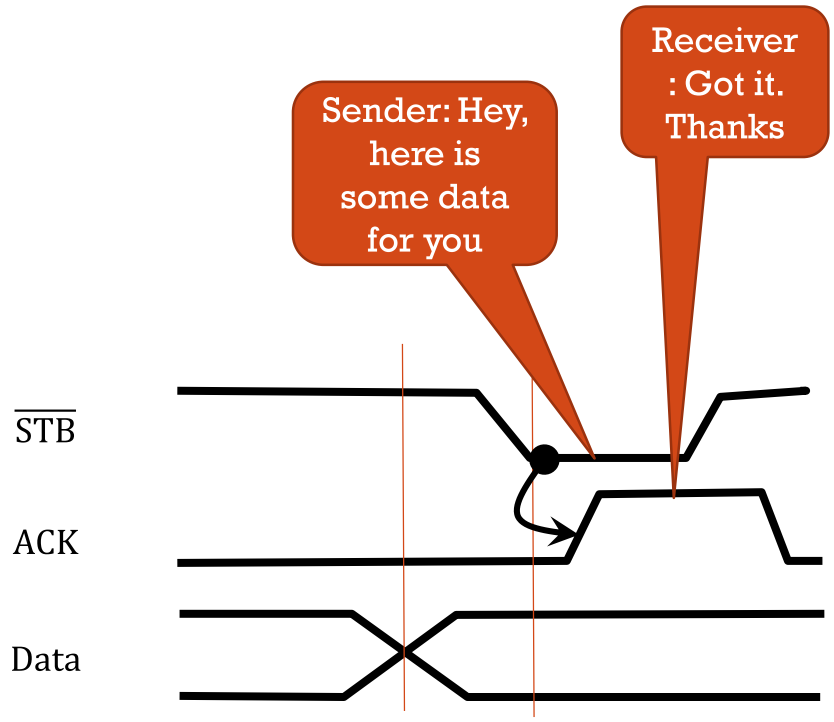


SIMPLE STROBE I/O LIMITATIONS

- Assumption
  - CPU Sender: Receiver is always ready
- Works well for low rate data transfer
- No signal for the sender to know when it is safe to send the next data byte
- Sending and receiving speeds on both ends are often different

SINGLE-HANDSHAKE I/O

- After reading data, an receiver send Acknowledge signal
- Also called "strobed input/output"
- Limitation??

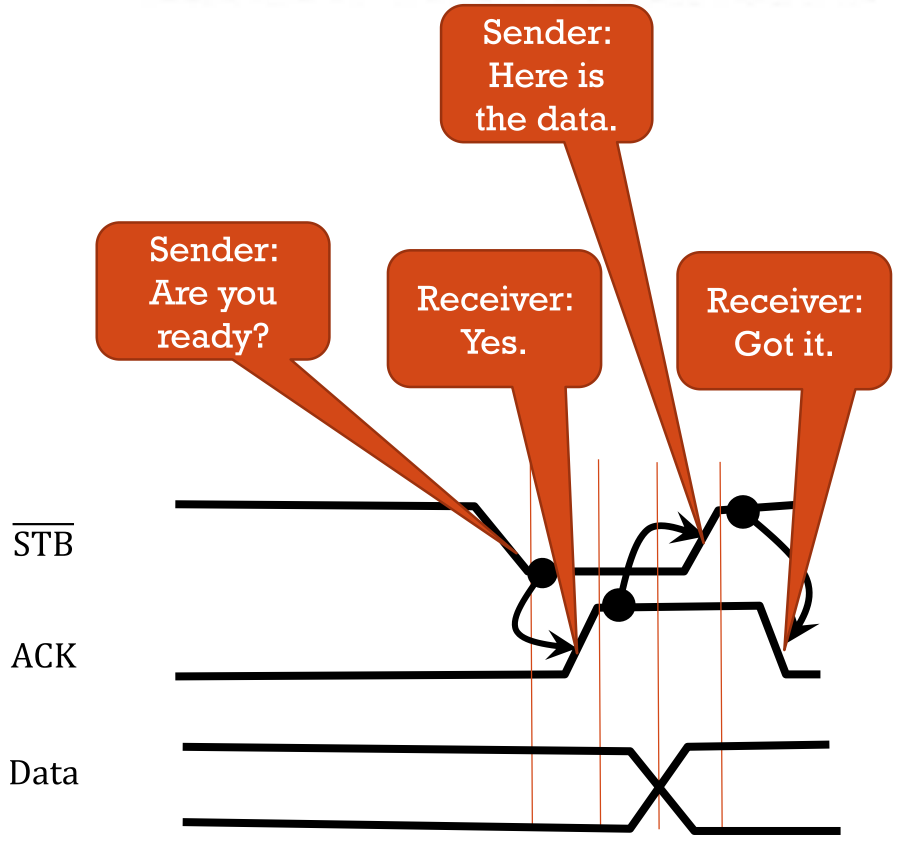

Sender: Hey, here is some data for you
Receiver : Got it. Thanks
$\overline{\text{STB}}$
ACK
Data

SINGLE-HANDSHAKE I/O LIMITATIONS

- Sender still doesn't consider whether the receiver is ready to receive
- In certain situations (such as multiple device communication)
  - sender should not send the information to the data bus unless the receiver is ready to receive

DOUBLE-HANDSHAKE I/O

- More coordination
- Data is put after receiver is ready
- Double handshake
  - Each signal edge has meaning
  - Sender sends the data after the first handshaking
- When applicable?



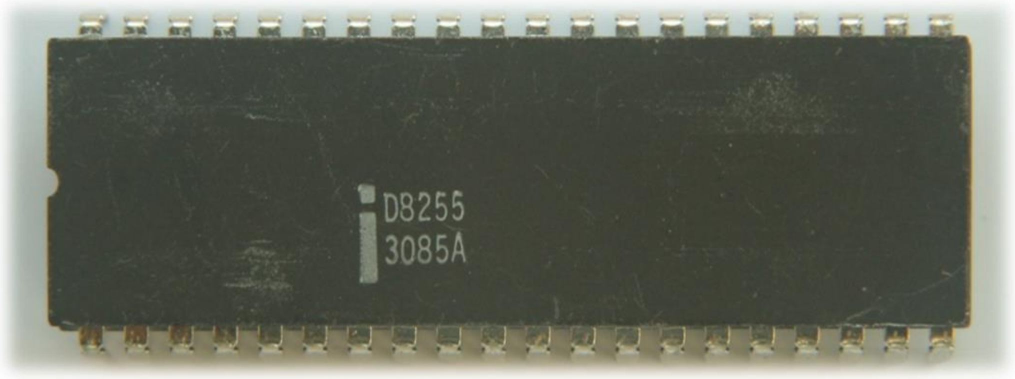
HOW TO IMPLEMENT HANDSHAKE?

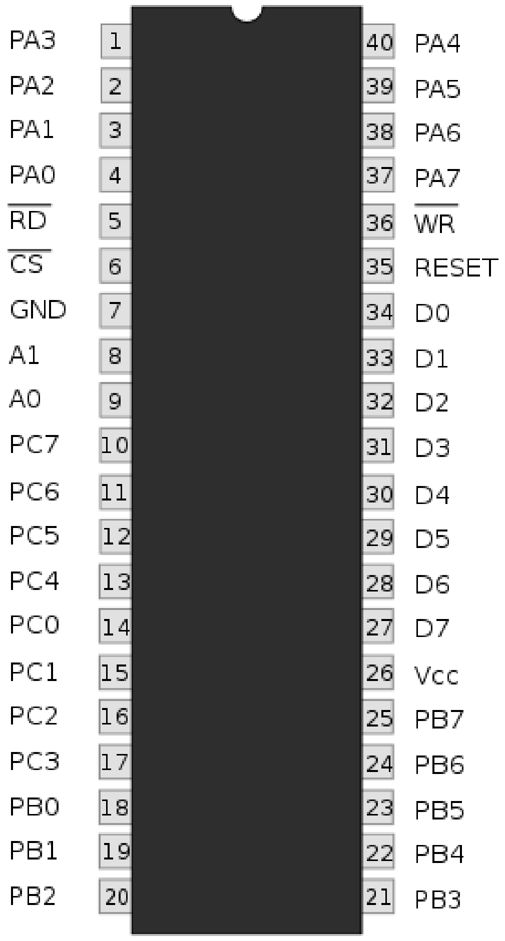
- The simplest idea
  - Directly connect the CPU to the I/O devices
  - CPU determines when it is time to send/receive next data byte
    - Polling
    - Interrupt (Better??)
- But in that case, CPU has to manage handshake operations
  - Produce $\overline{\mathrm{STB}}/\mathrm{ACK}$ signals by executing instructions
- Should we assign these tasks to CPU?

HOW TO IMPLEMENT HANDSHAKE?

- Better alternative
  - Connect the CPU to the I/O devices through a “interfacing” device which can
    - Automatically manage the handshake operation
  - Such an “interfacing” device is Intel 8255

INTEL 8255: PPI

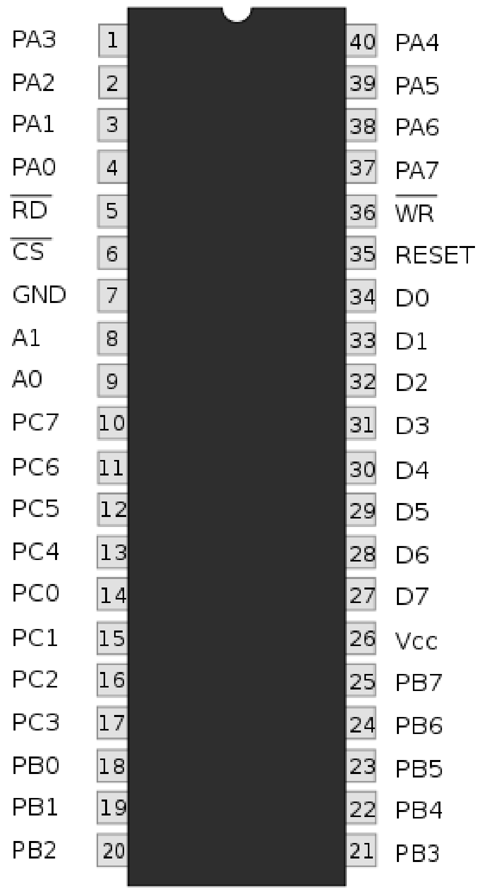


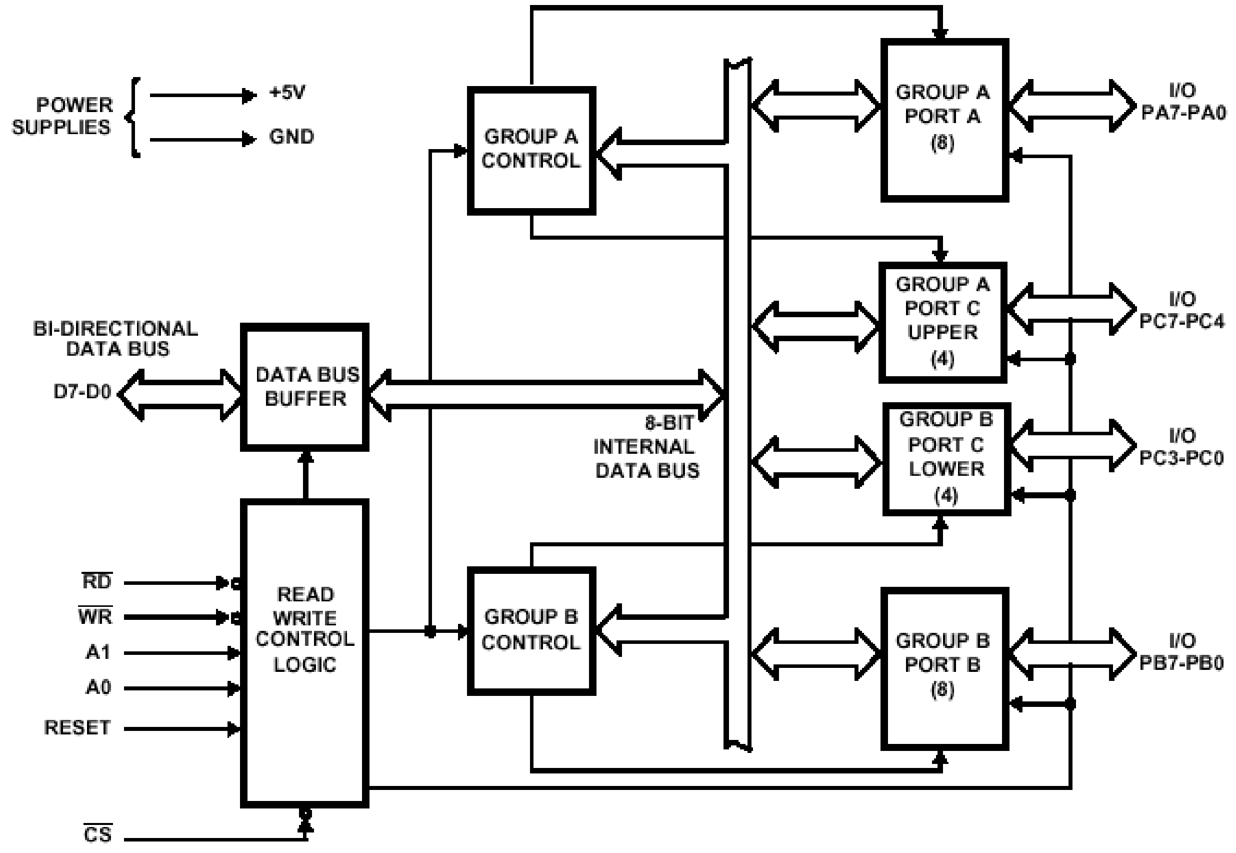


PPI: PROGRAMMABLE PERIPHERAL INTERFACE

- Let’s think about the name
  - Programmable?
  - Peripheral?
  - Interface?
- Can be programmed to automatically
  - Receive $\overline{\mathrm{STB}}$ signal from a peripheral
  - Send interrupt signal to the CPU
  - Send ACKsignal to the peripheral at proper times

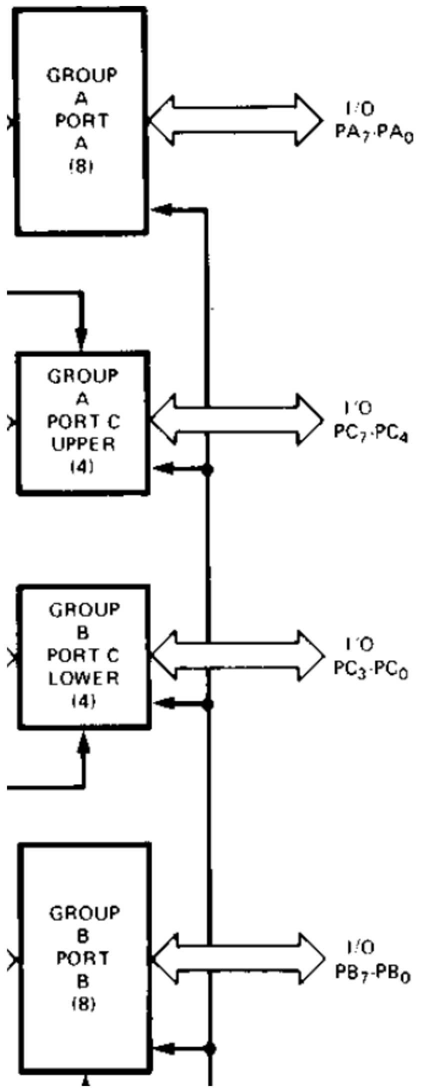
BLOCK DIAGRAM





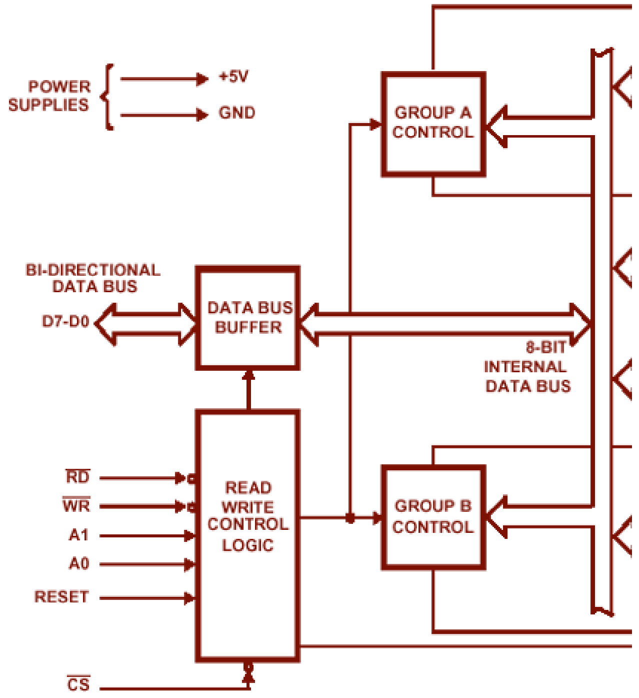
PORTS & REGISTER

- 24 input/output pins
  - 3 PORTS: A, B, C
  - Port A, B: I/O
  - Port C: I/O and handshake signals
- The three ports are further grouped as follows:
  - Group A : port A and upper part of port C
  - Group B : port B and lower part of port C



READ/WRITE AND CONTROL LOGIC

- Manage all of the internal and external transfers of both Data and Control or Status words.
- Accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.
  - **(CS)** Chip Select. A "low" on this input pin enables the communication between the 8255 and the CPU.
  - **(RD)** Read. A "low" on this input pin enables 8255 to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255.
  - **(WR)** Write. A "low" on this input pin enables the CPU to write data or control words into the 8255.
  - **(A0 and A1)** Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word register. They are normally connected to the least significant bits of the address bus (A0 and A1).
  - **(RESET)** Reset. A "high" on this input initializes the control register to 9Bh and all ports (A, B, C) are set to the input mode.



PORT SELECTION

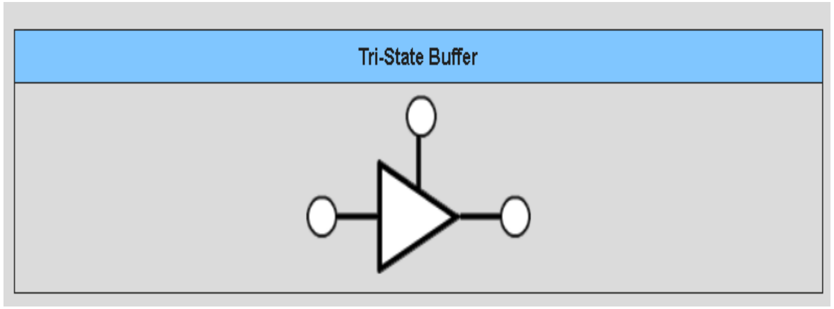
$\overline{CS}$	A1	A0	SELECTEDPORT
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	Control Register
1	X	X	8255 is not selected

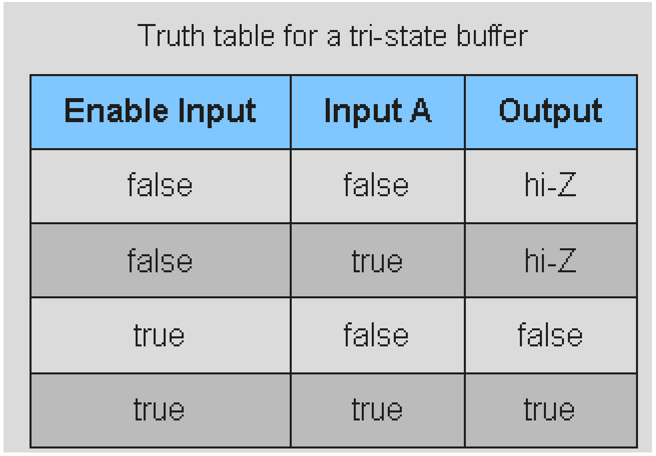
INTEGRATING WITH MICROPROCESSOR

- Three-state bi-directional 8-bit buffer is used to interface the 8255 to the system data bus.
- Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU.
- Control words and status information are also transferred through the data bus buffer.

TRI STATE BUFFER

- A tri-state buffer is similar to a buffer, but it adds an additional "enable" input that controls whether the primary input is passed to its output or not.
- If the "enable" inputs signal is true, the tri-state buffer behaves like a normal buffer.
- If the "enable" input signal is false, the tri-state buffer passes a *high impedance* (or hi-Z) signal, which effectively disconnects its output from the circuit.



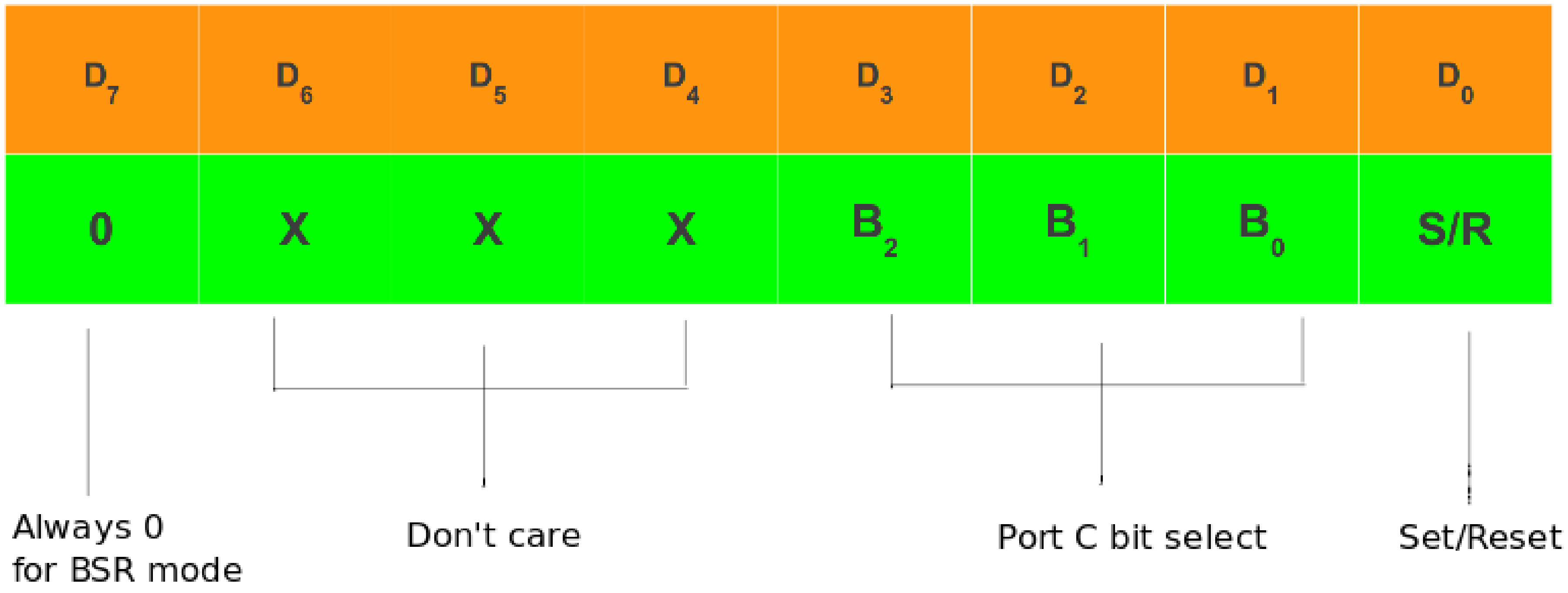

Truth table for a tri-state buffer

Enable Input	Input A	Output
false	false	hi-Z
false	true	hi-Z
true	false	false
true	true	true

OPERATIONAL MODES

- Bit set/reset Mode (BSR Mode)
  - used to set/reset the bits in Port C
- Input/Output Mode (I/O Mode)
  - MODE 0 – Simple I/O
  - MODE 1 – Single-handshake
  - MODE 2 – bidirectional handshake I/O
- BSR mode and I/O mode are independent
  - Selection of BSR mode does not affect the operation of other ports in I/O mode
- Functions of I/O pins and modes of I/O ports can be programmed by
  - writing proper control word in the control register

BSR MODE: CONTROL WORD



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	X	X	X	B_2	B_1	B_0	S/R

8255 Control Register format for BSR Mode

WHAT WILL BE THE CONTROL BITS ??

- Can you set PC5?

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

I/O MODE: CONTROL WORD

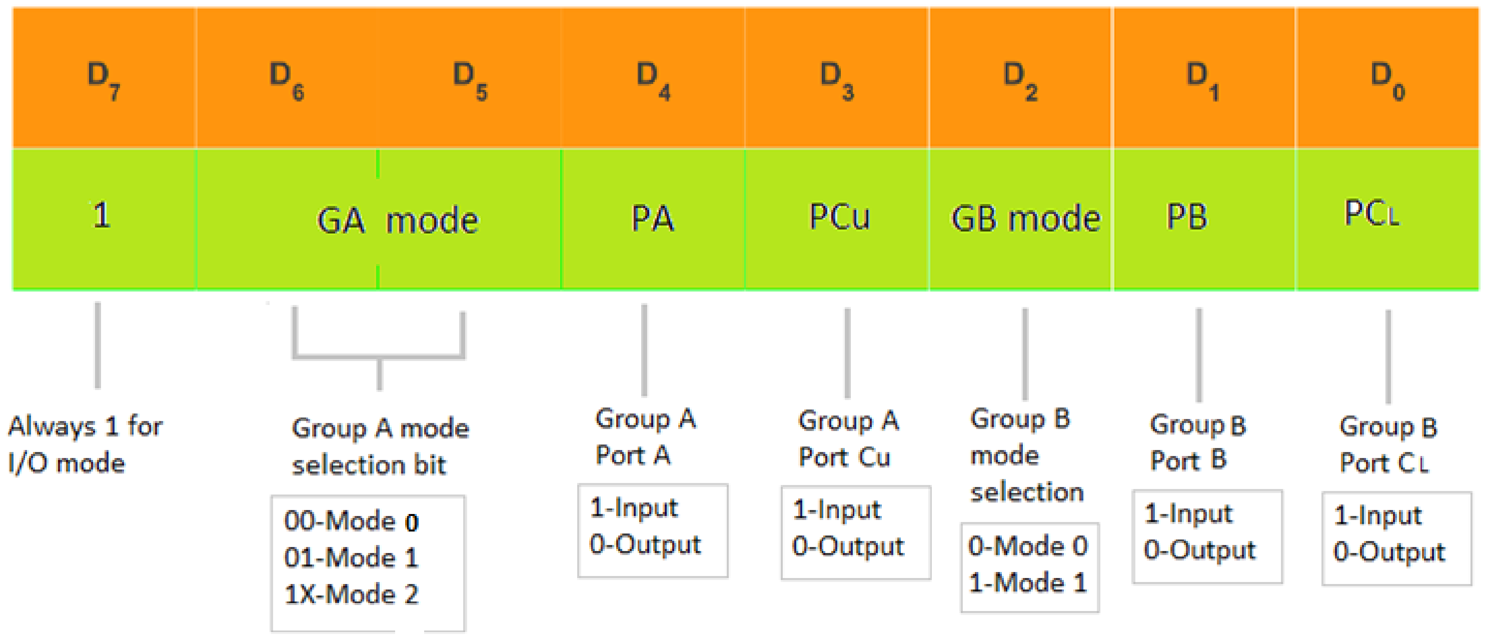
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	GA mode		PA	PCu	GB mode	PB	PCL
Always 1 for I/O mode	Group A mode selection bit 00-Mode 1 01-Mode 2 1X-Mode 3		Group A Port A 1-Input 0-Output	Group A Port Cu 1-Input 0-Output	Group B mode selection 0-Mode 0 1-Mode 1	Group B Port B 1-Input 0-Output	Group B Port CL 1-Input 0-Output

PCu-Port C upper
PCL-Port C lower

8255 Control Word For I/O mode

WHAT WILL BE THE CONTROL BITS ??

- A – input , B – output and C(lower) input ??



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	GA mode		PA	PCu	GB mode	PB	PCL
Always 1 for I/O mode	Group A mode selection bit		Group A Port A	Group A Port Cu	Group B mode selection	Group B Port B	Group B Port CL
	00-Mode 0 01-Mode 1 1X-Mode 2		1-Input 0-Output	1-Input 0-Output	0-Mode 0 1-Mode 1	1-Input 0-Output	1-Input 0-Output

8255 Control Word For I/O mode

PCu-Port C upper
PCL-Port C lower

I/O MODE: MODE 0

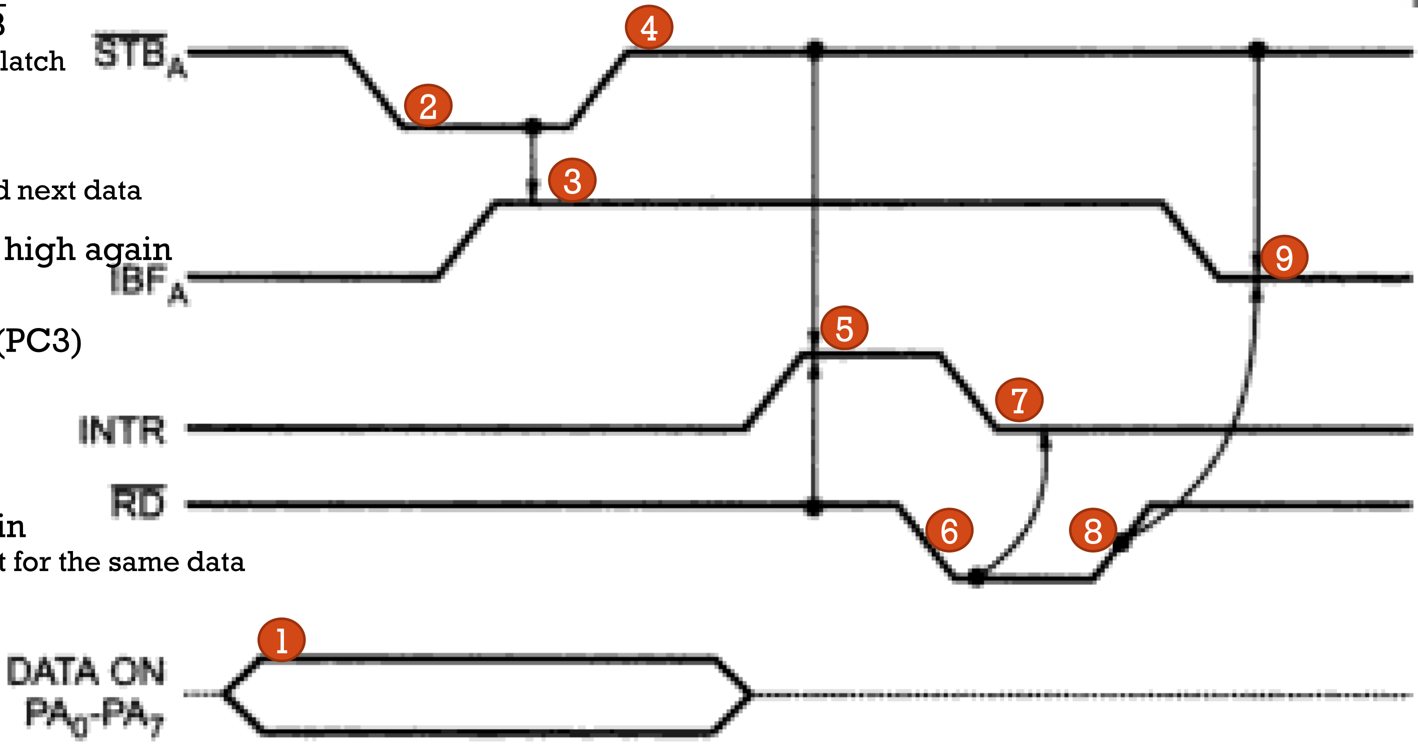
- SIMPLE I/O
  - A, B, and C(upper and lower) all can be used as I/O independently
  - 16 possible configurations

I/O MODE: MODE 1

- PORT A, B : Single handshake I/O (Strobed I/O)
- Handshake signals for PORT B
  - PIN PC0 – PC2
- Handshake signals for PORT A
  - PIN PC3 – PC5 when PORT A input
  - PIN PC3, PC6, PC7 when PORT A output
- Remaining lines of PORT C can be used for I/O

INPUT MODE 1 ON PORTA

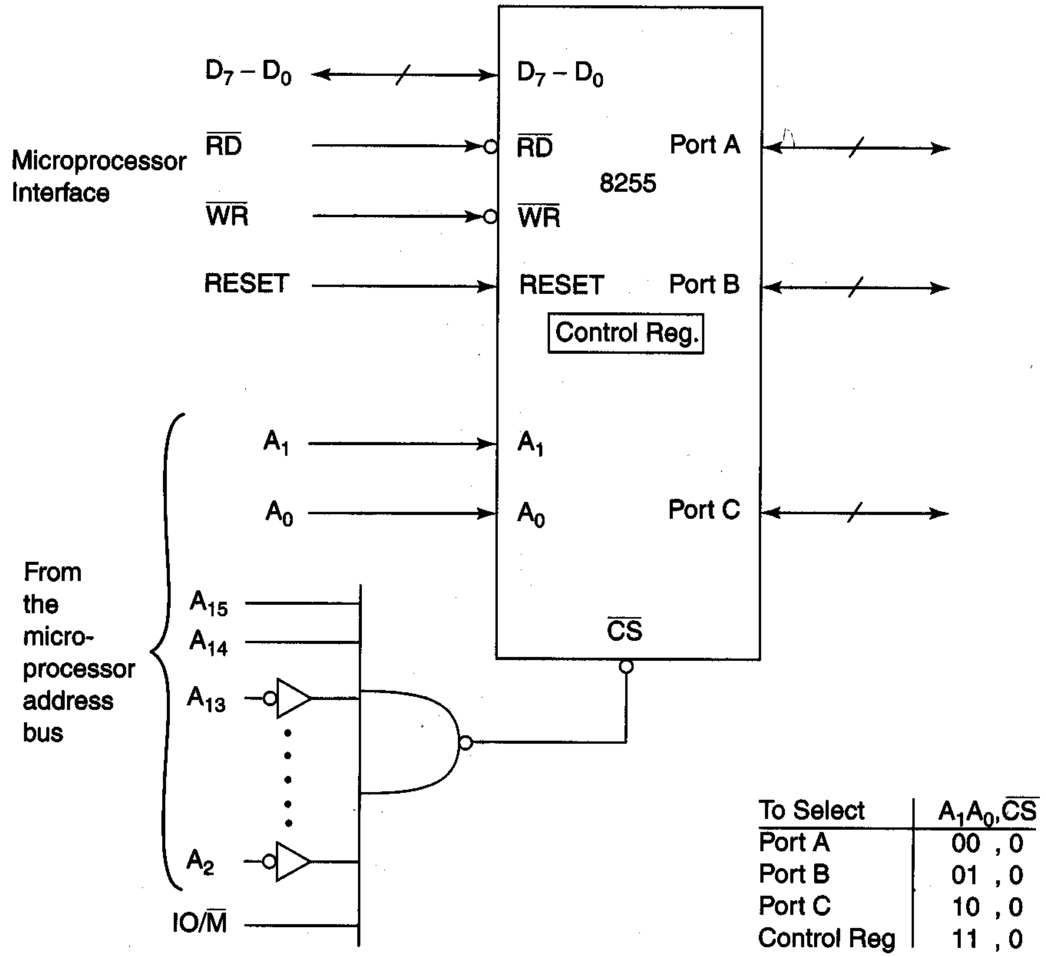
1. I/O device sends data to port's data line
2. I/O device lowers $\overline{STB}$
   - 8255 loads data into input latch
3. 8255 raises IBF (PC5)
   - ACK
   - Forbids I/O device to send next data
4. I/O device raises $\overline{STB}$ high again
   - Data is no more valid
5. 8255 generates INTR (PC3)
   - Inform CPU
6. CPU lowers $\overline{RD}$
   - Starts reading
7. 8255 lowers INTR again
   - Prevent a second interrupt for the same data
8. CPU raises $\overline{RD}$ again
   - Read complete
9. 8255 lowers IBF again
   - Data transfer complete



I/O MODE: MODE 2

- Port A can be used as a bi-directional 8-bit I/O bus
- Port B can be programmed in Mode 0 or in Mode 1.

INTERFACING 8255 WITH MICROPROCESSOR



To Select	$A_1A_0,\overline{CS}$
Port A	00 , 0
Port B	01 , 0
Port C	10 , 0
Control Reg	11 , 0

EXAMPLE:

- What is the addresses of port A, port B, port C of the 82C55A device?

Solution:

To access port A, $A_1A_0 = 00$, $A_{15} = A_{14} = 1$, $A_{13} =$

$A_{12} = \ldots = A_2 = 0$, which gives the port A address

as

$$1100\ 0000\ 0000\ 0000_2 = C000_{16}$$

Similarly, it can be determined that the address of

port B equals $C001_{16}$, that of port C is $C002_{16}$,

and the address of the control register is $C003_{16}$.

RESOURCES

- https://en.wikipedia.org/wiki/Intel_8255
- Intel 8255 datasheet (http://www.csee.umbc.edu/~reza2/courses/310/Slides/8255.pdf)
- Microprocessors And Interfacing 2E, Douglas V Hall
  - Chapter 9
- Microprocessors And Interfacing 1E, A.P. Godse
  - Chapter 7

THANK YOU