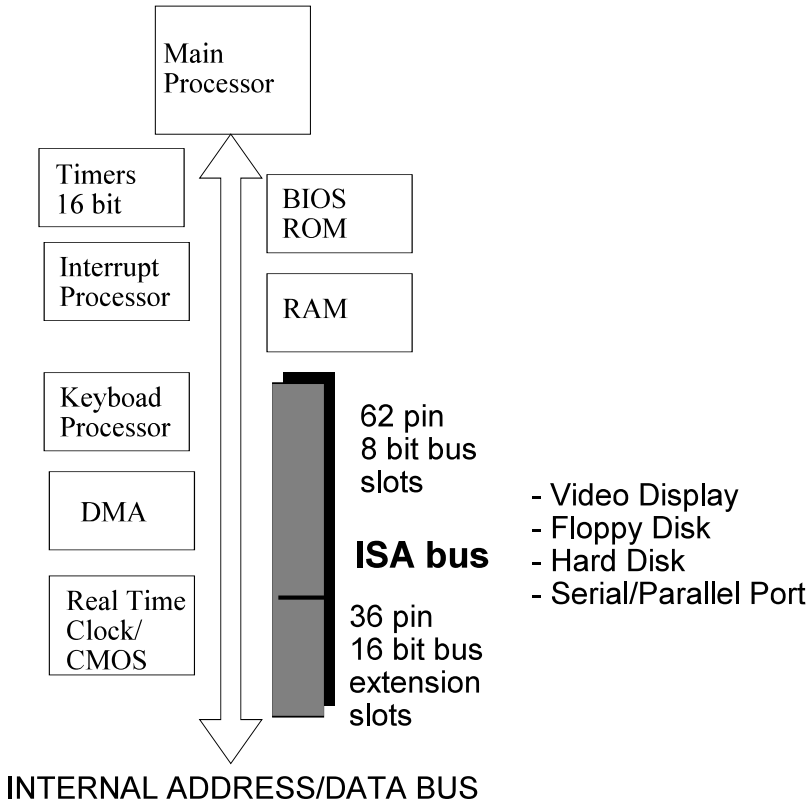
2.1 Introduction

As processors continue to advance, older models no longer in mainstream use find fruitful employment in embedded systems. The IBM PC, introduced in 1982 was followed by the XT. After that came the AT which used the ISA bus architecture which exerted great influence on desktop computer design in the 1980's. Although largely superseded, it is in use in embedded systems. For this reason, we will look at this bus in some detail.

Also, some of the hardware features still lives on in today's systems, which build upon the existing standards with advanced techniques in computer architecture. Here we will focus on the hardware interface aspects of the PC architecture. We will also cover the bus which allows devices to interface with the PC and the most common kinds of devices, namely memory chips, buffers and latches.

2.2 PC AT Hardware architecture

In the PC AT, the video display, secondary storage, serial and parallel ports are accessed



through the PC AT bus. This bus became known as the Industry Standard Architecture or ISA bus.

2.3 PC AT Memory and I/O Map

In any microcomputer system, the basic design will use up certain memory resources in order to provide basic function to the user. This places a constraint on users who wish to expand the capability of their system as these resources are used. We will consider the memory map of the PC and the areas used. This is shown in the appendix to the chapter. The appendix also lists the I/O addresses and interrupts that are already used by the basic system.

If you examine the *memory* map, you will see that the first 640 (256+384)K of memory is only available as system memory. From A0000H onwards, the memory is dedicated for system uses, such as providing memory for various vendor cards.

Thus a system designer should avoid using these resources which the system has reserved.

2.4 Timers and Interrupts

Two of the more important hardware functions in a typical PC are those of timing and interrupt processing. Timing is used to synchronize various activities, while interrupts provide an efficient way of signaling when external devices are ready to transfer data to and from the system.

2.4.1 Timers

There are two 16 bit timers running at 1.8432 MHZ which provide the timing for several system functions. The real time clock is used to keep track of the date and time and is not normally used for timing events.

2.4.2 Interrupts

The PC architecture allows up to sixteen devices to communicate with the processor, indicating that it is ready for a data transfer. Unfortunately, much of these devices are used by the system already, leaving only a few available for general use.

2.5 The ISA - PC/104 Bus

From the very earliest commercial microcomputer systems, vendors have always provided an interface bus. They realized that it was not possible to anticipate all the needs of users who

needed to interface to all kinds of devices. In fact this would cause a higher demand for their microcomputer systems as add-in boards could significantly increase the number of applications that could use their computer systems.

The ISA bus was simple to interface to, but designed to transfer 16 bit data at 8 Mhz. This proved too limiting to modern systems. However, current bus technology is complex to interface with.

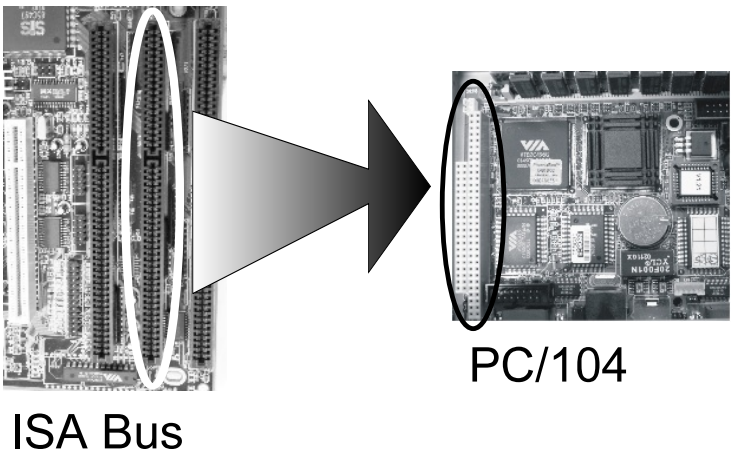


Figure 2 Comparison of ISA and PC/104 sizes

In 1992, Ampro Computers put up a specification for computer circuit boards that are about 9 cm by 9 cm in size and allows them to be connected up using "Molex" pin headers. The connecters increased the 98 (62+36) signals of the ISA bus to 104, thus giving the PC/104 standard, widely in use in embedded systems.

From here, we will use the term ISA and PC/104 interchangeably. The ISA bus contains various features that allow it to transfer data at high speed and to interface to slower devices. Briefly, they are DMA and wait-states. AEN goes high to indicate this condition.

When the AEN signal is low, it indicates that the *processor* is outputting the address and should be decoded. Thus we need to include this signal in decoding.

2.5.1 Wait States

Some peripherals are known to be slow. That is, it cannot perform an operation within one cycle of operation. By setting the I/O CH RDY low, the system processor will wait up to eight clock cycles for the data transfer to be made.

2.5.2 Description of Bus

When the IBM PC first appeared, it featured a 62 pin bus. The PC AT featured a processor with larger address and data size and so the bus was extended to 62+36 pins. Some signals were added so that cards with only 62 pins could still be used.

We will only cover those signals that are of interest to us. A full description of the signals on this bus can be found in various reference texts dealing with PC architecture.

Signal name	Description
A0-A23	This is the address bus and is made up of SA0-19 and LA17-23, giving a total of 16 MB of memory Only SA0-15 are turned on during I/O and only SA0-9 are actually used.
SD0-15	These are the data bus bits for devices. SD7 to SD0 are used for transfer of data with 8-bit devices. SD15 to SD0 are used for transfer of data with 16-bit devices.
AEN	Address Enable is used to inform system processor and other devices from the bus during DMA transfers. When this signal is active the system DMA controller has control of the address, data, and read/write signals. This signal should be included as part of ISA board select decodes to prevent incorrect board selects during DMA cycles.
-IOR	*I/O Read* indicates that the user of the ISA bus is doing a read and the selected I/O should output data to the data bus.
-IOW	*I/O Write* is driven by the owner of the bus and instructs the selected I/O device to capture the write data on the data bus.
-MEMR	*Memory Read* instructs a selected memory device to drive data onto the data bus. It is active on all memory read cycles.
-MEMW	*Memory Write* instructs a selected memory device to store the data currently on the data bus. It is active on all memory write cycles.

Fig 2-**1** Description of selected signals of the ISA Bus

In many cases, the full width of the ISA address and data busses are not used as this requires more hardware for interfacing. It is common practice to use only a subset.
In summary,

ISA Bus width (bits) ==>	Address		Data
	Memory	I/O	
Theoretical sizes	24	16	16
Common use	20	10	8

The ISA bus makes a distinction between addressing memory and I/O devices by having separate MEMR and IOR, MEMW and IOW signals. This is because the Intel microprocessors have different instructions for performing them. These are known as

memory-mapped I/O and I/O mapped I/O, respectively. Memory mapped I/O allows one to use the I/O as if it were a memory device using standard load/store/move instructions in assembler, or just normal high level language assignment statements.

I/O mapped instructions take the form of "input" or "output" style of instructions. The comparison between the two modes of addressing is shown below:

Memory mapped	I/O mapped
Uses microprocessor main memory address space	Uses separate memory space apart from processor's main memory.
May interfere with allocating memory to programs as we have to note that there are "special" locations that are not standard memory.	No concerns about special address ranges
Use flexible addressing modes to access data.	Normally all data must pass through accumulator - can be slow.

2.6 Memory Devices on the ISA bus

A PC has relatively large amounts of memory on its main board. What is the use of having extra memory on a card connected through the ISA bus? Furthermore, the memory on the card is normally quite small and the speed of operation of the ISA bus is quite slow.

There are two main uses for these memory:

1. The PC architecture allows for plug-in cards to install software that can be used by the processor to operate the card. For example, a display card typically uses the memory range B8000H to store software required to operate it. This software will execute at power up and they allow programs to interface to it through software interrupts.

2. Plug-in cards may need their own memory to store data or even run their own programs. Typically, this is RAM and acts as a buffer, so it can be used to quickly transfer data between itself and a PC.

Thus we use standard static RAM and EPROM devices on plug-in cards meant for the ISA bus rather than dynamic RAM which is used on the main board.

In the original PC architecture which had only 1 MB of memory, the addresses from A0000H to FFFFFH was allocated for this memory. Although the ISA bus can address 24 bits, we will only consider the first 20 bits.

2.6.1 Introduction to Types of Memory Devices

There are many different types of semiconductor memories, but we will focus on two: namely, Random Access Memory, or RAM, and Read Only Memory, or ROM.
Each has its own set of technologies and uses.

RAM - Random Access Memory
The term RAM has come to mean a memory device where data can be stored as well as retrieved, and where the process of storing data at a particular location takes about the same time as the process of retrieving it from that location.

Most RAMs are volatile devices i.e. the stored data is lost when the power source is removed. However, there are non-volatile RAMs on the market which contain either a capacitor, or a small re-chargeable battery cell, so that the stored data can be retained for some time after the power is removed.

2.6.1.1 ROM - Read Only Memory

The logic function is stored in the circuit permanently without the need of electrical power to sustain the memory it is non-volatile memory). Also, the input combination has no effect on how long it takes to read the output combination random access memory). So, a ROM can be defined as a fixed memory whose contents cannot be altered during normal operation.

Below we summarize some of the main features of ROMs, comparing them with RAMs.

	RAM	ROM
Volatile	yes	no
Random Access	yes	yes
Read	yes	yes
Write by processor	yes	no

There are a number of types of ROM, based on the methods of storing the data in the ROM, and whether or not that data can be erased.

In ROM devices, the data is stored as part of the manufacturing process, and can never be changed after that. Thus it requires costly manufacturing processes and is not economical unless many thousands are required.

Applications :
ROMs are used to store unchanging data like user prompts, error messages, character generators for video displays, or as the bootup program for computers. But ROMs are widely used in simpler forms, for example, even as a simple 4 line to 16 line decoder or a BCD to 7 segment decoder.

EPROM - Erasable PROM/One Time Programmable ROMs

An EPROM or Erasable PROM is a device in which the data can be erased after it has been stored and so it can be reused many times. The erasing is done by shining ultra-violet light on to the chip through a quartz window for 20 to 30 minutes. As a result, the entire data contents are erased each time. Because of the quartz window, the chip has to be made of ceramic. In order to make using of existing processes and lowering the cost of the package, manufacturers offer EPROMs in plastic packaging that cannot be erased. These are known as One Time Programmable ROMs (OTPROM).

Applications :

For design testing and development, it may be necessary to revise and test a program many times before it is considered error free. Because a PROM can only be programmed once, it is much too expensive for this purpose. An EPROM or Erasable PROM, which can be re-used many hundreds of times is used in this case. Although the initial cost of the IC package is greater than for a PROM, it becomes effectively much cheaper the more it is re-used. Because they are now relatively cheap. EPROMs have replaced PROMs in many applications.

PROM - Programmable ROM

To overcome the problem of expense when only small quantities are required, programmable ROMs are produced. Here, the data is not stored during manufacture, but later, in the field, by a user with the necessary programming equipment. Once stored, the data can never be changed. The process involves the “blowing” of small nichrome fuses or links in the device, and is thus non-reversible. Hence these devices are often called fusible link PROMs. They are normally bipolar devices, requiring a different manufacturing process.

Applications :

The main use of PROMs is for small-scale production where the economies of scale are not available. Because a chip manufacturer can produce the basic (unprogrammed) device in large quantities, they are comparatively inexpensive, but the user of the PROM must then pay the additional costs of programming the device. A custom-made address decoder is a very useful application of a small Field-programmable ROM.

EEPROM - Electrically Erasable PROM

An EEPROM or E^2PROM (Electrically Erasable PROM) is a special type of erasable semiconductor memory where the data can be erased and re-written electrically in circuit. This class of devices has recently become very important as storage devices.But it is not like a RAM. The process of storing data takes very much longer than reading it, usually about 30 milliseconds writing time. A later type of EEPROM is Flash memory. The main difference is that FLASH updates its contents in terms of group of bytes or sectors whereas EEPROM does so a byte at a time. FLASH can therefore store large amounts of data relatively quickly. Also, EEPROM is available with a serial interface, which makes it smaller, both physically and in storage capacity. Parallel devices are larger on both areas, thus addressing different applications.

Flash memory can be purchased as bare memory chips, which gnerally use NOR gate technology. However using NAND gate technology, a commonly available form is as "memory cards". Examples are Compact Flash (CF), Secure Digital (SD), MultiMedia Card (MMC) formats and others. Hardware on the cards make interfacing easier. For example, the CF card appears like a hard disk.

Flash memory also appears in standalone portable storage devices. For example, portable music players can appear as a hard disk when attached through the USB interface.

Applications :

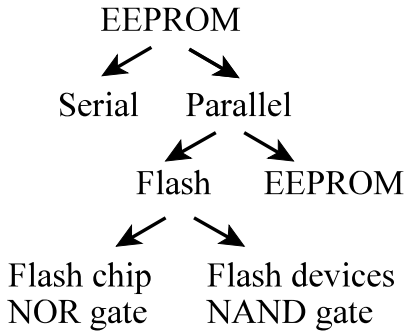
The form of flash memory to use is dependent on the application. As part of a system, it is natural to use flash memory for updates to programs as data stored within can be done easily and automatically. An example of this is the ability of modern equipment to update their firmware by "flashing" an update from the manufacturer.

However, if we wish to use flash memory for storing of large amounts of data, it makes more sense to use a CF card and access it like a hard disk.

Serial EEPROM is mainly used to store settings and configuration information where its low cost, low power consumption and small size are advantages here.

This illustrates the fundamental difference between RAM and EEPROM. In a RAM the processes of storing and retrieving data take about the same time - in the order of tens to hundreds of nano-seconds. While an EEPROM can be read from in about the same time, the process of storing data takes much longer.

In summary, the diagram describes the EEPROM family. The type used will depend on the applications



2.6.1.2 RAM Devices

RAM devices can be classified into two types: static and dynamic. RAM's are manufactured using both MOS and bipolar technologies. MOS uses NMOS and CMOS. These are usually simpler devices, and yield higher densities than bipolar process. Bipolar devices include Schottky TTL, ECL, and IIL types. Usually faster than MOS devices, and accordingly have higher power dissipation.

Static RAM's

Static RAM's use a flip-flop as the storage element, one F/F for each bit of data storage.

So an 8K RAM has 8192 F/Fs plus the necessary address decoding and I/O buffering circuits. Information is retained in the F/F for as long as power is applied.

Dynamic RAM's

Dynamic RAMs use the ability of MOS capacitors to store charge as the basic storage element. 1's or 0's are indicated by the presence or absence of charge on the capacitor. If the capacitor is charged above a certain level (that is, its voltage is above a certain threshold), it represents a logic 1, otherwise it is a logic 0. An ideal capacitor, once charged, maintains a constant voltage, but charge will leak away in a practical capacitor, and so the voltage will change.

The capacitor's voltage (the data) must be read regularly, and if it is a logic 1, it must be restored to full voltage, before too much charge leaks away. This is called refreshing. If a dynamic RAM is not refreshed often enough, the voltage of the logic 1's will fall below the threshold and all stored bits of data will revert to 0's. For example, an 8K dynamic RAM contains 8192 MOS capacitors plus associated decoders, buffers, etc. All these capacitors must be refreshed regularly, typically every 2 milliseconds.

2.7 I/O Devices on ISA bus

A more common use of plug-in cards is to provide many more ports to be used for interfacing with external devices. These ports normally interface with devices that are much slower than the main processor so that the relatively slower speed of data transfer is not an issue. As we saw earlier, the Intel architecture uses 16 bit addressing for I/O. On the PC, only 10 address pins are actually used, to reduce the hardware decoding requirements.

2.7.1 BASIC INPUT/OUTPUT HARDWARE

Various hardware elements can perform some of the functions of the I/O section. Simple devices include flip-flops (latches) and buffers. In this chapter we will look at some of the basic elements used in interfacing circuitry.

2.7.1.1 Buffers

Most microprocessors are MOS products, hence the driving capabilities of such devices are usually restricted to at most 1 TTL load or 5 MOS loads. This severely restricts the operation of the microprocessor in driving other devices, for example, RAMs, ROMs and other support devices. In order to increase the driving power, or driving current, buffers are used.

Buffers are TTL ICs which allow a signal to pass through them without altering the logic value of the signal. However, the buffer increases the DC drive characteristics of the

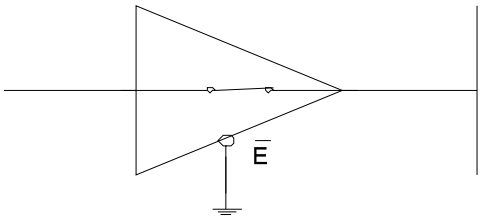
signal. Hence, the microprocessor, with the aid of a buffer is capable of driving many more circuits.

Another advantage of using buffers is that the microprocessor is isolated from any outside signals. If there is a change in voltage along the signal lines, the buffers are first affected, thereby preventing any damage to the microprocessor. Being mainly TTL ICs, they introduce a small delay in the transmission of data.

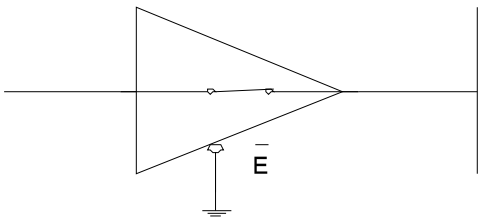
Two variations of the buffer exist. The first is simply a one-in one-out buffer which takes an input and increases the driving current of the signal at the output for all logic levels.

The second type of buffer is called a Tri-state buffer. This device has an additional control signal. When the control is active, the device performs exactly as a buffer. When the control is de-activated, the device prevents any output from appearing. The output of the device turns into a high-impedance state giving neither a logic 1 or O. The table below shows the logic states of such a buffer :

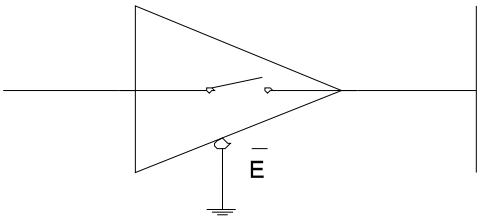
Input	Control	Output	State
0	Active	0	buffers input
1	Active	1	buffers input
0	Non-active	High impedance	Tri-state
1	Non-active	High-impedance	Tri-state



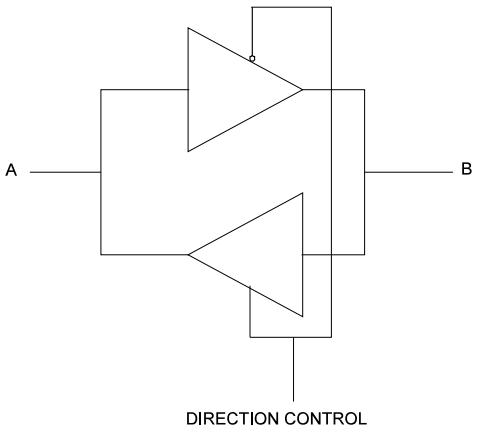
Tri-state buffers are useful in circuits which only allow the passage of signals when a particular state is active. This state can be connected to the Control input of the buffer.



A variation to the tri-state buffer is the bi-directional buffer. This buffer allows the passage of data from one point to the other depending on a Direction Control Signal. This allows a system to control the direction of flow of data from one point to the other. The figure below shows the operation of such a buffer.



When the Direction Signal is ‘low’, data flows from Point A to Point B. When the signal is ‘high’, the data flows from Point B to Point A.



A typical application of the bi-directional buffer is in the buffering of the microprocessor's data bus. Using the _RD line as the Control Signal, the microprocessor is able to control the flow of data into the microprocessor only when data is required. At all other times, the direction of flow of data is from the microprocessor to the other devices.

2.7.1.2 Latches

Latches are mainly made up of D-type flip-flops. The latch takes in a signal via a strobe (clock pulse). The latch then holds the signal until another strobed signal is applied or until the power is removed.

The latch has an advantage over the buffer in the sense that the signal is held at the particular logic level until a change is requested. The buffer holds the logic level as long as the signal is also at that level. Latches are thus used in preference to buffers in situations where the receiving end requires a little more time to process the signal. Figure below shows the operation of different latches in response to changing signals.

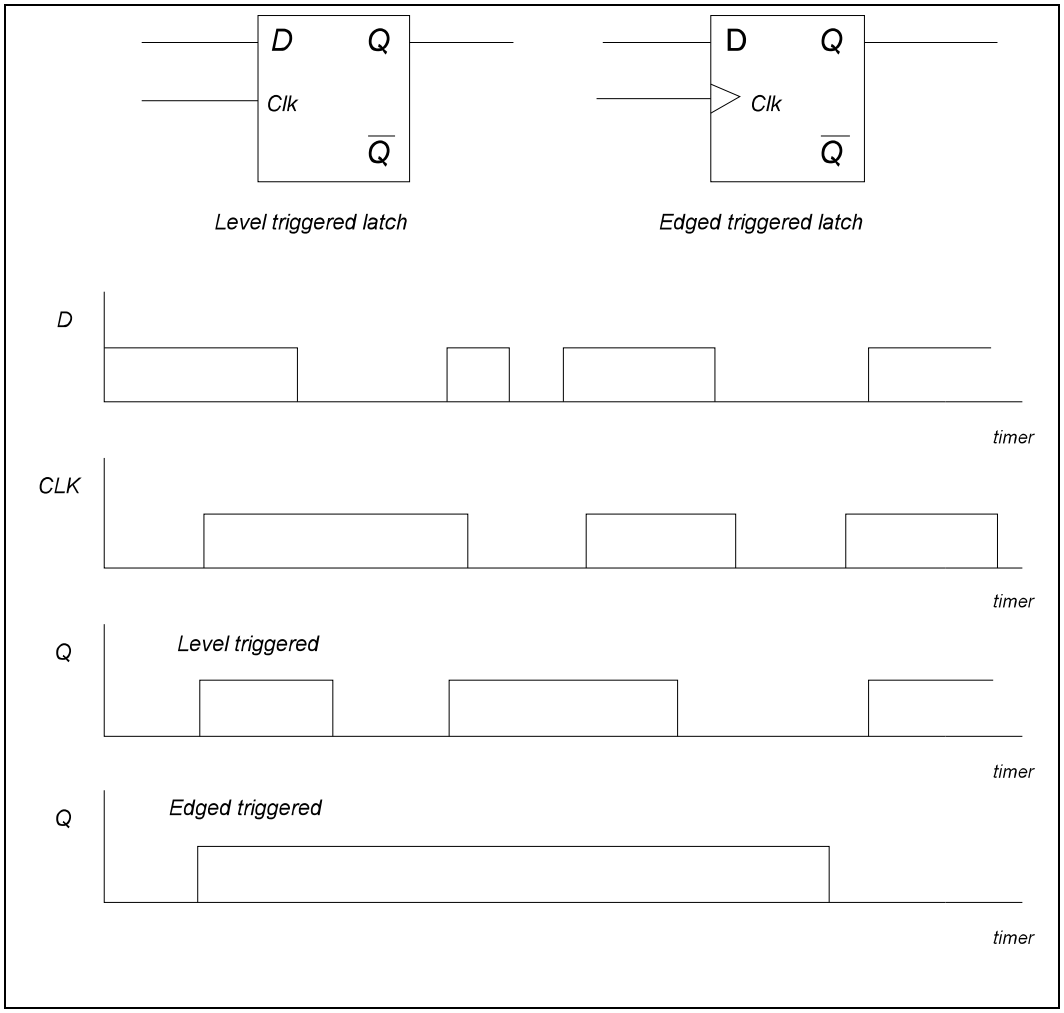


Fig 3.5 Signal Response of Latches

Latches can be level-triggered (changes output as long as the strobe is active) or edge-triggered (changes output only on either a rising or falling edge of the strobe signal). In cases where ambiguity is to be avoided, edge-triggered latches are used.

2.7.1.3 Input/Output Ports

Basic Input Ports

The basic input port is a tri-state buffer connected to a data bus line. The control signal of the buffer is usually the combination of _IORD and the selected port address, which is obtained using address decoding.

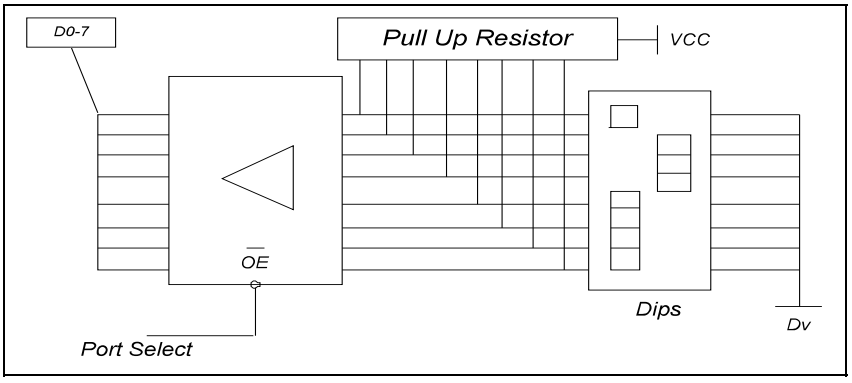


Fig. 3.8 An Example of an Input Port using a Buffer

Basic Output Ports

The basic output port is a latch. The CPU places the necessary data to be output on the data bus lines and then clocks the data into the latch with the control signal which is usually the combination of the _IOW and the selected port address, which is obtained using address decoding.

The latch is preferred over the buffer as external devices are usually slower than the CPU. Hence, the port has to hold the data until the device is ready to read it.

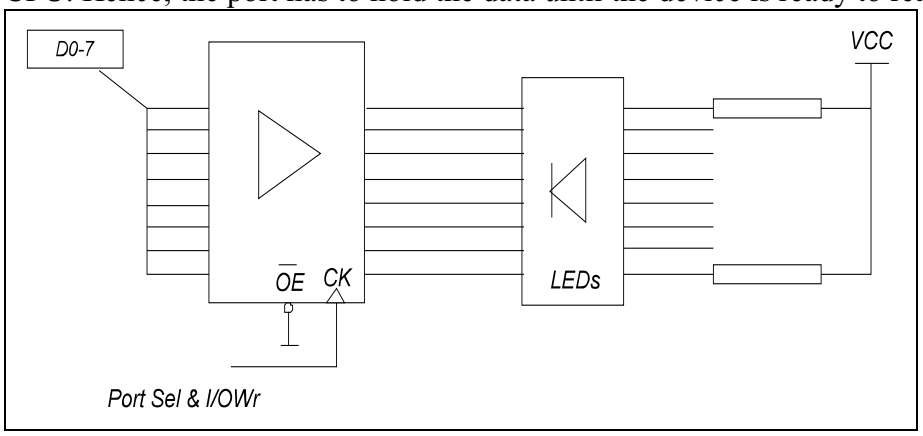


Fig. 3.9 An Example of an Output Port using a Latch

2.8 The BIOS and boot up process

As a final note and application, we see how ROM is used in a PC where memory is mainly DRAM. We will also see how a basic "monitor" program considerably enhances the use of a bare computer system.

The Intel microprocessors follow a given sequence when it is RESET. Recall that ROM is nonvolatile, so that if a processor starts, it is guaranteed to have valid code when it

jumps to that location.

In most other microprocessors, what happens after that is purely a matter of what the programmer puts into the ROM code. Starting from the early microprocessor systems available, the vendor would often put in basic routines in a so called "Monitor Program" and include it in the ROM.

These allowed users to do fundamental tasks like simple output to the screen, keyboard handling, output to printer, serial communications and so on. This allowed users to quickly bring up their system to productive use and allowed a measure of standardization to programs as they used common routines to do input and output.

With the coming of the PC, this monitor program became known as the Basic Input Output System or BIOS. More routines were added in this time, for secondary storage devices like hard and floppy disks. Also there was support for Time-Of-Day routines.

A BIOS may occupy as much as 64K of ROM which is not practical for small processors, but monitor programs can be as small as 2K. However if the system is simple enough, a programmer can make do without any built in monitor program. Consider the boot up process for Intel processors:

Microprocessor function
i) Performs a jump to location 0FF000H.
ii) Executes the code found there, which is normally contained in ROM.

Monitor program (BIOS) function:
i) It searches for a secondary storage device like a floppy or hard disk.
ii) It reads in the boot sector (normally 512 bytes in length) for the device.
iii) Puts the contents into memory location 7C000H.
iv) Jumps to that code and executes it.
v) This code will have instructions to load other larger files which will continue to initialise the system.

This process is called "bootstrapping".

Chapter Appendix

System Memory Map

Address (HEX)	Usage
00000-3FFFF	256K R/W Memory on System Board
40000-9FFFF	384K R/W Memory Expansion in I/O Channel
A0000- AFFFF	128K Reserved
B0000-B3FFF	Monochrome Display Adaptor
B8000-BBFFF	color Graphic Display Adaptor
BC000-BFFFF	
C0000- C7FFF	192K Read Only Memory Expansion and Control
C8000-CBFFF	Fixed Disk Controller
F0000-FFFFF	64K Base System ROM BIOS and BASIC Interpreter

I/O Address Map

Address (HEX)	Usage
000-00F	DMA Chip 8237A-5
020-021	Interrupt 8259A
040-043	Timer 8253-5
060-063	PPI 8255A-5
080-083	DMA Page Register
0A0-0AF	NMI Mask Register
0C0-0CF	Reserved
0E0-0EF	Reserved
200-20F	Game Control
210-217	Expansion Unit

220-24F	Reserved
278-27F	Second printer
2F0-2FF	Reserved
2F8-2FF	Serial Comms (Secondary)
300-31F	Prototype Card
320-32F	Fixed Disk
378-37F	Parallel Printer
380-38F	Other Comms
3A0-3AF	Reserved
3B0-3BF	Monochrome Display/Printer
3C0-3CF	Reserved
3D0-3DF	Color/Graphics
3E0-3E7	Reserved
3F0-3F7	Diskette
3F8-3FF	Serial Comms (Primary)

PC AT Hardware Interrupt Listing

Number	Usage
NMI	Parity
0	Timer
1	Keyboard
2	From 2nd Interrup Controller
3	Serial Comms (Secondary)
4	Serial Comms (Primary)
5	Parallel Printer 2
6	Diskette
7	Parallel Printer 1
8	CMOS Real Time Clock

9	Replaces IRQ2
10	Not used by system
11	Not used by system
12	Not used by system
13	Numeric coprocessor
14	Fixed Disk
15	Not used by system