# Chapter 0 Introduction and Revisions

A course in Computer Peripherals is relatively uncommon. Most courses in Computer or Electronics Engineering do not offer such a subject. Some of the material covered in this course is found as parts of courses covering computer interfacing, but the interdisciplinary nature of the technology involved in most computer peripherals has caused the devices to be omitted from these courses.

However, peripherals comprise a most significant component in any computer system for three reasons. Firstly, it is the most visible part of the hardware, as peripherals provide the interface between the human user and the system. Secondly, it constitutes a significant portion of the total cost of the system, and thirdly, it is often a significant contributor to the performance constraints of the system.

In the typical personal computer, the user enters commands and data at an input peripheral, usually the keyboard and the results are presented to him at a CRT display or on hard copy output produced by a printer. From an economic viewpoint, for a total hardware value of the cost contribution of the central processor, memory, associated power supply and cabinetry would amount to approximately 30%. With Intel Pentium processors which run at more than 400 Mhz, performance is undoubtedly limited by disk accesses and printer speeds, as well as by the user himself.

Since the value of peripherals is a significant factor in any computer system, it is also a significant contributor to the computer industry as a whole. As a result of the large investments in manufacturing facilities made by disk drive manufacturers, Singapore has become the disk drive capital of the world. Companies like Seagate, Conner and Maxtor are among the largest disk drive manufacturers in the world, and most of their manufacturing capacities are based in Singapore. Local companies like Creatives who specialised in peripherals devices like sound blaster cards, video blaster cards, DVD ROMs and RAMs is growing and contributed a lot to our economy.

Apart from the above, other peripherals manufactured in Singapore include printers, CRT displays and keyboards.

## 0.1. Definition

This is a course on computer peripherals, so we have to begin by defining what is meant by the term peripheral. A computer system consists of a number of functional components, the most significant of which is the central processor unit or CPU. The central processor, whether a large mainframe or minicomputer or even a small microprocessor-based personal computer, performs its logic, memory access and computation operations at rates of millions of operations per second. In order for this capability to be harnessed and applied to the real physical world of slow humans like us the results of these processes must be slowed down, translated into words, images, or control signals. It is the job of peripheral devices or peripherals to provide information to and extract results from the central processor.

## 0.2. Classification

Peripheral devices are usually classified by their function:

(1) Input devices such as keyboards, mice, bar-code scanners and digitisers.
(2) Output devices like printers, plotters and displays.
(3) Storage devices, which include floppy and hard disks, optical disks and magnetic tape drives.
(4) Multi-media devices and others.

Most peripherals would fall into one of the above categories, although some may span two classes. We will not cover the multi-media devices and it will be covered in CE446: Multimedia Systems Design. If you look at a PC that you can get for slightly above $1,000, you will get a list of peripherals together with it. A basic PC will have a keyboard for input, a harddisk for storage and as well as a CRT display or output printer. Most system nowadays will come complete with a floppy disk drive, a CD ROM and some user also include backup devices such as diskette, tape drives an zip drives.

There are other classes of peripheral devices which will not be covered here. These are the range of A/D and D/A interfaces for control and instrumentation, cameras and frame grabbers for screen capturing and the various communications interfaces like LAN and ISDN devices which are more dealt with in courses on real-time systems and data communications respectively.

## Physical, Ergonomic, Safety and Environmental Requirements

In the selection and design of peripherals, a number of additional considerations have to be taken into account, considerations which are often given scant attention in the design or selection of the central processing unit. Usually the peripheral interacts with the user, thus defining the environment in which the peripheral device must operate. For example, in the case of the video display terminal, although we can package all the electronics into a very light and small unit, we need to realise that for fast and comfortable data entry, the keyboard must be of a certain size and the displayed characters must not be too small for easy viewing. Apart from meeting the technical specifications the following requirements should be taken into account.

(1) Environmental and safety requirements. The operator or the application may not been in a clean and air-conditioned room. Power supplies may not be well regulated. There is no computer room environment. Equipment designated to be installed in the home will have to meet stringent standards regarding audio noise and radio frequency interference. They also need to meet relevant safety standards.
(2) Ergonomics. Human factors should be taken into consideration. For keyboards, it is the tactile (touch) and/or audio feedback, the size, angle, travel and pressure of the keys best suited for fast, accurate and comfortable typing? For printers, what is the quality of the printed report, is it noisy in operation? In the case of displays, what is the best colour and brightness for viewing, is a darken room needed, is there an annoying glare from the screen, what about the amount of radiation emitted by the CRT?
(3) Many countries have national standards and most of these are based on international standards that have been defined and are continually modified. These standards authorities include:
    - UL Underwriters' Laboratory, US safety
    - FCC Federal Communications Commission, US regulatory authority on transmission, and radio interference
    - CSA Canadian Standards Association, safety
    - DIN German Standards, safety, ergonomics

ISO International Standards Organization, safety, ergonomics, industrial standards
JIS Japan Industrial Standards, safety, industrial standards.

## 0.3. Computer I/O Architecture

Before proceeding with the study of computer peripherals, a brief review of the architecture of the computer, the requirements of interface design, and the structure of computer buses will be given.[TC1]

The digital computer normally consists of the central processing unit (CPU), memory, input devices, output devices, and storage devices as shown in Fig. 1.1. Interconnecting these components are the address, data and control buses. It is seen that each of the peripheral devices is connected to CPU through the interface unit. These interface units generally comprise the following:

(i) *Transmit and receive data registers/buffer:* As the CPU and the peripheral operate asynchronously at different speeds, these registers and buffers (FIFO) are use to hold and transfer data to and from the peripheral device. The length of these registers is normally the same as the data-word length of the peripheral device.

(ii) *Control registers:* One or more control registers are used to capture and store the command received from the CPU. With programmable devices, mode registers are used to set the mode of operation of the device.

(iii) *A status register:* Each bit of the status register is used to indicate individual status conditions to the CPU. Sometimes the CPU also writes the status register. In this case the peripheral reads it to determine the status of the processor.

(iv) *An address decoder:* Irrespective of whether the device is interfaced using memory-mapped or I/O-port techniques, the device will still have to decode the address information from the CPU to determine whether it should respond. Again for programmable devices, a series of addresses is decoded.

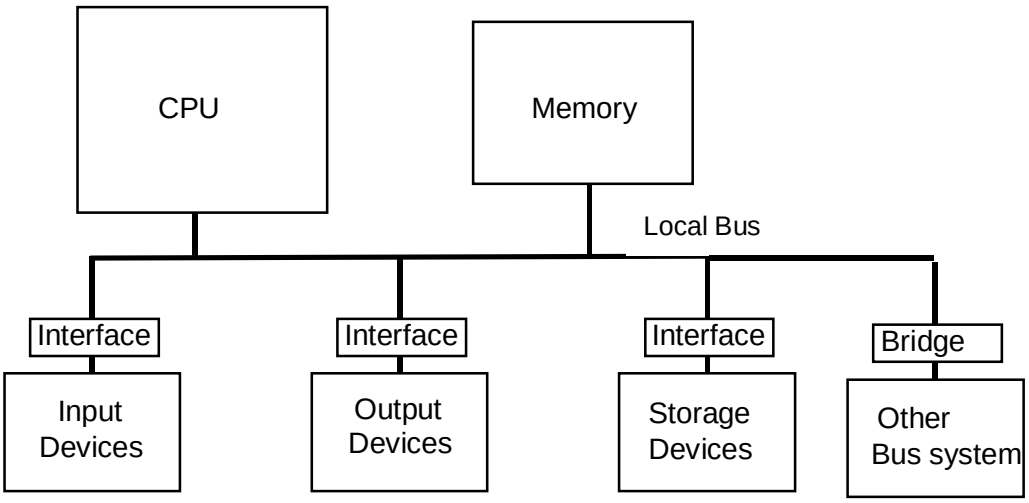


**Figure 1.1 Local bus**

(v) *Random logic:* For simple devices, random logic circuits may be used to check the status registers, read and write the data registers, perform timing, handle interrupt

signals and other functions. However, an embedded micro controller chip is frequently used.

## 0.4. Interfacing to Input/Output Devices

Interfacing between a CPU and a peripheral usually involves a trade-off between hardware and software. The advantage of hardware is speed, whereas the disadvantages are cost and inflexibility. The advantage of software is versatility, whereas its main disadvantage is its slow speed.

Interfacing can be handled entirely by the CPU (in software, with minimal hardware support), or at the other extreme, by a dedicated (intelligent) peripheral controller (in hardware, with minimal software support). Somewhere between these two extremes is the more usual master (CPU)-to-slave (peripheral) configuration, where the (dumb) peripheral support chip has a number of built-in functions, accessible by the CPU writing various bit patterns to its onboard control (command) register(s). The peripheral support chip handles many of the tasks with which the CPU would otherwise need to concern itself.

**Table 1.1 Typical peripheral support chips.**

| FUNCTION | MOTOROLA | INTEL | ZILOG |
|---|---|---|---|
| PARALLEL INTERFACE | MC6821 (8-bit)<br>MC68230 (16-bit) | i8255<br>i8256 | Z8420<br>Z8036 |
| SERIAL | MC6850 | i8256 | Z8440 |
| PROGRAMMABLE TIMER | MC6840 (8 bit)<br>MC68230 (16-bit) | i8253,54 | Z8430<br>Z8036 |
| CRT CONTROLLER | MC6845 (8-bit) | i8275 | |
| KEYBOARD | | i8279 | |
| BUS CONTROLLER | | i8286, 89, 289<br>i82188, 288 | |
| FLOPPY DISK CONTROLLER | MC6843 (8-bit) | i8271 | |
| PRIORITY INT CONTROLLER | MC6828 | i8259 | |
| DMA CONTROLLER | MC6844 (8-bit)<br>MC68440 (16-bit) | i8237, 57 | Z8410 |
| MEMORY CONTROLLER | | i8202, 3, 7, 8<br>(DRAM) | |
| MEMORY MANAGEMENT | MC6829 (8-bit)<br>MC68451 (16-bit) | | Z8010,15 |

A recent trend has been the emergence of coprocessor peripheral support chips which enable the CPU and coprocessor to carry out their respective tasks concurrently, communicating with each other upon completion. In many cases, these coprocessors are so intelligent that their processing ability rivals that of the host CPU. The resulting configuration more closely resembles a multiprocessor or distributed processing system, rather than the more traditional CPU-support chip master-slave relationship. For example, the PCI support chips that your have learned in CE302.

Another trend has been the emergence of custom peripheral chips where, rather than use a standard off-the-shelf support chip, a designer programs a dedicated single-chip microcomputer to perform the desired task (for example, the i8041 used in the keyboard of the IBM PC Keyboard). The success in the marketplace of specific microprocessor chips depends not only on the inherent qualities of the parent CPU, but also on the ready availability of both support chips and software for the processor in question. Ready access to data sheets, application notes and technical supports (within the local geographical area) are further factors that sway a user towards one particular processor over another.

In terms of functionality, it could be argued that one specific peripheral chip is superior to another; manufacturers can supply so-called benchmarks which purport to show the superiority of their products over their competitors. Such benchmarks should be regarded with a certain degree of healthy scepticism however, unless they originate from an 'independent' third party. As a rough rule-of-thumb, designers should choose support chips from the same processor family as the CPU itself; whether they be actually manufactured by the chip designer, or by a second-source company manufacturing the chips under license. After all, such support chips are *designed* to work with that particular microprocessor (although this did not prevent Apple from using Intel, Zilog and Synertek chips to support the Motorola CPU in their Macintosh computer!).

Generally speaking, 32-bit support chips from the same manufacturer are more powerful than 16-bit support chips, just as 16-bit chips are more powerful than 8-bit support chips. Comparing 8-, 16- or 32-bit support chips from different manufacturers is not that straightforward however; a lot depends on the particular application. Independent comparisons usually show that such support chips are functionally equivalent at least. Table 1.1 provides a representative sample of the more commonly available microprocessor peripheral support chips.

Nowadays, whenever a new microprocessor is released into the marketplace, it usually incorporates the ability to utilise the already (and often quite substantial) support chip and software base from the previous processor generation. For example, in the case of the MC68000, Motorola designed into this processor the ability to interface to the already existing, tried and proven family of MC68000 support chips; although the MC68000 is basically an asynchronous processor, it also has the ability to interface to synchronous support chips. Newer (asynchronous) 16/32-bit support chips were subsequently developed to interface specifically to the MC68000(20,30). A similar situation applies with the Intel family, which were able to utilise the earlier i8080(86) support chips.

The Motorola MC68000 families use memory-mapped I/O and run on a synchronizing E(nable)-clock. Intel and Zilog support chips do not depend for their operation on a synchronizing clock signal from the CPU. They also assume ported I/O rather than memory mapping. In recent years, support chips have been designed specifically to interface to the newer 16/32-bit processors released by Intel, Motorola, Zilog, National Semiconductor and others.

Intel processors support ported I/O, whereas Motorola processors support memory-mapped I/O. The advantage of memory mapping is that all I/O locations are addressed in exactly the same manner as memory locations; no special repertoire of I/O instructions is therefore required (along with the various addressing modes for each). Thus the overall size of the instruction set is reduced. A disadvantage with memory mapping is that some additional external address decoding is necessary to discriminate between memory and I/O devices (this is done within the CPU for ported I/O schemes). Another disadvantage is that some of the available address space is taken up with I/O device locations; not all of it is available for memory. Note, however, that with ported I/O systems, not all of the available I/O address space is always used. This means that memory-mapped I/O is more flexible.

It should be pointed out that memory mapping can be used with Intel processors, over and above the inbuilt ported I/O structure, assuming that the additional external decoding circuitry is provided.

When interfacing between a CPU and a peripheral device, it is usual to provide some form of *bi-directional buffering* between the two. This is often in the form of onboard registers within a peripheral support chip. Having the data buffered within the I/O port means that the CPU can, in effect, be reading from (or writing to) the I/O port at the same time as it is performing some other function. This leads to rudimentary concurrent operation. Such buffering also allows for different speeds between the CPU and peripheral device, for example, a printer will not be able to keep pace with characters reaching it from a CPU: a block of characters (say 256 bytes) is therefore buffered within the I/O section (in dedicated local memory).

Buffering is also often required between CPU and a peripheral device due to the different electrical characteristics of the two devices. The CPU usually operates at TTL logic levels (+5 V[HI] and 0V[LO]) whereas peripheral devices can have many different voltage and current characteristics.

Voltage level translation, scaling (up or down), current boosting (or limiting), or even analog-to-digital (and digital-to-analog) conversion might be required, depending on the peripheral in question.

An I/O interface can also act as a translator, adapting signals from one format (say ASCII or BCD) to another (binary).

Software input and output buffers will often is required over and above the hardware buffers just discussed. Double buffering is commonly used, while circular buffering is needed for highly asynchronous operation such as network communication protocol. Double buffering allows the peripheral device to transfer a block of data into one area of memory while the CPU is simultaneously fetching data placed into another area of memory on a previous occasion. Each buffer is used alternately by the CPU and disk. At times, the disk will be transferring data into buffer A, at the same time as the CPU is fetching data from buffer B. At other times, the disk will be loading buffer B, and the CPU will be fetching data from buffer A. Double buffering thus allows for concurrent operation.

In a circular buffer, the PC and the peripheral device are simultaneously filling and removing items from the First-In-First-Out (FIFO) buffer. In practice, care must be taken to select a buffer length sufficiently large so that buffer overflow does not occur (pointer movements are restricted to *modulo of buffer length*). The circular list is empty whenever the PC and the peripheral device pointers become equal. A typical application of such a data structure is in a serial data communications link, where characters arrive at random intervals, but where the CPU removes them at regular intervals. (The circular list thus acts as a buffer between the asynchronous peripheral device and the synchronous processor.)

## 0.5. I/O Interfacing Techniques

The times at which data reaches a computer from the outside world can be quite unpredictable. The processor therefore needs some means of synchronising itself to external events, for scheduling I/O transfers. There are two main methods of achieving this synchronisation, namely polling and interrupts.

### 0.5.1. Polling

Polling is a software technique whereby the processor continually asks a peripheral device if it needs servicing. During input, the I/O device sets a flag in the status register (SR) when it

has data ready for transferring to the CPU. Subsequently when the device is polled, the processor reads this status and goes on to read the data in. Similarly when the CPU has data to output, it polls the device to check for a ready status, upon which data is then written into the peripheral device. Several such I/O devices can be polled in succession, with the processor jumping to different I/O software routines, depending on which flags have been set.

Polling works well if a minimum amount of processing is required in response to each set flag. For slow I/O transfers, the processor will be spending most of its time in its polling loop, asking each I/O device in turn if it has any data ready. This amount to wasted time, during which the CPU could be carrying out other useful tasks. In applications where there is little else for the processor to do (such as in keyboard scanning, for example), this is fine, but for applications which involve substantial calculation as well, this amounts to inefficient use of the processor. The advantage of the polling technique is its simplicity.

### 0.5.2. Interrupts

Rather than have the CPU continually asking I/O devices whether they have any data available (and finding most of the time that they have not), a more efficient method is to have the I/O devices tell the processor when they have data ready. The processor can be carrying out its normal function, only responding to I/O transfers when there is data to respond to. On receipt of an interrupt, the CPU suspends its current operation (storing the contents of its program counter, SR and other registers), identifies the interrupting device, then jumps *(vectors)* to the appropriate interrupt service routine (interrupt handler).

The advantages of interrupts, compared with polling, are the speed of response to external events and the reduced software overhead (of continually asking I/O devices whether they have any data ready). Its main disadvantages are software commissioning and maintenance, and the additional hardware overhead required (in the form of device identification, supplying vector information and priority resolution).

A typical interrupt interface requires two handshake controls. The two handshake control lines that interface between the CPU and I/O port are Interrupt Request and Interrupt Acknowledge. Most systems will have more than one I/O port, with its associated set of interrupt handshake lines, so that some form of external (hardware) arbitration will be necessary in order to resolve priorities. This priority resolver decides in which order the devices will be serviced in the event of multiple interrupt requests occurring.

With more than one possible interrupting device, a priority must be allocated to each. Within the CPU it will therefore be necessary to mask off lower priority interrupts, and to respond only to those of higher priority (non-maskable interrupts, however, must always be responded to by the processor, a typical example being a system control console). In cases where several devices are allocated the same priority level, additional external hardware (in the form of a PIC) will be necessary.

There will be some means within the CPU whereby interrupts can be enabled and disabled. For example, the very first function performed by the interrupt handler is to 'disable further interrupts', so that the current interrupt can be attended to first. Any interrupts that occur during the servicing of this first one will usually be latched into the I/O port, and remain pending until the processor is ready to deal with them (that is, providing their priority level was higher than the current mask).

Some processors also provide an auto vector facility, whereby the starting location of the interrupt service routine can be set automatically within the CPU, depending on its priority level.

### 0.5.3. Direct Memory Access (DMA)

Interrupts provide fast response to a peripheral device, but the servicing of the interrupt is performed in software. Sometimes I/O transfers need to occur faster than interrupts can manage, such as with disk I/O, high-speed graphics, or interfacing to a Local Area Network (LAN). The servicing previously carried out by software can be performed faster if it is done by specialised hardware. Such a dedicated controller is designed to perform one specific task only, namely the high-speed transfer of data between the I/O device and memory (and vice versa), but bypassing the CPU. Hence the technique is referred to as Direct Memory Access (DMA).

With DMA, the device actually takes control of the system bus for the time required to transfer the data, then hands back the bus to the CPU upon completion. Most harddisk controller uses a DMA Controller (DMAC) for data transfers between disk and memory. The disk is connected to a peripheral controller which communicates to the DMAC via the Transfer Request and Transmit Acknowledge handshake lines. The DMAC in turn interfaces to the CPU via the Bus Request, Bus Acknowledge and Transfer Complete lines.

It is important to realise that the DMAC itself is accessed by the CPU as a typical I/O device, with its own unique port ID, vector and interrupt priority. However, once the DMAC has been granted the bus in 'burst' mode, it cannot be interrupted by the CPU (in this sense the DMAC has priority over the processor). With 'interleaved' or 'cycle steal' DMA, the DMAC and CPU share alternate bus cycles.

DMA transfers are inherently faster than either interrupts or polling, since they bypass altogether the imbedded instruction fetch-decode-execute cycles of the CPU. Rather than fetching instructions sequentially from memory, a DMAC has inbuilt (firmware) instructions. Moreover, these instructions can be executed concurrently, for example, transferring data at the same time as decrementing a byte counter.

## 0.6. Summary

In this introductory chapter, we revised interfacing of Input/Output (I/O) devices to a CPU. In current development, this is usually facilitated by the use of user-programmable peripheral support chips. Such peripheral chips are connected to the CPU in master-slave configuration, with their onboard command and status registers accessible via the system bus. More recent peripheral support chips comprise dedicated I/O processors, some of which employ a coprocessor philosophy, which leads to concurrent operation of the peripheral chip and the CPU.

Memory-mapped and ported I/O schemes were compared and contrasted, and the advantages and disadvantages of each mentioned.

The need for both hardware and software buffering between CPU and peripheral device was highlighted - the former for matching electrical signal levels, and the latter for the temporary storage of data until the CPU (and/or peripheral) can deal with it.

The basic I/O techniques of software polling, interrupts and Direct Memory Address (DMA) were introduced, and their respective advantages and disadvantages pointed out.

## 0.7.Reading Guides

WILKINSON & HORROCKS, Sections 2.2, 2.4
FULCHER, Sections 1.2, 1.5, 1.6, 3.4