
Chapter 3

Principles Of Data Communications

A Summary...

Basic concepts of computer to computer communications. Models for communications links and problems with long distance communications. Cross-talk in parallel conductors. Extension of internal communications concepts to external communications (parallel communications). The Centronics and IEEE-488 (GPIB) communications ports.

Read This Chapter If...

- ◆ You need to understand the basic concepts associated with making computer-based devices communicate with one another
- ◆ You need to understand the physical problems and limitations of the conductors used in communications
- ◆ You need to learn the basics of communications "protocols"
- ◆ You would like to come to terms with parallel communications and hardware hand-shaking.

3.1 Eight Fundamentals of Computer to Computer Communication

In order for two, computer-based devices to communicate with one another, even at the lowest level, a number of fundamental criteria must be satisfied:

- (i) A "physical" link between the two devices must be established. This is referred to as the transmission "medium" and may be realised through conducting cable, optic fibre cable or through electromagnetic wave propagation at radio, micro-wave or millimetre wave frequencies.
- (ii) Both devices must use the same representation for binary data on the external communications link. For example, both devices may need to accept that a binary "1" is represented by voltage or current level "A" and binary "0" is represented by voltage or current level "B" on the link.
- (iii) Both devices must use the same physical reference level with which to interpret data on the communication link. For example, if voltage is used for signal representation then a receiving device "X" must measure voltage on the link with respect to the same reference voltage that a device "Y" uses to transmit.
- (iv) If transmitted bit streams represent alpha-numeric characters, then both devices must interpret the characters in the same way. For example, if one device transmits bits representing ASCII characters, then the receiver must decode the bits as ASCII characters and not as EBCDIC characters.
- (v) The transmitting device must actively run software that sends data through its communications port and the receiving device must actively run software that reads in data through its communications port. The software must be co-ordinated so that transmission does not occur until the receiving device is ready to handle incoming data.
- (vi) The transmitting device must not send out data at a rate which is too high for the receiving device to handle.

- (vii) There must either be some pre-defined programming of the receiving device, so that it always handles incoming data in the same way, or the instructions for the handling of data must be sent by the transmitter, with the data. For example, a printer directly outputs all characters that enter its port, unless they are identified as special strings, which command the printer to perform a special function.
- (viii) If there is any possibility of the receiver being unable to handle the rate of incoming data, because of the time it has to take to process that data, then there must be some form of "hand-shaking" implemented. This should enable the receiver to signal the transmitter to stop and re-start transmission at any time, thereby preventing the loss of data.

These fundamental criteria may appear to be self-evident, but they are frequently mis-understood or not understood at all. Unless all of the above conditions are met, then there is no scope whatsoever for sensible data transfer to occur. You should therefore look upon these criteria as the 8 prerequisites of data communications.

The next question to be examined is, whether or not error-free data communications can be achieved by meeting only these 8 criteria. The answer to this is no. Satisfying the above criteria will allow data communications to take place, but much more work needs to be done in order to provide links that are both bi-directional and reliable.

3.2 Resolution of Conflicts in Communication - Protocol

Internal data communication within a single computer system is very well co-ordinated and synchronised. Devices are specially selected and integrated into a system to perform in a unified and ordered manner. Regardless of the autonomy of any one device or chip, within a computer system, a Central Processing Unit (microprocessor or processor board) is responsible for supervising the activities of all other devices. Hence a master-slave relationship exists and provides a stable platform for co-ordination of data transfer.

Once we decide to provide data communications outside the shell of a single computer then many of the luxuries of internal communications are lost. We face an environment of multiple vendors, differing electrical interfaces and differing forms of data representation. Moreover, we often have two equally intelligent devices that cannot automatically be configured into a master to slave relationship.

Within a single computer system, we have a situation where many of the devices (chips) are passive and need to be activated by the CPU in order to transmit data. Conflicts for use of the transmission media do not arise. However, when two or more computers wish to communicate, then it is the human user (system designer) who must ensure that conflicts for use of transmission media are resolved through appropriate hardware and software.

Within a single computer we generally have a sealed system, where the likelihood of fatal transmission errors is minimised by the absence of human intervention. If a device, such as a memory chip, fails within a computer then a fatal error occurs and the computer may completely lock-up. In an engineering control environment, this is probably more stable than a situation where a computer misinterprets data from an incorrectly functioning remote device and acts inappropriately upon that data. It is therefore essential that when two, or more, computerised devices communicate with one another externally, they can react to errors caused by external (remote) equipment. Typically, the sorts of errors that can arise are:

- the transmission medium can be physically broken
- the remote computer may be switched off or inoperative
- the transmission medium may be subject to electro-magnetic interference
- the remote device may attempt to transmit on the same transmission medium at the same time as the local device.

None of the errors, listed above, can be corrected by solely complying with the basic communications criteria outlined in 3.1. Therefore, in order for the normal process of error detection and correction to take place between communicating devices, the software and hardware on all computerised devices must be made to adhere to a common set of rules. These rules outline the methods for data transmission, error detection, resolution of conflicts for use of the transmission media and so on. When combined with a specification for the fundamental communications criteria of 3.1, the total set of rules is referred to as a "communications protocol".

At this point, you may think "so far so good", but where can I find this set of rules for communications, so that I can interconnect any two, computerised devices? Needless to say, there is no universal communications protocol for computer communication. There are countless numbers of specifications and protocols, some proprietary to computer manufacturers, others evolved from historical tele-type transmission techniques and still more that have been generated by various professional computer bodies around the world.

There is no question of one, particular protocol being universally "better" than all the others. Individual protocols have strengths and weaknesses, primarily because they are designed for specific applications and specific areas. For example, "Brand X" mainframe to "Brand X" mainframe office communications or PLC to PLC factory communications, etc. The real problem is that even where communications requirements are such that it would be sensible to use the same protocol to interconnect similar devices, the multi-vendor environment has made this difficult. For example, Brand X office computers cannot simply be "plugged into" Brand Y office computers, even though both are designed to perform a similar task, are essentially similar devices, transfer similar data and live in a similar environment.

We now look, qualitatively, at the way in which protocols work in regard to a simple, one-to-one, computer to computer link (a point to point protocol). Figure 3.1 shows the link. Let us say that we wish to transfer an ASCII file from device 1 to device 2. Let us also assume that the physical aspects of the link protocol have already been matched. That is, both devices satisfy common mechanical interfaces, voltages, binary data representation, etc. How does a protocol enable us to reliably transfer data between from one device to another?

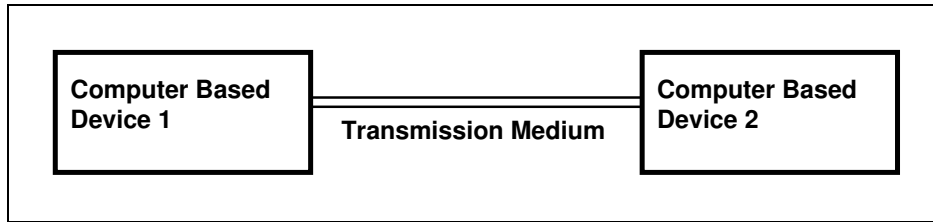


Figure 3.1 - A Point to Point Link

There are a number of phases that both devices must pass through in order to perform the common communications function of file transfer (from device 1 to 2). These phases ensure that the software on each device is structured to correct for errors or inconsistencies from the corresponding, remote device. The rules for each of these phases are clearly defined by a protocol and typical phases are as follows:

(i) Establishment of Link

Device 1 checks to see if Device 2 is present on the link by sending a specific "enquiry" message. If the link is active and device 2 is active then it should respond by sending back an "acknowledgement" message. Device 1 must track the time that device 2 takes to respond. If device 2 does not respond within a time interval (defined by the protocol) then device 1 assumes that the link is not active. This is called a transmission "time-out" error.

(ii) Issue of a Command and Command Qualifier

Device 1 sends device 2 a message, in a predefined format, which tells device 2 that a file is to be transferred. As a qualifier within the message, device 1 tells device 2 what to do with the file. For example, device 1 may tell device 2 to place the incoming file onto disk storage, with the file-name "FRED".

(iii) Acknowledgement of a Command

If device 2 has correctly received the command and qualifier from device 1, and is capable of carrying out the command, then it sends device 1 an acknowledgement message. The acknowledgement message tells device 1 that it can now proceed with further action needed to fulfil the command. If device 2 is unable to act upon the command from device 1, then it must respond with an error message. An error could occur on the receiver if, for example, the disk on which the incoming file is to be stored, is already full. The error response message would tell device 1 that it should not proceed with its proposed course of action.

(iv) ***Dissection of Messages***

All messages, command and otherwise, must be broken down into packets of manageable size for transmission. Thus if an error should occur in a packet, then only that packet needs to be re-transmitted (and not the entire message). Therefore, when device 1 wishes to transfer a large file to device 2, the file is broken up into packets and transmitted packet by packet.

(v) ***Error Detection and Correction***

When device 1 sends a message packet to device 2, it performs a mathematical calculation (manipulation) on every unit of data transmitted. This calculation is transmitted to device 2 immediately after the message. Device 2 performs exactly the same mathematical calculation on its incoming data as device 1. Device 2 also reads in the calculation sent by device 1 and compares it with the local calculation. If the two calculations provide an identical result, then it is assumed that the incoming message was not corrupted on the link. Device 2 can then issue a positive acknowledgement to device 1 to indicate that it is ready for the next message. If the two calculations are inconsistent, then it is assumed that incoming data has been corrupted, and device 2 issues a "negative acknowledgement" message to device 1, which indicates that the previous data message must be re-transmitted.

(vi) ***Termination of Transmission***

Device 1 transmits a file, piece-wise, ensuring that each packet is correctly received by device 2, using the technique described in (v). After the last piece of the file is transmitted to device 2 and positively acknowledged, then device 1 must terminate the transmission. Device 1 sends an "end of transmission" message to device 2. This allows device 2 to close the stored file and return to other duties.

The various phases of communication for a file transfer, under this typical protocol, assuming no error conditions, are illustrated in Figure 3.2. If you can understand the inherent uncertainties with each phase, then you should begin to realise the number of things that can go wrong in the above process. For example, what should device 1 do if device 2 is switched off in the middle of file transfer? What should device 2 do if device 1 is switched off in the middle of data transfer? What happens if both devices wish to transfer files simultaneously - which device should have preference? And yet, if the rules of protocol are inadequate or if the software on any one device is inadequate, then one of the devices will misinterpret incoming information or just "lock-up".

Just how rigorously we define a communications protocol and the means of escape from error conditions, largely depends upon how important it is to achieve accurate information transfer and "lock-up" free software operation. If for example, device 1 is a PC and device 2 is a printer, then the protocol illustrated in Figure 3.2 is probably far too rigorous and sophisticated to be worthwhile. On the other hand, if device 1 is a PC and device 2 is a CNC machine, then we might say that the protocol described is a minimum requirement because of:

- the length of the communications link
- the link's susceptibility to electro-magnetic interference
- the importance of accurate data transfer.

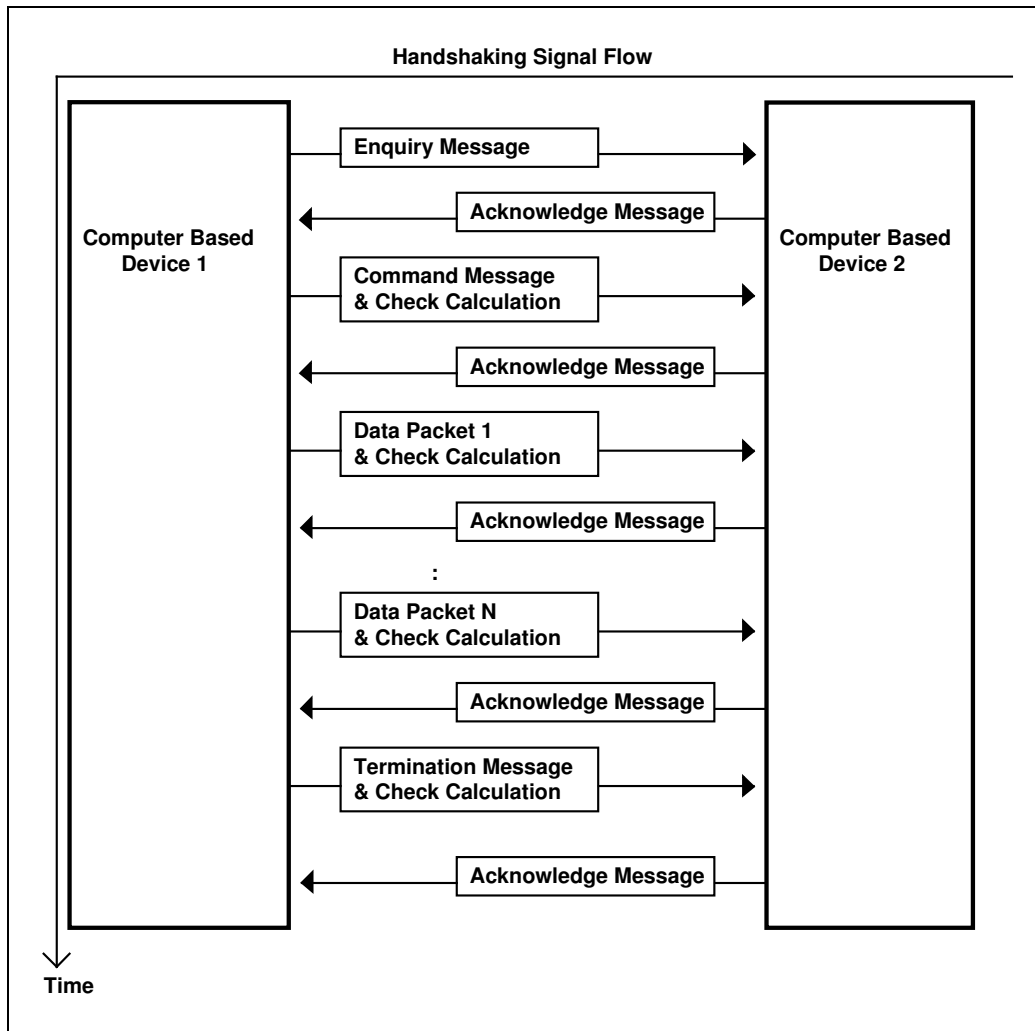


Figure 3.2 - File Transfer Sequence under Typical Protocol

We therefore choose simple, loose protocols for an electro-magnetically "clean" environment, where communication is short distance and errors are not crucial. This minimises the amount of work required in interconnecting intelligent devices. Conversely, the more important it is to minimise or eliminate errors and lock-ups, such as in a computer control environment, the more secure and rigorous the protocol must be. The penalty for data security is complexity of communications software and subsequently, the speed of the link itself.

3.3 Modelling Conducting Communications Links

In order to understand why it is a complex matter to make two computerised devices communicate with one another, it is first necessary to examine the physics of a conducting communication link. We often view a conducting cable (wire), between two computers, as an ideal element in an electrical circuit. This is shown in Figure 3.3. We assume that the wire has no resistance to the flow of current and that therefore, the signal emanating from device A is the same as the one reaching device B.

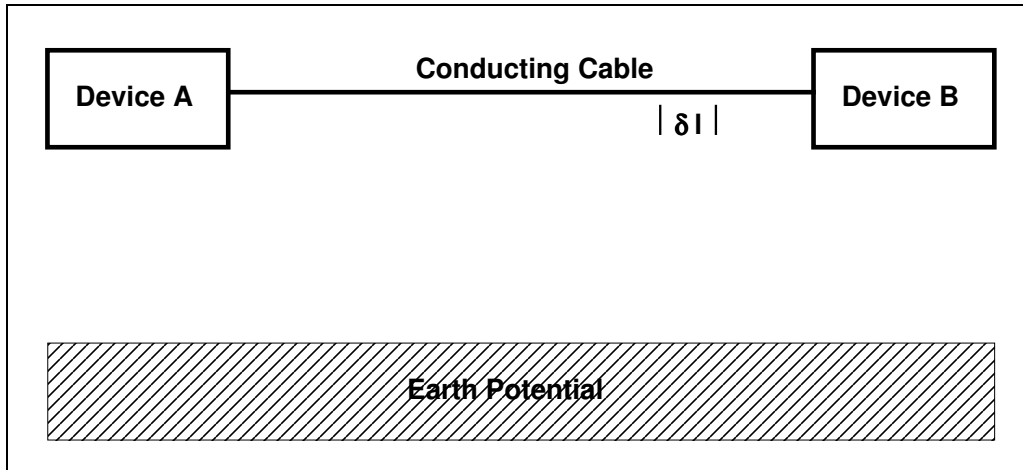


Figure 3.3 - A Single Wire Communications Link

The idealised model serves our purposes well for a number of levels of analyses. For example, when we are examining rules of protocol, as in section 3.2, we ignore the effects of imperfect conductors. However in order to understand why errors occur in communications links, we need to look at a more appropriate model of the conducting material. Firstly, we look at an infinitely small section of the wire (δl in length) and examine its physical properties.

There are no "lossless" conducting materials. All materials have some resistance to current flow, and energy is converted to heat within the conducting medium. So the circuit model for our infinitely small length of wire has a series resistor "R" to reflect the loss of energy at the receiving end of the conductor. Equally, the air between the conductor and earth is not a perfect insulator and therefore provides an alternative path in which current can flow through to earth. The conductance (inverse of resistance) of this alternate path to earth is "G" and reflects that current that does not appear at the receiving end of the conductor as a result of charge flow through the alternate path.

Since the conductor has current flowing through it, a magnetic field is produced around the conductor, and the resultant magnetic flux linkage of the infinitely small section of wire is represented by a series inductor "L". The conductor will also have a certain voltage (and net charge), with respect to earth, causing an electric field between the conductor and earth, thereby giving rise to a capacitance "C".

The series inductance and the shunt capacitance reflect energy storage and release within the line. Since both devices store and release energy at differing rates, the voltage at the receiving end of the infinitely small length of wire will not generally be in phase with that at the transmitting end. The complete transmission line is built up from these infinitely small sections and hence we could draw the circuit model for the transmission line as shown in Figure 3.4.

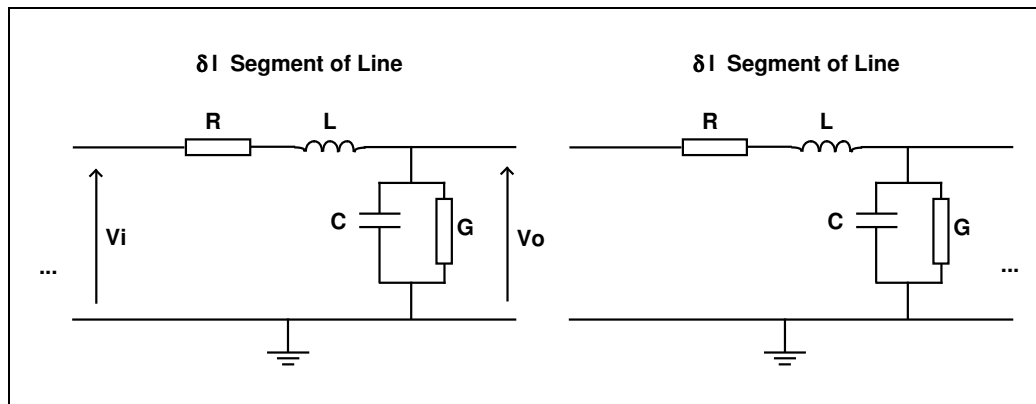


Figure 3.4 - Lumped Parameter Approximation of a Conducting Cable

This is referred to as a "lumped parameter" approximation of the transmission line because all the physical properties that, in reality, are distributed evenly along the line are represented by simple "lumped" circuit elements. Nevertheless, the approximate circuit provides us with some insight into what happens when digital signals pass through the transmission line.

A typical digital waveform that may pass through such a conducting cable, during both internal and external computer communications is shown in Figure 3.5.

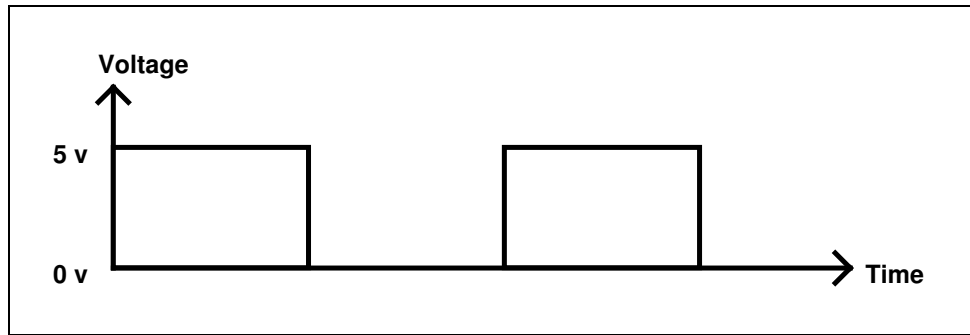


Figure 3.5 - Typical Digital Waveform

Mathematical analysis (Fourier Series) tells us that any waveform, regardless of its shape can be represented by the sum of a number of sinusoidal waveforms of differing frequency and amplitude. So looking at the digital waveform in Figure 3.5, we could say that the high-frequency sinusoidal components of the waveform, contribute to the sharp vertical edges, whilst low frequency sinusoidal components contribute to the overall shape and the horizontal edges.

For any of the individual sinusoidal components of the digital waveform, the ratio of output voltage (at the end of an infinitely short length of wire δl) over input voltage can be obtained using the traditional "phasor-method" of circuit analysis.

The impedance of the parallel branch is given by:

$$Z_p = \frac{1}{j2\pi fC + G} \quad \dots(1)$$

The impedance of the series branch is given by:

$$Z_s = j2\pi fL + R \quad \dots(2)$$

The ratio of output voltage to input voltage is then obtained through voltage division and is given by the expression:

$$\frac{V_o}{V_i} = \frac{Z_p}{Z_p + Z_s} \quad \dots(3)$$

Substituting equations (1) and (2) into (3) gives us the complex number expression:

$$\frac{V_o}{V_i} = \frac{1}{(j2\pi fL + R) + \left(\frac{1}{j2\pi fC + G}\right)} \quad \dots(4)$$

The magnitude ratio of output voltage over input voltage is given by:

$$\left| \frac{V_o}{V_i} \right| = \frac{1}{\sqrt{(2\pi fRC + 2\pi fLG)^2 + (1 + RG - 4\pi^2 f^2 LC)^2}} \quad \dots(5)$$

The phase difference of the output voltage with respect to the input voltage is given by:

$$\Phi_{V_o-V_i} = -\text{Arc tan} \frac{2\pi fRC + 2\pi fLG}{1 + RG - 4\pi^2 f^2 LC} \quad \dots(6)$$

Both expressions (5) and (6) are frequency dependent. Therefore we can say that the phase and magnitude-ratio of output voltage to input voltage will be different for each sinusoidal component of the digital waveform. Substituting some "limit" values into these expressions, we can observe that in the infinitely small section of line:

- very high frequency ("f" tending to infinity) sinusoidal components will be attenuated (diminished) to zero
- low frequency ("f" tending to zero) sinusoidal components will be attenuated by a factor of (1 + GR)

- low frequency ("f" tending to zero) sinusoidal components will be slightly "shifted" in phase with respect to the input.

If we were to exaggerate this effect, then the output voltage at the end of the infinitely small section would look as the voltage waveform illustrated in Figure 3.6.

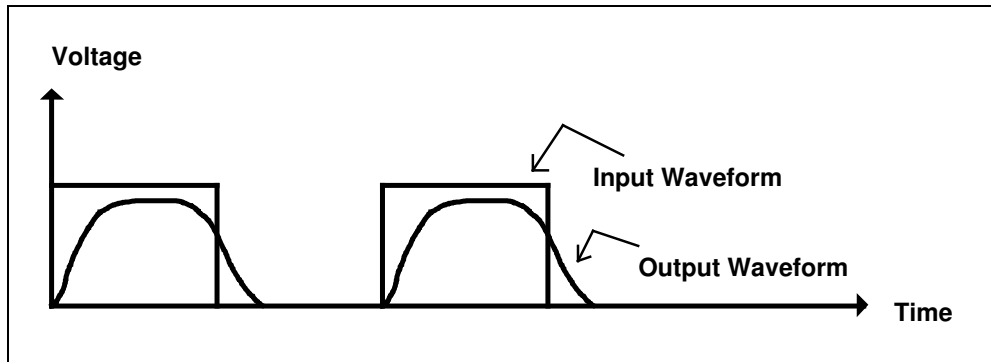


Figure 3.6 - Exaggerated Output Voltage at the end of Δ Segment

The attenuation of high frequency sinusoidal components, in the output waveform, means that the edges lose their sharpness. The phase shift in low frequency components, means that the output waveform appears to "lag" behind the input waveform. As a result of attenuation in some of the sinusoidal components, the output waveform also appears attenuated in some areas.

Since a conducting transmission line is composed entirely of these infinitely small sections, the distortion and attenuation of the output voltages, with respect to the input voltages, is increased with length. If the length of the transmission line is sufficiently large, then the output waveform will be attenuated and distorted to such an extent that it cannot be discerned from noise. This is shown schematically in Figure 3.7.

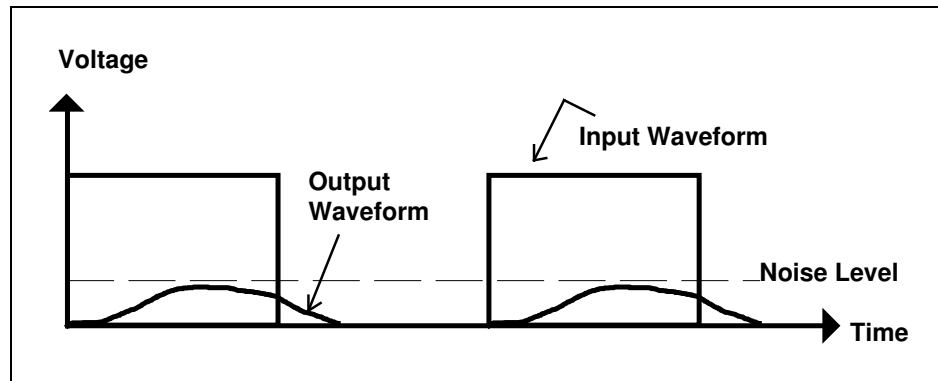


Figure 3.7 - Degenerative Effects of Long Transmission Lines

The lumped parameter circuit is not in itself adequate to describe the complex physical phenomena that take place along a conducting transmission line, particularly at high data transmission frequencies. In reality, accurate modelling of a conducting transmission line requires consideration of a number of other factors. These include the phenomenon referred to as "skin effect". As the frequency of the voltage waveform on the transmission line increases from 0, current flow tends to occur more predominantly on the outside of the conductor, rather than uniformly through the cross-section. At very high frequencies (Megahertz and Gigahertz), current flow occurs on the surface of the conductor, thereby giving the cable an apparently higher "resistance".

Another phenomenon that occurs along the transmission line at very high frequency transmission rates is that of "travelling waves" of voltage. Since conducting transmission lines are not ideal, it takes a finite time for the voltage at one end of the line to "ramp up" to the value at the other end. If high frequency transmission lines are not terminated with an appropriate impedance (matching impedance), then voltage waveforms travelling down the line can be reflected from the target end, back to the source, thereby causing signal distortion.

The net result of all these phenomena is that we cannot assume that the signals at the end of a conducting transmission line are the same as those at the source. The longer we make a transmission line, and the higher the frequency at which we transmit digital data, the more difficult it is to ensure that data integrity is maintained.

As a corollary of the above, there are a number of techniques that we can employ, to ensure that data integrity is secured on the transmission link. In simple links, these measures may include the restriction of data transmission frequencies and line lengths to levels that are not deleterious to transmission accuracy. In more complex links, signals need to be corrected or physically altered (amplified or modulated) prior to transmission to ensure that they are not degraded by the link.

3.4 Electro-Magnetic Interference and Cross-Talk

The signal at the output end of a conducting, data transmission cable is not only distorted because of the cable itself. Output signals can also be affected by magnetic fields that induce additional voltages in the cable. These induced voltages can lead to data errors at the receiving (output) end of the transmission line.

In order to understand this phenomenon, we examine the simple case of two infinitely long, parallel conductors (each carrying a current that varies with time). These are shown in Figure 3.8.

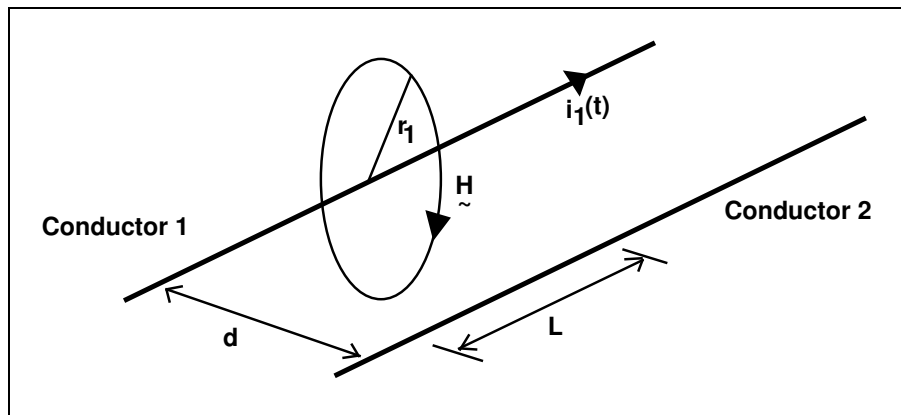


Figure 3.8 - Two Parallel, Current-Carrying Conductors

Ampere's Law states that the magnetic field intensity generated around conductor 1, as a result of the current flowing through it, is given by the closed-line-integral expression:

$$\oint_{\sim \sim} H \cdot dl = N i_1(t)$$

...(7)

where N is the number of "turns" in conductor 1 (clearly equal to 1) and $i_1(t)$ is the current flowing in that conductor.

As a function of radial distance (r_1) from conductor 1, the magnitude of the magnetic field intensity becomes:

$$H = \frac{i_1}{2\pi r_1} \quad \dots(8)$$

The magnitude of the magnetic flux-density as a function of radial distance from the first conductor is then given by:

$$B = \frac{\mu i_1(t)}{2\pi r_1} \quad \dots(9)$$

where μ is the "permeance" of the medium between the two conductors. The magnetic flux over a length "L" of conductor, passing through the distance "d", separating the two parallel conductors, is given by:

$$\Phi = \int_{\sim \sim} B \cdot ds \quad \dots(10)$$

$$\Phi = \frac{\mu i_1(t)}{2\pi d} \cdot d \cdot L \quad \dots(11)$$

Therefore the flux, per unit length, which passes through (links) the second (parallel) conductor is given by:

$$\frac{\Phi}{L} = \frac{\mu i_1(t)}{2\pi} \quad \dots(12)$$

Faraday's Law, tells us that the voltage induced in the second parallel conductor, as a result of the flux linked from the first, is proportional to the rate of change of the flux linkage:

$$V \propto \frac{d\Phi}{dt} \propto \frac{di_1(t)}{dt} \quad \dots(13)$$

Hence we can see that a time-variant current, passing through one conductor is capable of generating (inducing) a voltage in the second conductor. This voltage induction occurs whenever a changing magnetic field (regardless of its source) passes through the conductor. If the magnetic field is generated by the flow of electrical current (as above), then the induced, "unwanted" voltages in conductors are specifically referred to as "Electro-Magnetic Interference" or "EMI".

The above example, which shows long, straight and parallel conductors is a special case, lending itself to simple field calculations as shown in equations (7) to (13). It highlights the effects of electro-magnetic voltage induction. The magnetic field distributions of shorter, non-parallel conductors can be far more complex to deal with mathematically, but the principles are similar. The two key points to note are that:

- (i) Current flow in a near-by conductor can induce unwanted signals in a data carrying conductor
- (ii) The larger the magnitude of the current flow in a near-by conductor and the larger its rate of change with time, the larger the induced voltage in the data conductor.

It is therefore important to watch for high power conductors, in the vicinity of data communications cables, particularly where currents are being "switched". In an industrial environment this is a situation that commonly arises when high-powered electric motors (especially a.c. induction motors) are either switched on and off regularly, or else have widely changing mechanical loads.

Induced voltages in data conductors are not always the result of high-current cables in the vicinity. Sometimes, two low-current, parallel data conductors, in close proximity, can induce voltages within one another. This situation is referred to as "cross-talk" because data in one conductor is cross-linked into a parallel conductor.

The actual magnetic-field (flux) distribution in one conductor as the result of a current in another is dependent upon the relative geometry of the conductors. The "cross-talk" effect is most pronounced, with the field distribution arising from two, long, parallel conductors. Cross-talk can therefore be reduced by altering the field distribution between two data carrying transmission lines. This is in turn achieved through changing the geometry of the conductors, by simply "twisting" pairs of insulated cables around one another. The "twisted-pair" cable arrangement is shown in Figure 3.9.

The susceptibility of data-carrying conductors to external magnetic (and electro-magnetic) fields is minimised by wrapping insulated conductors in a conducting cage or "shield". The shield can be a foil wrapping or a braided wire cage (or both). Commercially available data transmission cables normally have a number of insulated, twisted-pairs shielded with a braided cage and all wrapped in a plastic insulating sleeve. Cable manufacturers provide a range of options so that end users can select the number of conducting wire pairs to suit their needs.

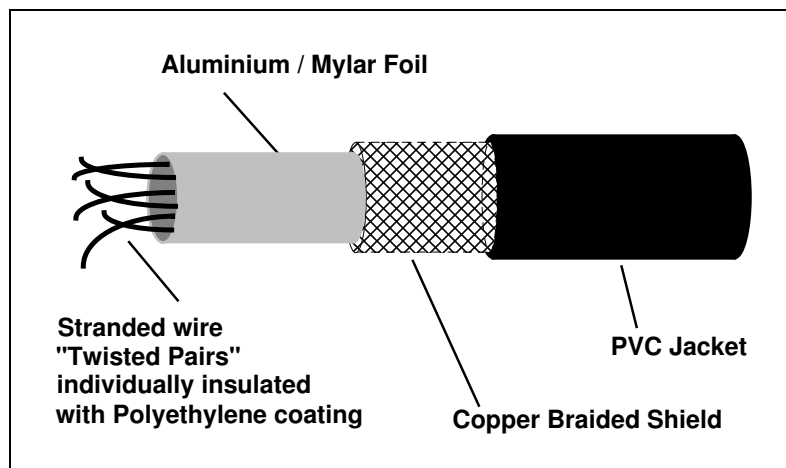


Figure 3.9 - Twisted-Pair Cable Composition

3.5 Parallel Data Transmission and Communications Ports

Data within a computer system is transferred in a "bit-parallel" manner. This means that all the binary digits (which together represent a basic unit of computer information) are essentially transmitted at the same time and received at the same time. If the basic internal unit of computer information is say, an 8-bit byte, then at least 8 conductors are required to link the two devices for "bit-parallel" operation.

The objective of data-communication is to transfer binary information from the data bus of one computer to the data bus of another computer, from which it can then be directed to any other internal location (such as memory, disk, etc.). At a first glance, therefore, it may appear sensible to extend the concept of "bit-parallel" communications beyond the boundaries of a single system, to enable "computer to computer" communications to occur. This concept is shown schematically in Figure 3.10.

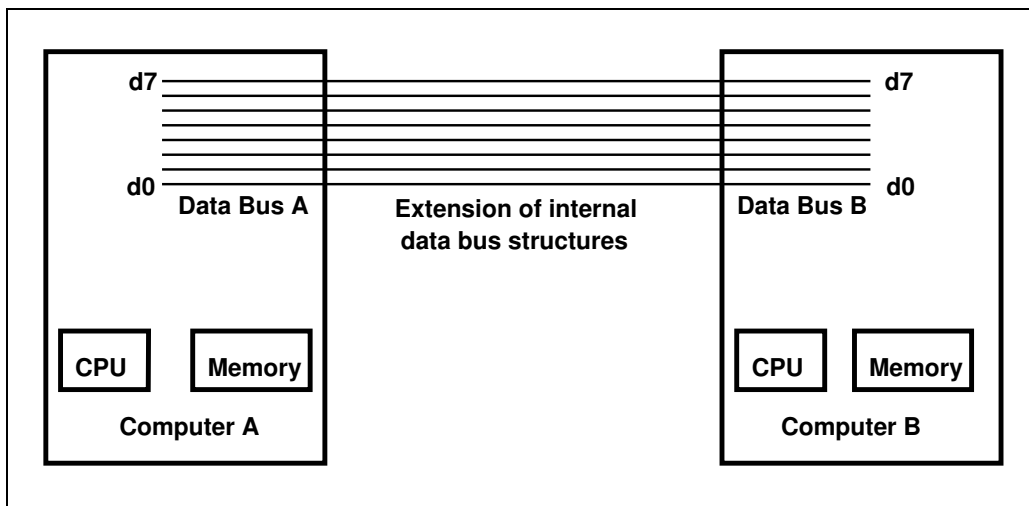


Figure 3.10 - Expanding "Bit-Parallel" Communications

It is actually feasible to enhance the simplistic link shown in Figure 3.10 in order to establish a realistic link between devices. This is referred to as a parallel "point to point" link.

The problems with implementing a simplistic parallel link, such as that shown in Figure 3.10, are numerous. Firstly there is the problem of physical incompatibility. Computer "A" may use one set of voltages to represent binary digits, whilst computer "B" may use a completely different set of voltages. Secondly, the data bus of computer "B" may be of a different size to the data bus of computer "A".

Finally, even if we could link the two devices in this way, we have a situation where two intelligent devices (CPUs) may both attempt to act as masters over the use of the data bus. For example, a contention situation could arise where device "A" sets a line low, while device "B" tries to set the same line high, thereby causing a temporary short-circuit.

Some of these problems are overcome by providing an electronic interface or "buffer" between each device and the external communication link. This is shown in Figure 3.11. In order for the two devices to then communicate, the interface on each device must perform a two-way conversion between the internal data bus signals and the common external representation.

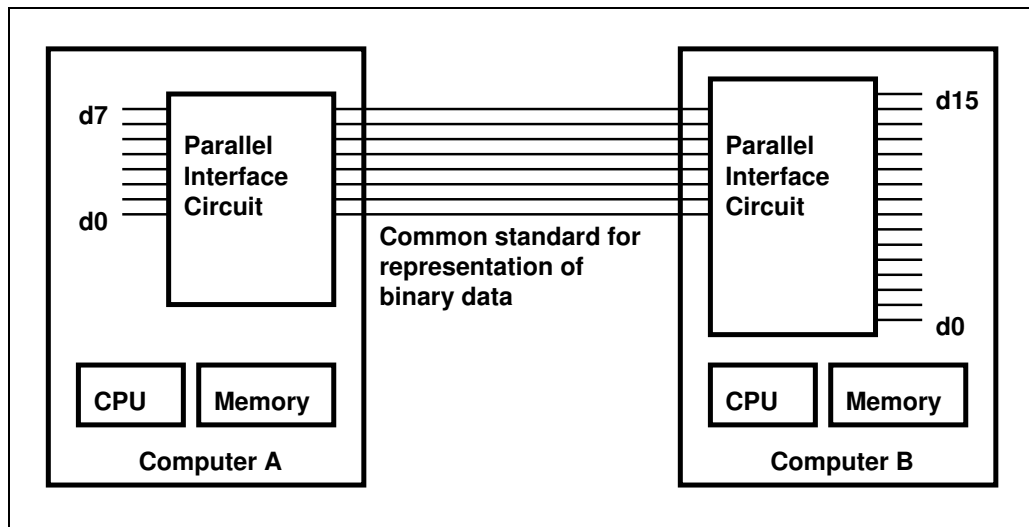


Figure 3.11 - Electronic Buffering of Parallel Communications

The circuitry in such a buffer needs to be designed with a view to withstanding possible short-circuits because the interfaces are normally not capable of resolving contentions for use of the external communication medium.

When we connect two devices for parallel communication, they are generally not of the same intelligence level. For example, device "A" may be a Personal Computer, whilst device "B" is a Printer (a dedicated, low-level computer). It may therefore be necessary to resolve contentions on such a link by providing additional lines on the interface circuits for external "hand-shaking" purposes. This is shown in Figure 3.12.

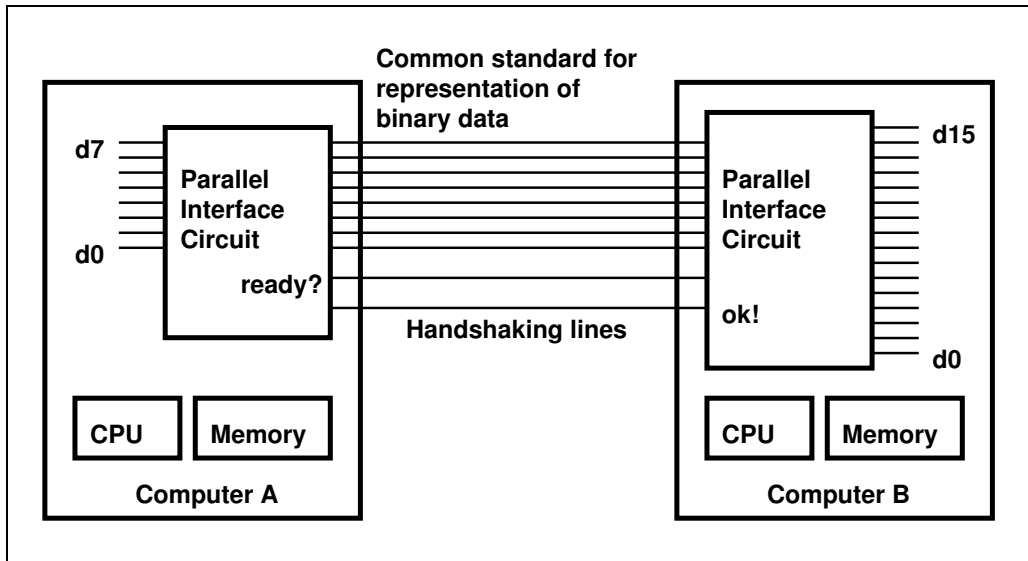


Figure 3.12 - Hand shaking Lines in Parallel Communications

Hand shaking lines on communication links are used in order to synchronise and/or co-ordinate the flow of data between two intelligent devices. They are sometimes referred to as "hardware protocols" because they are designed to achieve (using hard-wired links) the same ends as software protocols. Hardware protocols are not unique to external communications. Within any individual computer system, the address bus structure is the defacto hardware hand-shaking system that effectively controls the flow of data from the CPU to memory chips and vice-versa.

Figure 3.12 shows the most common form of hardware hand-shaking between intelligent devices. If we continue with the scenario where device "A" is a Personal Computer and device "B" is a printer, then the "ready?" and "ok" lines serve the same function as the "Enquiry / Acknowledge" sequence in a software protocol. Device "A" may assert the "ready?" line to a "true" (enable) state, and if device "B" is ready (on-line), then it should respond by asserting the "ok" line to a "true" (enable) state. This would allow people to develop software on device "A" that takes into account the possibility of device "B" being inoperative or disconnected.

Additional "control lines" are often used in communication links for the coordination of data flow and device to device signalling. A good example is a parallel link between a computer and a printer. The parallel link contains hand-shaking and control lines that pertain to the status of the printer. An "out of paper" line is a common hand-shaking line in such links.

Having established how simple parallel links can be made, we are again confronted with the problem of standards for the links between devices. A number of conformance issues are immediately apparent with respect to the parallel link:

- the number of data lines to be used
- the voltage representation of binary data
- the representation of characters
- the number and role of hand-shaking lines.

These issues are addressed by a number of common specifications and standards for parallel links.

3.6 The Centronics Parallel Port

A number of specifications and so-called "standards" exist to define the parallel communication process between intelligent devices. As with so many other aspects of computing, there are no universal standards or ubiquitous solutions.

A parallel communications specification that has become very widely accepted for computer to printer links is called the "Centronics" Parallel Interface. Centronics is a registered trademark. Its specification has been widely accepted by printer manufacturers, primarily because it is the common parallel interface on IBM (and compatible) Personal Computers. The Centronics connector plug and pin allocation are shown in Figure 3.13.

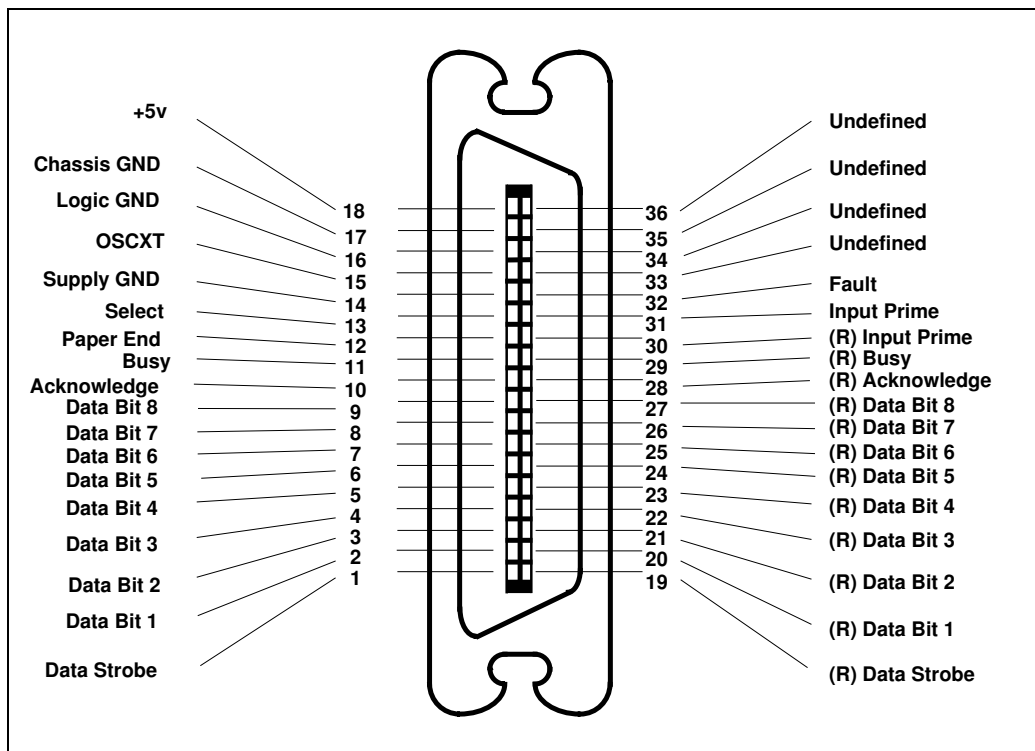


Figure 3.13 - Centronics Parallel Interface Connector

The Centronics parallel interface is a 36 line link. Eight of these lines (pins 2 - 9) are used to carry data. Pins 19 to 30, labelled with an "(R)", are signal ground return lines for the corresponding data lines. Binary data in the Centronics parallel link is represented with standard Transistor to Transistor Logic (TTL) voltage levels. These are shown in Table 3.1. Note that under this representation, there is an allowable error margin of 0.4 volts between the output from TTL circuits and the input to TTL circuits.

<i>Quantity</i>	<i>Item</i>	<i>Voltage Range</i>
Binary 0	Inputs to Circuits	0.0 → 0.8 V
	Outputs from Circuits	0.0 → 0.4 V
Binary 1	Inputs to Circuits	2.0 → 5.0 V
	Outputs from Circuits	2.4 → 5.0 V

Table 3.1 - TTL Voltage Levels for Binary Representation

Data flow on the Centronics parallel link can be governed by the hand-shaking lines that are provided. You should particularly note that these hand-shaking lines have been established under the assumption that the device on one end of the link will be a computer and the device on the other end will be a printer.

Pin (line) 1 on the Centronics link is a "strobe" signal that is supplied by the computer to "clock" data out through the parallel data lines. The receiving device (printer) examines the 8 bits of parallel data, from the computer, only on the negative going edge of the strobe pulse waveform. This gives the receiving device a timing mechanism for differentiating between successive data bits on each line. The negative-edge triggering technique is shown in Figure 3.14. This illustrates the receiving device interpreting bit number 3 of incoming data as a "0" at t_1 and as a "1" at t_2 .

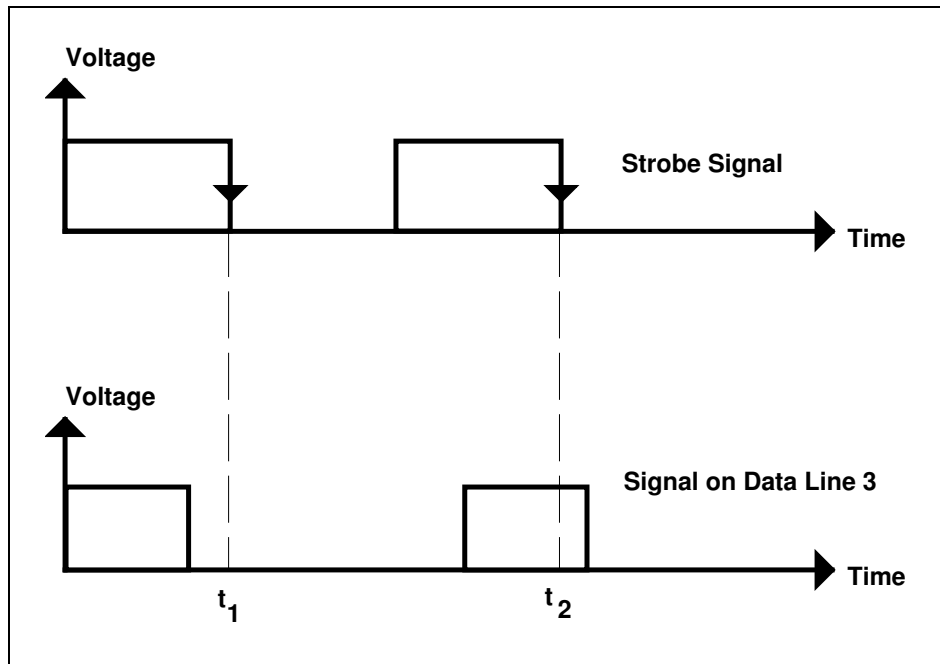


Figure 3.14 - Negative Edge Triggering Using a Strobe Signal

Pin (line) numbers 10 (ACKNLG) and 11 (BUSY) are two hand-shaking lines that co-ordinate the flow of data. If for example, the printer is not ready to receive data, then it sets pin 11 high - otherwise it is kept low and the computer is free to transmit. If the printer resets pin 10 to low, then it is acknowledging receipt of the last 8 bits of parallel data and indicating that it is ready to receive more - otherwise the printer is not ready to accept data. Table 3.2 lists the various Centronics line functions and their direction of information flow. Note however that the functions of lines 12, 13, 14, 15, 31, 32, 34, 35 and 36 vary according to individual applications, and they are often used to convey additional printer status information (such as out of paper, etc.).

The Centronics Parallel link is designed for fast, one way data flow, from a master device (computer) to what is generally a slave device (printer). It does not readily lend itself to an environment, where two, equally intelligent computer devices can both talk and listen to each other at the same time.

<i>Line or Pin</i>	<i>Corresponding Return Line</i>	<i>Signal</i>	<i>Direction</i>
1	19	Strobe	From Computer
2	20	Data Bit 1	From Computer
3	21	Data Bit 2	From Computer
4	22	Data Bit 3	From Computer
5	23	Data Bit 4	From Computer
6	24	Data Bit 5	From Computer
7	25	Data Bit 6	From Computer
8	26	Data Bit 7	From Computer
9	27	Data Bit 8	From Computer
10	28	Acknowledge	From Printer
11	29	Busy	From Printer
12		Paper End	Application Dependent
13		Select	Application Dependent
14		Supply Ground	Application Dependent
15		Oscxt	Application Dependent
16		Logic Ground	
17		Chassis Ground	
18		+5 Volt Rail	
31	30	Input Prime	Application Dependent
32		Fault	Application Dependent
33		Undefined	
34		Undefined	
35		Undefined	
36		Undefined	

Table 3.2 - Centronics Pin Configuration and Common Data Flow

The Centronics link is primarily intended as a point to point system, with only one device at either end. However, its capabilities can be extended through commonly available "Parallel Exchange Network Units". These enable a single computer to feed a number of Centronics compatible devices as shown in Figure 3.15. Alternatively, a number of computers can use the exchange to share high cost printers and other peripherals. This is referred to as "resource sharing".

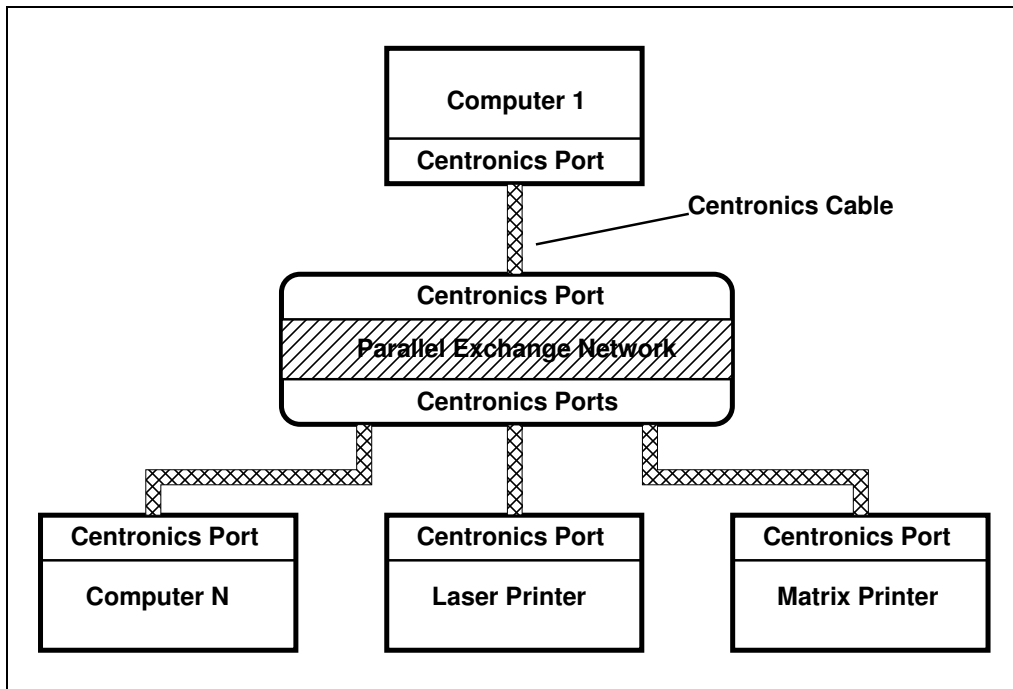


Figure 3.15 - Connecting Multiple Devices Through Centronics

The parallel exchange network unit is simply a switching (multiplexing) device, which can make a direct, point to point link between any two devices that are connected to it. The operation, sophistication and number of ports on these exchange units vary markedly between vendors but in relative terms the devices are all inexpensive. This is therefore a common and simple technique for sharing equipment through a relatively simple Centronics "Star" Network.

A common resource-sharing implementation might be as shown in Figure 3.15. This arrangement would enable either of the two computers to use either of the two printers. The majority of exchange units use "software switching" to facilitate this. This means that in order for one device to access another, through the exchange, the originator first sends a simple command string to the exchange. The command string tells the exchange which device is to receive information from the originator. The exchange then makes the physical "point to point" connection and thereafter becomes transparent. The originator (computer say) can then talk to the receiver (printer) as if the exchange was not present. The receiver (printer) is therefore never aware that it is connected to the exchange and functions as normal.

It is possible in such a system that a "contention" will arise when both computers wish to use the same printer at the same time. The more sophisticated exchange units can resolve these contentions and "queue" requests for connection in their own built-in memory buffers.

3.7 Networked Parallel Data Transmission and IEEE-488

In a scientific or engineering environment it is often necessary for a number of "intelligent" devices to be linked to one another so that data can be transferred and shared. For example, in a metrology laboratory, a Personal Computer may need to be linked to a number of microprocessor controlled data-acquisition devices so that it can process data and issue control signals. A simple point to point link is clearly inadequate for this purpose. There needs to be some form of "network" by which data interchange can occur.

Within an individual computer system, a parallel network exists in the form of the data and address bus. A Central Processing Unit (CPU) can readily communicate with other chips by selectively setting address bus lines high and low. Each and every memory location in such a system is activated by a unique pattern of high and low bits on the address bus. This is referred to as "addressing". Each chip device in the system has a unique range of addresses allocated to it. Each memory location or register in each chip has a unique address within the range allocated to the chip itself. All the chips share the same conductors (bus) for data transfer and contentions for use of the conductors inevitably arise. These are resolved either through the CPUs "masterly" use of the address bus or through special "bus controller" chips.

The internal parallel data transfer network can be expanded for external use, provided that the following parameters are standardised:

- the physical representation of data
- the size of the external data bus
- contention resolution (for media usage)
- device addressing.

The most common specifications for parallel data communication, in network form, are those referred to as the IEEE-488 "Instrumentation Bus" or Hewlett Packard Instrumentation Bus (HPIB) or General Purpose Instrumentation Bus (GPIB). These are all generic or proprietary terms for essentially the same specification. The instrumentation bus is very widely used on scientific, metrological and general laboratory equipment for short distance communications.

The GPIB specifications help us to define:

- the number of data lines to be used
- the shapes and pin configurations of connecting plugs
- the values of voltage required to represent binary digits
- device addressing
- the hand-shaking lines
- the control lines.

The concept of the GPIB is not unlike that of the Centronics parallel system, except that it is far more flexible (in terms of the devices that can be connected to it) and incorporates a number of features that make it amenable for use as a network as well as a point to point link. Connecting plugs in the GPIB system have a "plug through" facility so that by plugging one connector into the back of another, a daisy-chained, parallel network can be generated. This concept is illustrated in Figure 3.16.

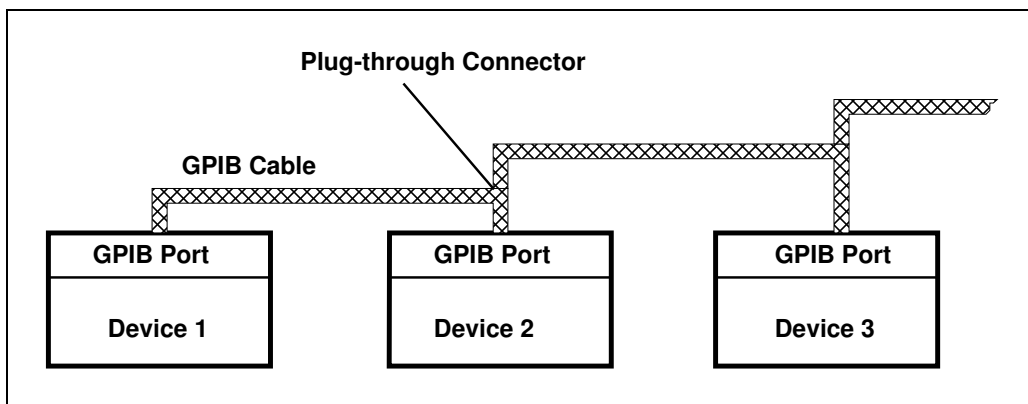


Figure 3.16 - Daisy-chaining to Form a Parallel Network

The GPIB system is intended as general purpose, network in which data-flow can be bi-directional, unlike the uni-directional, computer to slave (printer) relationship in the Centronics link. The IEEE-488 (GPIB) connector plug and pin allocation are shown in Figure 3.17.

The GPIB connector is a 24 pin device with 16 active lines. Each device in the GPIB system must be identifiable through a unique address to avoid bus conflicts. Each device in the system must be aware of its own address in order for the GPIB to function correctly.

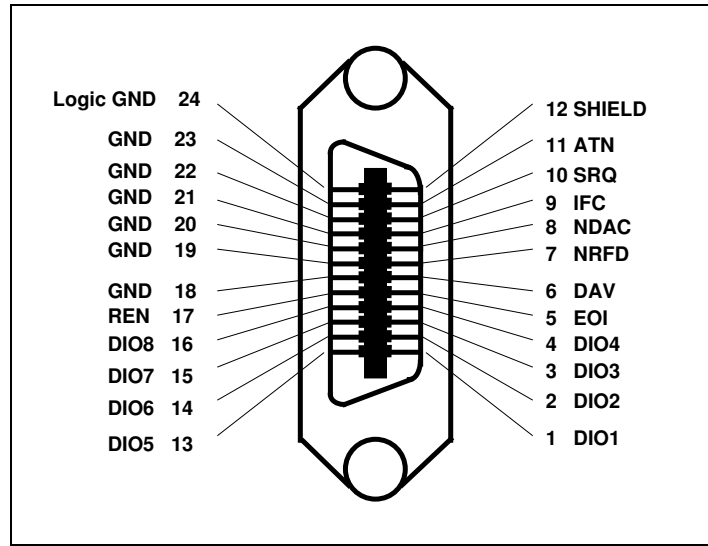


Figure 3.17 - IEEE-488 Connector and Pin Configuration

Pins 1-4 and 13-16 in the GPIB are used for data lines. The data on these lines may either represent unrelated binary information or ASCII character types. The data bus lines (DIO1- DIO8) allow the transfer of data, control words and device addresses. Note however, that in contrast to the Centronics representation of binary information, a binary "0" is represented by +5 Volts d.c. and a binary "1" is represented by a voltage of 0 Volts.

Within the IEEE parallel network system, there are essentially three types of devices that can be connected:

(i) Controllers

The controller is the device that is capable of getting other devices to accept commands. It does this by asserting the "Attention" (ATN) line (pin 11). Only one controller is permitted to exist on the parallel data bus at any one time.

(ii) Talkers

A talker is a device that is configured to transmit data on the data bus to other devices. Normally, only one talker is permitted to transmit on the data bus at any one time.

(iii) Listeners

Listeners are devices that read in data from the data bus, utilising a pre-defined hand-shaking sequence. More than one listener can exist and be active on the bus at any one time.

Hand shaking lines work on a similar principle to those in the Centronics system except that a number of devices may assert or monitor the lines. In the GPIB there are three hand-shaking lines, two of which can be asserted by listeners and the third by a controller.

The "Data Valid" (DAV) line (6) is one that is asserted by the controller to indicate that it has placed a control byte on the data bus. The "No Data Accepted" (NDAC) line (8) is one that is asserted by a listener on the GPIB to indicate that it has not yet accepted the last byte that was placed on the bus. The "Not Ready for Data" (NRFD) line (7) is used by an active listener to prevent new data or control bytes to be placed on the bus. Talkers all monitor the NRFD line and wait until it is de-asserted before sending the next 8 bits of information.

The control lines are similar to hand-shaking lines except that they are used to define the way in which binary information on the data lines is to be interpreted. The "Interface Clear" (IFC) line is used by a controller on the GPIB to place the entire interface system, and all its connected devices, into a defined "quiescent" state.

The "EOI" line is used by talkers to indicate the end of a multiple byte, data transfer message. In this mode it acts as an "end of data" indicator flag. A GPIB controller also asserts the "End or Identify" (EOI) line in conjunction with the "Attention" (ATN) line, when conducting a "poll" on the status of other devices in the system.

The "Service Request" (SRQ) line is asserted by a GPIB device to indicate to other devices the need for some specific service program. The "Remote Enable" (REN) is asserted by a controller to force a listening device to ignore its "front panel" controls.

The IEEE-488 network is a relatively fast and efficient means of transferring data between devices over short distances. This makes it particularly suitable for the laboratory environment, which is electro-magnetically "clean". As with all parallel data networks and links, the ability to transfer 8 bits of data simultaneously from source to destination is a major factor in performance.

A number of computer and scientific equipment vendors, including Hewlett Packard, have used the IEEE-488 system as the back-bone system for interconnecting devices over limited distances. Some of these equipment manufacturers take advantage of the parallel system performance and use the GPIB for functions such as memory to disk-drive transfers and so on. The "generalised" structure and specification of the IEEE-488 system make it an extremely useful tool for many short distance applications.