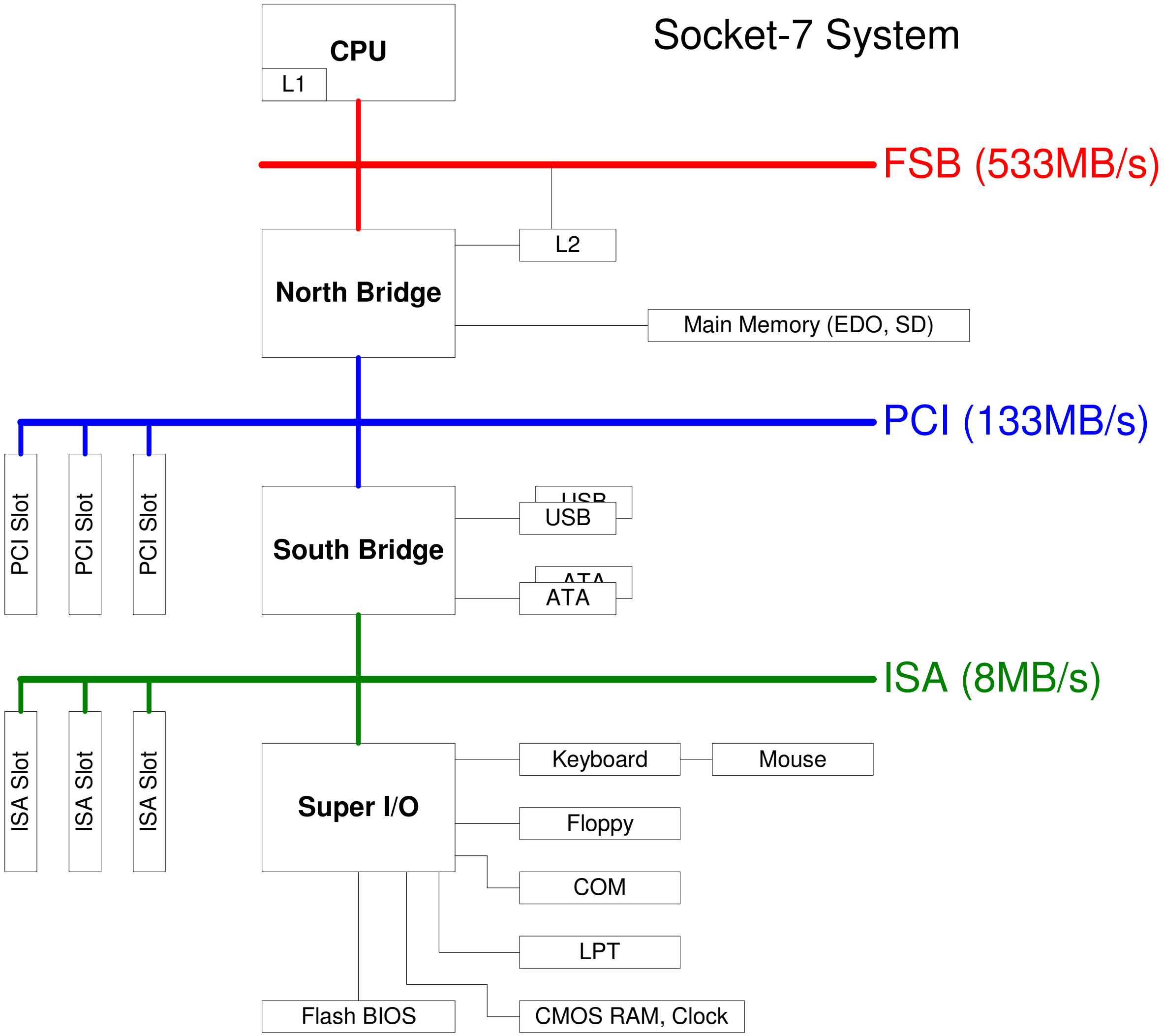
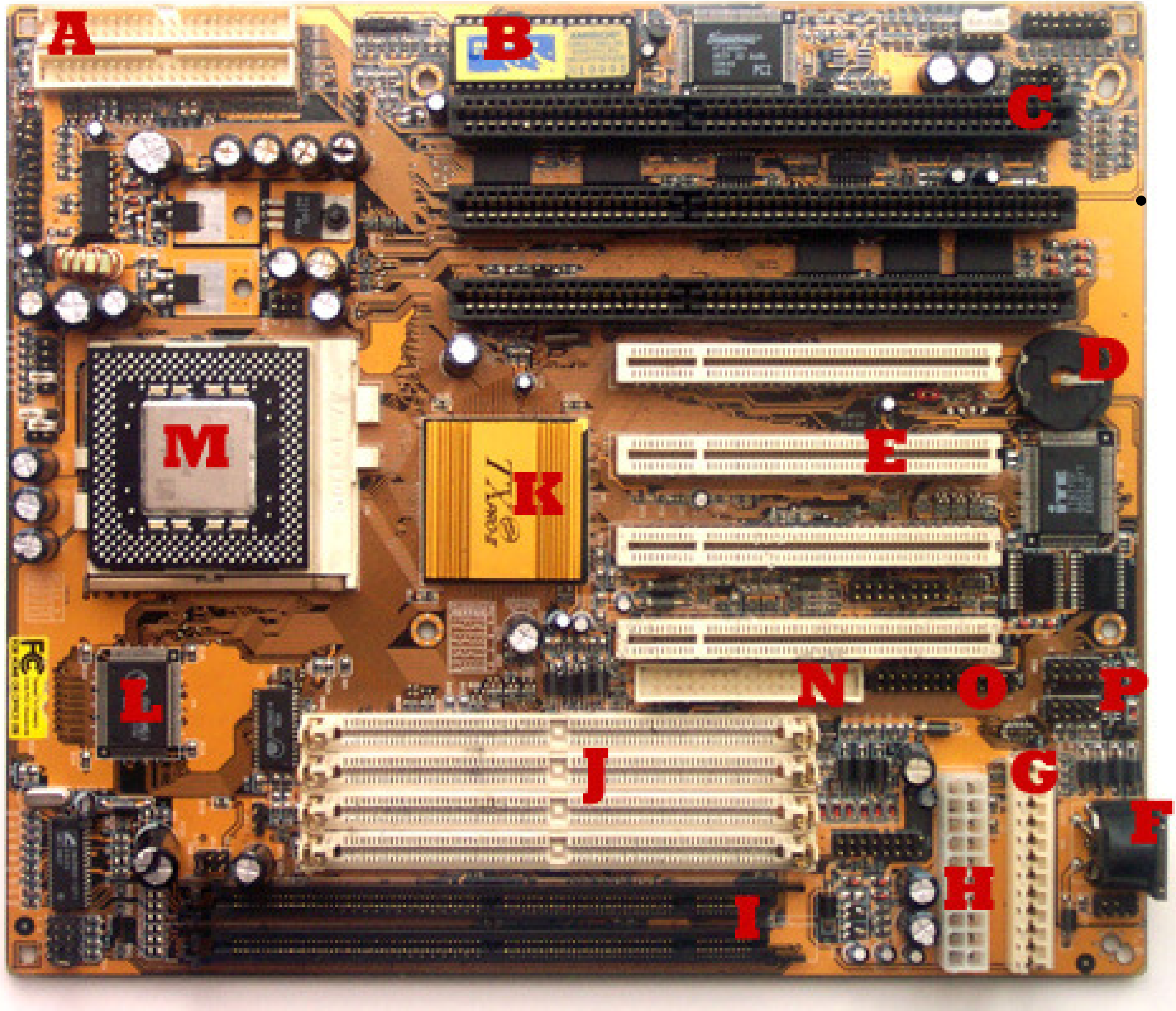
CO315 Computer Interfacing
interfacing through computer busses
XT and ISA Buses

Kamalanath Samarakoon

North/South Bridge Architecture

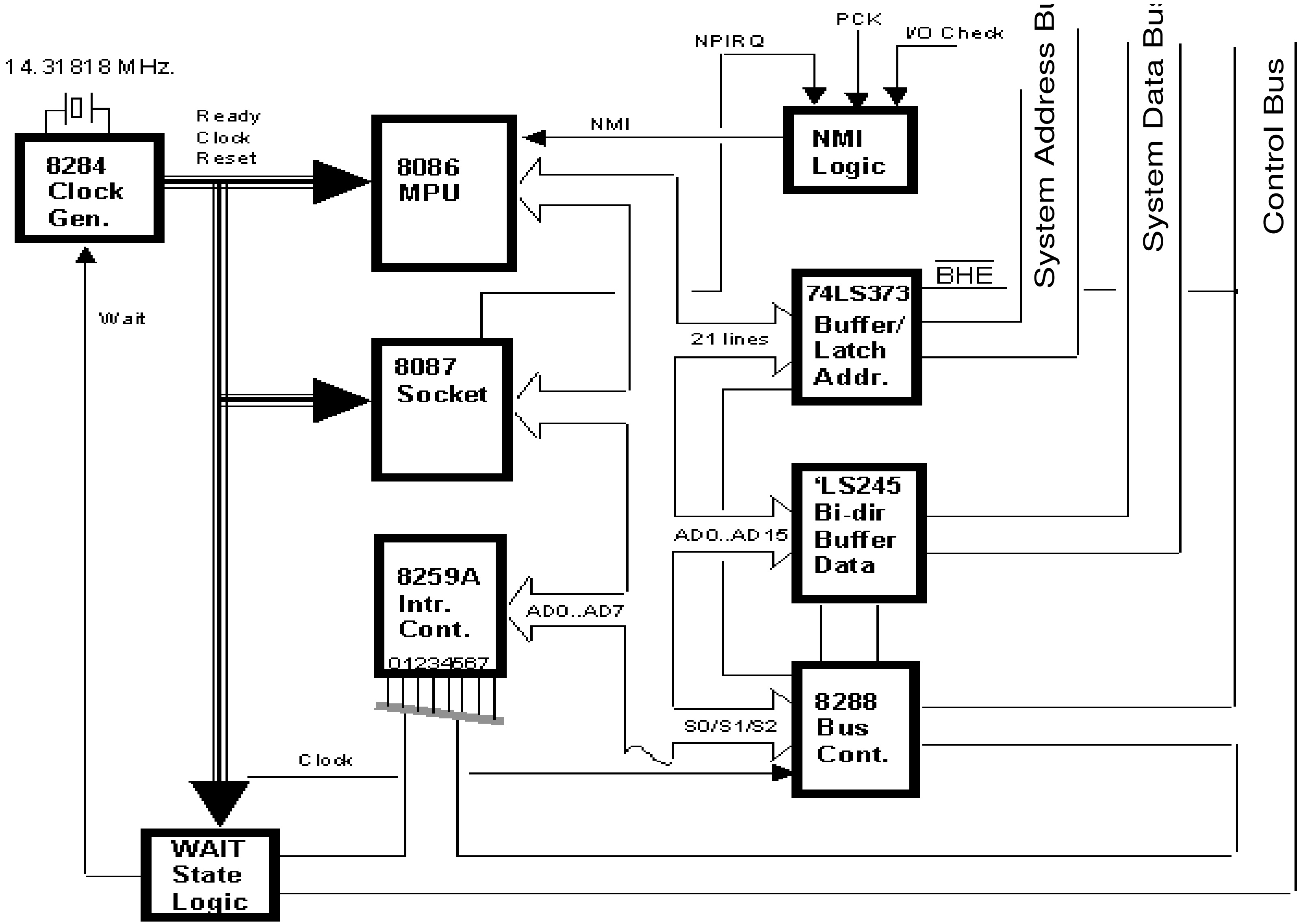


A Motherboard

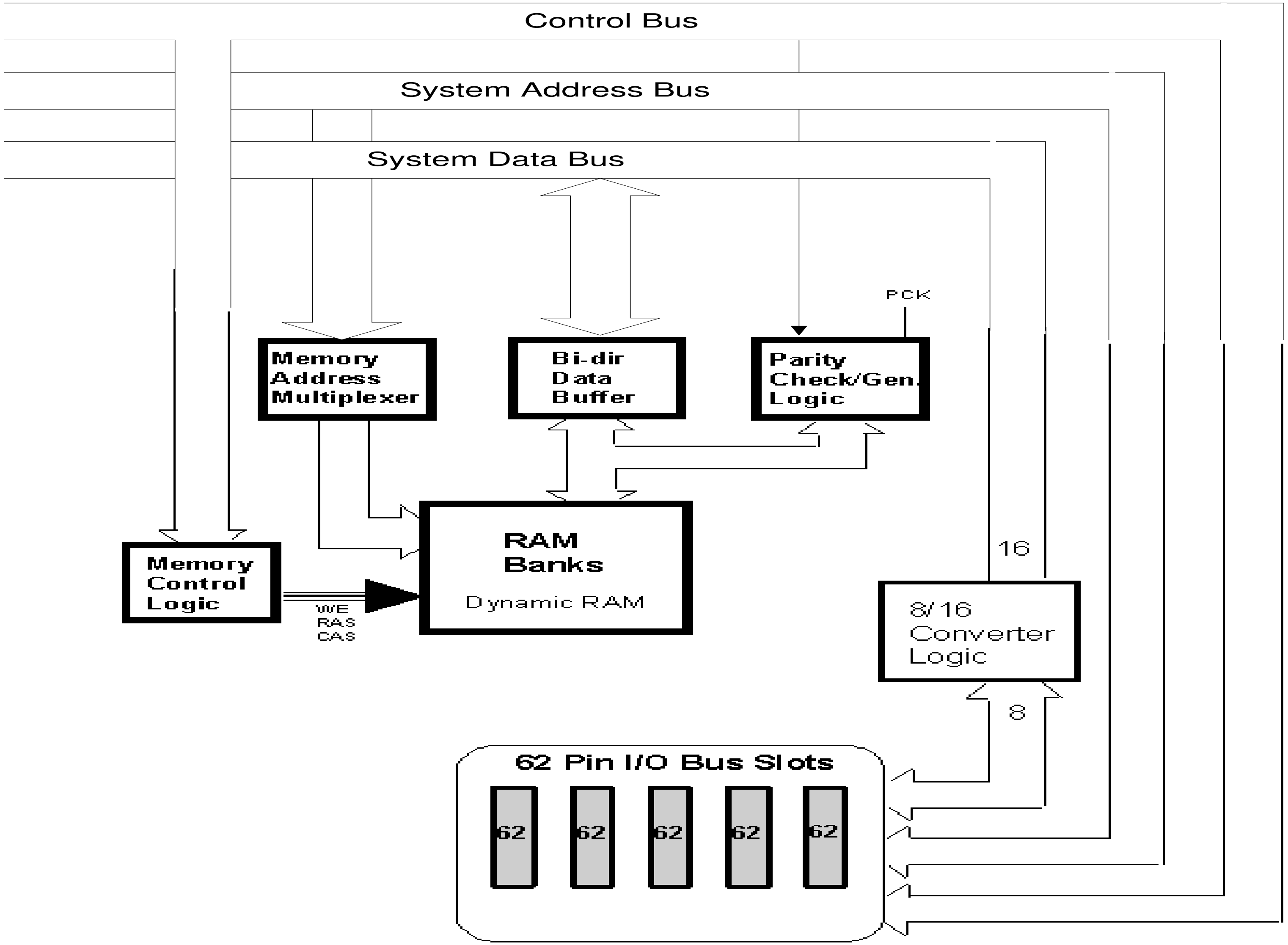


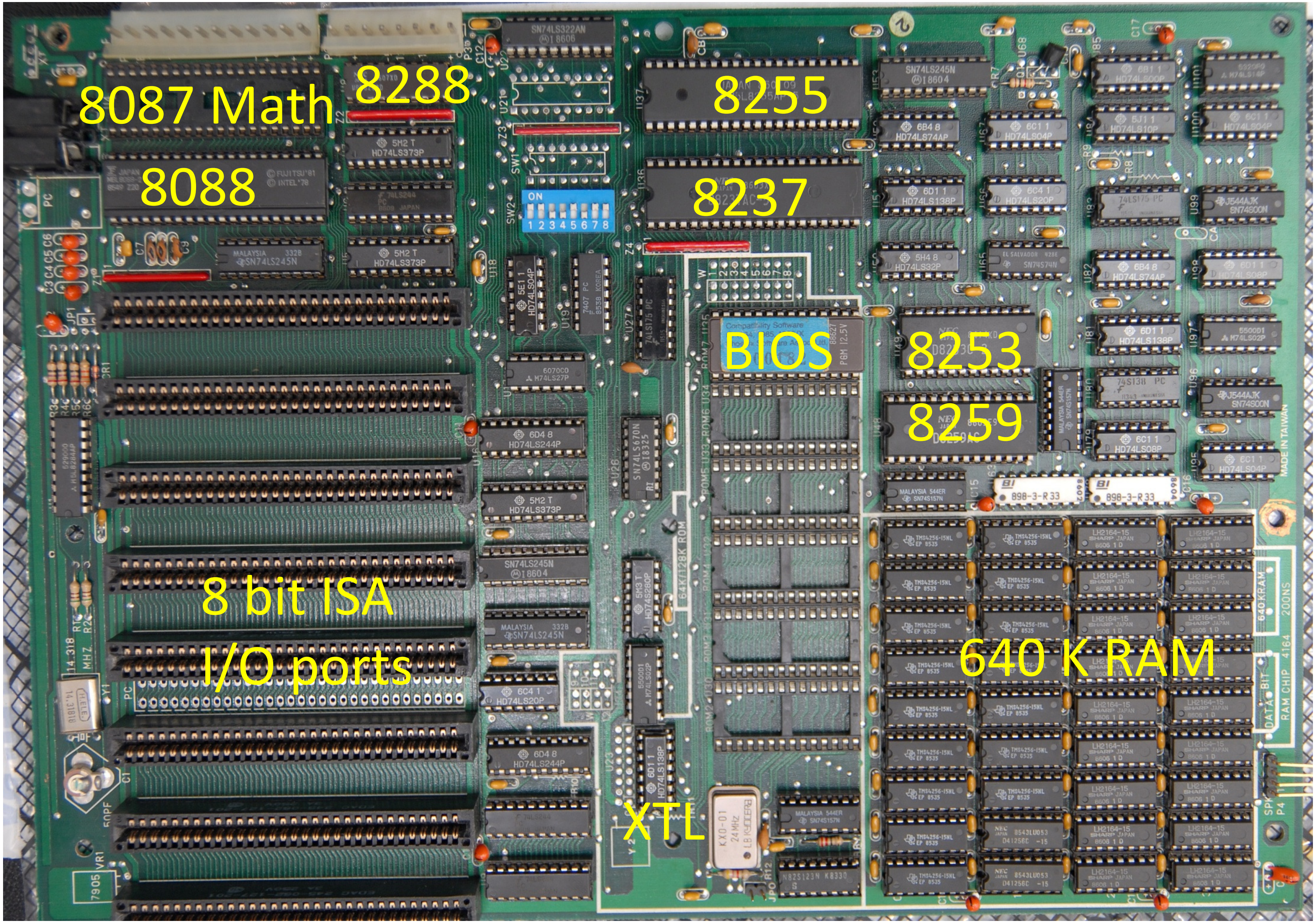
- A. Primary and Secondary IDE Controllers
B. ROM/BIOS Chip
C. ISA Slot
D. CMOS Battery
E. PCI Slot
F. AT DIN/5 Keyboard Connector
G. AT Power Socket
H. ATX Power Socket
I. 168 DIMM Socket
J. 72 SIMM Socket
K. Heat Sink Chip Set
L. L2 Cache Chip Set
M. CPU PGA on a CPU Socket 7
N. Floppy Drive Controller
O. PRN for Parallel Port
P. COM for Serial Ports

Early PC Processor Circuitry

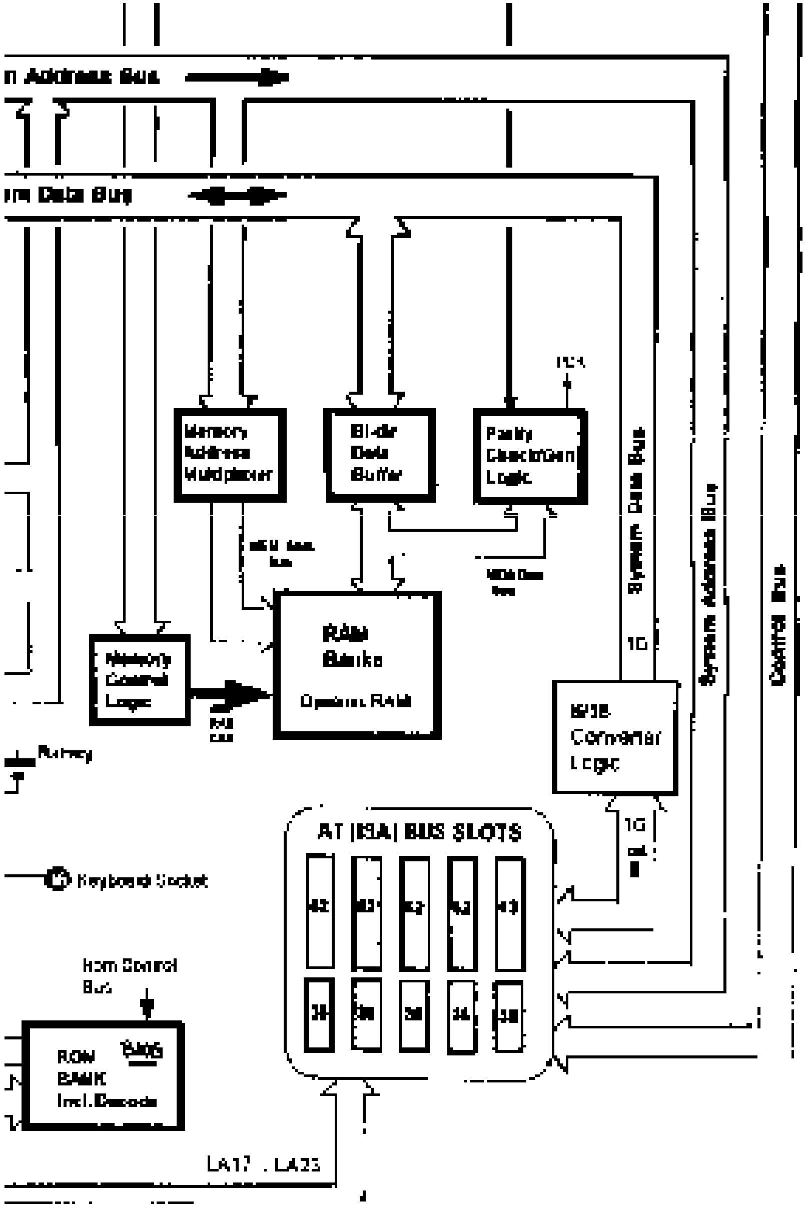


PC/XT: RAM and I/O Bus Slots




8087 Math
8288
8255
8088
8237
BIOS
8253
8259
8 bit ISA
I/O ports
640 K RAM
XTL

PC/AT RAM Memory and I/O Bus Slots

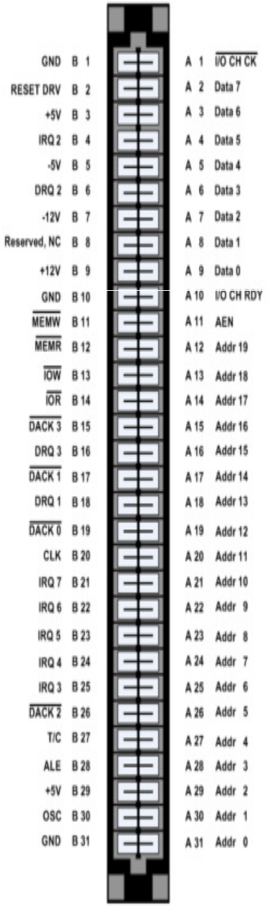


PC/AT Bus Standards

- The **AT-Bus** is a 16-bit (data) bus and is based on the 8-bit **PC-Bus**
- The **ISA** (Industry Standard Architecture) standard formalised the AT-Bus (16-bit) standard as an industry standard, and this standard is commonly referred to as the '**ISA Bus**' standard
- A more advanced bus, the **EISA** bus (Enhanced Industry Standard Architecture) is a standard, which is upwards compatible to the ISA bus

8 Bit XT Bus

8 Bit XT Bus – top view



8-Bit ISA Bus connector

1	GND	IO CHK
2	RESET	
3	+5V	
4	IRQ9	
5	-5V	
6	DRQ2	
7	-12V	
8	OWS	
9	+12V	
10	GND	IO RDY
11	MEMW	AEN
12	MEMR	
13	IOW	
14	IOR	
15	DACK3	
16	DRQ3	
17	DACK1	
18	DRQ1	
19	DACK0	
20	CLOCK	
21	IRQ7	
22	IRQ6	
23	IRQ5	
24	IRQ4	
25	IRQ3	
26	DACK2	
27	T/C	
28	ALE	
29	+5V	
30	OSC	
31	GND	

D0-D7

A0-A19

ISA Bus Connector Contains

8- bit Data Bus

Demultiplexed 20-bit address Bus

I/O and Memory Control Signals

Interrupt Request Lines (IRQ2->IRQ9)

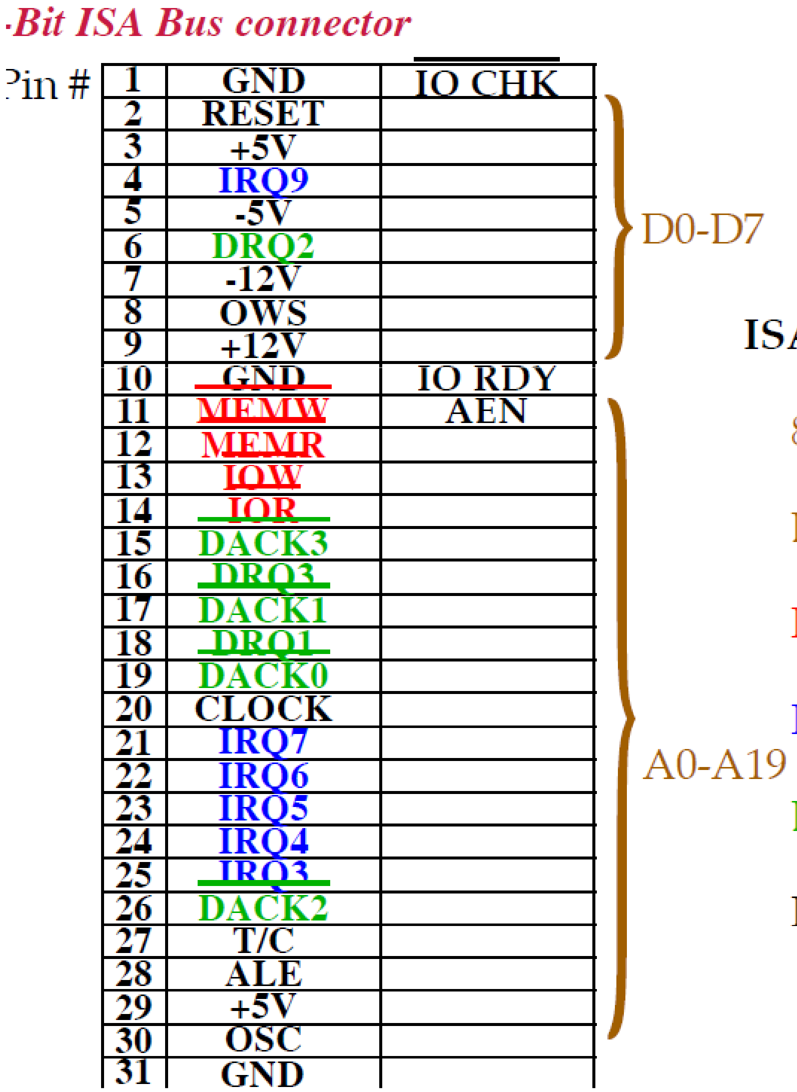
DMA channels 1-3 Control Signals

Power, RESET and misc. signals

Pin configuration

Pin	Name	Description
A1	/I/O CH CK	I/O channel check; active low=parity error
A10	I/O CH RDY	I/O Channel ready, pulled low to lengthen memory cycles
A11	AEN	Address enable; active high when DMA controls bus
B2	RESET	Active high to reset or initialize system logic
B4	IRQ2	Interrupt Request 2
B6	DRQ2	DMA Request 2
B8	/NOWS	No WaitState
B11	/SMEMW	System Memory Write
B12	/SMEMR	System Memory Read
B13	/IOW	I/O Write
B14	/IOR	I/O Read
B15	/DACK3	DMA Acknowledge 3
B16	DRQ3	DMA Request 3
B17	/DACK1	DMA Acknowledge 1
B18	DRQ1	DMA Request 1
B19	/REFRESH	Refresh
B20	CLOCK	System Clock (67 ns, 8-8.33 MHz, 50% duty cycle)
B26	/DACK2	DMA Acknowledge 2
B27	T/C	Terminal count; pulses high when DMA term. count reached
B28	ALE	Address Latch Enable
B30	OSC	High-speed Clock (70 ns, 14.31818 MHz, 50% duty cycle)
C1	SBHE	System bus high enable (data available on SD8-15)
C9	/MEMR	Memory Read (Active on all memory read cycles)
C10	/MEMW	Memory Write (Active on all memory write cycles)
D1	/MEMCS16	Memory 16-bit chip select (1 wait, 16-bit memory cycle)
D2	/IOCS16	I/O 16-bit chip select (1 wait, 16-bit I/O cycle)
D8	/DACK0	DMA Acknowledge 0
D9	DRQ0	DMA Request 0
D10	/DACK5	DMA Acknowledge 5
D11	DRQ5	DMA Request 5
D12	/DACK6	DMA Acknowledge 6
D13	DRQ6	DMA Request 6
D14	/DACK7	DMA Acknowledge 7
D15	DRQ7	DMA Request 7
D17	/MASTER	Used with DRQ to gain control of system

8 Bit XT Bus



Bit ISA Bus connector

Pin #		
1	GND	IO CHK
2	RESET	
3	+5V	
4	IRQ9	
5	-5V	
6	DRQ2	
7	-12V	
8	OWS	
9	+12V	
10	GND	IO RDY
11	MEMW	AEN
12	MEMR	
13	IOW	
14	IOR	
15	DACK3	
16	DRQ3	
17	DACK1	
18	DRQ1	
19	DACK0	
20	CLOCK	
21	IRQ7	
22	IRQ6	
23	IRQ5	
24	IRQ4	
25	IRQ3	
26	DACK2	
27	T/C	
28	ALE	
29	+5V	
30	OSC	
31	GND	

- IRQ Interrupt request

ISA Bus Connector Contains

8- bit Data Bus

Demultiplexed 20-bit address Bus

I/O and Memory Control Signals

Interrupt Request Lines (IRQ2->IRQ9)

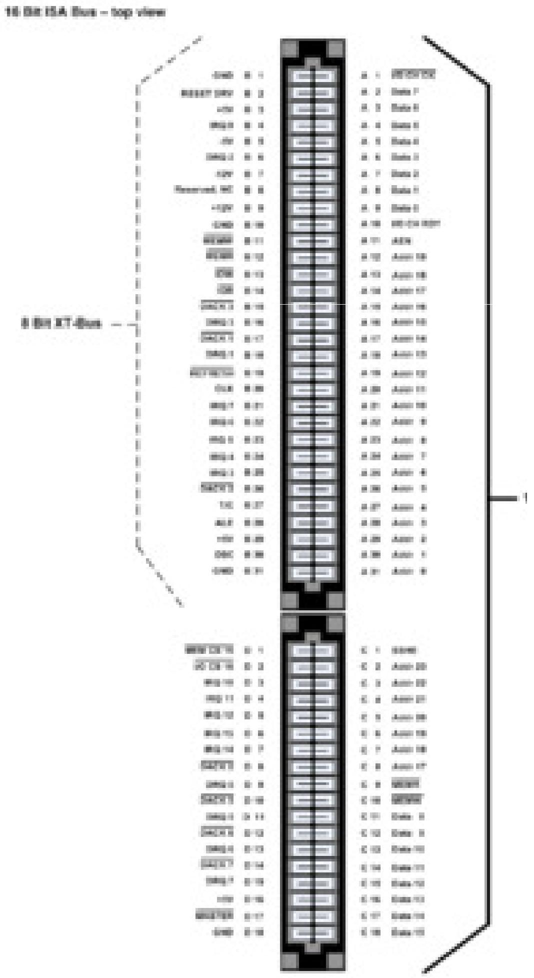
DMA channels 1-3 Control Signals

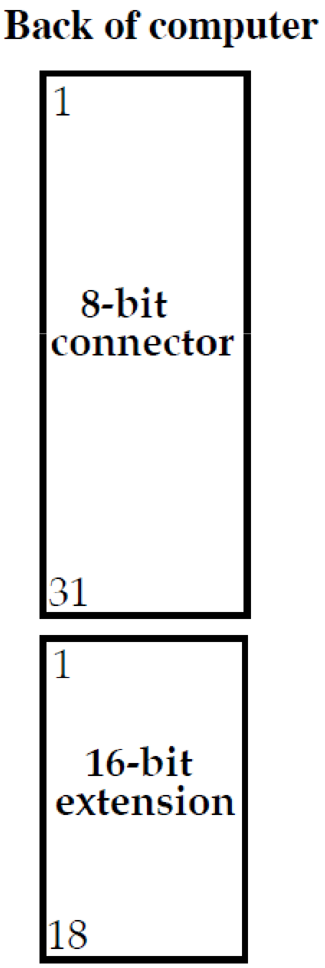
Power, RESET and misc. signals

8 Bit Bus Interface

- 4, 8-bit latches interfaced using an ISA interface for 32 bit parallel data.
- 74LS244 buffers used to ensure only **one lower power TTL load on the bus.**
- Loading is important as many cards can be connected on the bus.
- The DIP switch can be used to change the address thus avoiding address conflicts with other cards in the system.
- 16-bit ISA bus has an additional connector attached behind the 8-bit connector.
- Although 8 additional data bits, D8-D15, are available, the features most often used are the additional interrupt request and DMA request signals.

ISA Bus Connector





16-bit connector

1	$\overline{\text{MCS16}}$	BHE
2	$\overline{\text{IOCS16}}$	A23
3	IRQ10	A22
4	IRQ11	A21
5	IRQ12	A20
6	IRQ15	A19
7	IRQ14	A18
8	$\overline{\text{DACK0}}$	A17
9	DRQ0	$\overline{\text{MEMR}}$
10	$\overline{\text{DACK5}}$	$\overline{\text{MEMW}}$
11	DRQ5	D8
12	$\overline{\text{DACK6}}$	D9
13	DRQ6	D10
14	$\overline{\text{DACK7}}$	D11
15	DRQ7	D12
16	+5V	D13
17	MASTER	D14
18	GND	D15

ISA Pin configuration

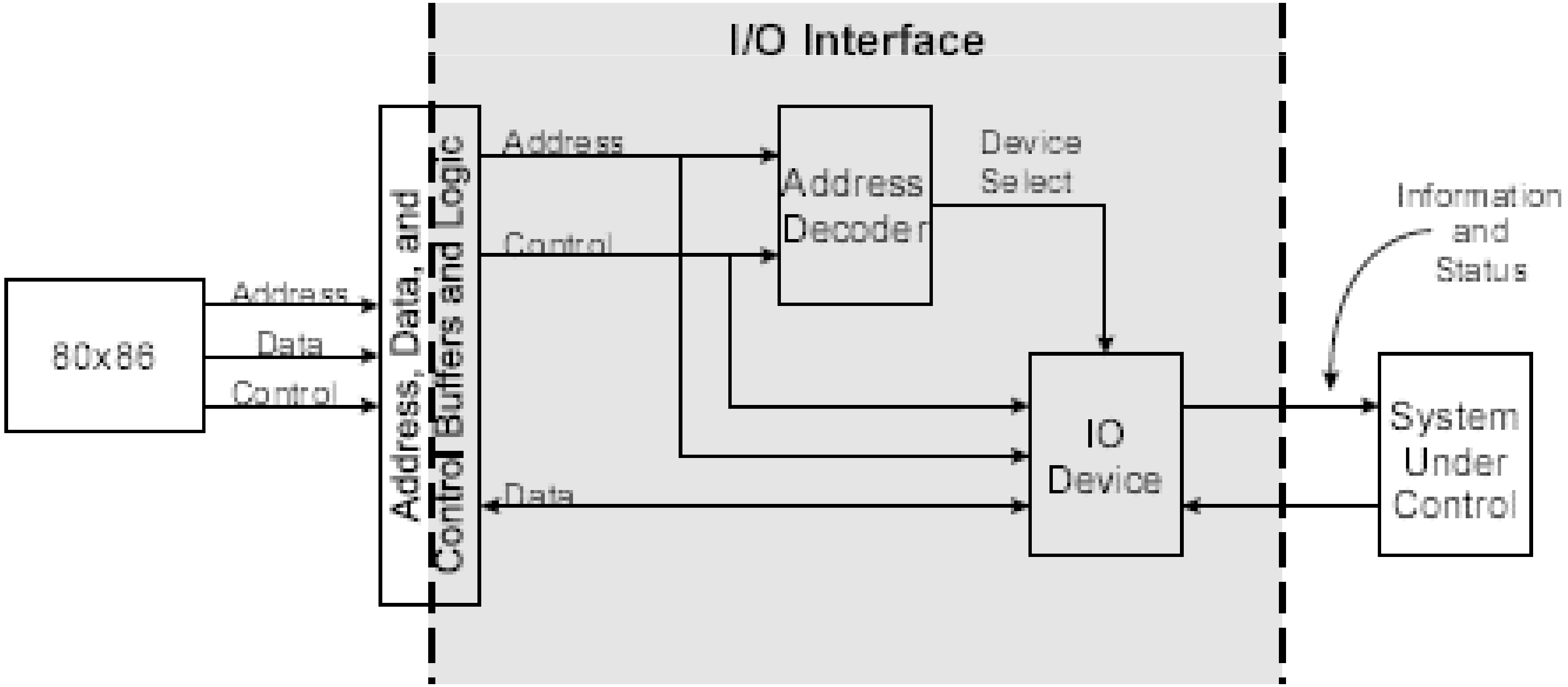
ISA Pin Descriptions
DATA - D0-D7 bi-irectional data bus.
The data bus is driven by a bidirectional buffer that is only enabled at times corresponding to data availability (iowc/, iorc/, memw/, memr).
ADDRESS - A0-A19 fully demultiplexed and stable during a full bus cycle.
ALE - Address Latch Enable - This output signal comes directly from the bus controller IC and provides timing information for decoding the address lines. It is not needed for bus decoding since the address lines are already demultiplexed on the ISA bus.
AEN - Address Enable - This output signal allows the IO device to distinguish between processor bus cycles and DMA bus cycles. A high on AEN indicates that a DMA cycle is occuring and that the address, data and control lines are under the control of the DMA controller. Peripheral IO devices that do not have DMA capability should insure that they only decode address that are generated by the processor (AEN='0') and not a DMA controller.
IO CHANNEL RDY - This normally high input line can be pulled low by a slow device to inserve wait states.
IO CHANNEL CHECK - This normally high input line is pulled low to indicate a memory or IO device pari6ty error. In turn, the parity error will cause a non-maskable interrupt (NMI) of Type 2 to occur.
RESET DRV - This output signal is active high during power-on and can be used to reset or initialize IO devices.
DRQ1-DRQ3 - These input lines are connected to the corresponding DMA request pins on the DMA controller. Raising a selected line generates a DMA request. DMA channel 0 is reserved for memory refreshing.
DACK0-DACK3 - These four active low output lines provide DMA acknowledge signals for the four DMA channesl. DACK0 can not be used by other devices but is useful since it indicates that a DRAM memory refresh cycle is occuring.
IRQ2-IRQ7 - Interrupt request lines are connected directly to the PC interrupt controller. A line should be held high until the request is serviced by the appropriate interrupt service routine. IRQ0 and IRQ1 are reserved for use on the system board by the time of day and keyboard interupts and are not generally available. Other lines may be used by other devices as well.
OSC - The output of the system oscillator, typically around 8-20MHz (traditionally 14.318MHz).
CLK - Traditionally the processor clock signal with a 33% duty cycle (4.77MHz). More recent PC systems will have frequencies in the range of 4-10MHz. The frequency of this clock output should always be measured to insure that it is used properly.
Power Supply Lines - All voltages available on the system board are available on the ISA bus. These include +5Vdc and ground, +12 and -12Vdc, and -5Vdc. The current draw from each of these lines should be carefully controlled.

Designing an ISA interface

At this point, we know a lot about how the software architecture of the IBM PC works. We've explored the whole operating system hierarchy and understand how to access hardware elements.

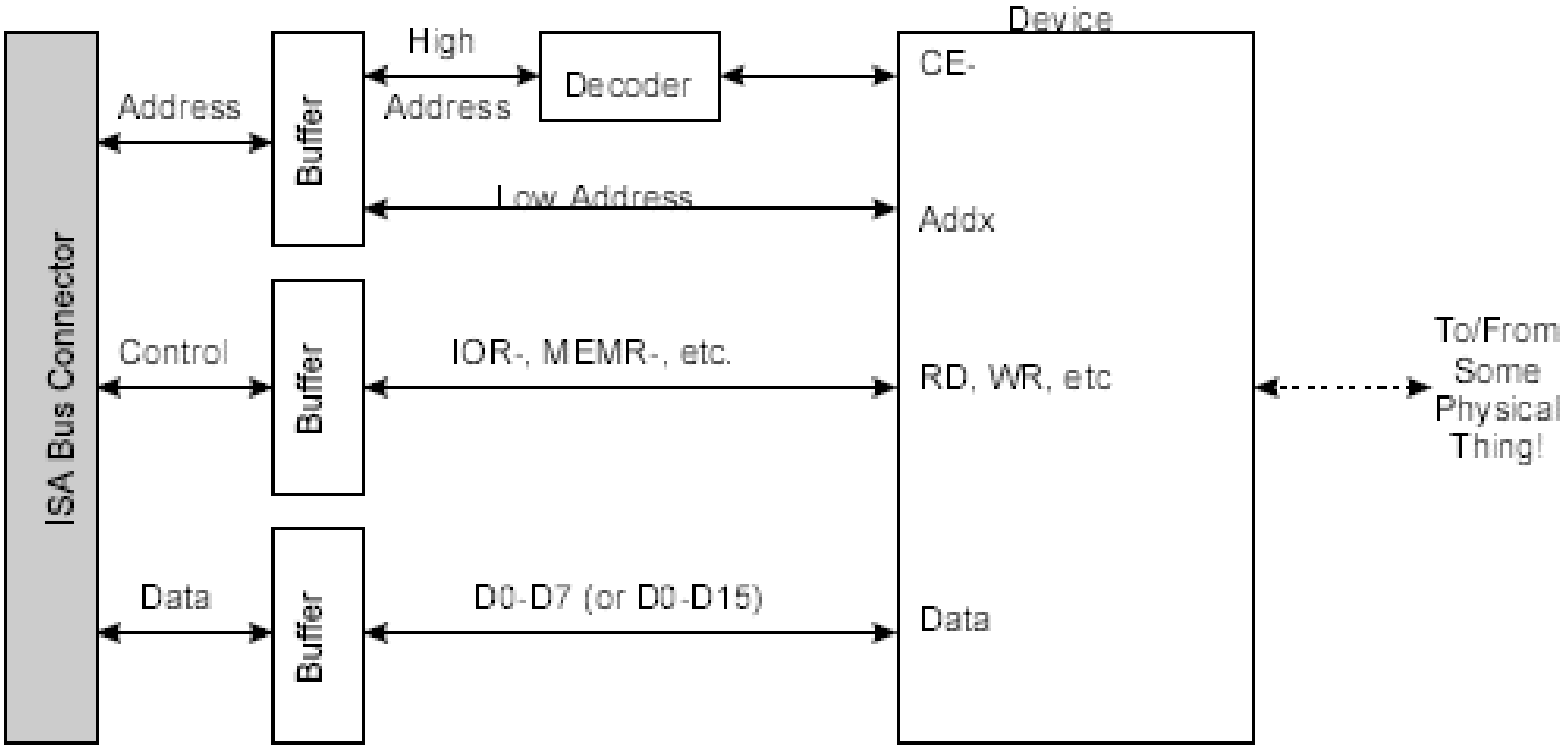
We've also taken a quick look at the ISA Bus and at the idea of Address Decoding as a method for locating hardware devices within the memory space of the computer.

Now, let's put it all together by building a (very) simple IO port!



A Generic ISA-Bus Interface Card Design

The basic block diagram of an ISA-Bus interface includes buffers (to isolate the card from the ISA Bus), Address and control decoding (to make sure the card responds to the desired bus condition) and the stuff being controlled (memory or other devices).

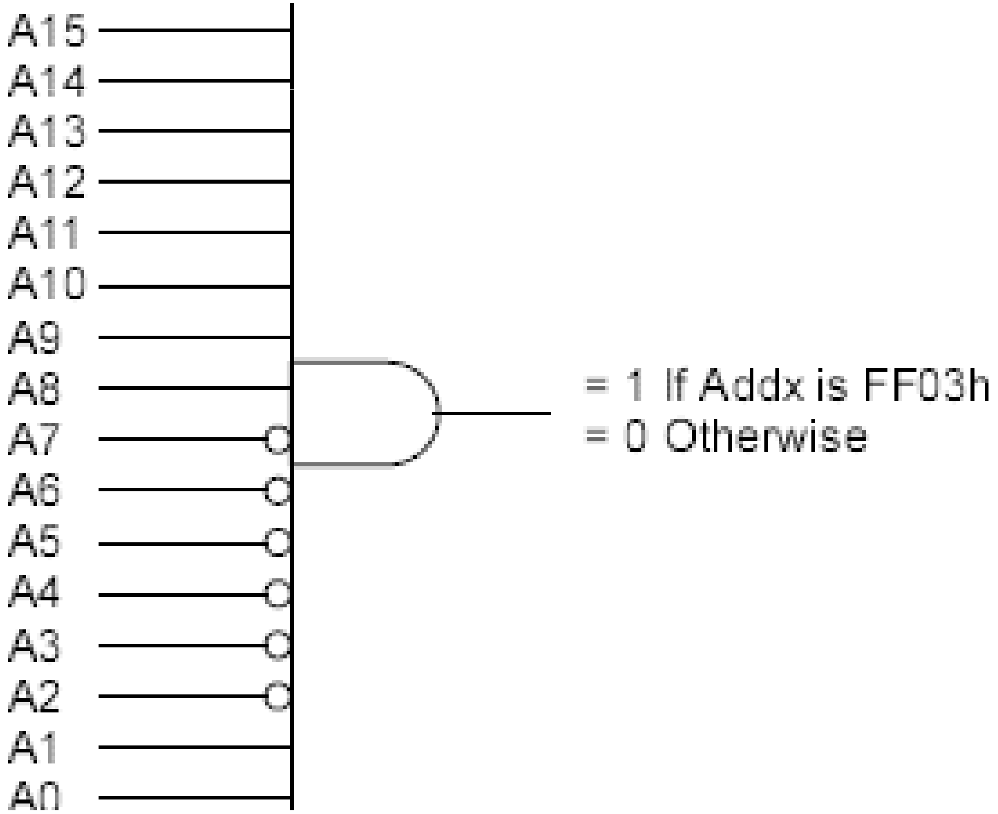


Address Decoding

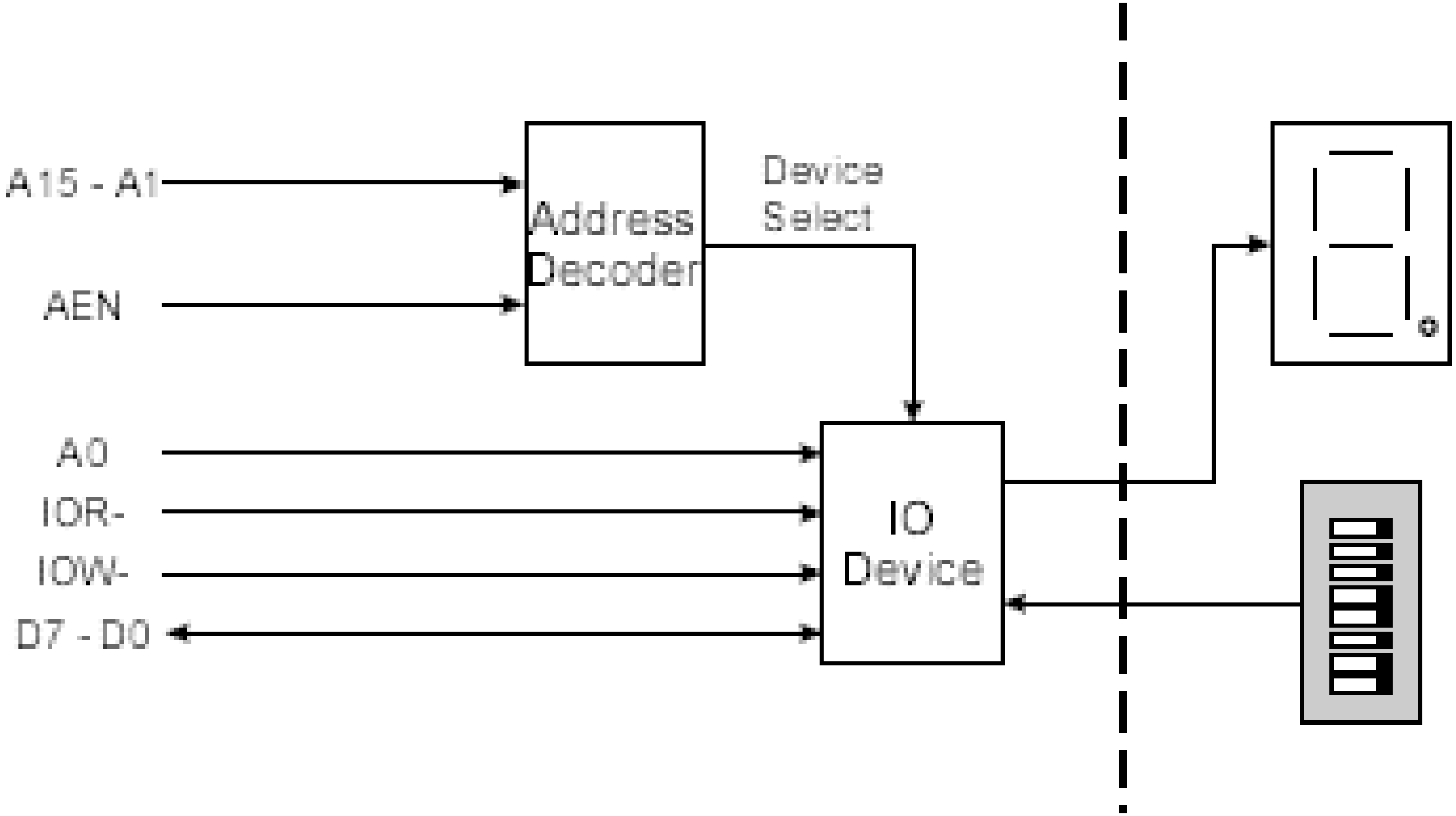
Address decoding is the way we can locate hardware devices at particular locations in the address space (memory or port) of a microprocessor. Once again, this looks complicated, but it really isn't if you take it in bits (pun intended).

Typically, an address decoder is a combinational circuit that enables an address select line if an appropriate access condition exists.

Suppose we want to decode address FF03h:

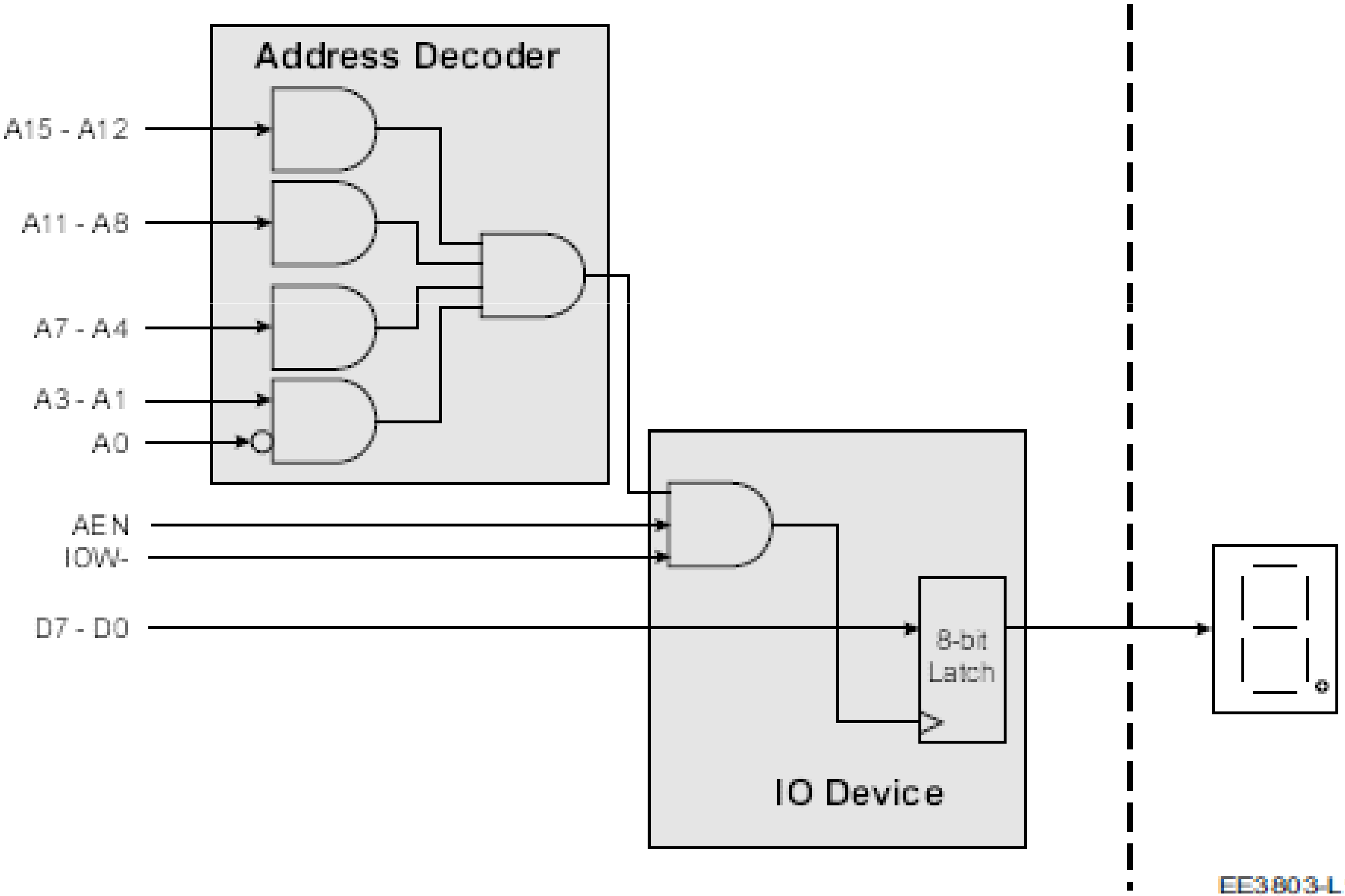


Use of ISA control lines

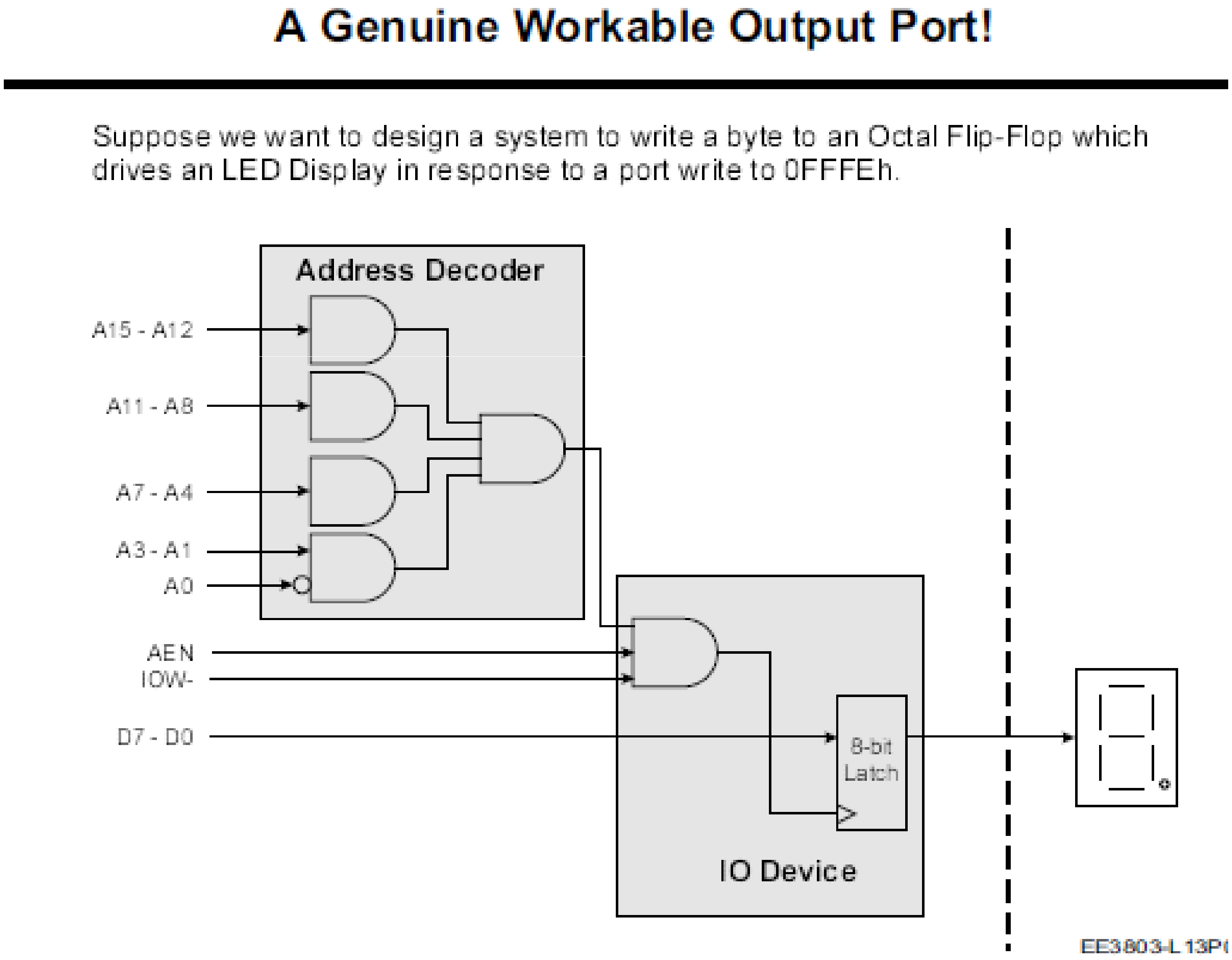


A Genuine Workable Output Port!

Suppose we want to design a system to write a byte to an Octal Flip-Flop which drives an LED Display in response to a port write to 0FFFEh.



Make it input port



A Genuine Workable Output Port!

Suppose we want to design a system to write a byte to an Octal Flip-Flop which drives an LED Display in response to a port write to 0FFFEh.

EE3803-L13P(

How to make bidirectional?

•Up to now all interfces have been unidirectional. i.e, data was transferred in only one direction.

•That's fine if we don't need to get any feedback from our system.

•If we do, we have to find a way to communicate in both directions on the same bus.

•Remember, only one source can drive the bus at any given time.

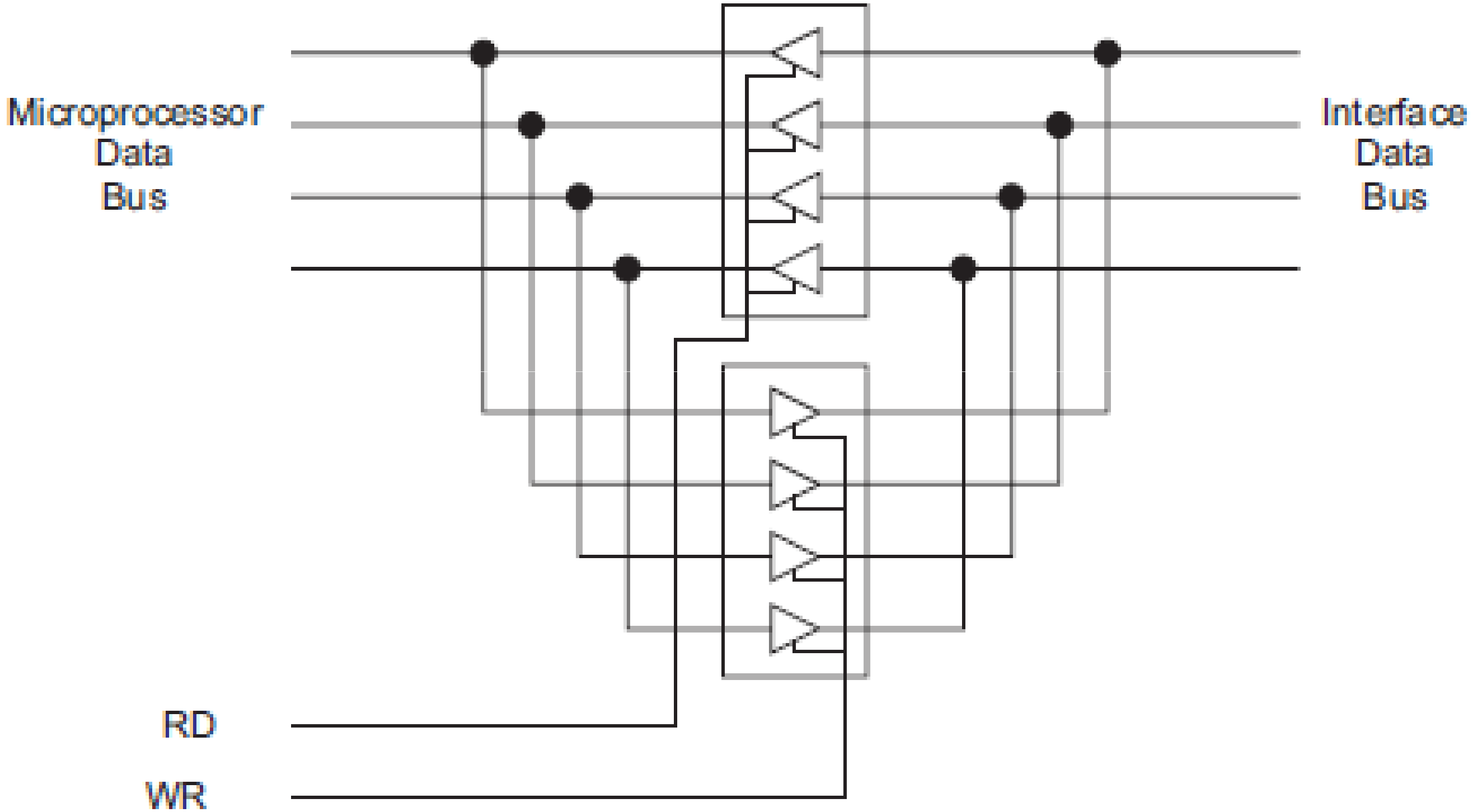
•If two drivers are on the bus at the same time it is called contention.

•If contention is allowed to last too long, devices on the bus may be destroyed.

•The usual way to construct a bi-directional interface is to use a tri-state buffer on the data lines:

Using Bi-Directional Buffers

Consider the following example:

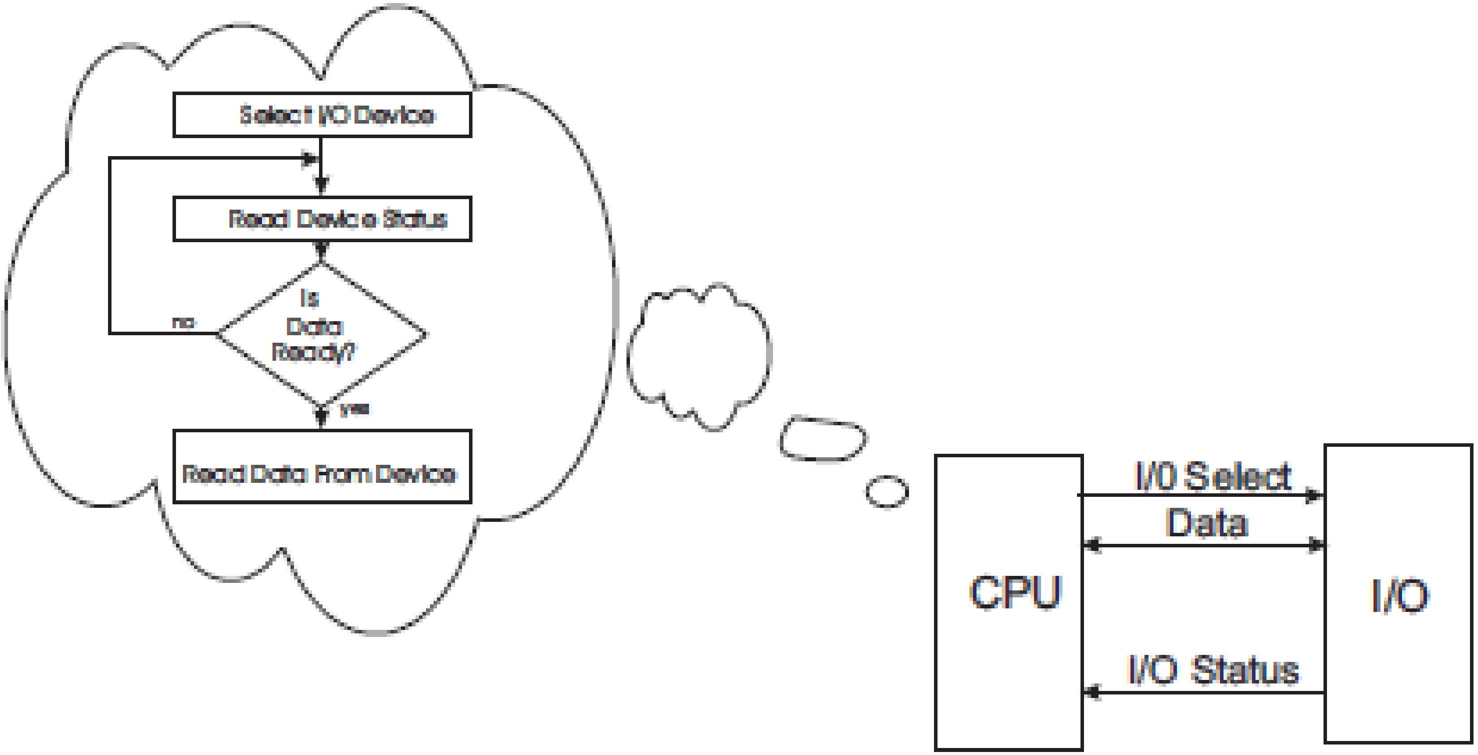


In this case, RD and WR are active high. If RD = 1, data goes from the interface to the processor. If WR = 1, data goes from the processor to the interface. What if WR = RD = 1??

Using Bi-Directional Communications

Now that we can send and receive data, we can design an interface that interacts with the outside world. However now we have a problem. Somehow, we must synchronize the events in the outside world with the events inside our processor.

A very simple way to do this is to use a technique called polling:

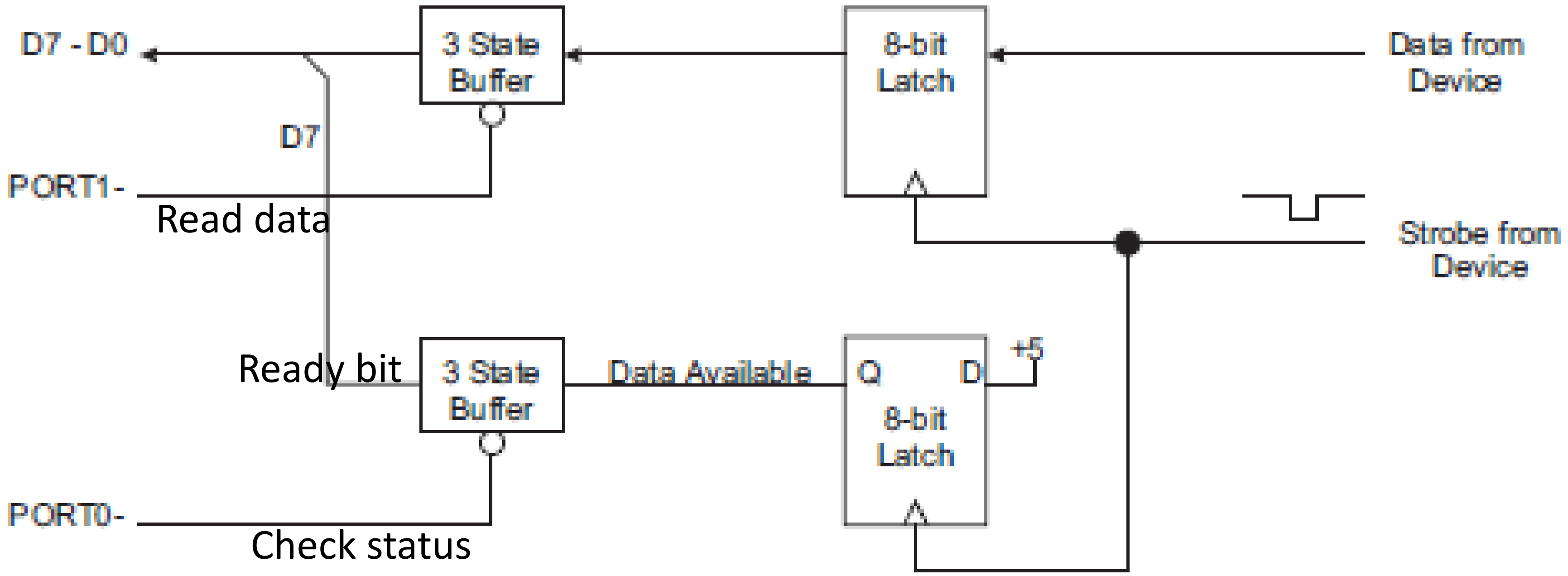


Handshaking Between The CPU And An IO Device

Ok, the idea of polling is pretty straightforward, but how does this "status" reporting work from a hardware level?

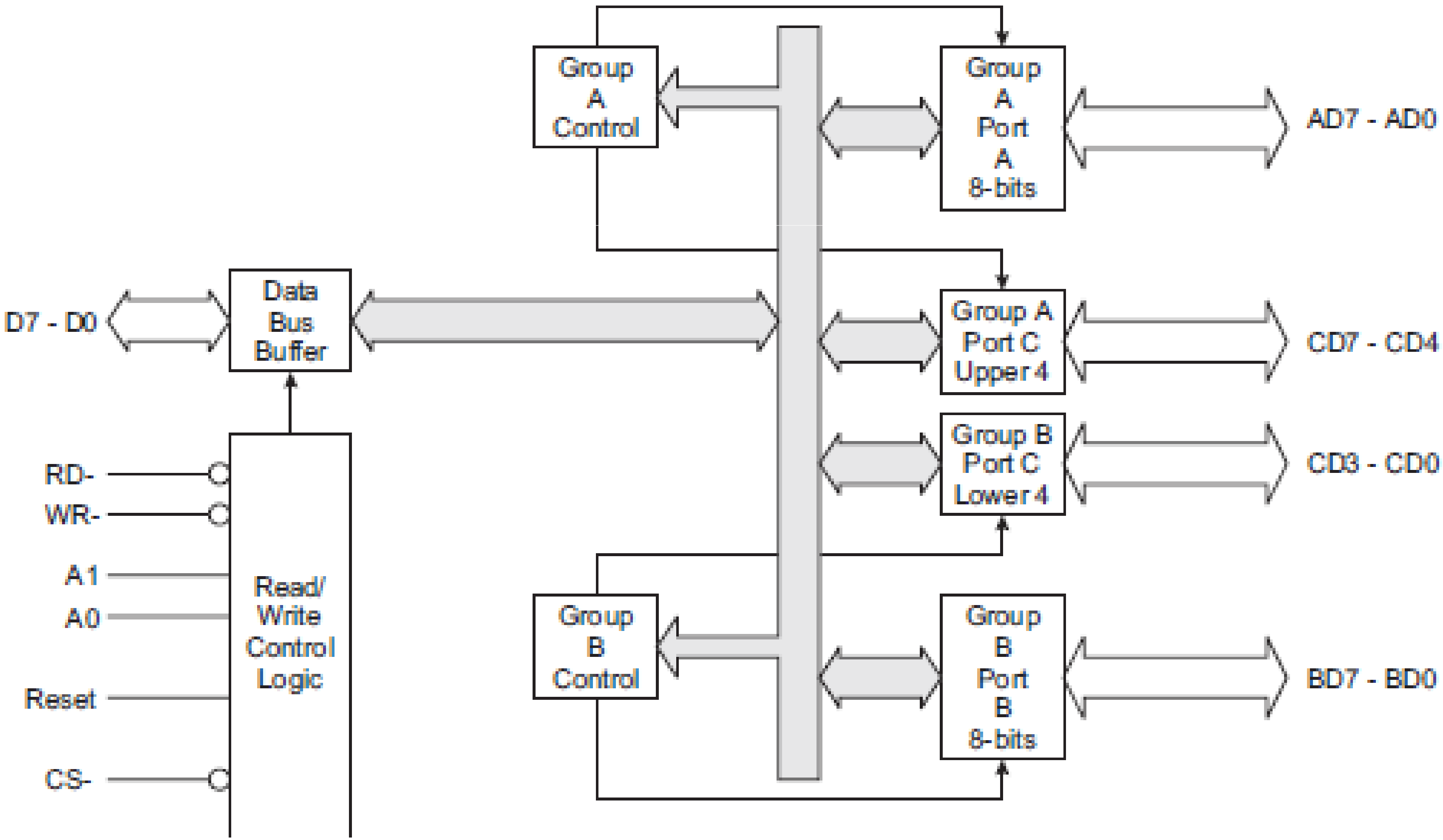
Well, somehow the IO device has to be able to "tell" the CPU when there is data that needs processing. That way the CPU just has to check a "ready" bit every once in a while.

Consider the following:



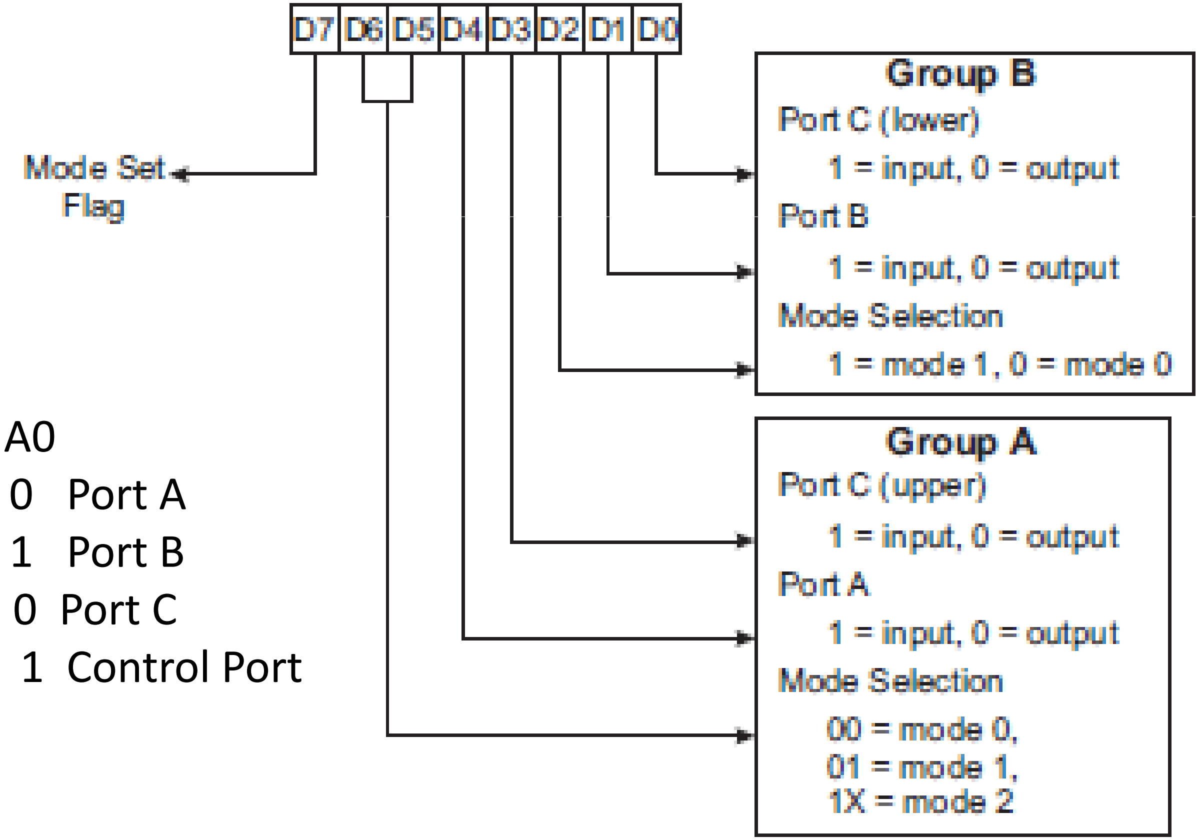
8255 PPI

Bidirectional IO is so important that there are devices that have been developed specifically to make the job easier. These devices can be programmed to act in a variety of ways. Consider the Intel 8255:



Programming The 8255

The ports of the 8255 are accessed based on the condition of a1 and a0. Address 0 accesses port A, 1 accesses B and 2 accesses C. If a1 and a0 are both 1, then the control port is accessed. This port determines exactly how the three ports will behave:



Group B

Port C (lower)
1 = input, 0 = output

Port B
1 = input, 0 = output

Mode Selection
1 = mode 1, 0 = mode 0

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

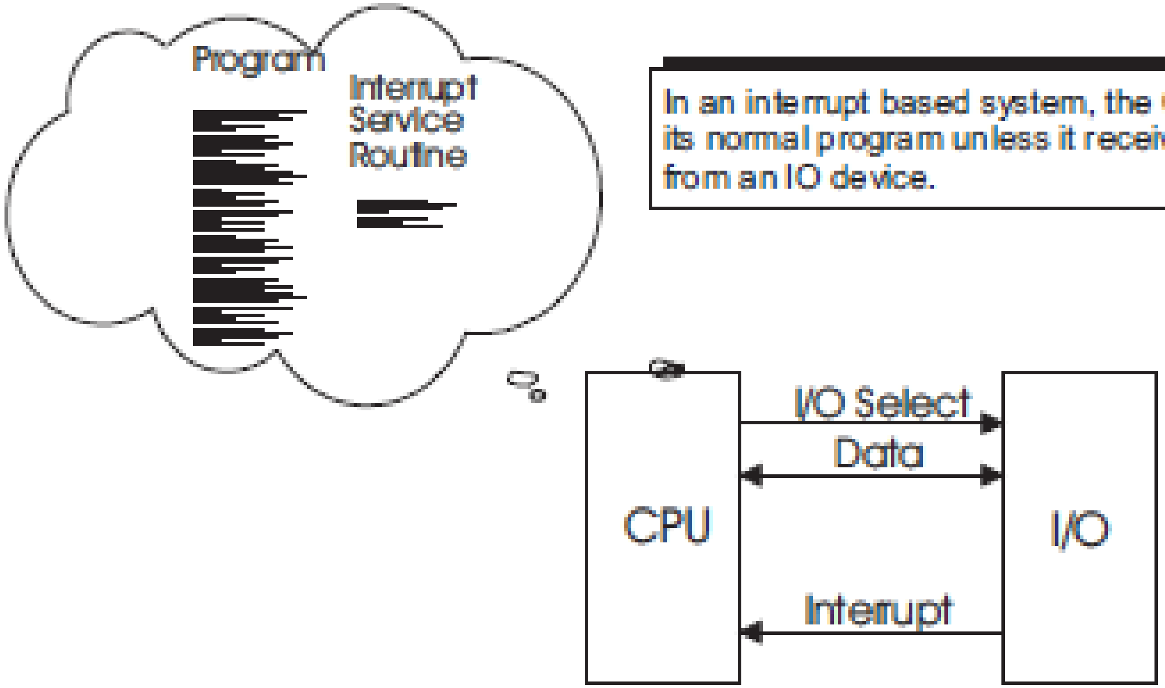
A1	A0	
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control Port

Interrupt Driven I/O: Redirecting the CPU On Demand

The problem with polling is that most of the CPU's resources may be spent waiting (for example, scanning for keystrokes in a word processor. Thus, it is desirable to devise a scheme where the CPU can do other things while it is "waiting" for input.

This ability is possible through the use of interrupts. Like the name suggests, interrupt-driven IO is a scheme where the CPU is temporarily interrupted from its normal sequence of tasks to go and perform some special operation. Once complete, the CPU returns to where it was before the interrupt was received.

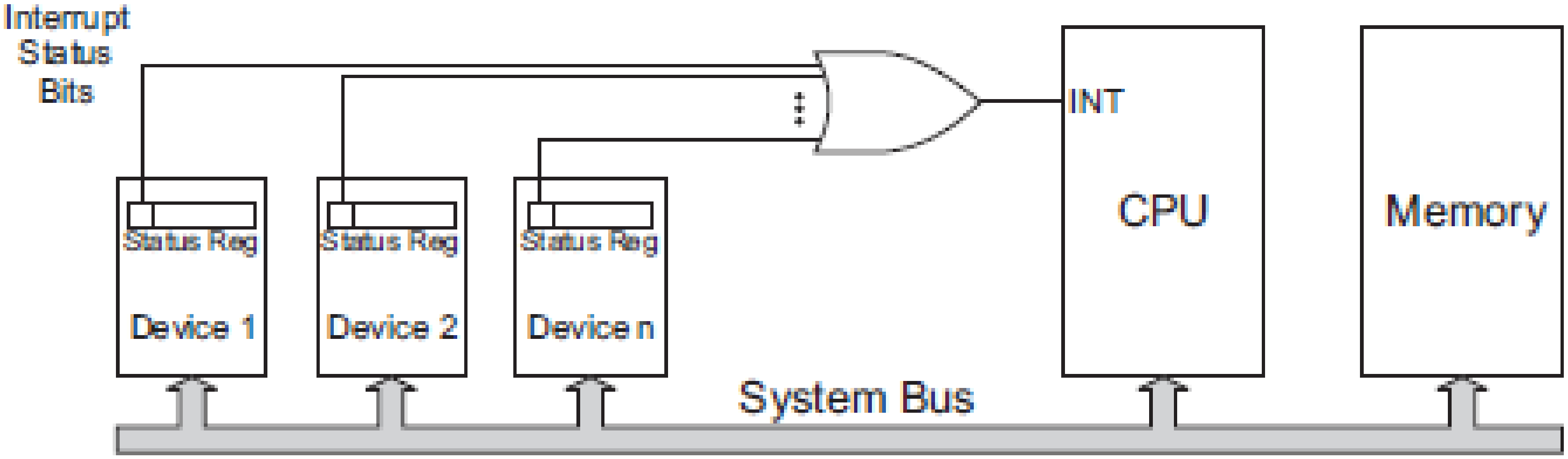
In an interrupt based system, the CPU executes its normal program unless it receives an interrupt from an IO device.



Two Flavors Of Interrupts

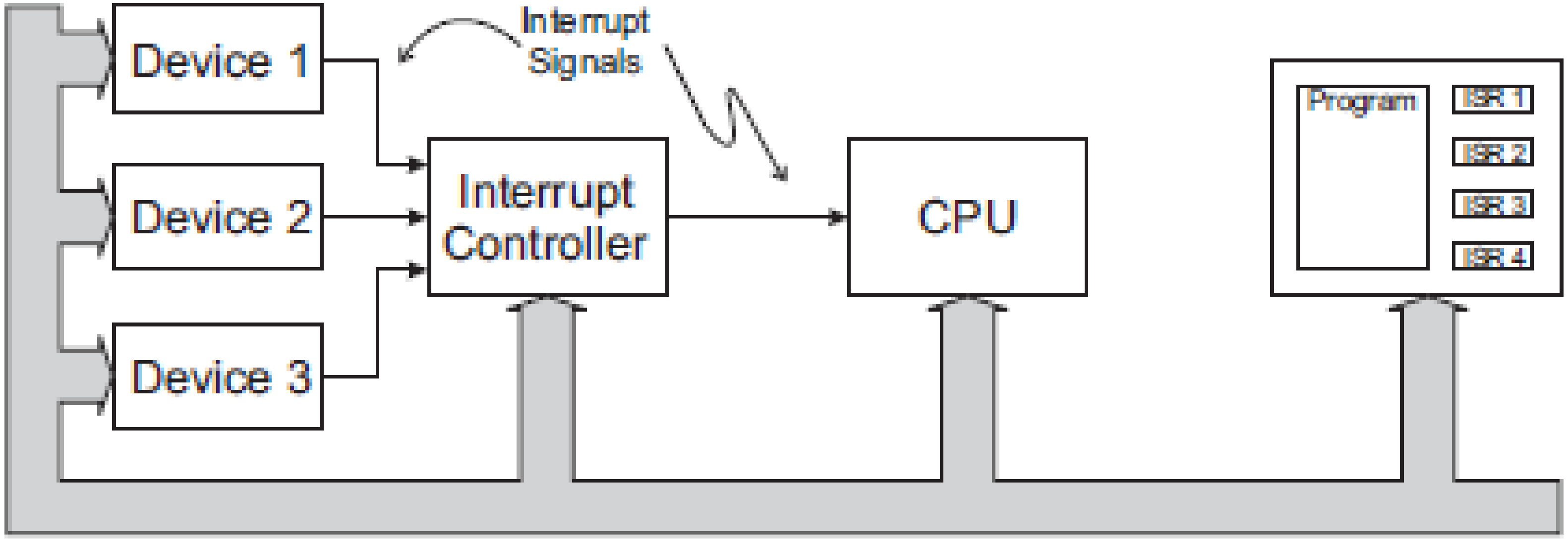
There are two basic ways of accomplishing interrupt-driven I/O. One is called the polled interrupt method, the other is called the vectored interrupt method. The major difference is the way in which the CPU determines where to find the proper interrupt service routine.

An example of polled interrupts is shown below:



A Vectored Interrupt Scheme

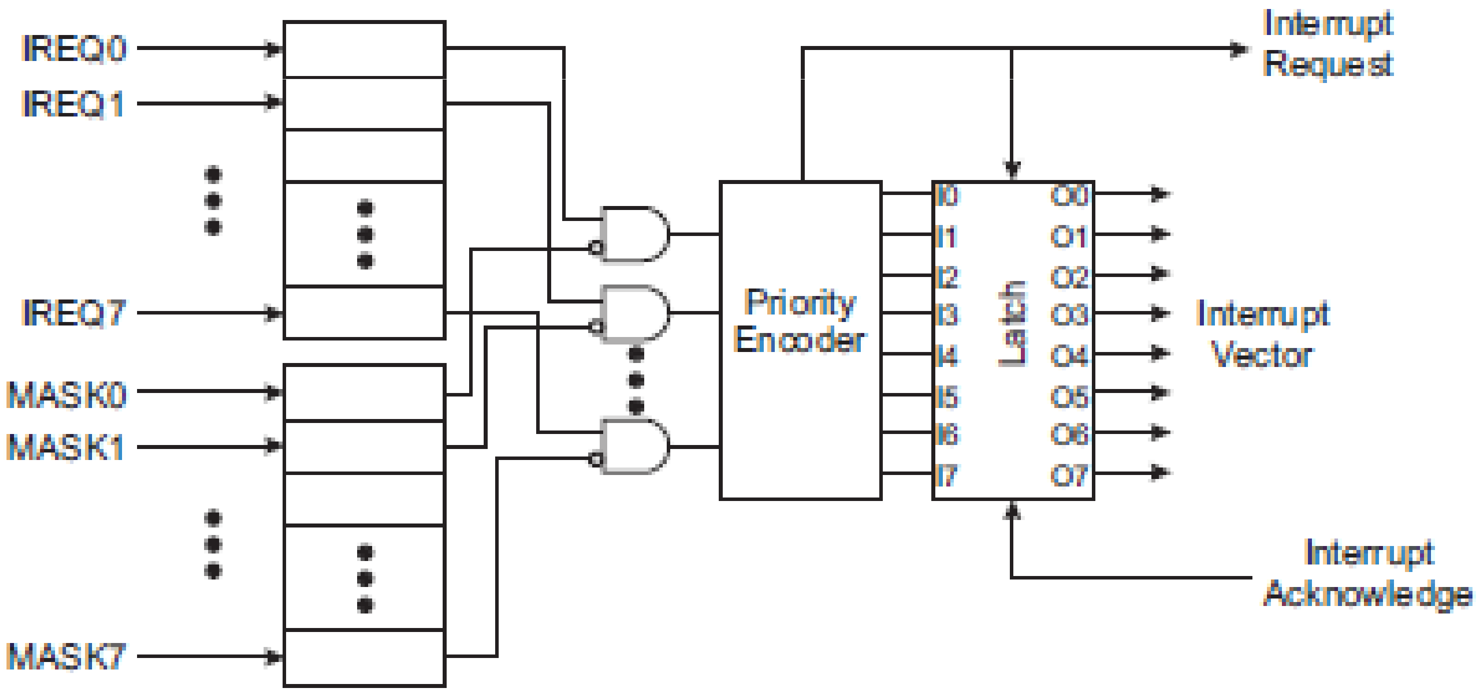
Rather than searching through each device to see where an interrupt came from, in a vectored interrupt system the interrupt controller tells the CPU where to locate the interrupt service routine:



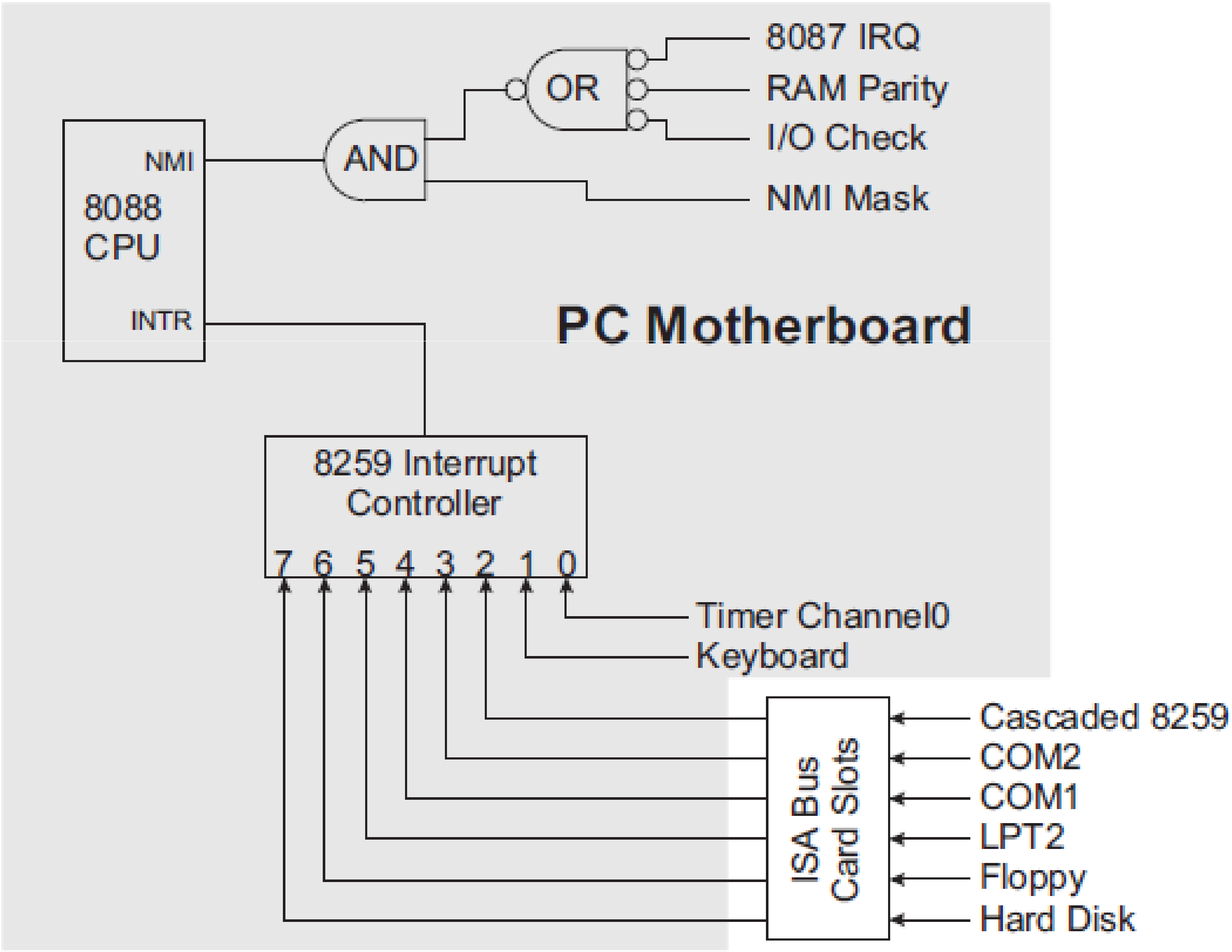
A Vectored Interrupt Controller

Usually, in a computer system, different interrupts will have different priorities. For example, a disk may have higher prioity than a typist. Why??

Therefore, it may be necessary to both tell the CPU where to go and to enforce some relative priority in the event that multiple interrupts happen at the same time.



The IBM PC Interrupt System



8059A PIC

- Is already covered
- Background http://en.wikipedia.org/wiki/Intel_8259
- Three registers
  - Interrupt Mask Register (IMR)
  - Interrupt Request Register (IRR)
  - In-Service Register (ISR)
- IRR maintains a mask of the current interrupts that are pending acknowledgement
- ISR maintains a mask of the interrupts that are pending an EOI
- IMR maintains a mask of interrupts that should not be sent an acknowledgement.

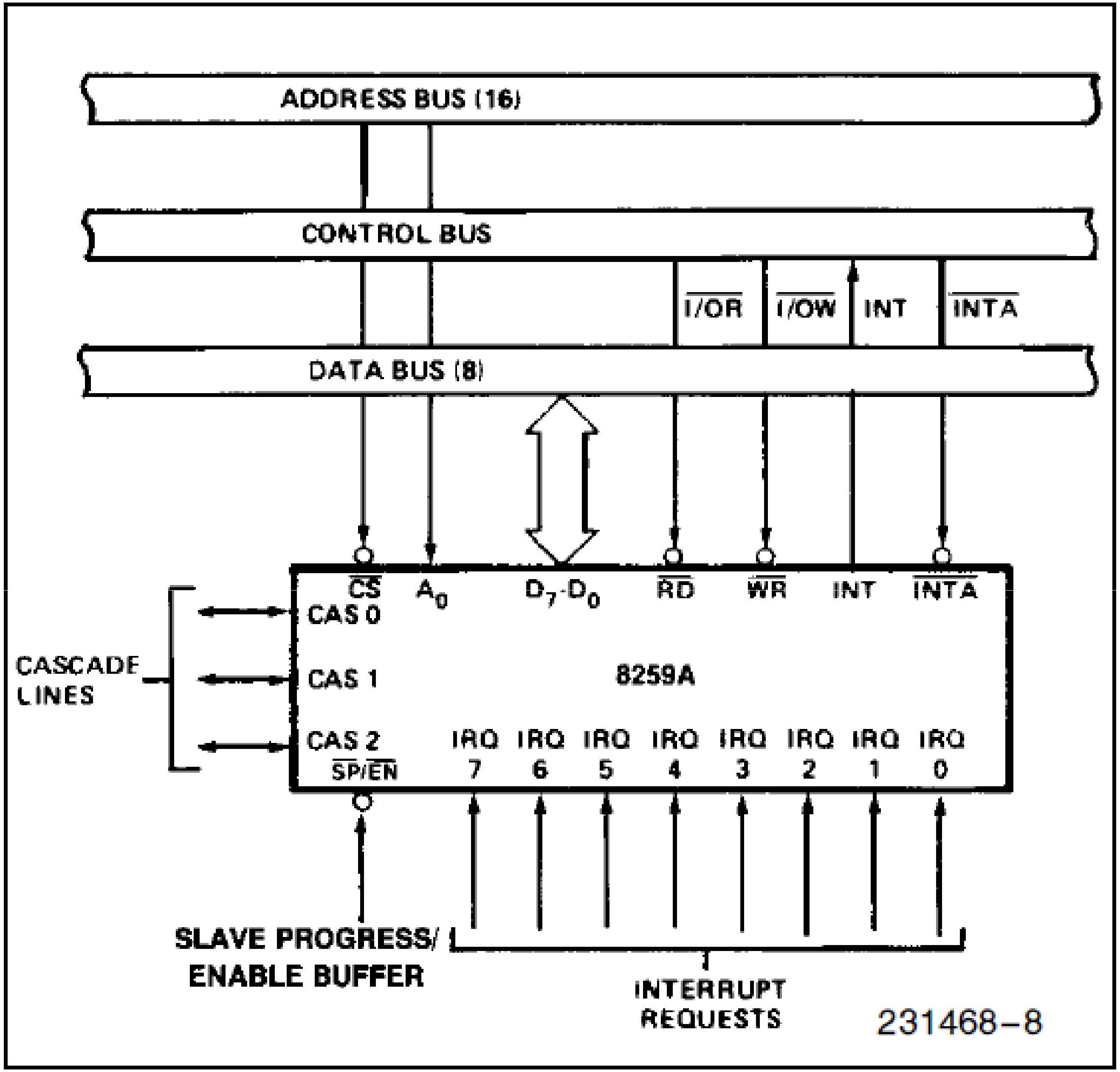


Figure 5. 8259A Interface to Standard System Bus

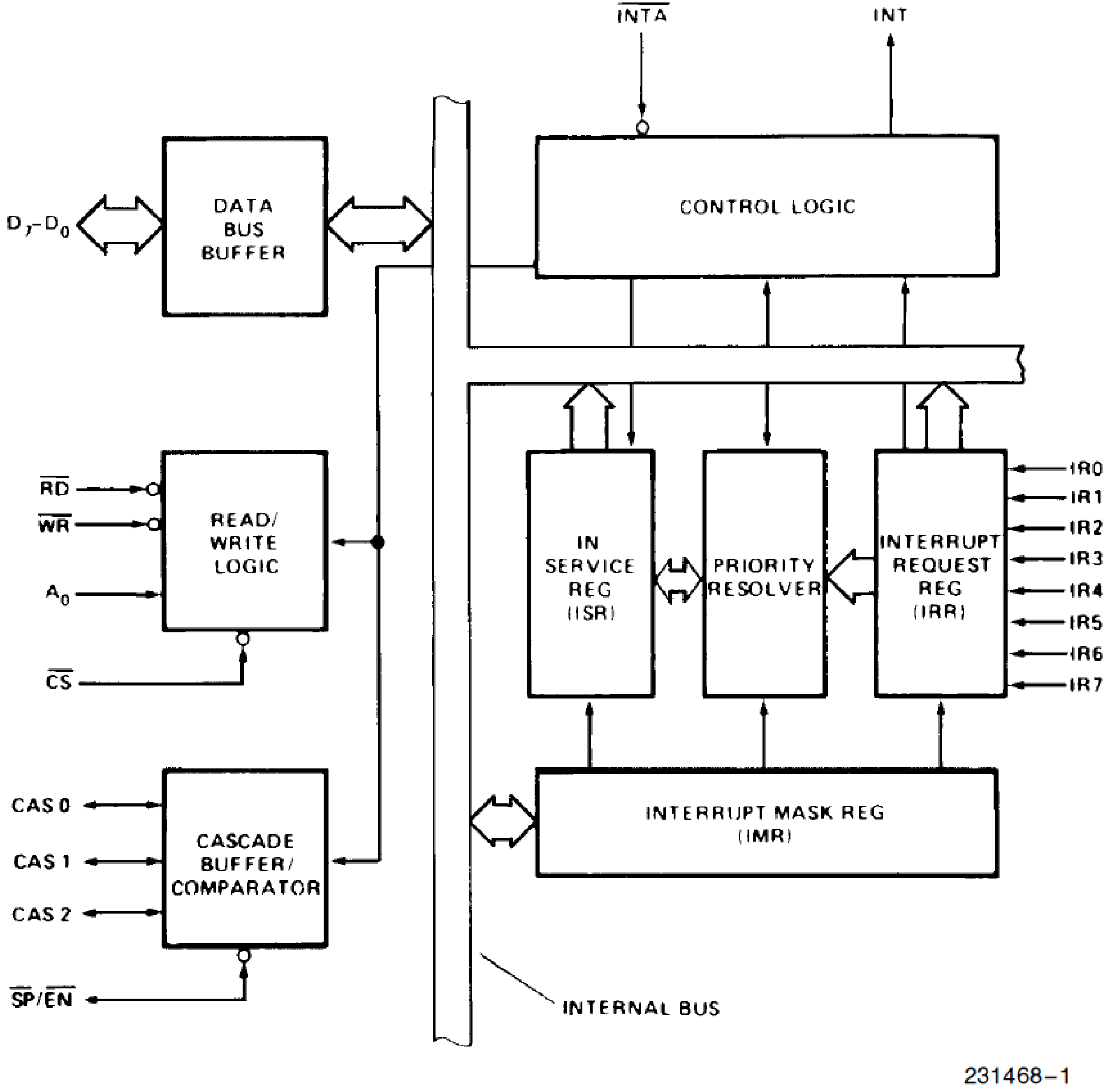


Figure 1. Block Diagram

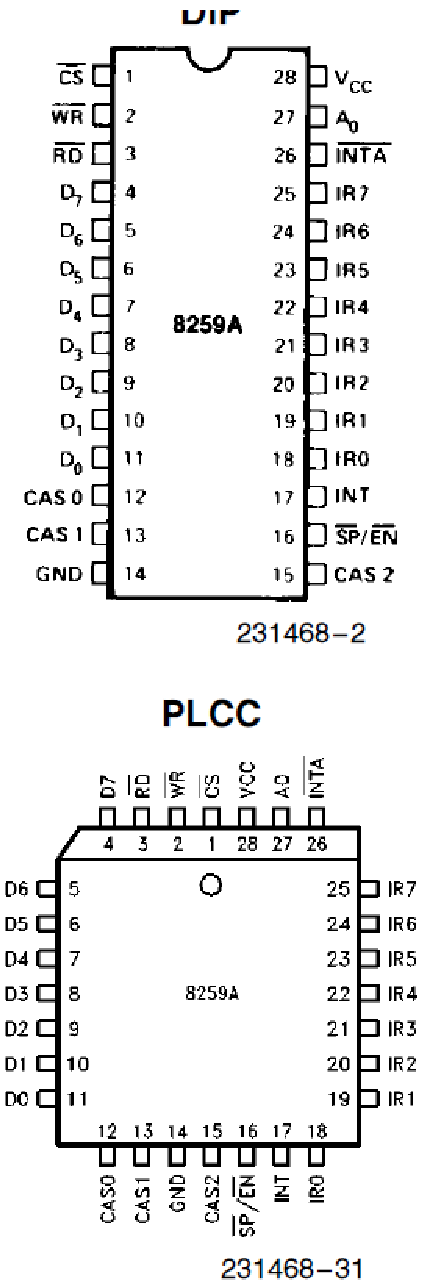


Figure 2. Pin Configurations

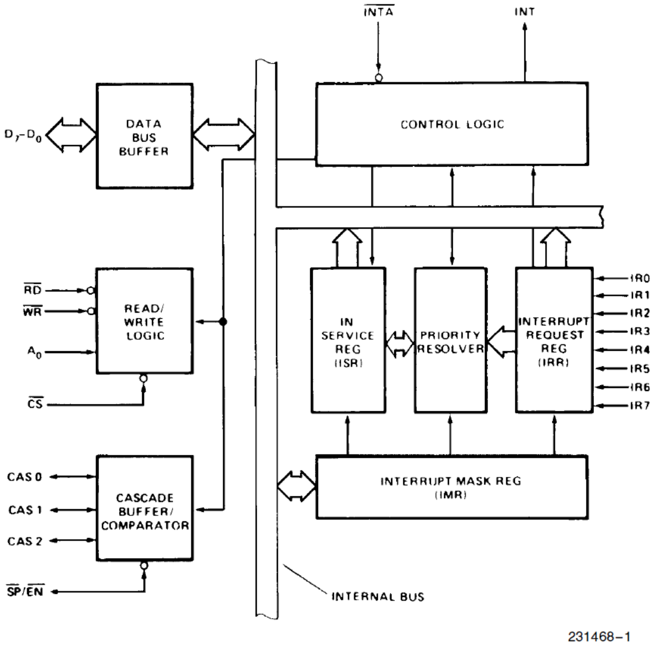


Figure 1. Block Diagram

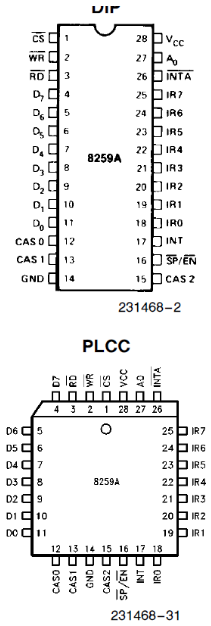


Figure 2. Pin Configurations

Interrupt sequence

1. One or more of the INTERRUPT REQUEST lines (IR7±0) are raised high, setting the corresponding IRR bit(s).

2. The 8259A evaluates these requests, and sends an INT to the CPU, if appropriate.

3. The CPU acknowledges the INT and responds with an INTA pulse.

4. Upon receiving an INTA from the CPU group, the highest priority ISR bit is set, and the corresponding IRR bit is reset. The 8259A will also release a CALL instruction code (11001101) onto the 8-bit Data Bus through its D7±0 pins.

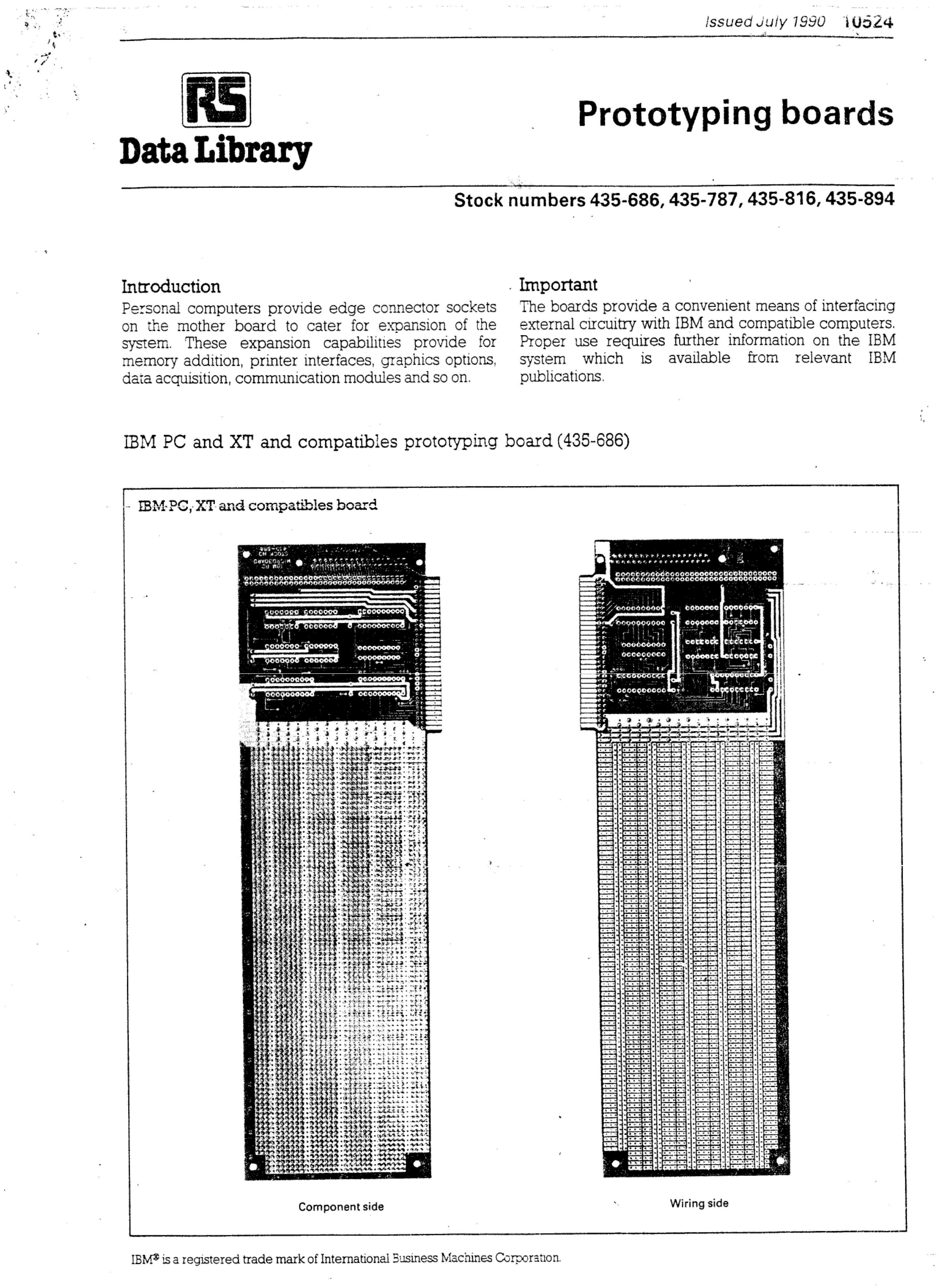
5. This CALL instruction will initiate two more INTA pulses to be sent to the 8259A from the CPU group.

6. These two INTA pulses allow the 8259A to release its preprogrammed subroutine address onto the Data Bus. The lower 8-bit address is released at the first INTA pulse and the higher 8-bit address is released at the second INTA pulse.
7. This completes the 3-byte CALL instruction released by the 8259A. In the EOI mode desides how to end the inturrpt

EOI

- End Of Interrupt (EOI) operations support
  - specific EOI
  - non-specific EOI
  - auto-EOI.
- Specific EOI specifies the IRQ level it is acknowledging in the ISR.
- Non-specific EOI resets the IRQ level in the ISR.
- Auto-EOI resets the IRQ level in the ISR immediately after the interrupt is acknowledged.

ISA Prototyping board



Issued July 1990 10524

RS Data Library

Prototyping boards

Stock numbers 435-686, 435-787, 435-816, 435-894

Introduction

Personal computers provide edge connector sockets on the mother board to cater for expansion of the system. These expansion capabilities provide for memory addition, printer interfaces, graphics options, data acquisition, communication modules and so on.

Important

The boards provide a convenient means of interfacing external circuitry with IBM and compatible computers. Proper use requires further information on the IBM system which is available from relevant IBM publications.

IBM PC and XT and compatibles prototyping board (435-686)

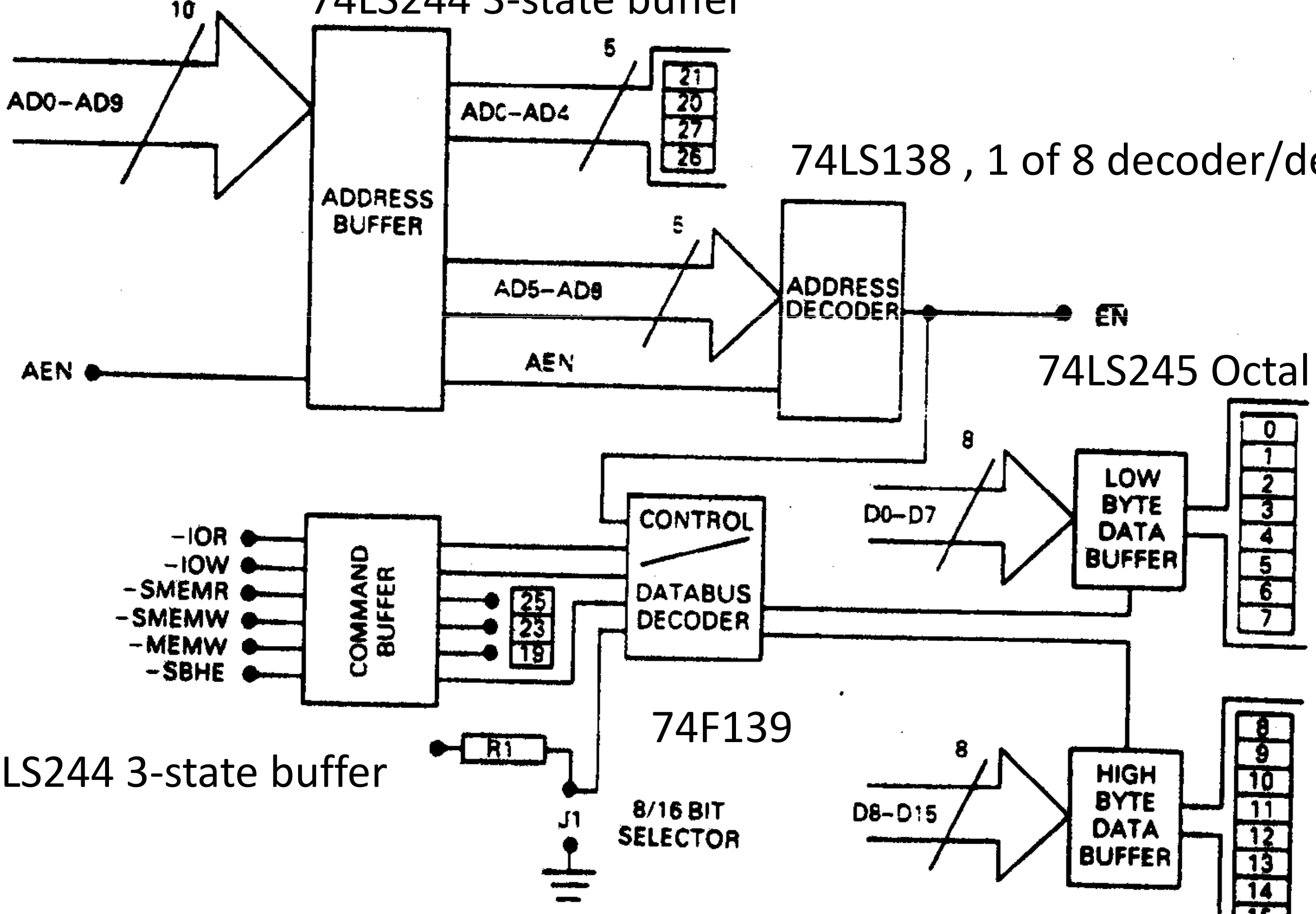
IBM PC, XT and compatibles board

Component side

Wiring side

IBM® is a registered trade mark of International Business Machines Corporation.

Input/output Interface block diagram



74LS244 3-state buffer

74LS138 , 1 of 8 decoder/de-mux

74LS245 Octal transceiver

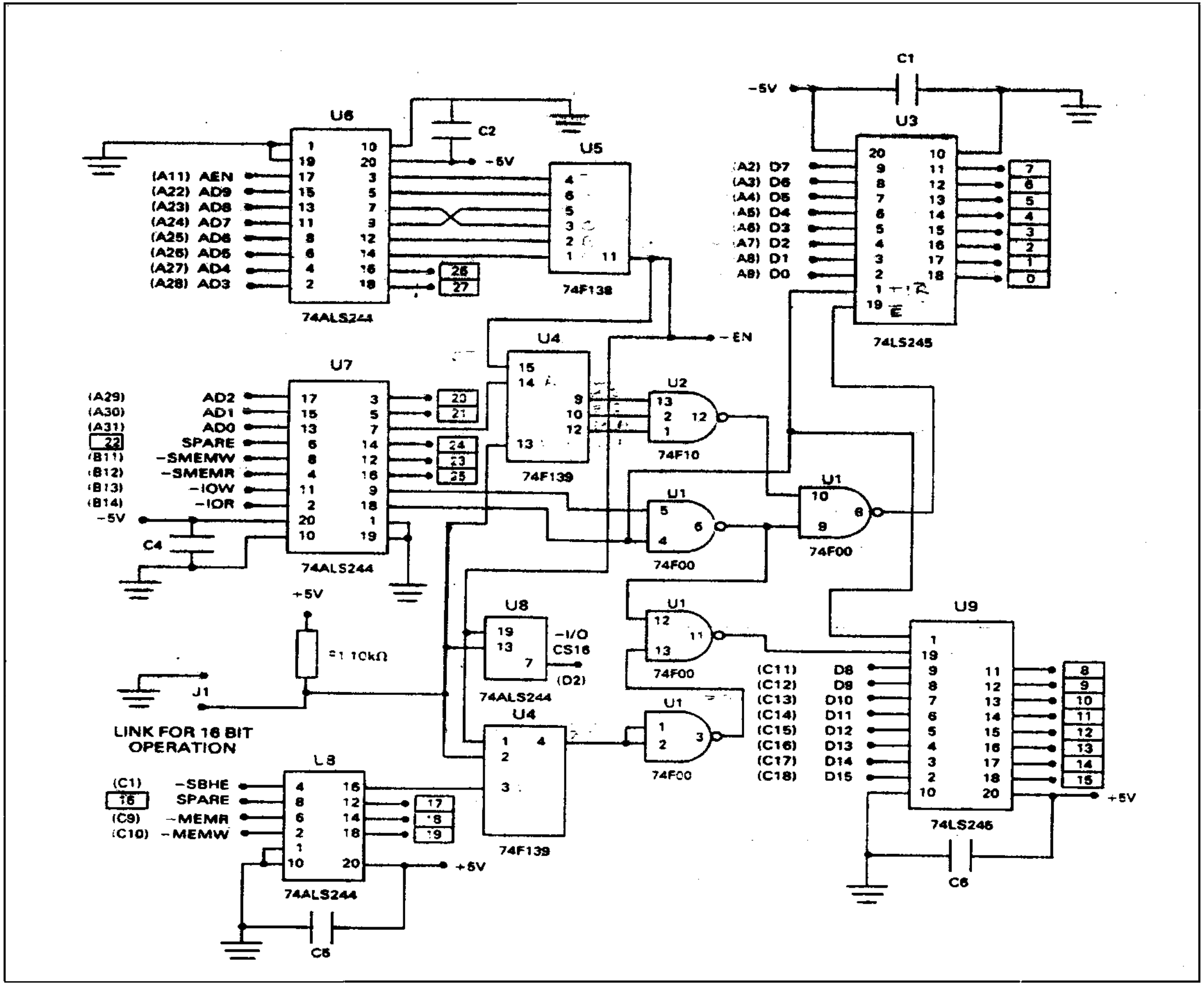
74F139

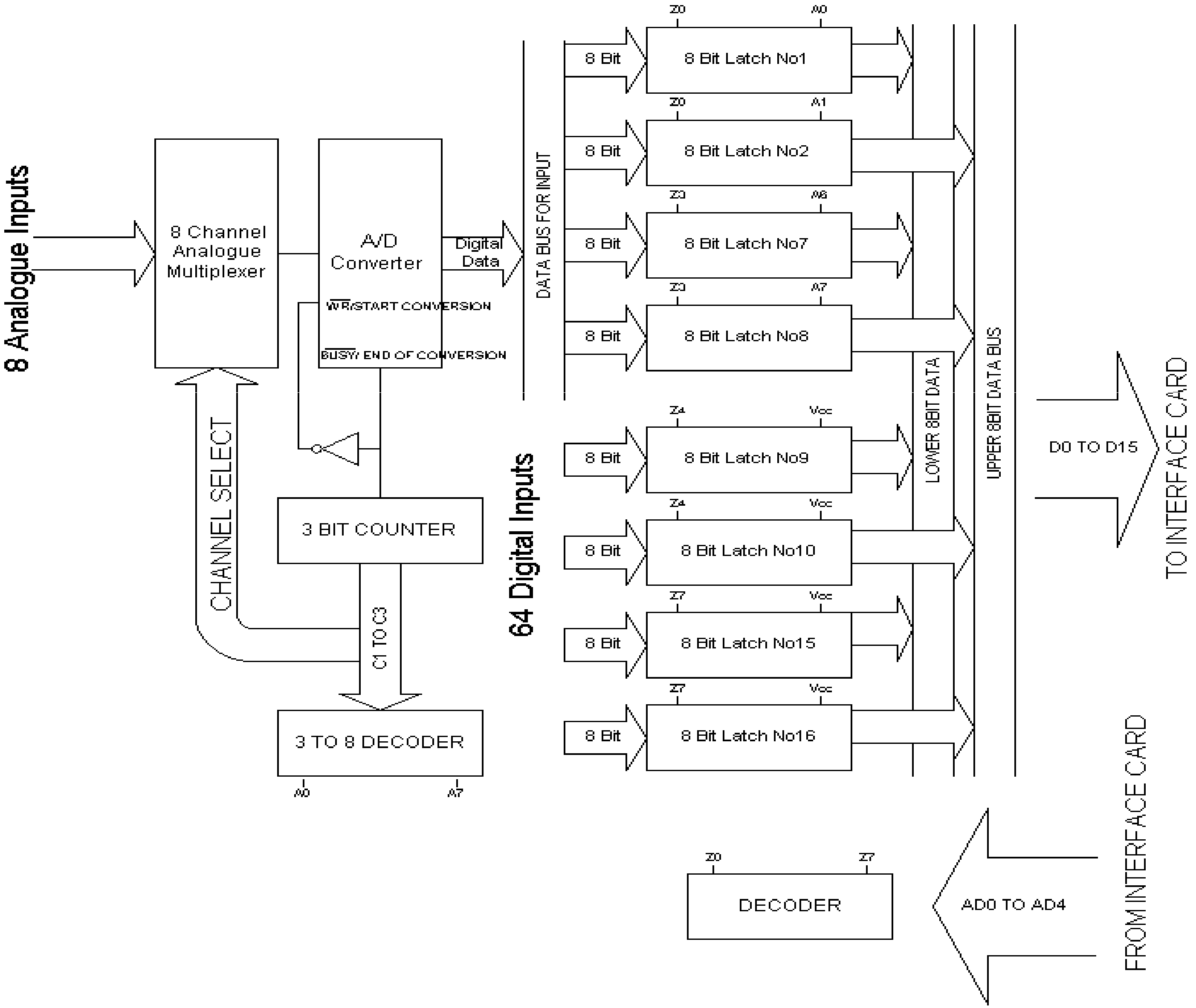
74LS244 3-state buffer

1 of 4 decoder/de-mux

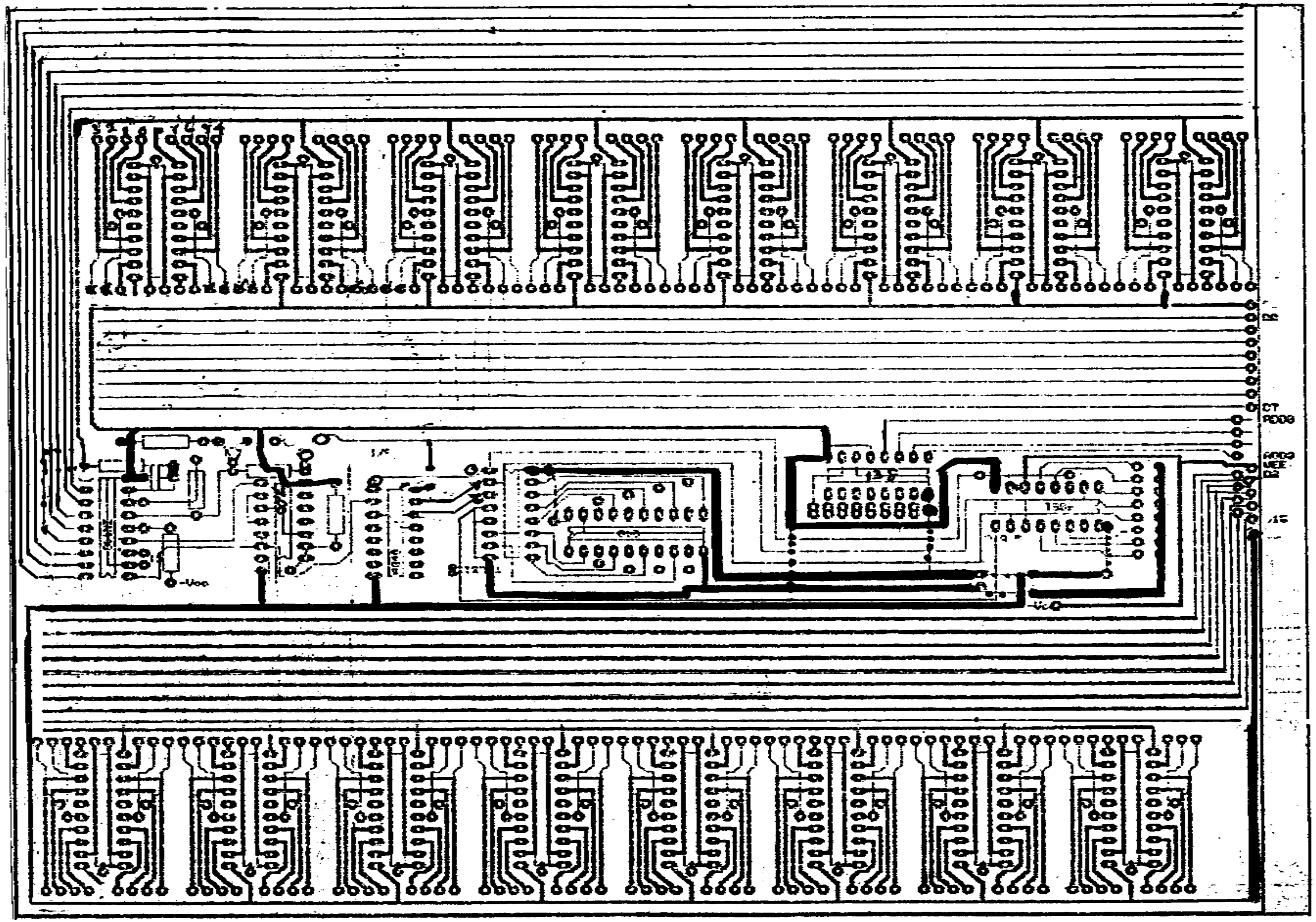
74LS245

Circuit diagram




8 Analogue Inputs
8 Channel Analogue Multiplexer
A/D Converter
Digital Data
WR/START CONVERSION
BUSY/ END OF CONVERSION
CHANNEL SELECT
3 BIT COUNTER
C1 TO C3
3 TO 8 DECODER
A0
A7
DATA BUS FOR INPUT
64 Digital Inputs
8 Bit
Z0
A0
8 Bit Latch No1
Z0
A1
8 Bit Latch No2
Z3
A6
8 Bit Latch No7
Z3
A7
8 Bit Latch No8
Z4
Vcc
8 Bit Latch No9
Z4
Vcc
8 Bit Latch No10
Z7
Vcc
8 Bit Latch No15
Z7
Vcc
8 Bit Latch No16
LOWER 8BIT DATA
UPPER 8BIT DATA BUS
D0 TO D15
TO INTERFACE CARD
Z0
Z7
DECODER
AD0 TO AD4
FROM INTERFACE CARD

PCB of DAQ Card

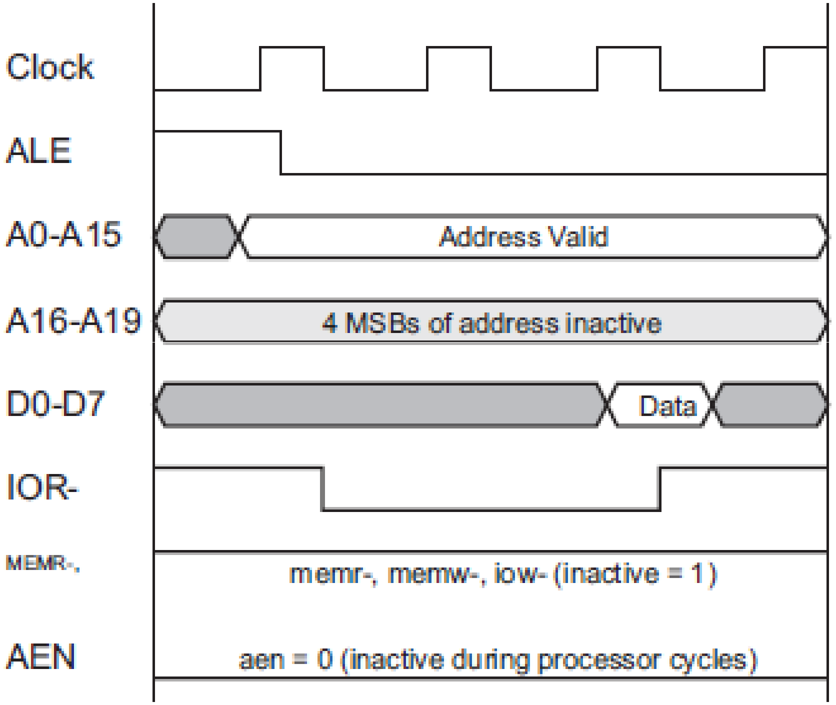


PCB lay out of the alarm circuit

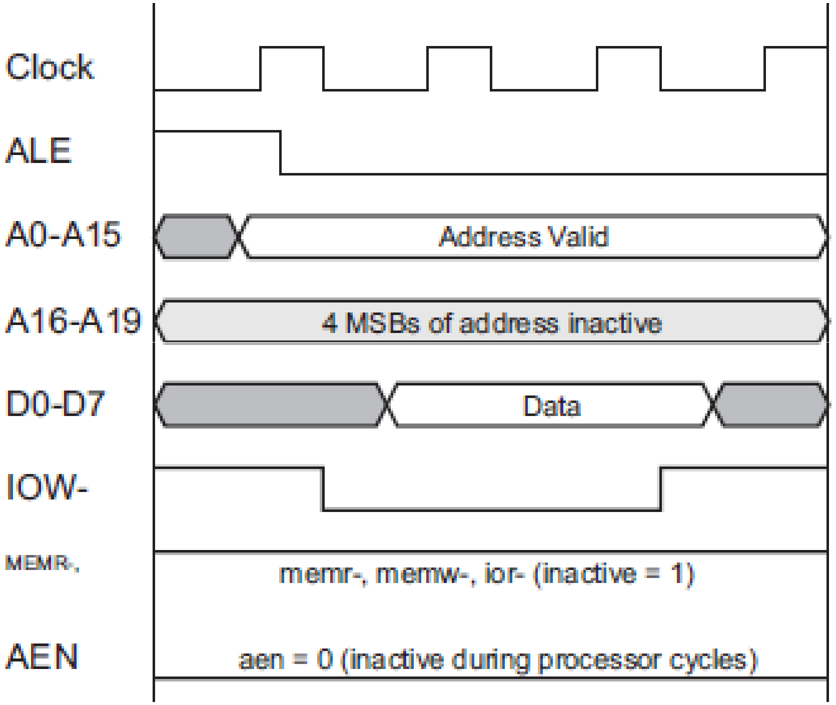
Figure 13

I/O Timing cycle

I/O Port Read Cycle


Clock
ALE
A0-A15
Address Valid
A16-A19
4 MSBs of address inactive
D0-D7
Data
IOR-
MEMR-,
memr-, memw-, iow- (inactive = 1)
AEN
aen = 0 (inactive during processor cycles)

I/O Port Write Cycle

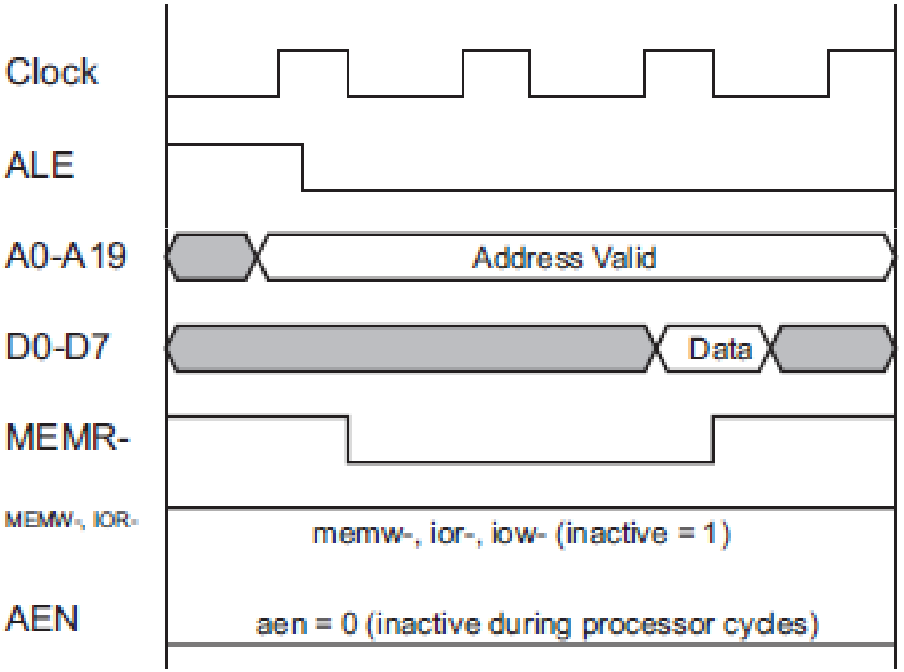

Clock
ALE
A0-A15
Address Valid
A16-A19
4 MSBs of address inactive
D0-D7
Data
IOW-
MEMR-,
memr-, memw-, ior- (inactive = 1)
AEN
aen = 0 (inactive during processor cycles)

IOR- goes low when the processor initiates a read from the port address space.

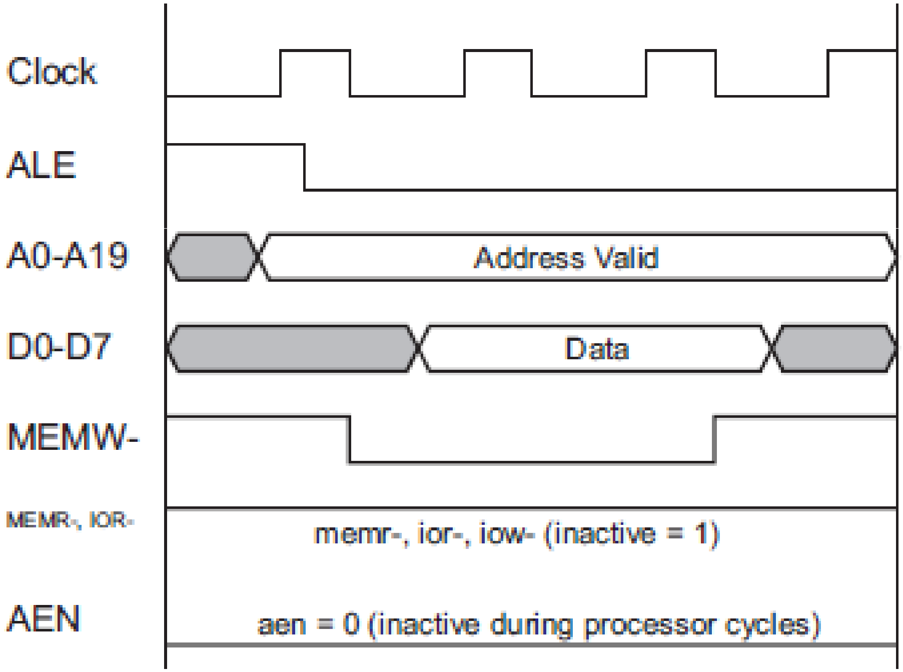
IOW- goes low when the processor initiates a write to the port address space.

Memory timing

Memory Read Cycle

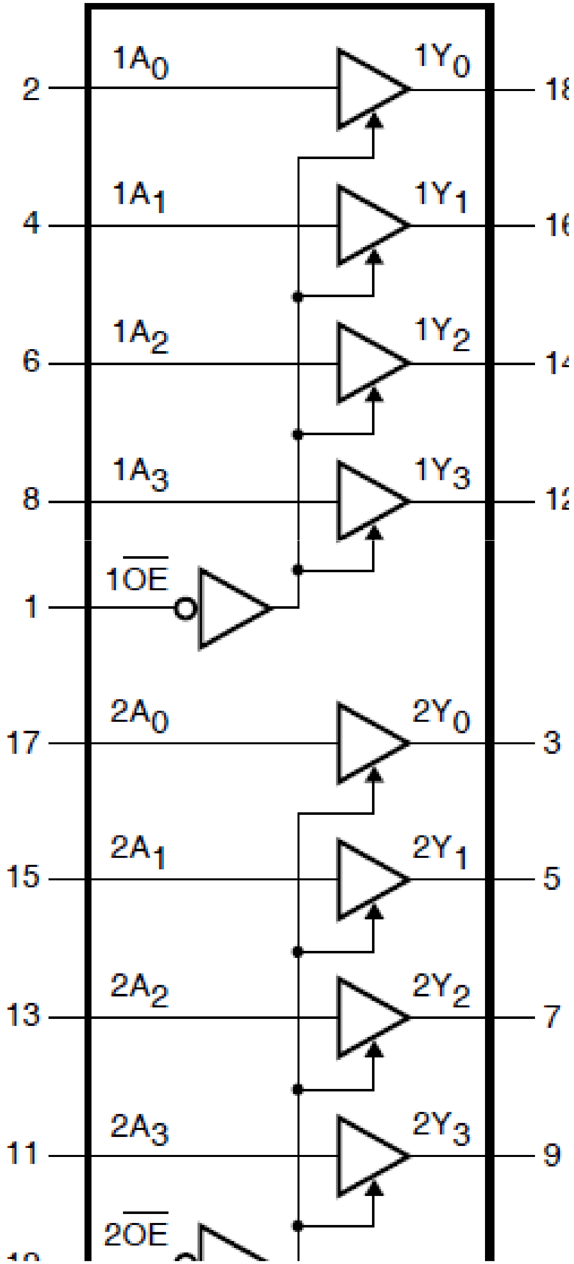

Clock
ALE
A0-A19
Address Valid
D0-D7
Data
MEMR-
MEMW-, IOR-
memw-, ior-, iow- (inactive = 1)
AEN
aen = 0 (inactive during processor cycles)

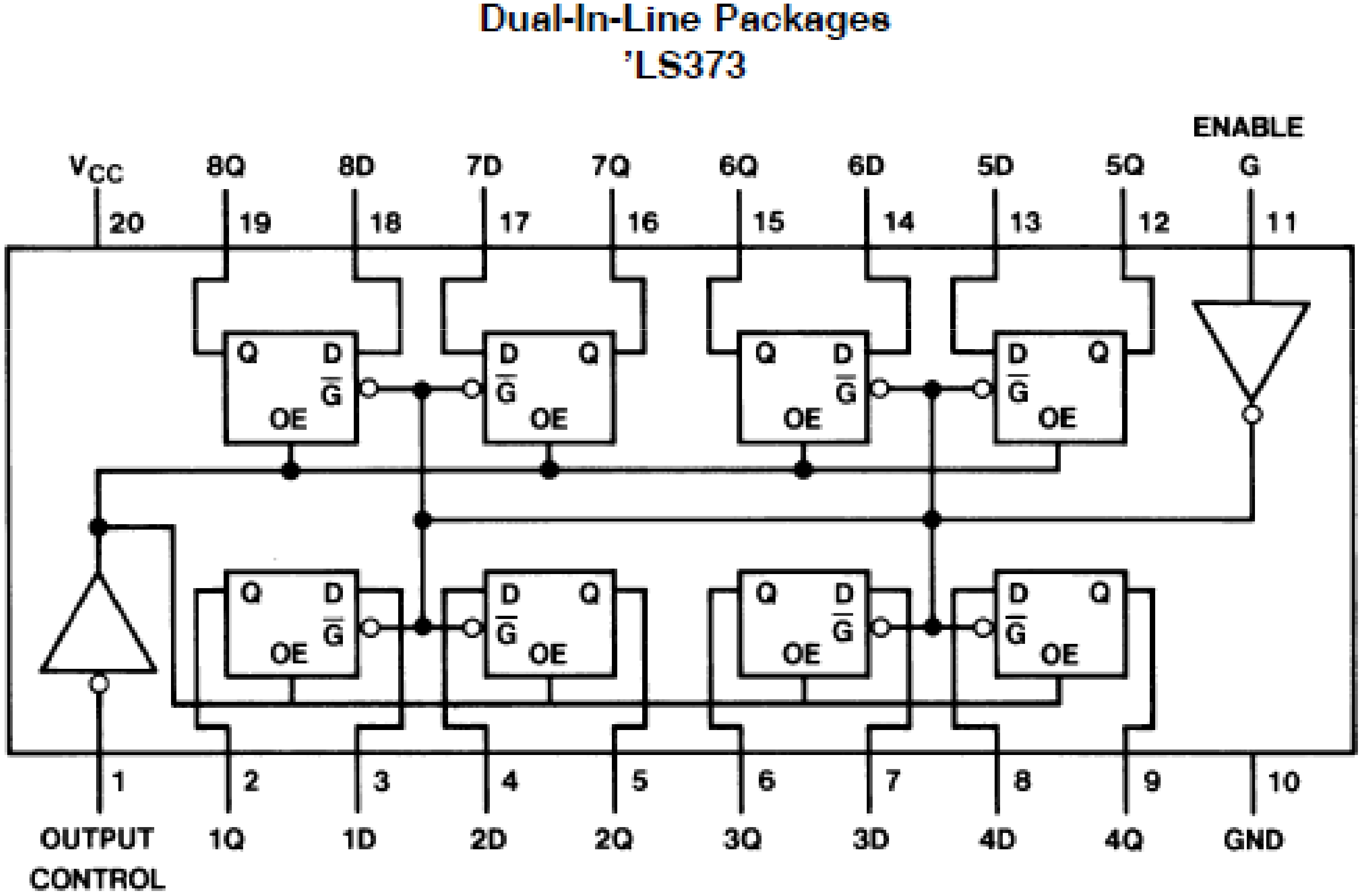
Memory Write Cycle


Clock
ALE
A0-A19
Address Valid
D0-D7
Data
MEMW-
MEMR-, IOR-
memr-, ior-, iow- (inactive = 1)
AEN
aen = 0 (inactive during processor cycles)

MEMR- goes low when the processor initiates a read from the memory address space.
MEMW- goes low when the processor initiates a write to the memory address space.

74LS244
3 state octal buffer/line driver

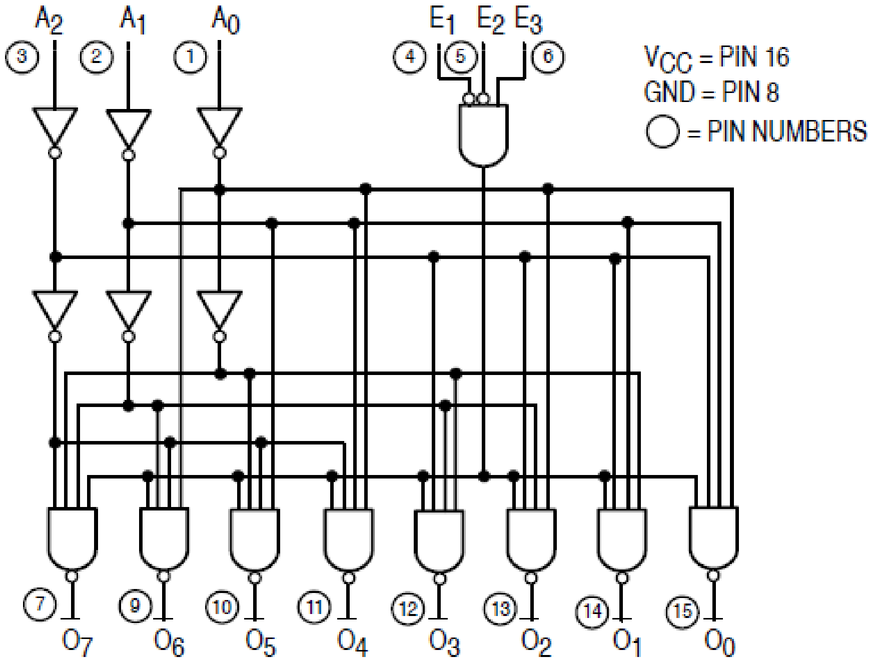




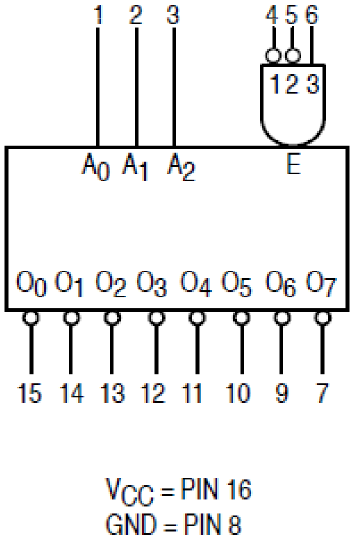
3-STATE Octal D-Type Transparent Latches and Edge-Triggered Flip-Flops

74LS138 1-OF-8 DECODER/ DEMULTIPLEXER

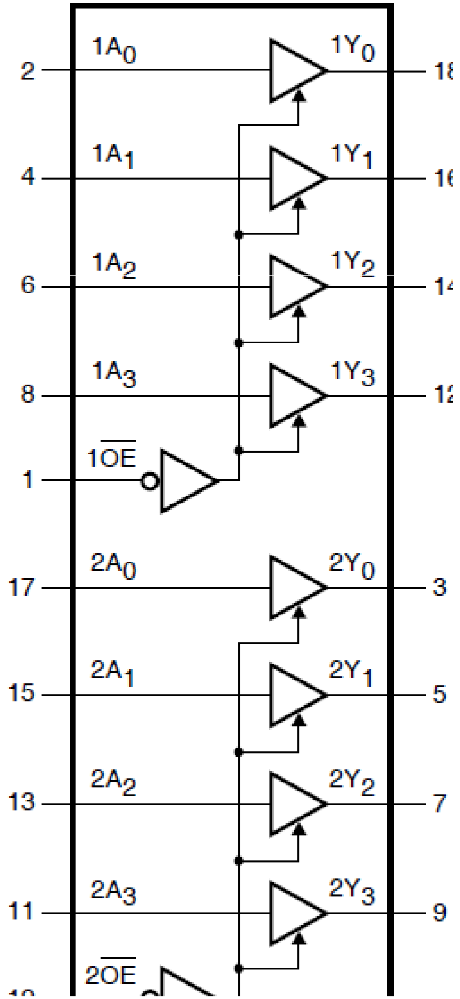
LOGIC DIAGRAM



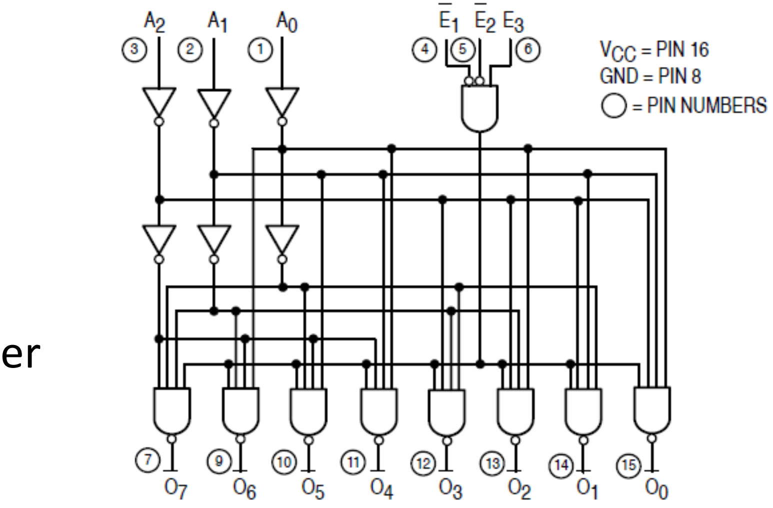
LOGIC SYMBOL

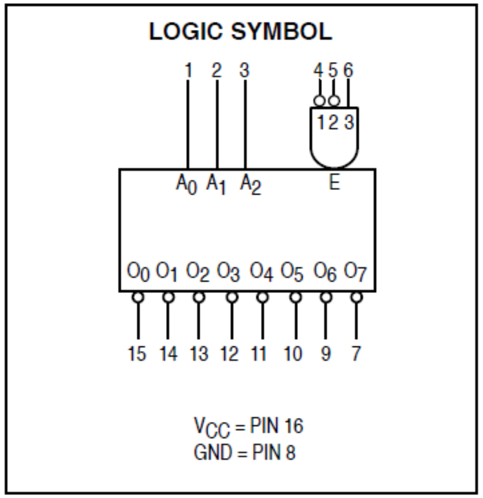


74LS244
3 state octal buffer/line driver



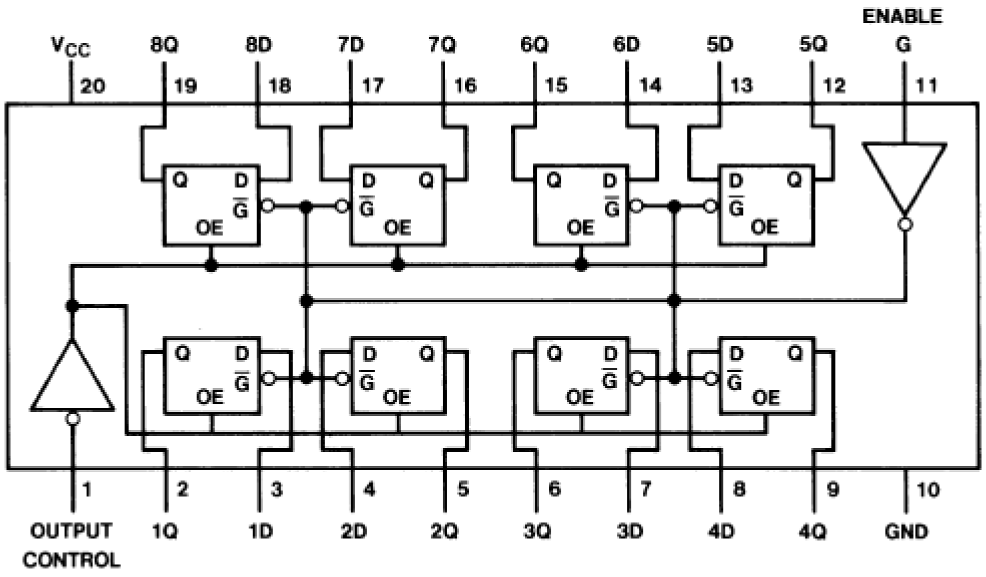
LOGIC DIAGRAM





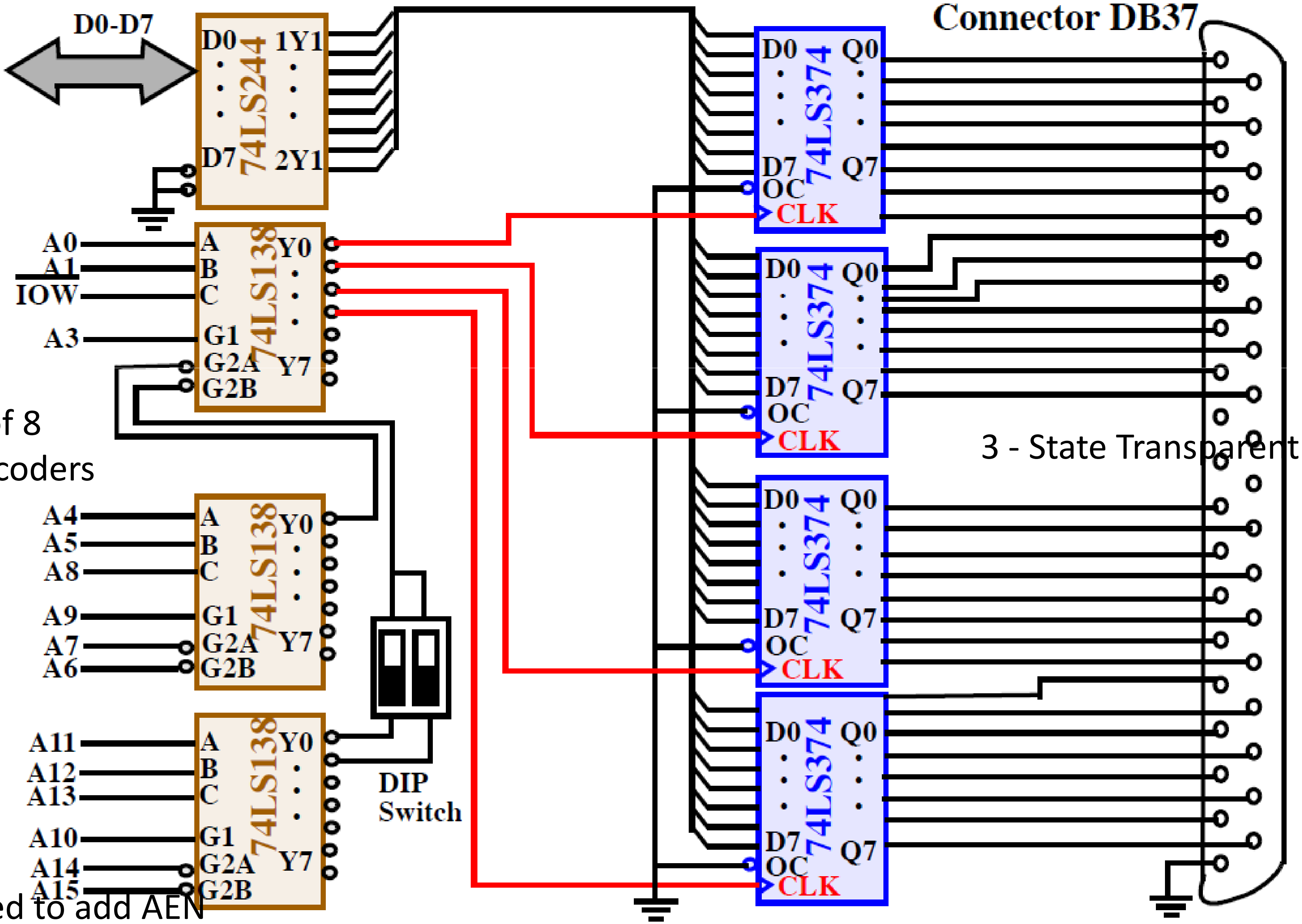
74LS138 1-OF-8 DECODER/ DEMULTIPLEXER

Dual-In-Line Packages
'LS373



3-STATE Octal D-Type Transparent Latches and Edge-Triggered Flip-Flops

Example 8 Bit Bus Output Interface
Decoder could besimpler



Buffers

1 of 8 decoders

3 - State Transparent Latches

Need to add AEN