Hard disk know-how

Types, performance, upgrades and SCSI *vs* IDE. Eleanor Turton-Hill helps you get to know your hard disk.

The hard disk is that part of your system which holds all the programs, documents and data when your PC is switched off. The longer you have your computer, the more documents you create and the more data you store, the more valuable your hard disk becomes. In fact, hard disks which crack up can put small companies out of business in a flash.

In short, your hard disk is the storage place for your valuable work, so it's important to have a good understanding of how it works. Here, I'll be answering some common questions about hard disks including what to look for when shopping for one, how to upgrade and how the hard disk affects the performance of your PC.

Virtually all hard disks today use a set of read/write heads that float above a rapidly-rotating rigid metal plate. Here you can see that the hard disk is made up of multiple platters and multiple heads which increase the capacity and speed of the disk drive

How many types of hard disk are there?
It is now well known that the internal parts of the average home PC do not (alas) fit together like the bricks in a Lego set. However, as the personal computer market matures, universal standards are gradually making an impact on the design and compatibility of PC hardware. But it's a slow process and, as with many other consumer products, standardisation is a difficult and messy process.

One of the earliest and most significant standards introduced into PC hardware was IDE (Integrated Drive Electronics): a standard which controls the flow of data between the processor and the hard disk.

The IDE concept was initially proposed by Western Digital and Compaq in 1986 to overcome the performance limitations of earlier subsystem standards like ST506 and ESDI. The term IDE itself is not an actual hardware standard but the proposals which had been put forward were incorporated into an industry-agreed interface specification known as ATA (Advanced Technology Attachment). ATA defines a command and register set for the interface which creates a universal standard for communication between the drive unit and the PC.

One of the major innovations brought about by IDE was the integration of the disk controller functions onto the disk drive itself. The separation of the controller logic from the interface made it possible for drive manufacturers to enhance the performance of their drives independently — there were no performance-boosting features incorporated into the ATA interface itself.

Mass acceptance of the IDE standard hinged on its ability to serve the needs of the market in terms of the two most important criteria: cost and compatibility. Over the years, these two factors have been more significant to mainstream PC users than high performance and thus IDE has become established as a mass-market standard.

How is disk performance measured?
The speed of a hard disk can be measured in different ways and it is important to know exactly what figures are being quoted when you're shopping for a new one. The performance of your hard disk is very important to the overall speed of the system. A slow hard disk will hinder a fast processor like nothing else in your system can.

As an initial gauge, look for the drive's "average access time": the time taken by the drive to locate the track on which a piece of data is stored, and the specific place on that track where the data is sitting. This is usually quoted in milliseconds (ms).

In addition to "average access time", also look out for "transfer rates". The transfer rate is the speed at which the drive can deliver the data from the disk platters to the CPU. This is generally described in megabytes per second (Mb/sec).

In order to get an accurate view of a hard drive's performance, the average access time and the transfer rate should be looked at in tandem. Drive makers and dealers have a reputation for bending the truth on such issues and are often found to quote the fast access time of a drive, without any mention of the transfer rate —you'll see this in advertisements, too. Unfortunately, a high access time coupled with a slow transfer rate produces a slow drive.

Because access time is measured in milliseconds and transfer rate is measured in Mb/sec, the overall drive performance can be difficult to get your head around. Essentially, you're looking for the lowest possible access time and the highest possible transfer rate.

Another measure of hard disk performance, of which you should be aware, is "seek time" which is conveniently confused (by some people) with the access time. Seek time is also measured in milliseconds and defines the amount of time it takes a hard drive's read/write head to find the physical location of a piece of data on the disk. The seek time says absolutely nothing about the speed of a hard drive.

The importance of the access time and transfer rate is that they tell you how long a hard drive takes to locate and retrieve data.

How do I upgrade my hard disk?
Computer technology changes quickly. Every year, processor speeds increase and hard drive capacity grows. Before you know it, there's a new generation of feature-rich software waiting to cripple your poor aged PC. Sooner or later you'll have to face up to the fact that your machine is becoming outmoded, and find some way of dealing with this.

The speed of your hard disk has a major impact on the overall performance of your machine. Hard drives found in old computers tend to be physically large, slow, power-hungry and of limited capacity. If your machine is really ancient, a modern IDE hard disk would greatly improve performance.

Before splashing your money around, there are a few basic facts you need to know about your PC. First, take the lid off and look at the arrangement of components. The first and most obvious thing to find out is whether you actually have room for another hard disk.

Check up on the manufacturer of your hard drive and the drive's type (if you've lost your manual, look at the machine's setup screen) before you go shopping for a new hard disk because the BIOSs (Basic Input/Output System) in some older machines do not officially support IDE. Ask the dealer whether the new drive will work in a "master/slave" configuration with the old one and finally, cover yourself by checking that the drive you buy has a "no questions asked" return policy.

There are essentially two types of modern drive interface: SCSI and IDE (*see the panel*). We'll concentrate on the more common IDE variety. Unfortunately, adding a second IDE drive is not always a simple procedure, because not all IDE drives work to the same standard. If both your drives adhere to the ANSI standard (ATA), both drives should happily co-exist. If they are incompatible, you could end up throwing away your old one.

IDE drives can control two hard disks on the same cable and, in order to make them work together, one must be set up as a slave and the other as a master. This can be done fairly simply by moving a jumper at the back of the drive from one position to another. When you plug in the drive, make sure that the cable is plugged in the right way around, otherwise your machine will appear to be dead when you turn it on. Pin 1 is usually marked so that you can line up the cable in the correct way.

The hard disk which you buy will generally be faster than the one you've already got so set this one up as the master and your existing one as the slave. They'll work more efficiently together if you store your applications on the faster disk and data on the slow one.

Once you've physically connected your new hard drive to the machine, you will have to configure the PC's BIOS . The BIOS contains a series of entries such as number of heads, cylinders and sectors per track which define the type of hard drive in the machine.

Generally, you can get into the BIOS setup utility by pressing a key combination when you boot up. Here you'll need to configure the hard drive type number as well as other system configuration details. Make sure you have all the information you need before you go anywhere near your BIOS or you could end up frustrated for hours, or even days, trying to put it right.

IDE and SCSI

If you've ever leafed through one of our *PCW* computer group tests you couldn't fail to have noticed the many seemingly-incomprehensible acronyms like those I've already mentioned. They refer to interface standards which define the way in which the hard drive connects to your PC. In the first generation of computers, the electronics to manage the hard disk were placed on a separate controller card. Technology has moved on since then and the same advances in microchips, which have led to faster processors and cheaper memory, now enable the controller function to be placed on the disk itself.

- **Integrated Drive Electronics (IDE) is currently the most common hard drive interface and is also (not by coincidence) the least expensive. IDE disks are connected to an interface card by a cable which extends the signals from the bus inside the PC. The cable does not plug directly into the ISA bus so it either goes into an interface port on the main board, or into an interface card. The IDE standard supports two disks connected together, the first acting as a controller and the second as a slave, with both disks sharing a single I/O (input/output) address and interrupt.**
- **Enhanced IDE (EIDE) is a much-upgraded version of IDE. All computers built since 1994 should have an EIDE hard disk controller and this provides many advantages over IDE. Firstly, EIDE can support four devices (instead of two) which don't necessarily have to be hard disks. They can also be CD-ROMs or tape drives compatible with the EIDE standard. Secondly, IDE was always restricted in that it would not support hard disks larger than 528Mb. The third, but certainly not least, improvement was in the massive increase in data throughput compared with standard IDE. This massive speed enhancement puts EIDE on an equal level with SCSI as a high-end drive interface.**
- **SCSI (pronounced "scuzzy") stands for Small Computer Systems Interface and is another standard for connecting hard drives and peripherals to your PC. SCSI hard drives are very fast and very expensive. They act as good interfaces for high capacity hard drives used as network file servers and for very high-powered scientific and engineering applications. For the average user, it is not really worth spending lots of extra money on SCSI, especially as EIDE now provides similar performance results.**

•PCW Contacts

Eleanor Turton-Hill welcomes feedback and suggestions from readers. She is at **ellie@vnu.co.uk**

Programming primer

Pay attention at the back! Eleanor Turton-Hill provides an overview of programming principles and common structures.

A computer program is nothing more than a sequence of instructions which are designed to make your computer behave in a certain way. There are literally hundreds, if not thousands, of programming languages and they are all different. Some have only superficial differences, while others differ hugely. Nevertheless, all programming languages have *some* things in common: they all attempt to describe the processing of data, and because of this they all share certain basic facilities.

Here, I'll be giving an overview of programming principles and common structures (and if you're about to embark on a programming course, this will give you a good head start). Once you've understood the basic principles of programming, you should be able to get to grips with practically any language.

What is a computer program?

Computers are not very intelligent things. They can't philosophise on the meaning of life, but they are very good at performing boring, repetitive tasks very quickly and very accurately.

Deprived of a program written by a human being, a computer can do nothing. If a program is badly written, it can very obediently turn perfectly good data into complete gibberish… just one small error can alter the behaviour of an entire application.

There are three basic facilities which all programming languages must have. Firstly, they must have some way of representing data and performing operations on it. Most provide some form of primitive data and structured data, and some allow you to create your own data types.

Secondly, programming languages must provide some kind of evaluation mechanism; some way of describing the way in which you would like the data to be transformed.

Thirdly, every programming language must have a set of naming and declaration rules. These rules state when you can and cannot refer to other elements of a program.

Constants and variables

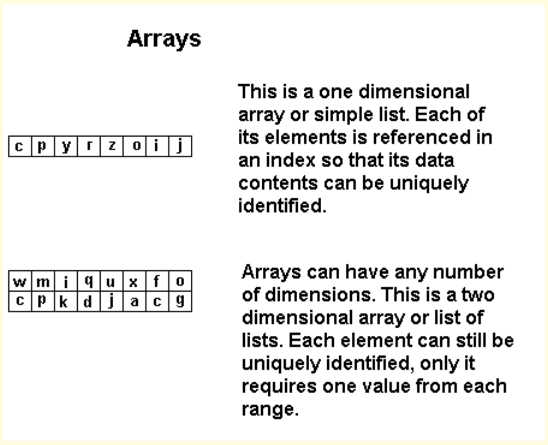
Most computer programs use some numbers which do not change their value throughout the duration of the program. These values are called constants and are usually declared at the beginning of the code.

The rate of VAT, for instance, is a figure which may be referred to many times in a program but always as the same value. This can be declared at the beginning of the code (e.g. VAT = 17.5). Then, every time you need to use the figure, you can simply refer to "VAT". What's more, when the rate of VAT alters, all you need to change is the constant declaration at the top of the code.

A variable, as you've probably guessed, is a value which changes throughout the program. The value of a variable can be altered and manipulated by the program, and certain operations may be performed, depending on its value. All constants and variables have names and values.

Data types

Before a variable can be referred to in code, it must first be declared and given a data type; the program needs to know the type of thing with which it is dealing.



This is a one dimensional array or simple list. Each of its elements is referenced in an index so that its data contents can be uniquely identified.

Arrays can have any number of dimensions. This is a two dimensional array or list of lists. Each element can still be uniquely identified, only it requires one value from each range.

Most programming languages acknowledge several data types, the major distinction being between text and numbers. In Pascal, there are four basic data types: char (for character strings), int (for integer values), and real (for double precision real numbers).

There is also a Boolean type which is used as a flagging mechanism. A Boolean variable can hold just one of two values like Yes/No, or On/Off, or 1/0.

Another data type used in most programming languages is the array type. This is used for managing lists. When an array is defined, it is given an index which enables you to uniquely identify any one of its elements *(see Arrays, above)*.

Controlling program flow

Every language has different conventions for beginning a program. In both C and Pascal, programs start with the reserved word "main". This makes it clear where execution should begin *(Fig 1)*.

The following text shows three basic control structures which are universal in procedural programming languages. Each structure has variations and each is written slightly differently, depending on the syntax rules of the language you're using. Here, I've made the examples simple and used pseudo-code to illustrate their structure.

Fig 1 A 'main' point

```
      C                                 Pascal
main () {                       PROGRAM main (input,output);
definitions;                            definitions
statements;                     BEGIN
}                                        statements
                                END.
```

● IF THEN ELSE

The IF statement is probably the easiest programming structure to understand. It

What is object-orientated programming?

There's been much confusion recently about the meaning of the term "object-orientated". This is largely because it has been bandied about by all and sundry to mean a multitude of different things. Put simply, object-orientated programming is a collection of design principles for writing code. It is only supported by some languages and aims to break programs down into manageable units called "objects". The core idea behind this is to make components which are sufficiently general purpose as to enable them to be re-used in other programs.

This method of designing code yields many advantages. Firstly, a program which is divided into independent chunks is easier to understand, easier to debug and generally easier to maintain. Secondly, if many of those chunks are re-usable, time will be saved in future projects. Thirdly, an application made out of many independent parts can be more easily created by teams, thus increasing productivity.

The first object-orientated programming languages (Simula and Smalltalk) were conceived more than 20 years ago, but it's only recently that people have started taking its principles seriously. C++ is now the most popular object-orientated language. Objects within C++ can correspond to real-world entities such as bank accounts, employees or customers. But they can also correspond to computer hardware and software components such as communications ports, or video display windows, or data structures such as stacks or lists.

What are classes?
Many of the objects that a program uses have the same structure. A program which simulates the operations of a bank, for example, will need many account objects and many customer objects. Once the structure of an object has been set up, it is possible to produce many copies of it. This is done by using "classes"; each contains a complete description of one kind of object. Truly object-orientated code must have the three essential characteristics of inheritance, encapsulation and polymorphism. This may sound frighteningly technical, but in fact the whole thing rests on three fairly simple concepts.

One: classes can be defined from scratch, or they can be created by modifying an existing class. Derived classes take on all of the characteristics of the existing class, plus any modifications. This is called inheritance, and can save you an incredible amount of time and effort in code writing. Two: objects are available to the programmer through an interface which responds to a limited number of different kinds of message. The internal structure of individual objects is hidden from the programmer and this data hiding, or encapsulation, simplifies the use of objects. And three: a major attribute of an object-orientated language is that all the objects of the derived classes of a parent class are type compatible. This means that a derived class can be used anywhere that the parent class is expected. This is called inheritance polymorphism and enables clients of a family to see a simple uniform interface.

will execute one or other group of statements depending on the value of a condition. We use this structure in normal language all the time: "If it's sunny, we'll go out, otherwise we'll stay at home".

In code, it looks more like this:

```
IF condition true THEN
  instructions
ELSE
  instructions
END IF
```

● WHILE DO

The WHILE statement is iterative rather than conditional. It will execute a statement continually until a condition no longer holds. This translates to normal language something like this: "While John is well, he will keep working. If he is unwell, he will stop".

In code, it looks something like this:

```
WHILE Condition is true
  DO Instructions
WEND
```

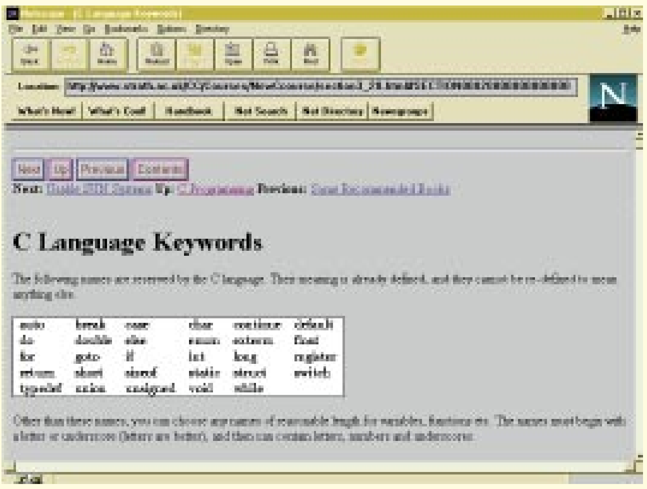
● FOR..NEXT

The FOR..NEXT control structure is also a repeating routine. It is used to execute a single statement, a specified number of times: "For the next five days, I'll be going to work".

In code, it looks rather like this:

```
FOR
 n=1 TO 5
Instructions
NEXT
```

There are plenty of good programming tutorial pages on the internet. Check out this one, written by Steve Holmes of the University of Strathclyde, for some lessons in C: http://www.strath.ac.uk/CC/Courses/NewCcourse/ccourse.html


Netscape - [C Language Keywords]

C Language Keywords

The following names are reserved by the C language. Their meaning is already defined, and they cannot be re-defined to mean anything else.

auto	break	case	char	continue	default
do	double	else	enum	extern	float
for	goto	if	int	long	register
return	short	sizeof	static	struct	switch
typedef	union	unsigned	void	while	

Other than these names, you can choose any names of reasonable length for variables, functions etc. The names must begin with a letter or underscore (letters are better), and then can contain letters, numbers and underscores.

Structure

Once you've got used to the idea of variables, data types, and basic control structures, you're ready to start writing simple programs. But when your code starts to become more complex, then you'll have to learn about scope rules and structure.

It's easy to turn a perfectly good working program into complete garbage if you don't follow a few design principles. Your program may still work quite well but will gradually become unreadable and, worst of all, unmaintainable.

Over the years there have been many theories about how programs should be designed. The idea of the procedure emerged in the seventies with C and Pascal. It attempted to break code down into manageable and well-specified chunks, making it easier to write and maintain, especially by large teams. This "modular" style of programming, which is based on the idea of packaging data and functions, developed into what is now known as "object-orientated" code (*see the box, above*).

If you are thinking of learning a programming language, there are plenty of ways to get started. Turbo Pascal and Turbo C++ are both available from Borland, in DOS and Windows versions. Microsoft offers a Visual Basic Pro and Visual C++ Student Pack which you can get for a street price of about £80. ▣

PCW Contacts

Eleanor Turton-Hill welcomes any feedback and suggestions from readers. She is at **ellie@pcw.ccmail.compuserve.com**

Borland 0990 561281
Microsoft 01734 270000

Sound principles

Eleanor Turton-Hill explains how to make the most of the sound-making capabilities of your PC.

These days, most PCs come with a sound card pre-installed but very few people fully exploit the sound capability of their machines.

Sound cards don't only make games and multimedia applications sound great but with the right software you can also compose, edit and print your own music as well as learn to play the piano, record and edit digital audio, and play audio CDs from your desktop.

Before you start fiddling with your recording software, however, it helps to understand some of the underlying principles of sound generation on PCs. Here, I've given a brief summary of the most important technical concepts relating to sound.

Why does a PC need a sound card?

Sound is a relatively new capability for PCs because no-one really thought about it when the PC was first designed. The original IBM-compatible PC was designed as a business tool, not as a multimedia machine, so it's hardly surprising that nobody thought of including a dedicated sound chip in its architecture. Computers, after all, were seen as calculating machines; the only kind of sound necessary was the beep that served as a warning signal.

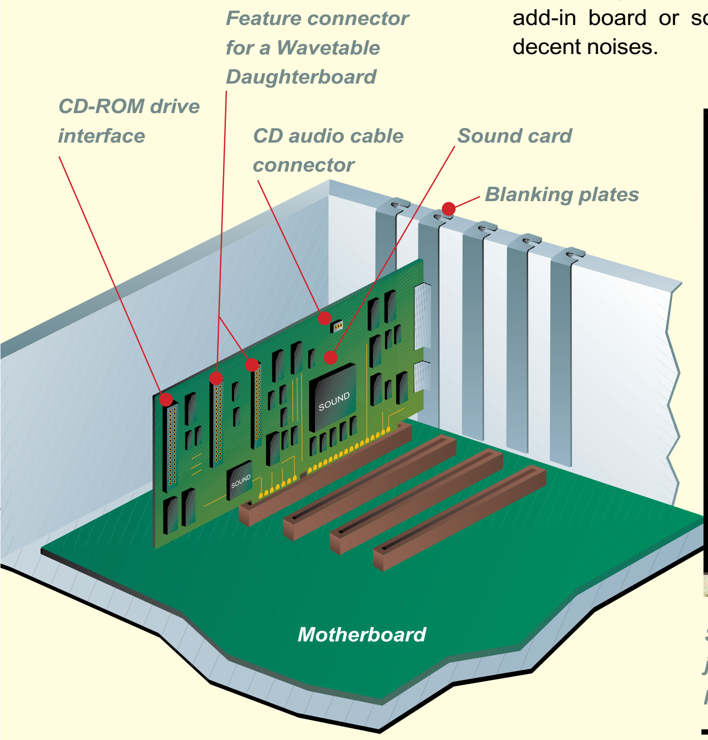
For years, the Macintosh has had built-in sound capabilities far beyond the realms of beeps and clicks, but PCs with integrated sound are still few and far between. That's why PCs continue to require an add-in board or sound card to produce decent noises.

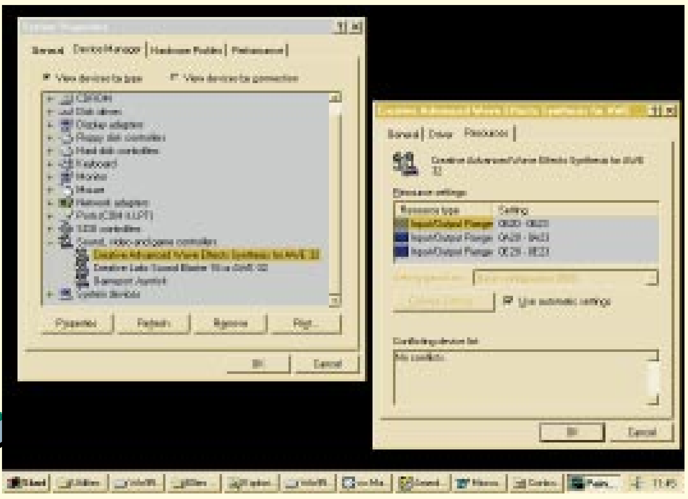
The popularity of multimedia applications over the past few years has accelerated the development of the sound card, and the increased competition between manufacturers has led to these devices becoming cheaper and more sophisticated.

Sound cards can cost from as little as £30 to as much as £300. Most modern boards now use WaveTable technology which uses sampled sounds of real instruments. WaveTable has greatly improved the quality of sound available on a PC compared to the synthesised sounds of a few years ago. Digital Signal Processing (DSP) technology is becoming more widespread on sound cards and this allows reverb, delay, and other digital effects to be applied to instruments or samples.

What is Plug and Play?

Plug and Play is a standard introduced by Microsoft in its latest operating system, Windows 95. Essentially, Plug and Play was introduced to make the installation of new devices easier by automating the whole process. Windows 95 includes drivers for a large number of sound cards and should automatically detect Plug and Play cards when they are installed.





Sound cards which conform to the Plug and Play standard have only just started to appear on the market. Those which live up to their promises should install automatically and report no conflicts

WaveTable daughterboards

If you're thinking of improving the sound capability of your PC, check your existing sound card first to see if it has a feature connector. If it does, then you could save some money by upgrading the card with a WaveTable daughterboard.

WaveTable daughterboards are compatible with any 16-bit sound card that has a feature connector. The connector is easy to find and will be located at the bottom left-hand side of the card near to the blanking plate. It looks similar to a CD-ROM interface, only smaller. Some cards, including the Value edition of the SoundBlaster 16, may not have this connector, so check first.

Installing a WaveTable daughterboard couldn't be easier. Simply remove your sound card and "sit" the daughterboard on top, making sure the connectors are firmly attached. Three plastic spacers will also be provided which prevent the two cards from damaging each other.

The most interesting thing about this type of upgrade is that you don't have to install any software for it to work and there are no dip switches to configure. If you want the daughterboard to be the default synthesiser in Windows, which is likely, you will need to edit the MIDI mapper, or MIDI output port if you're using Windows 95. This is straightforward and is explained in the accompanying manuals.

The quality of the instruments on each WaveTable daughterboard vary significantly. This is usually determined by how much ROM the card has. Most cards contain between 1Mb and 4Mb of samples and offer digital effects which include reverb, chorus and delay. Reverb gives the impression that the instruments are being played in large halls or churches, which is great for when you're playing Doom. When chorus is applied, the sound is similar to many instruments playing at once when only one is actually being used. Delay is just a posh word for echo.

The operating system reads your Config.sys and System.ini files and scans for existing drivers on installation. If the card's drivers are pre-bundled with the operating system, they'll be installed and configured for you. If not, you'll be prompted for an installation disk.

Sound cards which aren't Plug and Play-compatible must be installed manually using the Add New Hardware Wizard in Windows 95. As with many new Plug and Play devices, the "seamless integration" concept does not always work in practice; often, cards which claim to conform to the Plug and Play standard don't install smoothly.

WaveTable synthesis

WaveTable cards play back pre-recorded samples of real instruments. A WaveTable ROM, therefore, is an electronic table of waveforms. Whereas one FM sound card will sound much the same as the next, WaveTable cards differ significantly in quality. The quality of a card's instruments is determined by several factors: the quality of the original recording of each instrument; the sampling rate, or frequency, of the recordings; the number of samples used to reproduce each instrument; and the compression methods used to store the samples.

By using high ratio compression techniques, more samples (or instruments) can be "squeezed" into small amounts of ROM. There is a trade off with quality, however, as compression often results in loss of dynamic range and quality.

Every instrument produces subtly different timbres depending on how it is played. For example, when a piano is played softly, you don't hear the hammers hitting the strings. When it's played harder, not only does this become more apparent, but there are also changes in tone.

Many samples and variations have to be recorded for each instrument to recreate this range of sound accurately with a synthesiser. Inevitably, more samples require more ROM. A typical sound card may contain up to 700 instrument samples within 4Mb ROM. To accurately reproduce a piano sound alone, however, you're looking at between 6Mb and 10Mb of data. This is why there is no comparison between the synthesised sound and the real thing.

Digital signal processors

Digital effects can dramatically improve the overall quality of sound cards. Digital Signal Processors (DSPs) use complex algorithms to add reverb and other effects to give the impression that instruments are being played in large concert halls. Other popular effects include stereo choruses and delays.

Adding a stereo delay to a guitar part can "thicken" the texture and give it a spacious stereo presence. Chorus is also used to thicken instruments and gives the impression that many instruments are playing when, in fact, only one is being used.

What is MIDI?

MIDI stands for Musical Instrument Digital Interface. It was developed as a communications protocol so musical instruments could "talk" to each other. MIDI was first developed to allow keyboard players to "layer" the sounds produced by several synthesisers, although today MIDI is used mainly for sequencing. A sequencer is a piece of software that records and plays back MIDI information. It allows complex musical arrangements to be built up that would otherwise be impossible for one person alone to play.

MIDI doesn't transmit any sound, just simple binary information. The ones and zeros that are sent down the cables contain very specific instructions. The most common instructions tell the receiving instrument to play a particular note for a duration of time — a note-on message followed by a note-off message. The same instructions contain details of how loud to play that note.

The synthesiser knows which sound to play using a simple program change message. This message tells the synthesiser to select sound number 67, for example, which in the General MIDI specification is a saxophone. Before General MIDI came into effect, sequences containing program change messages were meaningless if played back on an instrument other than the one on which it was recorded. This was because program 3 on the original synthesiser may have been a piano, while on another synthesiser it may have been a trombone. The result is a tune that sounds nothing like the composer intended.

In much the same way as you can have seven SCSI devices in a chain, MIDI communicates over 16 channels allowing up to 16 MIDI instruments to be played from only one interface. Since the majority of sound cards are multi-timbral, 16 instruments can be played simultaneously from only one device. Adding a second MIDI interface opens up another 16 MIDI channels. Some MIDI interfaces offer as many as 16 outputs, making it possible to access 256 at the same time. This might sound ridiculous, but in large MIDI setups you can easily run short of channels.

PCW Contacts

Eleanor Turton-Hill welcomes any feedback and suggestions from readers. She is at **ellie@pcw.ccmail.compuserve.com**

Hardware **basics**

Eleanor Turton-Hill **looks under the bonnet to demistify and explain the workings of your PC's engine components.**

Judging by some of the emails and letters I've been getting recently, there's still a fair amount of confusion out there when it comes to understanding hardware.

All computers have four basic elements: a processor, memory, storage devices and I/O devices. Understanding the relationship between these four units is an essential starting point if you're trying to get to grips with hardware, so here's an overview.

CPU

One of the first things you'll hear people talking about is the type of processor in their machine. This is the central processing unit (CPU) and is the single most important component in the machine because it processes data and controls all other parts of the computer.

Even the simplest processor is an extremely complex device. I'm not going to go into great detail here, but it is useful to have an outline of its main functions.

If you take the lid off your computer you will see several flat, black, blocks stuck to a green board. The CPU is the big square one usually marked "Intel" but sometimes "Cyrix" or "AMD".

Essentially, what the processor does is to store, move and manipulate data. It can only do very simple things like move numbers from one place to another or perform very basic mathematical operations, but it does all of these things very fast.

The CPU works by continually retrieving instructions from memory that tell it where to get data, what operations to perform on the data and where to store the results. The particular instructions which the CPU is following at any one time are determined by the program it is running.

If I say that my computer is able to produce reports, what I mean is that it has a program which instructs the CPU to execute a particular group of instructions which create a report. If I say that my computer knows my friend's phone number, what I actually mean is that the number is stored on my PC's hard disk.

Often, the CPU performs several operations on the same data, or it may need to hold the result from one operation, to be used on the next. Such data needs to be stored somewhere close at hand so it is put into address registers and data registers on the CPU itself. This prevents the processor from having to access the memory every time it generates data.

RAM

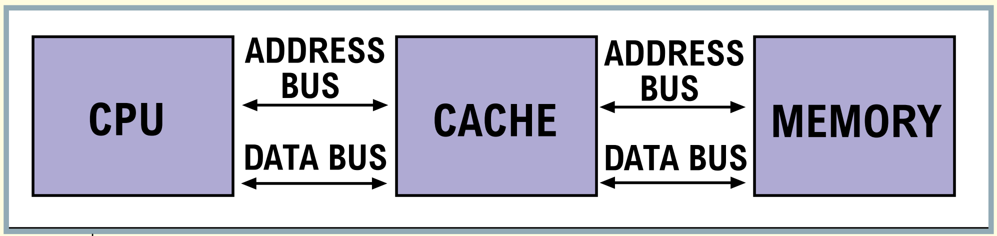
One of the concepts which confuses beginners is that of the location of data. The CPU spends its time fetching instructions and executing them according to what program is running. But where is the data? Is it in the hard disk?... the memory?... the cache? Well, the answer is that data is continually moved around. It's in different places depending on the particular stage of the CPU cycle.

The CPU can perform operations directly on data stored in its own registers, but it can also perform operations on the data in memory and on data stored on disks or tapes. But data on your hard disk or tape must first be brought into memory before the CPU can do anything with it.

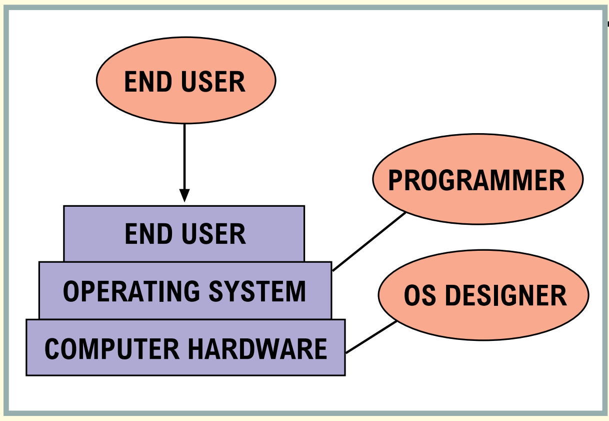
RAM stands for Random Access Memory. It's the working memory used by your computer to store instructions and data before they can be committed to the hard disk. Because RAM works much faster than the hard disk, it's used for handling all the data which is in constant use while programs are running. The hard disk is used for dumping any data which the system does not currently need.

Cache

Modern computers have a very large amount of memory compared with the first



A "cache hit" occurs when data required by the CPU is found in the cache. Because the cache provides data at high speed, it can dramatically improve the performance of the whole system



The hierarchical view of hardware can be extended to software. The ultimate aim of a computer is to provide a set of applications for the end-user. These applications are developed by the application programmer using a particular operating system (OS). The OS masks the details of the hardware from the programmer and provides the programmer with a convenient interface for using the system

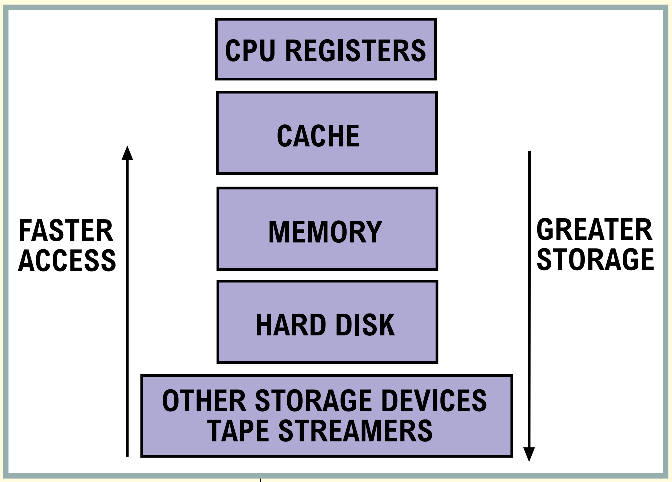
PCs of the early eighties and this has had an effect on the development of the PC's architecture.

Storing and retrieving data from a very large block of memory is more time consuming than from a small block. With a large amount of memory, the difference in time between a register access and a memory access is very great and an extra layer is required in the storage hierarchy.

A device called a cache sits in between the CPU's registers and main memory. This cache is much faster than main memory but slower than the CPU's registers. Its advantage is that it can hold more data than can be held in registers and can work faster than main memory.

When the CPU goes to read data from a certain address in memory for the first time, the cache goes to find it from memory. When it has retrieved the data, it records the address and data in its own fast memory. Eventually, the cache's memory fills up with records of addresses and data that the CPU has requested and when those same pieces of data are requested again, they are taken directly from the cache.

When the requested data happens to be in the cache, a "cache hit" is said to have occurred. Any requests which are made for data which is not already in the cache result in a "cache miss" and one of the records in the cache is then replaced.



In order to perform satisfactorily, the PC uses a hierarchy of memory/storage technologies. As you go down the hierarchy, the cost per bit decreases. Thus the smaller, more expensive memories are supplemented by the larger, cheaper, slower ones

Hard disk speed

The speed of a hard disk can be measured in lots of different ways, and it is important to know exactly what figures are being quoted when you're shopping for a new one. The performance of your hard disk is very important to the overall speed of the system: a slow hard disk will hinder a fast processor like nothing else in your system can.

As an initial gauge, look for the drive's "average access time". This is the time taken by the drive to locate the right track on which a piece of data is stored, and the specific place on the track where that data is sitting. This time is usually quoted in milliseconds.

As well as "average access time" look out for "transfer rates". The transfer rate is the speed at which the drive can deliver the data from the disk platters to the CPU. This is generally described in megabytes per second.

In order to get an accurate view of a hard drive's performance, the average access time and the transfer rate should be looked at together. Drive makers and dealers have a reputation for bending the truth on such issues and are often found to quote the fast access time of a drive without any mention of the transfer rate. You'll also see this in advertisements. Unfortunately, a high access time coupled with a slow transfer rate produces a slow drive.

Because access time is measured in milliseconds and transfer rate is measured in megabytes per second, the overall drive performance can be difficult to get your head around. Essentially, you're looking for the lowest possible access time and the highest possible transfer rate.

Another measure of hard disk performance of which you should be aware is "seek time", which is conveniently confused (by some) with the access time. Seek time is also measured in milliseconds and defines the amount of time it takes a hard drive's read/write head to find the physical location of a piece of data on the disk. The seek time says absolutely nothing about the speed of a hard drive. The importance of the access time and transfer rate is that they tell you how long a hard drive takes to locate and retrieve data.

Hard disk

The hard disk is the part of your system which holds all the programs, documents and data when your PC is switched off.

The longer you have your computer and the more documents you create and the more data you store, the more valuable your hard disk becomes. In fact, hard disks which crack up can put small companies out of business in a flash. Your hard disk is the storage place for all your valuable work.

The programs which you run (i.e. your word processor, graphics package or spreadsheet) are replaceable. When you buy your PC, you'll often get some of this software pre-installed on the hard disk, but you'll also get a set of floppy disks which you can use to re-install it if anything goes wrong. Anything else which you create should be instantly backed up onto a spare floppy disk.

The hard disk inside your PC is made of aluminium alloy covered with a magnetic coating. This makes the disk itself a pretty rigid plate: hence the name "hard" disk. Hard disks are completely sealed inside the disk drive and are not removable like many other media. They also spin very fast and have high recording densities, which means that they must be kept free from dust and any other kind of environmental contamination if they are to be maintained properly.

Thankfully, for the user, most hard disks look pretty much the same and people rarely know much about their internal workings. Hard disks have changed radically over the years, especially in terms of capacity. The smallest hard disks held a tiny 5Mb while these days 8Gb is the maximum hard disk capacity. The average PC bought today has between 500Mb and 1Gb in hard disk storage.

Data is recorded onto the magnetic surface of the hard disk in exactly the same way as it is on floppies or digital tapes. If you've ever defragmented your hard disk, then you probably have some mental image of how the surface of the disk looks. Essentially, the surface of your hard disk is treated as an array of dot positions, each of which can be identified and set to a binary "1" or "0". The position of each array element is not identifiable in an "absolute" sense, and so a scheme of guidance marks helps the recorder find positions on the disk. The need for these guidance markings explains why disks have to be formatted before they can be used. ▣

PCW Contacts

Eleanor Turton-Hill welcomes feedback and suggestions from readers. She is on **ellie@pcw.ccmail.compuserve.com**