

---

## ***Chapter 4***

### ***Serial Data Communications - Fundamentals***

#### ***A Summary...***

An examination of problems associated with using parallel links over long distances. Line-drivers and modulation techniques. Converting parallel data into serial formats. The UART and USRT. Synchronous and Asynchronous serial communications links.

#### ***Read This Chapter If...***

- ◆ You need to learn the problems of parallel communication
- ◆ You need to know why and where serial communications is used
- ◆ You need to understand about line-drivers and amplification
- ◆ You don't understand the difference between synchronous and asynchronous communications
- ◆ You need the concepts of modulation explained.

## 4.1 Serial Communications in Integrated Manufacturing

If it had not been for the advent of CNC machines, PLCs and robots, then many manufacturers could have lived their lives in complete ignorance of the problems associated with the lack of standardisation in data communications. Few people could have imagined just how difficult it would be to make a range of vastly differing industrial controllers communicate with one another. Indeed, when some of these devices were originally designed (notably CNC), there was little understanding of why industrial controllers should communicate at all and even less understanding of what they would need to say if they could communicate.

In the manufacturing office, a simple phone-call to a computer vendor would have been more than adequate to secure the installation, management and maintenance of a network for the computerised office. Manufacturers could have all been thankful to the great god of production problem resolution - volume. For no matter how badly designed a computer system and no matter how incompatible one form of computer architecture is with another, if there are enough of them in the market-place, an effective solution to every problem is never far away.

The majority of computers in the office fall into a relatively small number of vendor stables and architectures. And there are invariably an enormous number of office machines on the market with the same (or similar) architectures. Even when "Brand X" systems are not compatible with "Brand Y" systems the manufacturer of "Brand X" computers will try to break into the "Brand Y" market by providing a communication link (and a piece of interfacing software) that will facilitate compatibility.

The provision of links between office computer systems of differing architectures is often financially attractive on a volume basis alone. Over and above this however, we have a bonus situation where office computers are generally of the same vintage. The likelihood of finding computerised office equipment older than 10 years is relatively small. So much so that we could say the likelihood of finding an office computer of 20 years of age would be almost nil.

The manufacturing environment, on the other hand, is very different to that of the office. With the exception of "off-the-shelf" CNC machines, production equipment is generally "tailor-made" for specific applications. Control systems (although increasingly modular) are usually tailored for specific applications. Even "off-the-shelf" CNC mills and lathes are low volume entities in comparison with office computers and printers. To make matters worse, these low volume CNCs are produced by a large number of vendors, each with their own individual views on controller architecture.

Production equipment often has a far longer life-span than normal computer equipment. A significant proportion of production machines are still extremely efficient even after forty years of service. Within a manufacturing organisation we therefore contend with a wide range of control equipment and an equally wide range of architectures, ranging from state-of-the-art microprocessor control to hard-wired, relay-ladder-logic control.

The low volumes and large varieties of manufacturing control equipment make it extremely difficult (and unprofitable) for organisations to offer the same range of communications solutions that are available to the office world. Professionals within the manufacturing environment therefore do not yet have the luxury of selecting "communications solutions" from an electronics supermarket.

It is normally accepted that, for manufacturing management purposes, there is merit in integrating production equipment through communications links. This should, in theory, enable real-time monitoring of production data and allow for more rapid changes to production schedules and part designs. It is also accepted that effective data links between Computer Aided Design (CAD) systems and production machines are now vital to minimise human errors. With these goals in mind, moves have long been afoot to try to rationalise (standardise) communications within manufacturing. However, (at the time of publishing) this standardisation has still not been realised.

There is some hope that in the long term, communicating with a machine or controller, from a host computer, will be as simple as making a plug-in connection. However it must be realised that even if universal communications standards for manufacturing are adopted immediately, there are still many short term problems to be overcome.

A significant proportion of machines and production systems, with controllers built during the 1970s and 1980s, will still be in service for many years. For the many and varied PLC and CNC controllers in such systems, the prospect of plug-in compatibility with sophisticated communications networks is virtually non-existent. However, if these systems are to be integrated with other computers (in order to make manufacturing management software systems more efficient and direct links from CAD possible) then the traditional mechanisms for communication (and their limitations) must be clearly understood.

The only mechanisms for communication between many of the industrial control systems, developed during the 1970s and 1980s, and host computers are the so-called "point to point" serial links. Regardless of how these links are viewed, relative to networks, it must be accepted that they will play a major role in manufacturing for many years. Although these links are conceptually simple, they tend to be a major cause of problems and time wastage to manufacturing organisations. Each serial link between an industrial control system and a host computer represents a unique problem. Each link requires a unique hardware and software solution.

As a result of non-standardisation, point to point, serial communications links are amongst the most misunderstood areas in the manufacturing world. Many professionals within industry do not appreciate the considerable (and time consuming) problems involved in achieving reliable data transfer on these links. A sound understanding of the principles of point to point serial communication is therefore vital in order to reduce development times and costs.

## 4.2 The Role of Parallel Communications in Manufacturing

Following a discussion of the role of serial communications in manufacturing, one may ask about the prospect of parallel communications, for transfer of information to and from production controllers. In practice, parallel communication is rarely used outside a laboratory or metrology environment, where the distance between devices is less than (say) 15 metres.

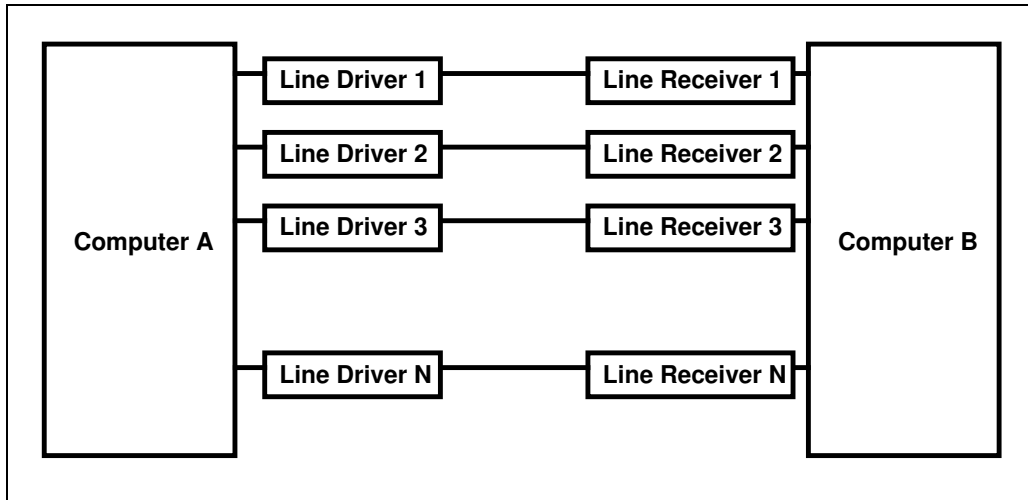
Even within a small manufacturing organisation, the distance between a host computer and an industrial controller is likely to be in the order of hundreds of metres and parallel communication techniques cannot readily be transferred to communications over longer distances. One fundamental reason for this is because of the voltage attenuation and distortion that occurs through a conductor. A simple rule of thumb would be to say that normal digital (TTL) signals cannot reliably be interpreted at the receiving end of a low grade conductor once its length exceeds several metres. There are two possible solutions to the problem:

- (i) Use a higher grade of conductor
- (ii) Amplify the signal before transmission.

The first solution only provides a marginal increase in the maximum transmission distance and does not, in itself, provide a solution for long-distance communication. The second solution requires an "interface" between the transmission line and the computer.

Devices that modify a signal at the transmission end of a link, to enhance reception, are referred to as "line-drivers". The complementary devices, which convert line signals back to the required internal form for the target device are referred to as line-receivers. Sometimes the line-drivers and line-receivers are internal to the computer system. Signal amplification is a good example of "line-driving" and is a way of overcoming the degenerative (attenuation) effects of a transmission line. However, in order to implement this solution in a parallel system, a number of line-drivers, as shown in Figure 4.1, would need to be used.

A parallel link may have more than 15 signal-carrying lines, each of which would have to be "driven" in order to increase the span of the link to more than a few metres. Although it would be feasible to provide these drivers, there is clearly an increased cost element involved.



*Figure 4.1 - Extending the Range of Parallel Communications*

In a medium to large scale organisation, the length of a link between a CAD system and a CNC machine could be in the order of a kilometre (taking into account cable deviations around obstacles and so on). Leaving aside the issue of "line-drivers" we need to look at the more obvious question of cable cost and utilisation. At any instant in time on a parallel communication link, each conductor carries only one binary digit of information. Consequently in the extended ASCII or EBCDIC representations of characters, 8 conductors (each of a kilometre in length) would be required to transmit each character. If you add to this the number of hand-shaking and control lines in a given link then you will appreciate just how much cabling is involved. Consider then the feasibility of linking computers on a nationwide or global basis - parallel techniques start to look very unattractive.

In a realistic system, it is often necessary to have the capacity for two intelligent devices to talk to one another simultaneously. Kirchoff's voltage law tells us that only one voltage level can exist at any given point in an electrical circuit, at any given time. Consequently, we cannot simply have two devices transmitting to one another on the same conductors at the same time. If we wish to have this simultaneous, two way data transfer between devices, then we either need two sets of 8 conductors or else find a suitable means of sharing the same conductors. In order for two transmitting devices to share the same conductors, a more complex voltage representation of binary data is required (this requires modulation techniques). While both these solutions are possible, neither is attractive - the former because of the increased number of conductors and the latter because of the complex line-drivers required.

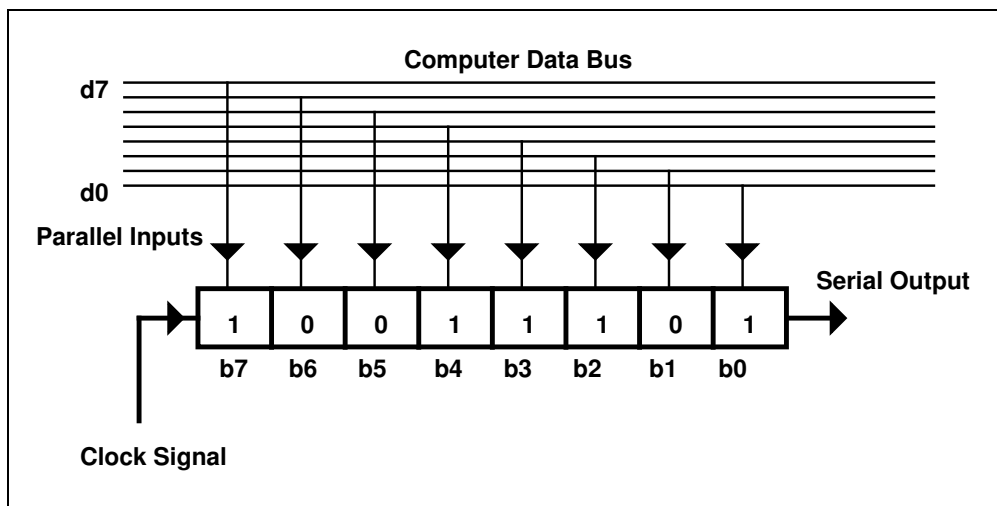
Parallel communication systems do have one major advantage over rival technologies. A major feature of parallel communication is its relative speed of transmission. All the data bits that form a single character are sent and received simultaneously. However, speed is not always a prerequisite in communications. For example, a CAD to CNC link does not need to be fast because the physical machining process is not comparably fast. Moreover, as the cost of the link itself increases (with distance), there must be a trade-off situation, where it becomes extravagant to continue with parallel links and an alternative must be used. The alternative is serial communications, where one line is used to transfer all data bits.

It should also be well noted that the cost of physical hardware in parallel links is not the only reason for their lack of use in long distance communications. Had there been a universally adopted standard for medium-distance parallel communications, then it is just possible that the cost of parallel hardware would have been overlooked in favour of "plug-in compatibility". In the final analysis, the costs associated with developing specialised hardware and software probably far outweigh the cost of line-drivers and extra cabling. However, despite the emergence of parallel communications specifications such as IEEE-488, standards are no more universal than in serial communications.

Parallel communications have therefore remained in the realm of short-distance office and laboratory communications.

### 4.3 Parallel to Serial Conversion

Common digital circuits, such as "shift-registers" allow us to convert parallel information, as found on an internal computer data bus, into serial information. This is illustrated in Figure 4.2.



*Figure 4.2 - Shift-Registers for Parallel to Serial Conversion*

Data bus bits, b0 - b7, enter the shift-register simultaneously (in a parallel fashion) and are then fed out of the register, in sequence, each time the clock-signal waveform reaches a pre-defined point (usually the negative edge). This is shown in Figure 4.3.

Once data is converted from parallel form into a serial bit stream, then it can be transmitted over long distances using only a single conductor. This provides significant savings over parallel transmission (in terms of cabling and line drivers).

It is an equally simple task to generate complementary decoding circuits for receiving devices, that will convert incoming serial bit streams into a parallel form for internal use. These digital circuits are conceptually similar to the ones used for parallel to serial conversion, except that data flow is in the opposite direction. When the conversion and decoding circuits are linked with a conducting cable, the net result is a serial communication link.

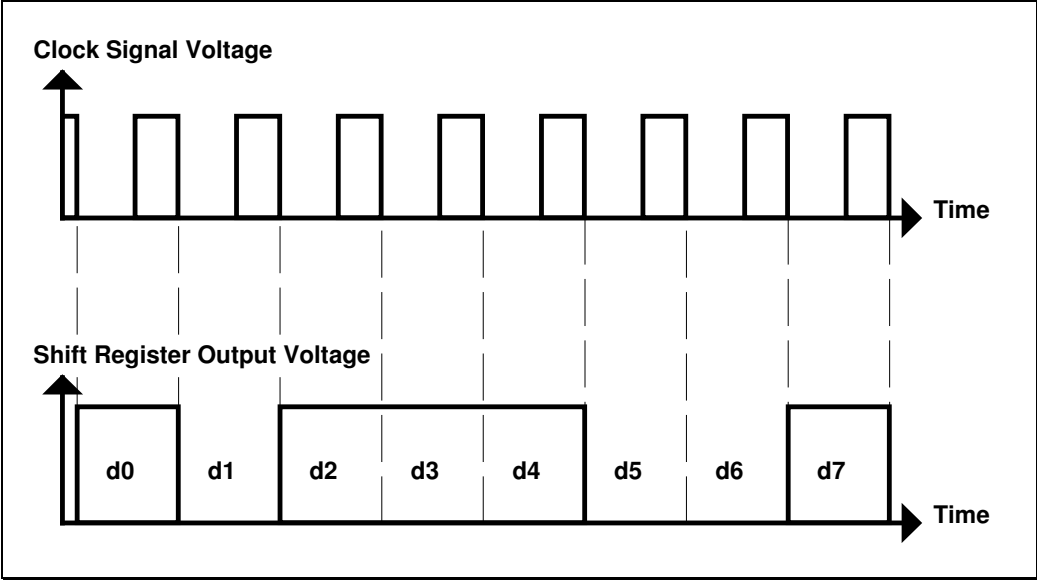


Figure 4.3 - Timing Diagram for Parallel to Serial Conversion

As with all data communications, both the transmitting and the receiving devices must agree on the form of binary data representation to be used. It is important to understand that voltage is a relative quantity and is not absolute. The representation of binary ones and zeros at a transmitting device will be through voltage levels, with respect to its local reference level. Similarly the interpretation of data at a receiving device will be through voltage levels with respect to its own local reference level. If these two reference levels are not the same then transmitted data cannot be correctly interpreted by the receiver. A "common" reference voltage line is often connected between communicating devices so that all devices interpret data with respect to the same reference. This is shown in Figure 4.4.

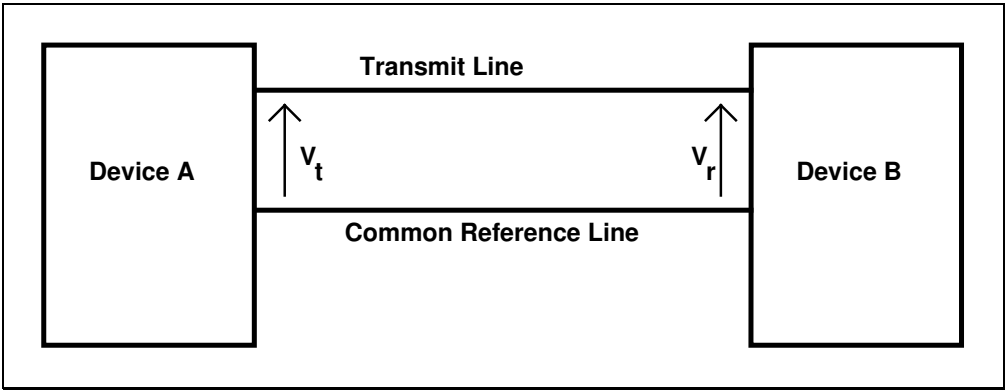
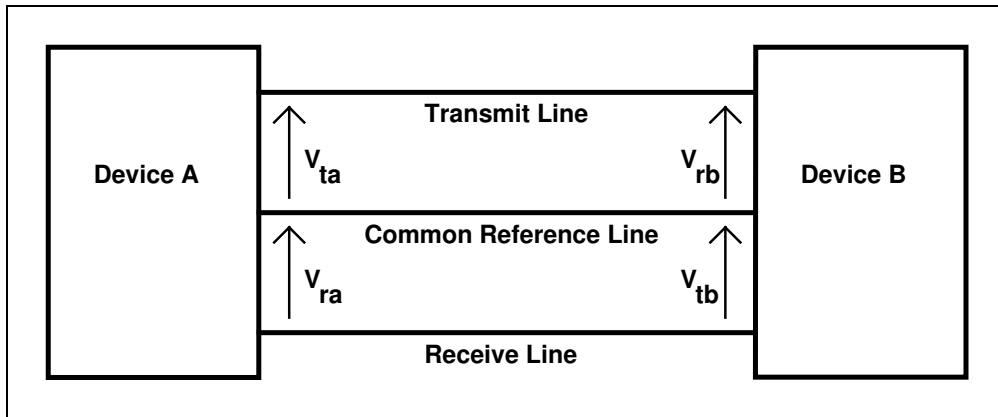


Figure 4.4 - A Common Voltage Reference for Transmission Signals

Once we have a serial mechanism by which we can transfer data on a single line, then it is also feasible to provide an additional line for simultaneous, two-way communications. This is shown in Figure 4.5.



*Figure 4.5 - Simultaneous Two-Way Serial Data Transmission*

A transmission system in which data communications can only ever occur in one direction (master to slave) is referred to as a "simplex" link. A system in which data communications can occur in two directions, but not simultaneously is referred to as a "half-duplex" link. When interconnected devices can simultaneously transmit and receive then the link is referred to as a "full-duplex" link.

When we transfer data in parallel systems, the minimum unit of information (byte or word) is defined by the number of data conductors. For an 8 bit, parallel system, an entire byte of information is transferred simultaneously from one device to another. If that byte represents alpha-numeric information then the minimum amount of information transferred is one character.

In serial systems the minimum unit of data that can be transferred is one bit. However, sometimes it is necessary to treat a group of bits as a single character, such as in the ASCII system. Therefore in a serial link, we define systems that treat each bit as a discrete unit to be "Bit-Oriented". Serial systems in which bits are only interpreted in clusters of 8, in order to represent alpha-numeric, are referred to as "Character-Oriented" systems.

To summarise the concepts of serial data communications between computer systems, we can say that it consists of 4 conversion phases:

- (i) Conversion from parallel to serial representation
- (ii) Conversion from internal form to external transmission form
- (iii) Conversion from external transmission form to internal form
- (iv) Conversion from serial to parallel representation

The four phases are shown schematically in Figure 4.6 for a simplex link. Phase (i) involves the clocking of parallel data out onto the transmission line, one bit at a time, with the rate determined by the clock speed. Phase (iv) is the complementary function that has to clock serial data back into a register for parallel transfer to the target device's internal bus. Phases (ii) and (iii) are performed by line-driver and line-receiver circuits, which act as the interfaces (marked I/F in Figure 4.6) between the internal data representation of the computer system, and the external data representation required on the transmission line. These two phases modify the transmitted signal into a form that minimises the effects of line distortion and attenuation.

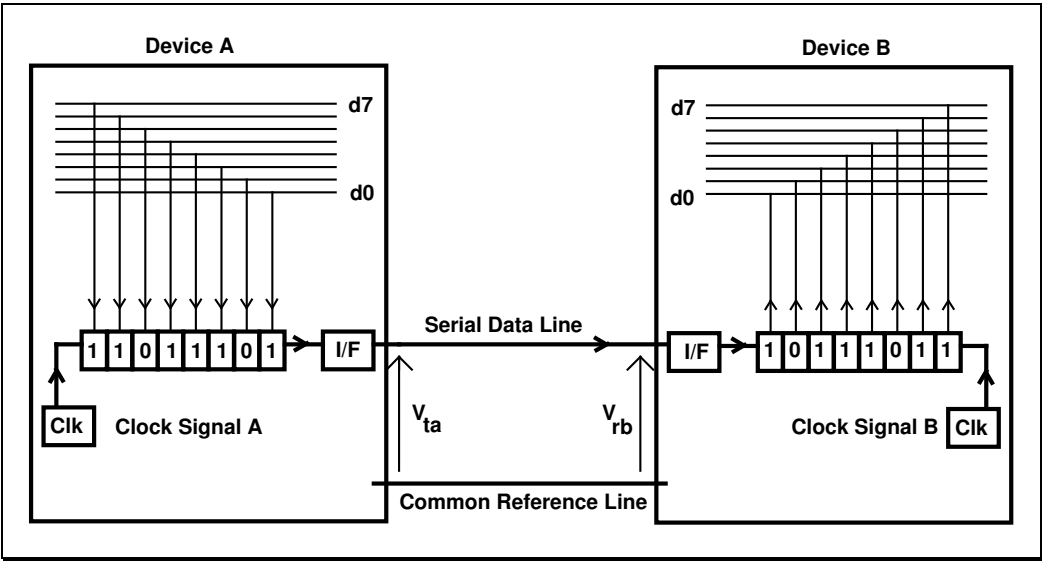


Figure 4.6 - Schematic of Serial Data Transfer Between Devices

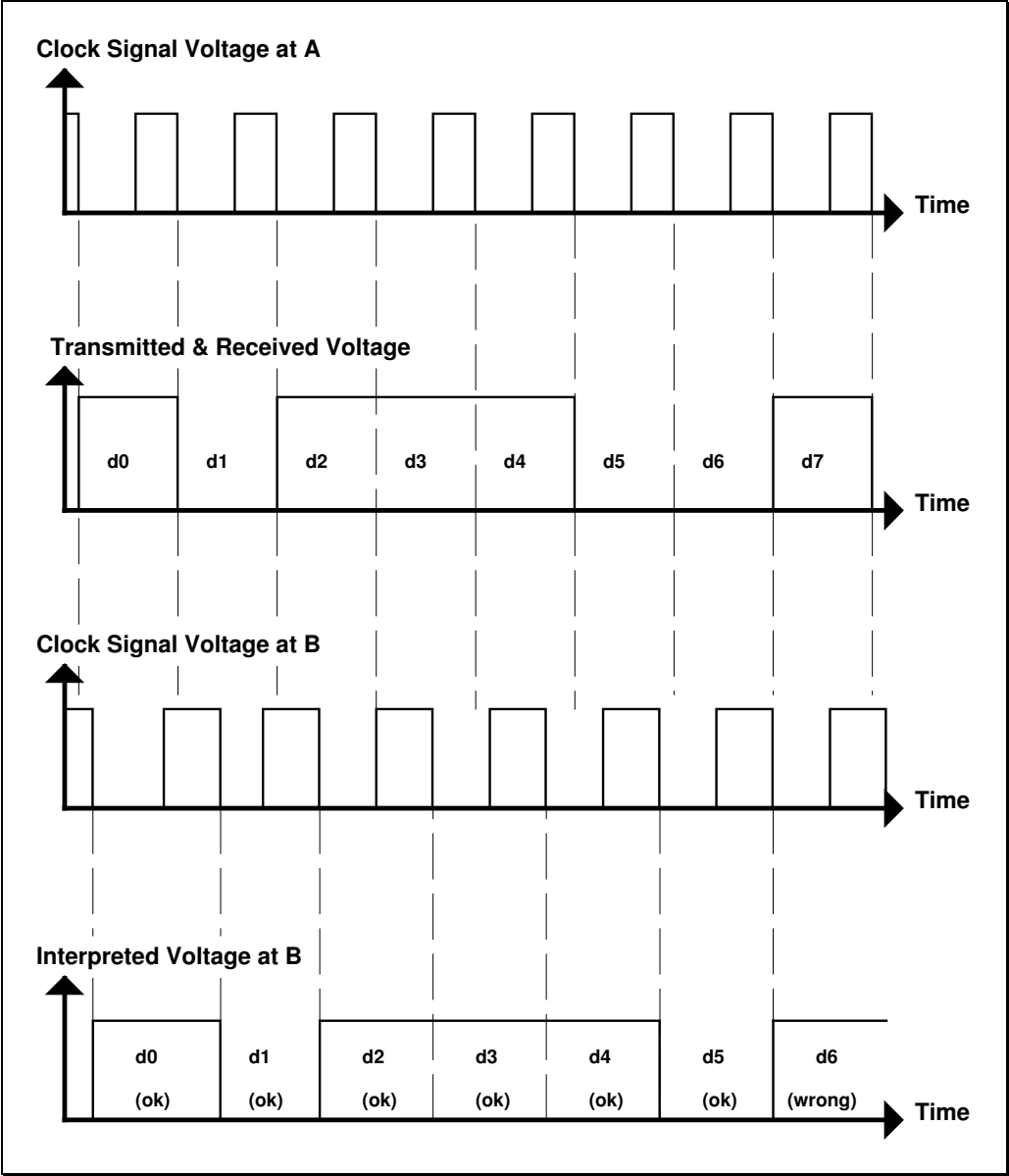
## 4.4 Synchronous Serial Data Communications

The schematic of Figure 4.6 shows two devices that are linked by a common pair of conductors. Device "A" clocks out voltage pulses (the data signal), whose widths are determined by the frequency of clock signal "A". It is assumed that the transmission medium is ideal and that the signal appearing at the receiver is exactly the same as that emanating from the transmitter. The receiving device, "B", reads incoming voltage pulses at time intervals determined by the frequency of its clock signal "B". That is, at a predefined point in its clock cycle (the negative going edge say), device B examines the input signal and accordingly places a high or low value into the register (and shifts other values along).

If the frequency of clock signal A and the frequency of clock signal B are not precisely the same, then it is possible that device B will not be able to interpret the incoming signal correctly. This is shown in an exaggerated form in Figure 4.7. As can be seen from this diagram, the mismatch between the transmitter clock and the receiver clock eventually causes the receiver to misinterpret the incoming signal - even though it has been correctly received.

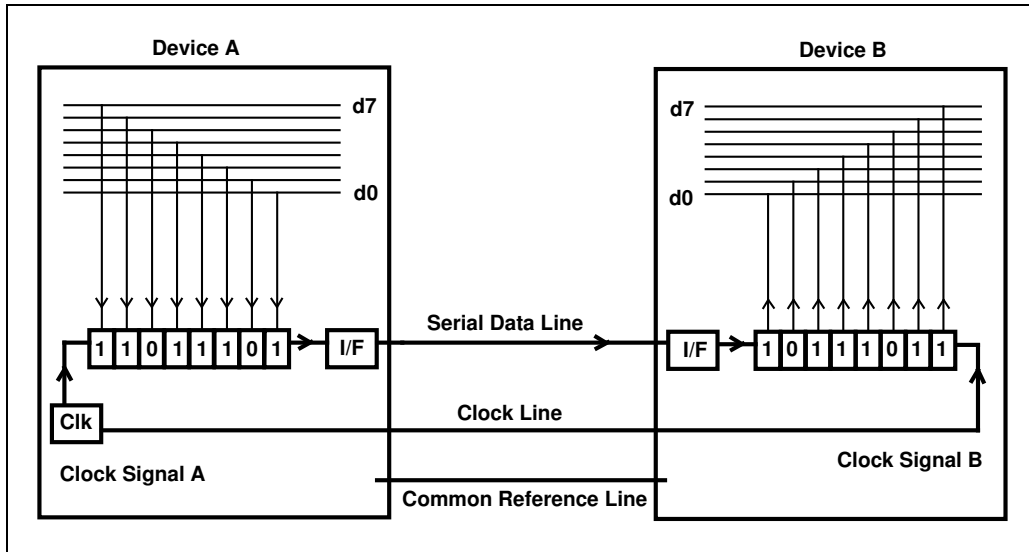
Although Figure 4.7 exaggerates the phenomenon of mismatched clocks, it illustrates that even slight mismatches, between the clock used to shift information out of the transmitter's register and that used to shift data into the receiver register, will ultimately lead to erroneous interpretation of data. Since it is not feasible to simply expect two independent free-running clocks to be in synchronism, a mechanism must be found to co-ordinate the clocking of data between a receiver and transmitter.

The first technique that we shall examine that enables a receiver to correctly decode incoming serial data is referred to as "synchronous" serial transmission. This involves "synchronising" the receiver's clocking signal to that of the transmitter.



*Figure 4.7 - Incorrect Interpretation of Data Due to Unmatched Clocks at Receiver and Transmitter*

The most obvious way to synchronise the receiver's clock to the transmitter is to provide an additional line, containing the transmitter's clock signal, in parallel with the signal line. This is shown in Figure 4.8.

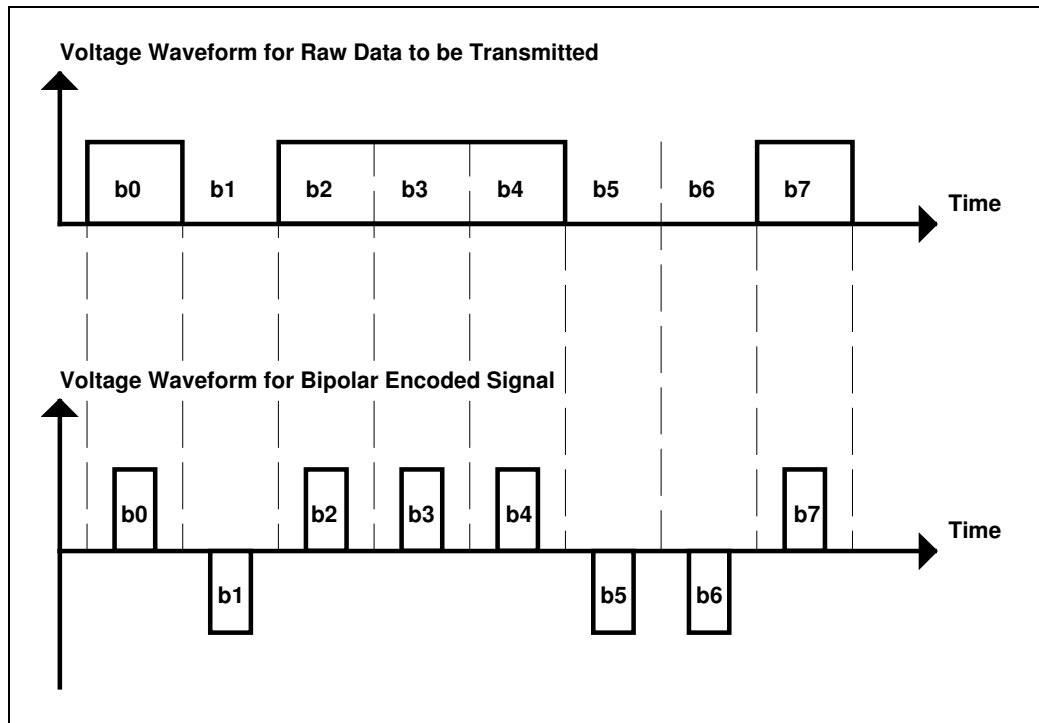


*Figure 4.8 - Synchronising a Receiver to a Transmitter*

Although this appears to be a good idea, in practice, this technique is not generally used. This is partly because the use of an additional line diminishes the value of serial transmission and partly because it is not always possible to have the additional line between the receiver and the transmitter. This often happens when using public communications networks that provide only a single line for data. An alternative is to somehow embed the transmitter's "clocking" information into the data signal itself. The receiver needs to "extract" or "recover" the clock signal in order to correctly decode the incoming signal. Whenever the receiver extracts clocking information from the transmitter, the scheme is referred to as "*synchronous transmission*".

There are a number of means by which synchronising information can be transmitted within a data signal. The first method is referred to as "clock encoding and extraction". Under this scheme, clocking information is placed into the data stream by use of special transmission techniques. The receiving device extracts this clock information and uses it as the reference clock for data decoding.

The simplest form of clock encoding and extraction is achieved through a bipolar encoding technique, where binary 1's are represented by a positive voltage and binary 0's are represented by a negative voltage. Between each pair of binary bits, the voltage waveform is at zero volts. This is referred to as an "RZ" waveform or a Return-to-Zero waveform. A bi-polar, RZ waveform is shown in Figure 4.9.



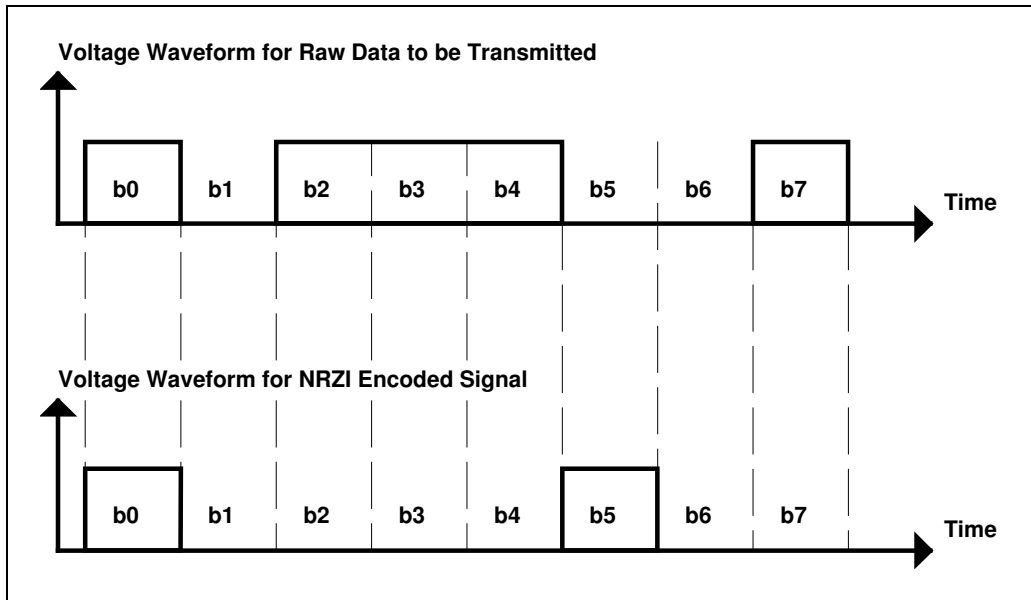
*Figure 4.9 Bi-Polar Encoded (RZ) Waveform*

The fact that we now have a zero voltage state in between successive bits means that the receiver can detect the time duration between bits and therefore extract "clock" information. Note how this is not possible with the raw waveform of Figure 4.9, because successive, contiguous streams of 0's or 1's make it impossible for the receiver to differentiate between bits.

Another, slightly more complex technique, known as the "Manchester Encoding", or Phase Encoding (PE) technique, produces a waveform with the normal, two voltage levels. It is referred to as a "Non Return to Zero" or "NRZ" encoding technique. The width of pulses in the Manchester scheme varies depending on whether successive bits are the same or not. This provides a mechanism for "extracting" clock information at the receiver.

If clocking information is not embedded into the transmitted data signal, there is a possibility that the receiver's own clock will allow it to drift out of synchronism whenever a long stream of zeros and ones are received. Therefore, the only other means by which a receiving device can synchronise itself to a transmitter, is to always ensure that there are enough transitions ( $0 \rightarrow 1$  and  $1 \rightarrow 0$ ) in the data signal. This means that the transmitted data must be encoded in a special format.

This encoding is referred to as a "Non Return to Zero Inverted" (NRZI) technique or a "differential encoding" technique. Under this scheme, the original waveform is transmitted raw, until a binary "0" occurs. From this point onwards, the waveform is inverted after every binary "0". This is shown in Figure 4.10.



*Figure 4.10 - Differential (NRZI) Encoding of Waveforms*

Differential encoding ensures that there will always be sufficient bit transitions in the transmitted waveform, to which the receiver's clock can synchronise, provided that there are no continuous streams of binary "1s". In practice, this does not occur, purely because of the way in which binary data is transmitted in synchronous links. Data in bit oriented schemes is broken up into groups of bits, referred to as "frames". In order to distinguish between frames, binary 0's are always strategically inserted into the data to generate special bit patterns.

## 4.5 Asynchronous Serial Data Communications

Asynchronous serial communication is perhaps the most common form of data communications that must be handled within a manufacturing environment. It is widely used for communication between:

- Production machines (CNCs) and computers
- Production Controllers (PLCs) and computers
- Computers and terminals
- Computers and printers
- Computers and plotters
- Computers and computers.

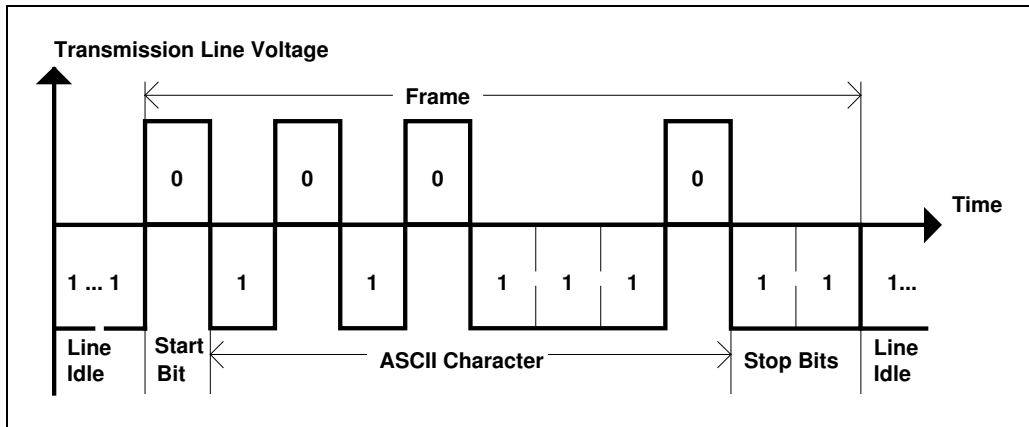
Asynchronous communication is of importance to us, if only because it is one of the most misunderstood topics in manufacturing (with good reason). It is also important because it is the area in which a large amount of specialised tailoring must be done by manufacturing professionals.

Historically, asynchronous serial communication was introduced for links in which data transmission occurred only at spasmodic intervals - say between a terminal (used by a human) and a computer. In these situations, the transmission line spends a large proportion of its time in an "idle" state, with no transitions between high and low. The receiver in these links therefore needs to be able to synchronise itself to the transmitter only at the start of an incoming data unit (and these are intermittent).

In as much as asynchronous links are designed for low volume data flow, the speeds at which transmission occurs are much lower than those associated with synchronous links. This means that it is not necessary to use complex encoding techniques to supply clock synchronising information to the receiver. The inclusion of a simple transition (1→0 or 0→1) between basic data units is enough to allow a receiver to decode the incoming waveform. In other words, asynchronous links are those in which the receiver does not attempt to extract the transmitter's clocking information.

Another feature that has been incorporated into asynchronous transmission, both for historical and practical reasons, is that the system is usually character-oriented. The basic unit of data transmission is normally an 8-bit quantity. This accommodates the ASCII (7-bit), extended ASCII (8-bit) and EBCDIC (8-bit) character sets. There is no reason why asynchronous systems must be restricted to character-oriented transmission, but in view of their common, practical application it has been a logical development.

Assuming an asynchronous ASCII system is used, Figure 4.11 shows how the character 'u', which has an ASCII value of 117 decimal (or 01110101 binary), is transmitted. This shows what is referred to as a "frame". A frame is a complete basic unit of information that incorporates data, plus the essential encapsulating bits that are used in the transmission process.



*Figure 4.11 - Asynchronous Transmission Format for the ASCII Character 'u'*

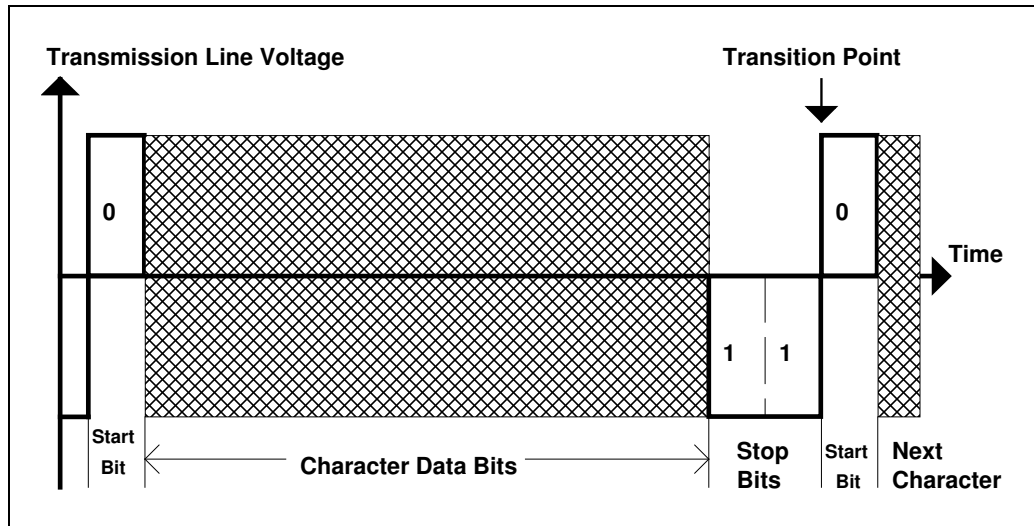
Figure 4.11 illustrates a number of important points in regard to asynchronous serial data transmission. These are:

- Binary 0 is represented by positive voltage
- Binary 1 is represented by negative voltage
- The line is kept at binary 1 when idling
- Each character is surrounded by start and stop bits.

It is a source of aggravation and confusion that the voltage logic for transmission of data on an asynchronous link is the exact opposite to what one would naturally expect. It is also in conflict with the voltage levels used in typical digital circuits such as TTL. We should also note that this representation is not a universal standard but it is in very common usage on asynchronous links.

In early electro-mechanical transmission, data was transmitted on a link by the flow and interruption of current on the line (and not by voltage variation). It was discovered that the reliability of such links could be greatly improved by maintaining a fixed current during idle periods. Data was then effectively transmitted by interrupting the quiescent state and stopping current flow. The idle state is when the system is "marking time" and hence is referred to as the "MARK" condition or binary 1 (current ON). The condition where current is absent is referred to as the "SPACE" condition or binary 0 (current OFF).

We must now accept that in asynchronous serial transmission, 1, MARK and Negative (no signal) are synonymous and similarly that 0, SPACE and Positive are synonymous. Once we come to terms with the inverted voltage logic, we can see how the start and stop bit/s allow a receiver to synchronise itself with incoming data. Figure 4.12 shows how the transition appears between successive characters.



*Figure 4.12 - Well Defined Transitions Between Frames in Asynchronous Communications*

The transition region caused by the succession from stop bits to a start bit provides enough information for the receiver to synchronise itself to the incoming data. Note particularly that there do not have to be 2 stop bits in order for this recognition to occur. The same result can be achieved with only 1 stop bit. In most asynchronous links, a transmitter and receiver can be set up to function on either 1 or 2 stop bits. As long as both devices use the same convention it doesn't make any difference.

At this point it is imperative to note another convention in asynchronous data communications. That is, START is signified by a binary 0 (positive voltage or SPACE) and STOP is signified by a binary 1 (negative voltage or MARK). This makes sense if we adhere to the convention of an idling line being in a MARK state. A change from MARK to SPACE indicates that data is being transmitted. This is again the exact opposite to what we have come to expect in digital circuits, where we normally set a value to binary 1 in order to enable and binary 0 to disable. You need to be extremely careful when reading literature related to this topic. You should try to deal with binary values (not voltage levels) when analysing asynchronous serial circuits and you can use Table 4.1 for equivalent terms.

As long as you remember that in order to start or enable any function in serial communications you must supply a binary 0 and to stop or disable any function you must supply a binary 1 then there is no confusion.

<i>Binary 0</i>	<i>Binary 1</i>
Space	Mark
Positive	Negative
Start	Stop
Enable	Disable
Low	High
False	True

*Table 4.1 - Equivalent Terms in Serial Communications*

An important question that needs to be examined is how and why synchronous and asynchronous links differ. If it is possible for an asynchronous receiver to accurately decode incoming information from a stop to start bit transition then why cannot the same be achieved in synchronous transmission?

The fundamental difference between the two schemes is speed. Asynchronous links generally transmit data at rates of either 110, 300, 1200, 2400, 4800, 9600 or 19200 bits per second (bps). These are relatively slow speeds in comparison to synchronous links that run at speeds in the order of Megabits per second. In order for an asynchronous receiver to extract timing information from the stop to start bit transition, its clock must run at many times ( $\approx 16$ ) the transmission frequency. This becomes a less reliable system as transmission rates increase toward those of a synchronous system.

The second, major difference between synchronous and asynchronous schemes is in the amount of data transmitted, and the overheads imposed by adding start and stop bits to a data frame. Since asynchronous schemes relate to low data rates, the insertion of start and stop bits (which increase overheads by a minimum of 2 bits for every 8 bits of data - 25%) is not of great consequence. However in synchronous schemes, which are designed for the high speed transfer of large quantities of data, these overheads are unacceptable.

Both synchronous and asynchronous schemes are more complex than might first be envisaged. Thus far we have introduced some of the essential concepts of the two methodologies. We shall build upon these to provide a practical working knowledge of serial communications links and particularly of RS-232, which is of particular importance in manufacturing.

## 4.6 Error Detection Techniques

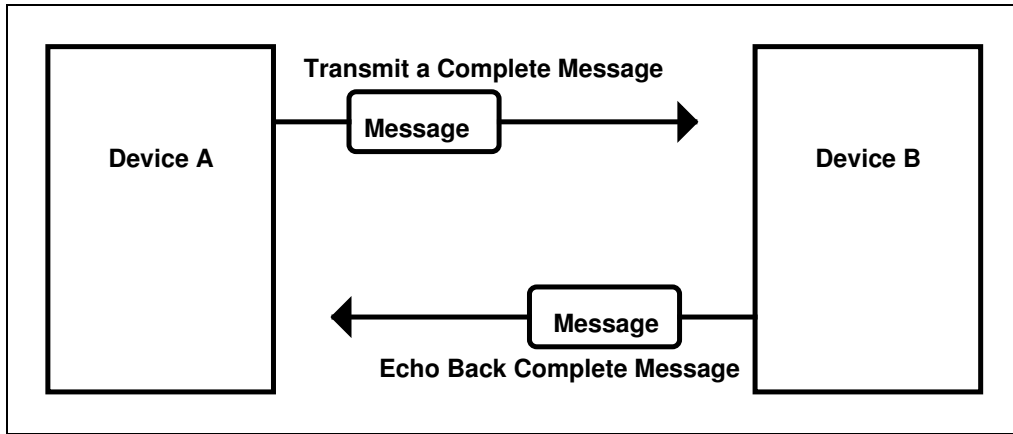
Regardless of whether we use synchronous or asynchronous communications techniques and regardless of how well we engineer a communication link, there is always the possibility that an error will occur when we transmit data from one device to another.

The gravity of an error in communications depends entirely upon the application. For example, if a communication link is established to transfer a part program from a CAD system to a CNC machine and an undetected transmission error occurs, then the corrupted program may ultimately result in damage to either the machine, cutting-tool or work-piece. On the other hand, an error in communications between a computer and line-printer may simply cause a spurious symbol to appear on a print-out. For each application, risk factors must be considered and appropriate error detection and correction measures taken to ensure that a required link reliability is achieved. Sometimes, such as in the case of a computer to printer link, these measures may amount to nothing more than accepting that errors will occur on rare occasions. In any event it is essential that we are aware of the risks involved.

The key point to note about all error detection techniques is that none of them are 100 per cent effective. In other words, no matter which technique we apply, a receiving device cannot be assured that it has received a message correctly unless it can verify the message against its own local reference. Obviously if a receiver already has a reference copy of the message, then there is no need for transmission in the first instance. Therefore, when we say that a link between two devices is secure, we really mean that the probability of an undetected error passing through the link is very small.

There are a number of error detection techniques in use. The scheme which one might intuitively expect to be simple and effective is called "echoing". This is shown in Figure 4.13.

The system relies upon the receiving device sending its incoming message back to the transmitter. The transmitter checks the echo message with the original and if they are the same, then it is assumed that the message at the receiver is correct. If the transmitter notes a difference between the original message and the echo, then the original message is re-transmitted. The echoing technique is sometimes used to check part program down-loads from a host computer to older CNCs, but is otherwise not in widespread use.



*Figure 4.13 - Error Detection and Correction by Echoing*

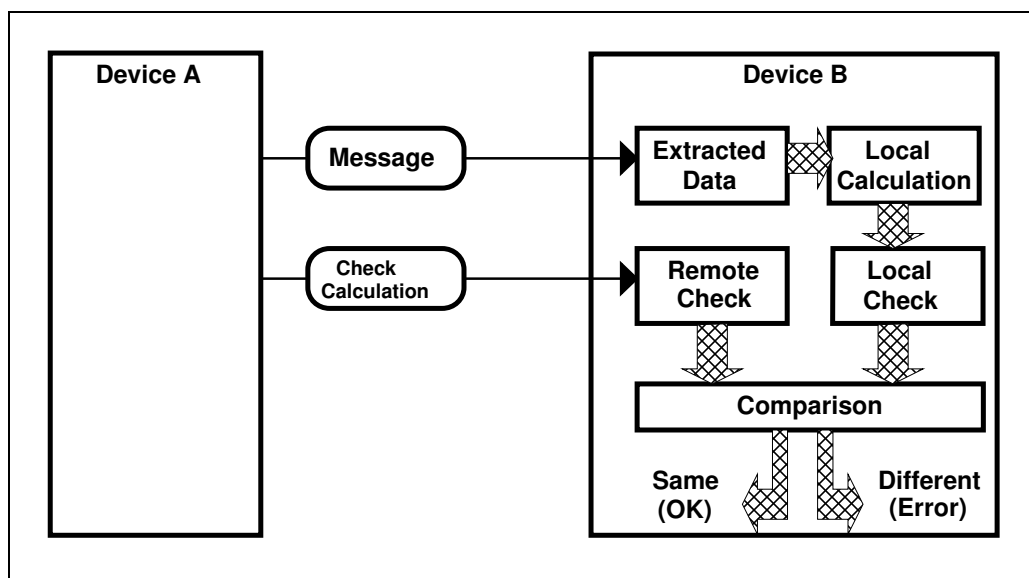
The most obvious problem with the echoing technique is the amount of time required to achieve error detection and correction. The system becomes unwieldily when the length of a message becomes large because we more than double the amount of time required to send data, even when transmission is correct. For example, the time taken to echo-send 1 Megabit of data at 10 kbps is greater than 200 seconds when the data is correctly received. If there is a single bit error in the data on the first transmission a correct transaction takes more than 400 seconds. Add to this the probability that the echo message may be incorrect (corrupted), while the original is correct and it becomes apparent that the system is not viable except when no other alternatives are available.

Two salient points emerge from an examination of the echoing system of error detection:

- (i) Large messages need to be broken down into smaller units for transmission so that if an error is detected in a unit then only that unit needs to be re-transmitted (and not the entire message)
- (ii) Complete echoing is not practical but a similar system (based upon smaller message units) can be implemented. An additional piece of "checking" information (which numerically summarises a message unit) can be transmitted with each message unit, so that the receiver can judge for itself whether or not it has correctly received the unit.

These two points are the basis for all error detection techniques used in communication.

Each unit of information that is transmitted on a link is followed by a piece of checking data that mathematically relates all the elements in the message unit. The checking data therefore "summarises" all the contents of the message unit in a single bit or group of bits. A receiving device takes in both the message and the piece of checking information. It then performs the same checking calculation on the incoming message (as the transmitter performed on the outgoing data) and compares this with the incoming check calculation. If they are the same then the receiver assumes the message unit is correct - otherwise it assumes the message unit is incorrect (corrupt). This is shown schematically in Figure 4.14.



*Figure 4.14 - Error Detection through Check Calculations*

There are two problems with these systems:

- A transmission error in both the check calculation and the message unit could cause the receiver to mistake an incorrect message unit for a correct one (and vice-versa).
- Since the checking calculation is physically smaller than the transmitted data unit it should be evident that it cannot uniquely describe that data. In other words, one check calculation can accurately describe a number of totally different data units.

It is because of these two problems that the receiver can never be 100 per cent accurate in determining the validity of an incoming signal. This is the penalty that we pay in order to increase the speed of transmission.

In simple statistical terms however, we can see that:

$$\Pr(E_{\text{unc}}) = \Pr(E_{\text{d}}) \quad \dots (1)$$

$$\Pr(E_{\text{uc}}) = \Pr(E_{\text{d}}) \cdot \Pr(E_{\text{c}}) \quad \dots (2)$$

Where:

$\Pr(E_{\text{unc}})$  = Probability of an undetected error in a system with no check calculations

$\Pr(E_{\text{d}})$  = Probability of a data error occurring

$\Pr(E_{\text{uc}})$  = Probability of an undetected error in a system with check calculations

$\Pr(E_{\text{c}})$  = Probability of a check calculation being correct when a data unit is incorrect

From the two probability expressions shown, it is clear that even if the chances of a check calculation being correct whilst the data unit is incorrect, are as high as one in a hundred, then we have still increased the reliability of a "checked" link by a factor of 100 over an unchecked link.

The simplest form of error-detection used in digital circuits and communication is referred to as "parity checking". When active, this comes in two different forms, namely "odd parity" or "even parity". A single parity bit is sent after each unit of data. In the "odd parity" system, the parity bit is either set or reset in order to make the total number of "1s" in the data unit (including parity bit) odd. In the "even parity" system, the parity bit is either set or reset in order to make the total number of "1s" in the data unit (including parity bit) even. The odd and even parity system waveforms are shown in Figure 4.15 for the serial transmission of the ASCII character "u", which has a bit pattern of 01110101.

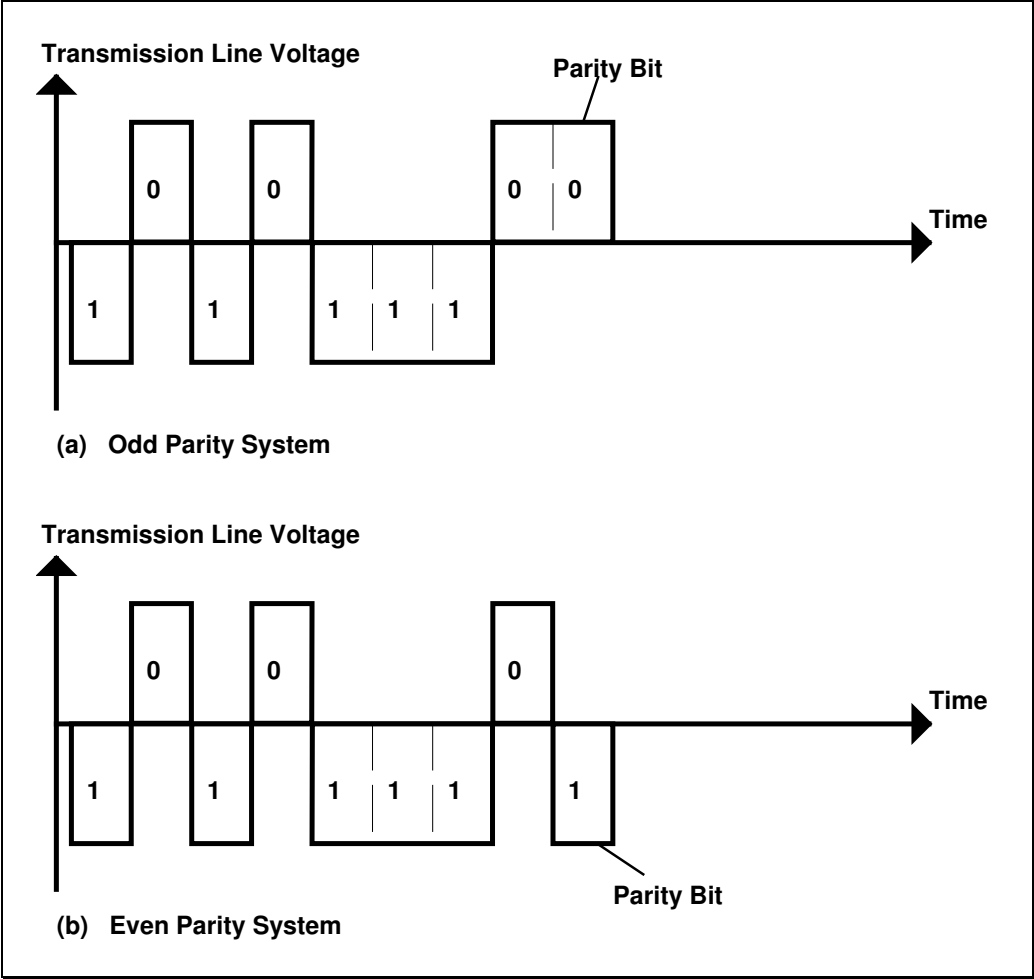


Figure 4.15 - Serial Transmission of ASCII 'u' Under Odd and Even Parity

Parity checking is yet another remnant from the early days of data communication. There are a number of problems apparent with the system:

- the corruption of 2 data bits or one data bit and the parity bit allow an error to pass through undetected
- parity checking is usually associated with character-oriented systems. Although a parity error can be detected with the system, few devices can automatically request for a re-transmission of a single character (it is too time consuming to disrupt information flow on a link for the sake of one erroneous character) and hence the system is of limited use.

The two problems noted above are typical of those associated with all error checking techniques, but they are more pronounced with the parity system. The probability of undetected errors slipping through a parity check is higher than with other techniques. More importantly, parity checking is at the opposite extreme to the echoing technique in that it breaks messages up into units of individual characters, which are too small for a practical link. Where the echoing technique demanded the re-transmission of an entire message in the event of an error, the parity system would demand a re-transmission of every erroneous character. This constant interruption to a transmission sequence, and demand for re-transmission of individual characters, adds unnecessary complexity and overheads to a link.

Despite the fact that the parity feature is seldom utilised, for error correction, it is still important to us because parity bits are very common items on asynchronous serial links. Parity checking is normally performed by hardware and therefore it is vital that two communicating devices are configured to the same parity settings in order for the link to function. Some devices allow parity to be either odd, even, switched off or configured so that the parity bit is always 1 (MARK parity / stick parity odd) or always 0 (SPACE parity / stick parity even). As long as communicating devices agree on parity, it does not need to become a major issue.

Having examined the two extremes of error detection, we can now focus on two important intermediate techniques. We now appreciate that a large message needs to be broken down into smaller units for transmission. In this way, if a unit is detected as being corrupt then only that unit needs to be re-transmitted, rather than the entire message. If the units are too small (such as individual characters), then the process becomes impractical for the reasons noted above. There is no single value of unit size that is universally adopted as being optimal for message transmission in serial links. A typical value for a unit may be in the order of a few hundred bytes. The common names for a basic unit are "block", "data frame", "packet" or "telegram".

In character-oriented serial links, the Block-Check-Sum, or BCS, is commonly used as the checking mechanism for blocks of data. Let us assume that we arbitrarily had a block size of 5 bytes and examine how a BCS might be calculated. Let us suppose that we wish to transmit an ASCII message "HELLO JACK" from one device to another. The message is broken up into two blocks, namely "HELLO" and "JACK" that are transmitted, in turn, with corresponding Block Check Sums. A common BCS calculation involves taking an "exclusive OR" of corresponding bits that make up the ASCII values of each data character and transmitting this as the check-sum. Table 4.2 shows how this is implemented.

<i>Character</i>	<i>ASCII Value</i>	<i>Block Check Sum (Running Tally of Exclusive OR of Characters)</i>
		0000 0000
H	0100 1000	0100 1000
E	0100 0101	0000 1101
L	0100 1100	0100 0001
L	0100 1100	0000 1101
O	0100 1111	<b>0100 0010</b> <i>(Transmitted Sum)</i>

*Table 4.2 - Calculation of an Exclusive OR Block-Check-Sum*

There is no single standard that defines the way in which a Block-Check-Sum calculation is to be performed. Table 4.2 shows only a simple example. However, the actual BCS calculation selected influences the effectiveness of the BCS in trapping errors. It is not difficult to perform some probability calculations in order to quantitatively ascertain the effectiveness of the BCS. That is, to determine the probability of the BCS character being apparently correct at the receiver, whilst the message is corrupt. This is done on an individual basis for each specification of a BCS checking scheme.

In the exclusive-OR scheme shown above, the BCS scheme would fail whenever the corresponding bits on pairs of characters were incorrect. The scheme would also fail in situations where a single bit in the data was incorrect and the corresponding bit in the BCS was also incorrect. In other words, for the above example, we can consider the data bytes and the BCS as a 6 byte message, transmitted in the following sequence:

<i>Character</i>	<i>Bit Pattern</i>							
	b7	b6	b5	b4	b3	b2	b1	b0
H	0	1	<u>0</u>	0	1	0	0	0
E	0	1	<u>0</u>	0	0	1	0	1
L	0	1	0	0	1	1	0	0
L	0	1	0	0	1	1	0	0
O	0	1	<u>0</u>	0	1	1	1	1
BCS	0	1	<u>0</u>	0	0	0	1	0

Multiple pairs of errors in any one of the bit columns shown in the example also make this particular BCS appear correct, even though both the data and the BCS are incorrect. Try this for yourself by reversing all the values of the underlined bits in the "b5" column. This form of reasoning can be converted, using permutations, combinations and the binomial theorem into an expression, which is ultimately dependent upon the probability of an error in a single bit occurring, or the mean Bit Error Rate (BER).

Therefore if we wish to ascertain the effectiveness of a BCS calculation, we need to determine the mean number of bit errors occurring per unit of time. This can be done by continuously transmitting a known stream of data to a receiver and having the receiver check each bit of incoming information against its local reference value. In a manufacturing environment, this needs to be carried out over a time period that accurately reflects the production environment. The test must have a time profile where factors such as switching of heavy current machinery, etc. are included. Errors can then be logged and the mean BER determined for the worst-case scenario.

Ultimately the objective for professionals, designing links in the manufacturing environment, is to minimise the mean bit error rate before even considering its influence on BCS calculations. Sometimes this involves the replacement of conducting links with optic fibre technology and so on. It must be made clear that check calculations are not a cure for the problem of bit errors - they are a secondary line of defence against data corruption. Minimisation of error occurrence must always be the primary defence.

Some communications links in the manufacturing environment are subjected to "error bursts". These are often caused through switching of high current equipment, and are characterised by a group (burst) of bits that are in sequence and corrupted. An otherwise low BER link may be affected spasmodically by these phenomena. If the duration of an error burst is more than 8 bits, then simple BCS calculations such as the one described above are unreliable because errors can occur in the same bit of consecutive characters. This can be visualised by examining the way in which the "HELLO" packet described above is actually transmitted on a serial link:

H	E	L	L	O	BCS					
01 <u>0</u> 01000		01 <u>0</u> 00101		01001100		01001100		01001111		01000010
◆◆◆◆◆◆◆◆										
Error burst										

Assuming that all the bits affected by the error burst are set high (say), then it is clear that corresponding (underlined) bits of consecutive characters may slip through the BCS.

We therefore use more complex check calculations that are bit-oriented but are also suitable for character-oriented systems. These are referred to as "Frame Check Sequences" (FCSs) or Cyclic Redundancy Checks (CRCs). The generic name for these check digits, which are appended to blocks of data is a "Polynomial Code". The number of binary digits in a CRC is chosen to reflect the worst case error burst conditions, but 16 and 32 bit CRCs are predominant in standards for calculations.

An extremely tedious mathematical analysis is required to quantitatively prove why CRCs are far more effective than BCS calculations in minimising the probability of undetected errors slipping through to the receiver. It is not the purpose of this book to enter into this analysis. Suffice to say that the following five principles are used in calculating a CRC:

- (i) An entire packet or block of data is treated as one long binary number "b"
- (ii) The number is multiplied by  $2^n$  by adding n zeros to the end of the number (resulting in the number "B"). The value of n depends upon the specification for the CRC calculation and is equal to the number of CRC digits to be transmitted.
- (iii) The new number "B" is divided by another selected binary number "g", which is referred to as a "generator polynomial". The value of "g" depends upon the specification for the CRC calculation, but it always contains one bit more than the number of CRC bits to be transmitted
- (iv) The remainder of the binary division is the CRC which is transmitted following the data.
- (v) The receiver takes the incoming bit stream (b'), multiplies it by adding "n" zeros (giving B'), adds the CRC digits (modulo 2) and divides each of them by the generator polynomial. If the transmission was successful, then the result of the division (r) is always zero - otherwise it is non-zero

Mathematically, the above steps are expressed by:

$$B = b \times 2^n \quad \dots(\text{ii})$$

$$CRC = \text{Remainder} \left( \frac{B}{g} \right) \quad \dots(\text{iii})$$

$$\begin{aligned} r &= \frac{B'}{g} + \frac{CRC}{g} \\ &= \frac{B'}{g} + \frac{B}{g^2} \\ &= \frac{B'}{g} + \frac{B}{g} \quad \dots(\text{iv}) \end{aligned}$$

All the calculations performed above are modulo 2, which means that there are no carry digits. Therefore "g<sup>2</sup>" is the same as "g". It also means that if B' and B have identical bit patterns, then their modulo 2 sum must be equal to zero since:

$$1 + 1 = 0 \quad \text{and} \quad 0 + 0 = 0$$

Note also that step (iii) in the above sequence is equivalent to performing an exclusive OR function, bit by bit between the modified frame contents and the generator polynomial.

The major characteristic of the polynomial code technique is that all error bursts with a duration of fewer bits than the generator polynomial are detected. It is also important to note the short-hand way in which the generator polynomial is expressed in standards. For example, the generator polynomial within the CCITT specification for CRCs is given by:

$$x^{16} + x^{12} + x^5 + x^0$$

This is equivalent to the number 10001000000100001. Clearly, the polynomial indices therefore represent the bits in the generator that are set to 1.

## 4.7 Signal Modulation

Up until now we have only examined communications systems where the conducting communication medium (cable) can only carry one bit of information at any instant in time. This observation followed on from a brief look at Kirchoff's Voltage Law, which says that at any one point in any electrical circuit only one voltage can be present at any instant in time. We also noted with interest, the wastefulness of long-distance parallel communication, where a single cable conveyed only one bit of information in any one data transfer. The logical progression was to examine serial communication, which was slower but more cost effective in terms of cabling.

We now need to accept that it is possible for a single cable to carry more than a single bit of information at any instant in time, whilst still obeying Kirchoff's Voltage Law. There are many ways to achieve this. The most obvious is just to add data voltage waveforms together and transmit the total voltage - but then how would we ever decode them at the other end? The answer is that we can't unless each of the individual voltage waveforms have been given unique characteristics *before* they are added together and transmitted. Then, at the receiving end, we can sort out the individual data voltage waveforms (from the total waveform) by searching for the unique characteristics.

We therefore need a more complex representation for binary data, that will allow many signals to share the same link, by effectively creating "channels" within the medium. This is achieved through signal modulation. This technique encodes each waveform with unique, frequency dependent characteristics that can be readily decoded.

The signal modulation used in digital communications is completely analogous to that used in radio or television broadcasting. You will note that between a television or radio transmitter and your receiver there is only one communication medium (the air) - yet many signals use the same medium simultaneously through the technique of modulation.

In order to understand how modulation works, we need to understand the relationship between a signal and its frequency spectrum. Mathematically, the frequency spectrum of a time dependent signal,  $x(t)$ , is obtained by taking its "Fourier Transform"  $X(f)$ . This is achieved through the following transformation:

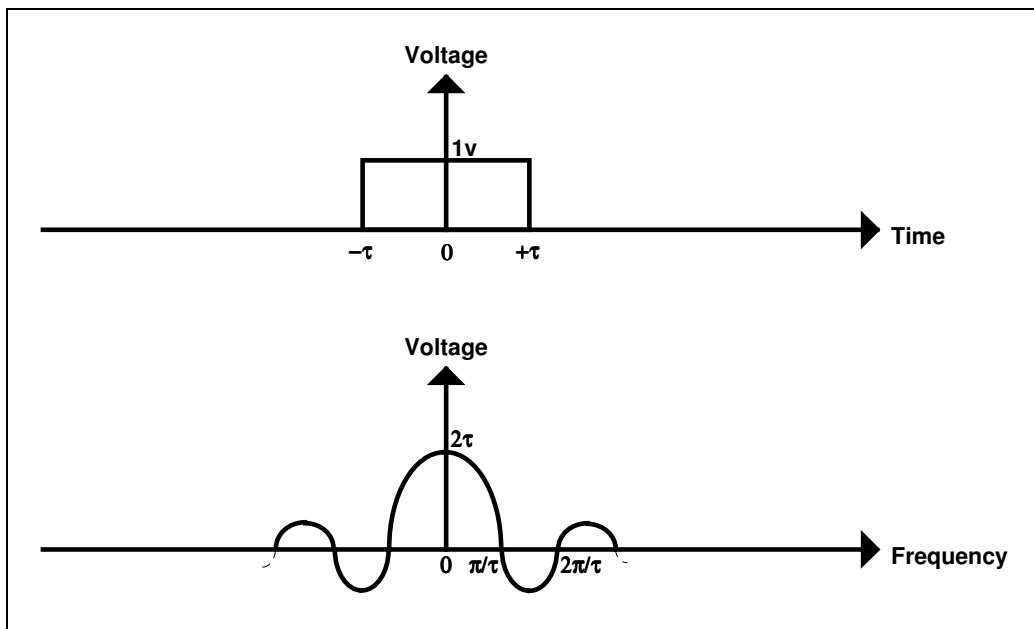
$$X(f) = \int_{-\infty}^{+\infty} x(t) \cdot e^{-j2\pi ft} dt$$

$X(f)$  is referred to as the "frequency domain" representation of the signal  $x(t)$ . The symbol "j" is used to denote the "imaginary number".

The "Inverse Fourier Transform" of the function  $X(f)$  takes us back to the original "time domain" representation of the signal  $x(t)$ :

$$x(t) = \int_{-\infty}^{+\infty} X(f) \cdot e^{j2\pi ft} df$$

This looks somewhat complicated, but in physical terms it really means that we can view energy (or voltage) from two different perspectives - frequency and time. As an example, the frequency spectrum of a simple square pulse, looks as shown in Figure 4.16.

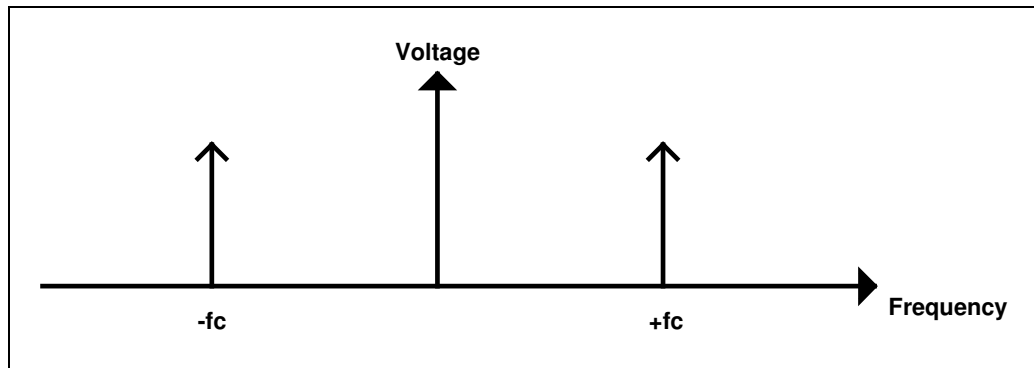


*Figure 4.16 - Frequency Spectrum of Square Pulse*

So, there are two ways in which we can identify voltage patterns on a transmission line - we can either look at their time domain or frequency domain representations. Up until now, we have only examined the time domain. In Figure 4.16, the frequency domain representation of the square pulse is much more complex than the time domain representation. Often the reverse is true, as for the unlimited cosine voltage waveform,

$$v(t) = \cos 2\pi f_c t$$

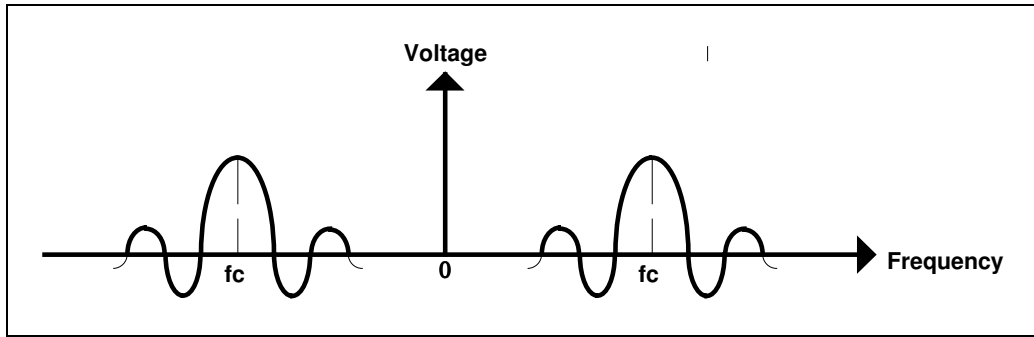
whose frequency spectrum is shown in Figure 4.17.



*Figure 4.17 - Frequency Spectrum of an Unlimited Cosine Waveform*

The frequency spectrum of a cosine voltage wave is composed of two infinitely tall and narrow peaks at frequency values corresponding to the frequency of the cosine waveform. If we were to be pedantic, each peak should have an " $\infty/2$ " height because the total wave energy is divided equally between the two peaks.

Does Figure 4.17 imply that if we output a cosinusoidal voltage waveform from a voltage oscillator, that we would measure infinite voltage peaks, using a voltmeter centred at that frequency? The answer is no. The frequency spectrum shown in the above example is for a cosine waveform that starts at negative infinity in time and ends at positive infinity in time. The frequency spectrum of a realistic time-limited cosine waveform, of frequency  $f_c$ , which starts at time  $-\tau$  and ends at time  $+\tau$  is shown in Figure 4.18.



*Figure 4.18 - Spectrum of Time-Limited Cosine Waveform*

The height and width of the main peaks, depend upon the time duration of the cosine waveform. The larger the value of " $\tau$ " the taller and narrower the main peaks. As the time duration of the waveform approaches infinity (reflecting infinite energy), the spectrum approaches that shown in Figure 4.17. However, we commonly use the spectrum shown in Figure 4.17 as an approximation when analysing systems.

Taking the argument further, we could look at a transmission line, where the voltage waveform (as a function of time) was the sum of 4 cosine functions, each of differing frequency and amplitude:

$$v(t) = A.\cos 2\pi f_1 t + B.\cos 2\pi f_2 t + C.\cos 2\pi f_3 t + D.\cos 2\pi f_4 t$$

The frequency spectrum for this waveform is the sum of the spectra of the individual waveforms. However, if one plots the time domain function, using some arbitrary values of frequency and amplitude, then a very untidy waveform emerges. If any individual component is significantly larger than the others, then the total waveform looks like a distorted version of the largest individual component. On the other hand, when we look at the frequency spectrum of the total waveform, it is surprisingly simple. This is shown in Figure 4.19, where  $f_2 > f_3 > f_4 > f_1$ .

What has been achieved in this example is of considerable interest to us. We have a number of different waveforms on the transmission line simultaneously (added together), giving us a frequency spectrum where individual components can be easily singled out.

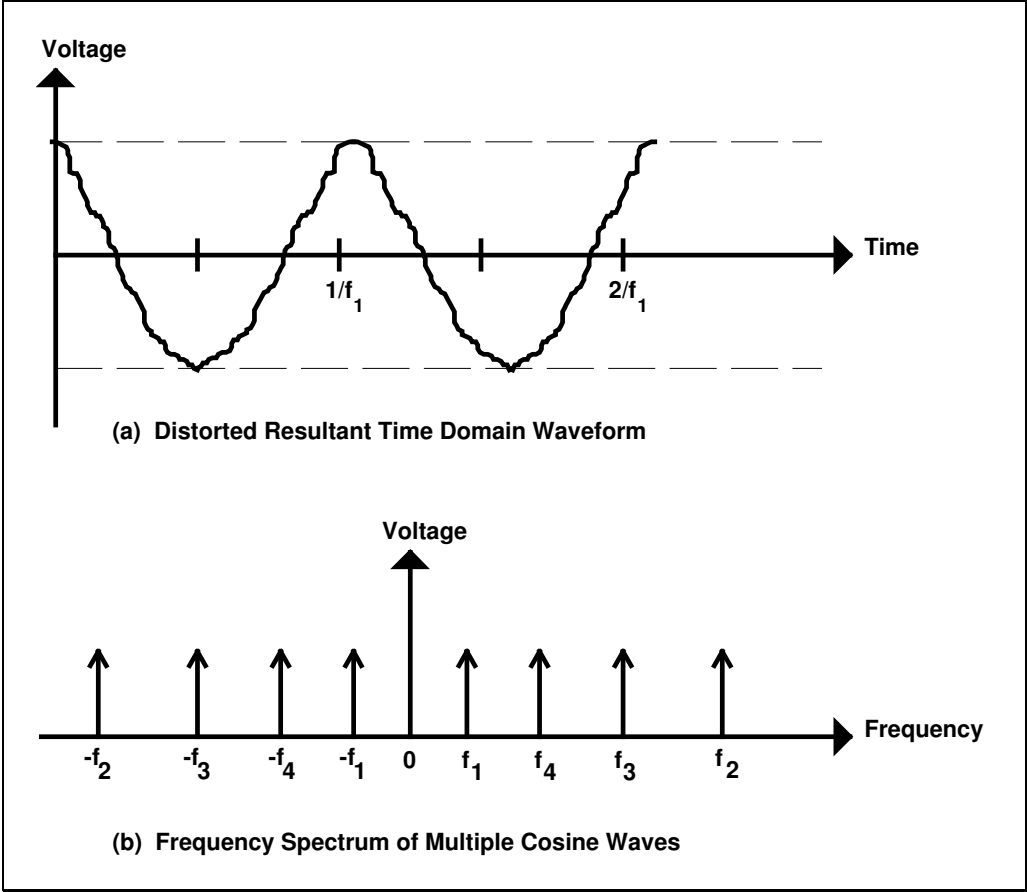
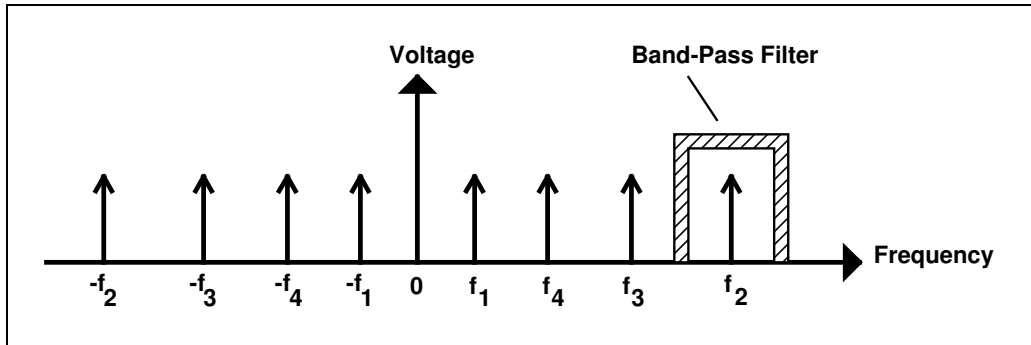


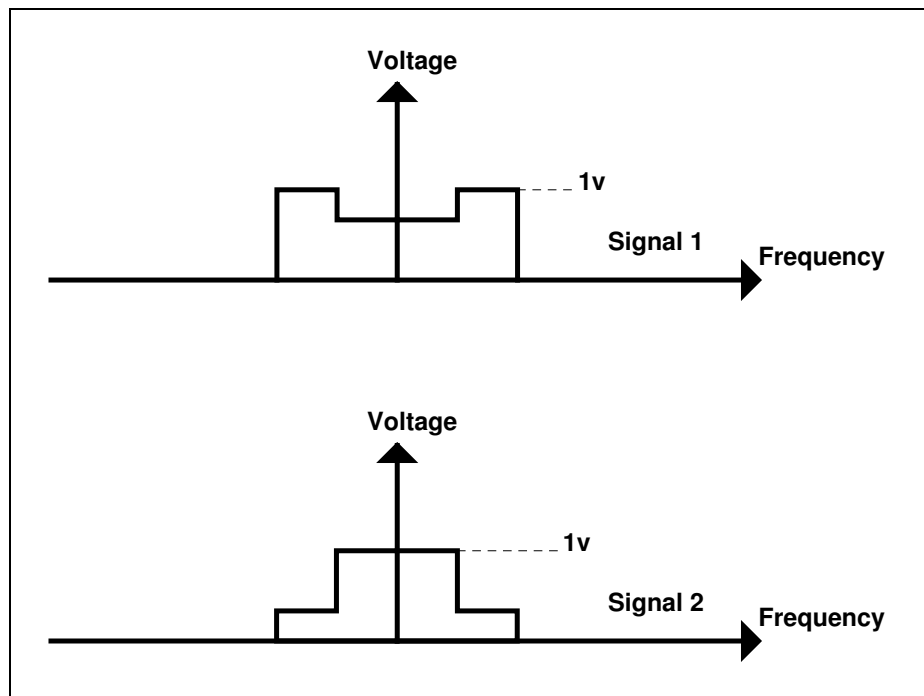
Figure 4.19 - Multiple Cosine Waves on Transmission Line

The conclusion that can be drawn is that if we choose our time domain signals appropriately, then it is possible to achieve a frequency spectrum in which individual components can be readily identified. For example if we transmitted the complex time domain signal of Figure 4.19, then we could recover any individual component of that signal at the receiver by "filtering" in the frequency domain. A "filter", in communications terms, is an electronic device that blocks out all frequencies except those within a desired range. The effect of a filter is shown in Figure 4.20, which schematically shows how the  $f_2$  component of our sample waveform can be retrieved.



*Figure 4.20 - Filtering out Desired Signals*

The question that now needs to be examined is how a signal, which does not give us a desired frequency spectrum, can be forced into a form where it does behave in an appropriate manner. For example, let us assume that we wish to transmit two signals on a communication link simultaneously, and that their spectra are as shown in Figure 4.21.



*Figure 4.21 - Frequency Spectra for two Different Signals*

Simply adding the two voltage waveforms (and spectra) together, as we did before, will produce a resultant waveform that is not only distorted in the time domain, but which is also not recoverable in the frequency domain, from filtering. This is shown in Figure 4.22.

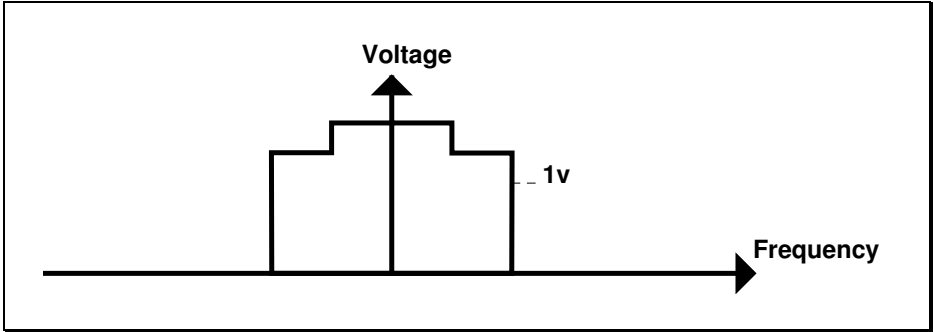


Figure 4.22 - Result of Adding Waveforms of Figure 4.21

However, by "treating" each of the signals prior to adding them, we can keep their respective spectra discrete. For example, if we electronically multiply signal 1 with a cosine wave of frequency  $f_1$  and signal 2 with a cosine wave of frequency  $f_2$ , (where  $f_2 > f_1$ ) the frequency spectrum of the combined signals is shown in Figure 4.23.

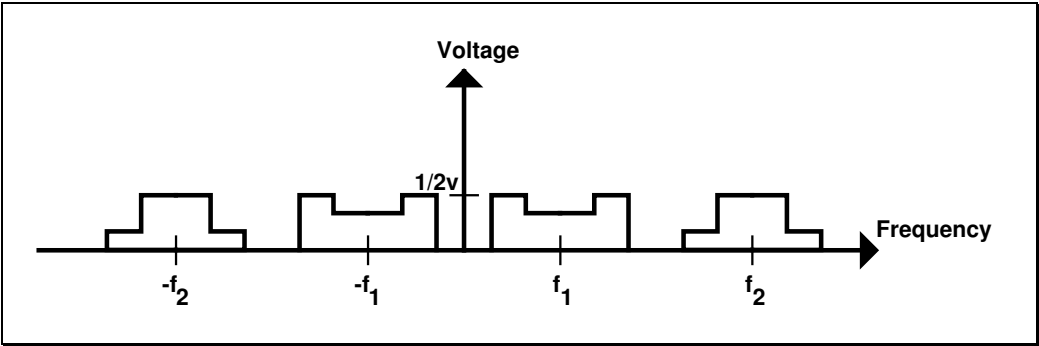
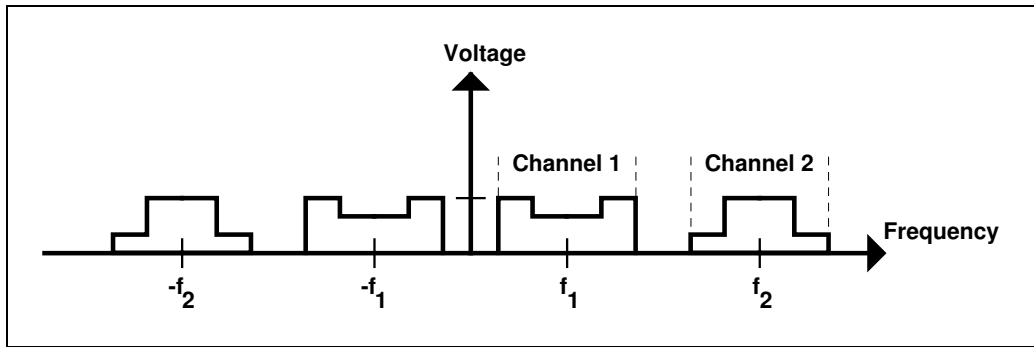


Figure 4.23 - Frequency Spectrum for "Treated" Signals

Direct multiplication of each of the signals with a cosine wave of a specific frequency (referred to as carrier frequency) is known as "Double Side-band Modulation" or DSB. It is easily achieved using fundamental electronic circuits. Note well the effect of this procedure. Apart from the fact that the original signals are "mirrored", clearly they have been "shifted" on the frequency spectrum away from their original "baseband" positions.

Provided that we limit the frequency range over which individual signals exist (the bandwidth), then we can use modulation to create many channels on the same communication medium. This is shown in Figure 4.24.



*Figure 4.24 - Channels Within Communications Media*

Mathematically, we can say the above phenomenon arises through the "convolution" of frequency domain functions that occurs when we multiply time domain functions. That is:

$$a(t) \cdot b(t) \leftrightarrow A(f) * B(f)$$

where the asterisk defines the convolution of the two frequency domain functions. The convolution of two functions is defined as follows:

$$A(f) * B(f) = \int_{-\infty}^{\infty} A(\Omega) \cdot B(f-\Omega) \, d\Omega$$

In other words, if two functions are multiplied in the time domain, then their spectra are convolved in the frequency domain. One can verify the result of Figure 4.23 by graphically convolving the waveforms of Figure 4.21 with the frequency spectra of two cosine waves of differing frequencies.

Graphical convolution is not as complex as the mathematical expression might suggest. Assume that the spectrum of the signal waveform is  $A(f)$  and that the spectrum of the cosine waveform is  $B(f)$ .  $A(f)$  is plotted on a new set of axes (voltage versus  $\Omega$ ) to represent  $A(\Omega)$ . The cosine spectrum is also redrawn on the new axes to represent  $B(\Omega)$ . In order to represent  $B(-\Omega)$ , we need to reflect the waveform about the zero frequency point. In the case of the symmetrical cosine spectrum, this results in the same waveform. If we substitute arbitrary values of "f" into the expression  $B(f-\Omega)$  we can move the  $B(-\Omega)$  along the  $\Omega$  axis. Whenever the two waveforms overlap, we can calculate the area of the product. If we plot the resultant area as a function of "f", then we have the convolution of the waveforms.

The "Dual" situation of the above convolution is also true and hence, if two functions are multiplied in the frequency domain, then their time domain equivalents are convolved. That is:

$$A(f) \times B(f) \leftrightarrow a(t) * b(t)$$

where

$$a(t) * b(t) = \int_{-\infty}^{\infty} a(\tau) \cdot b(t-\tau) d\tau$$

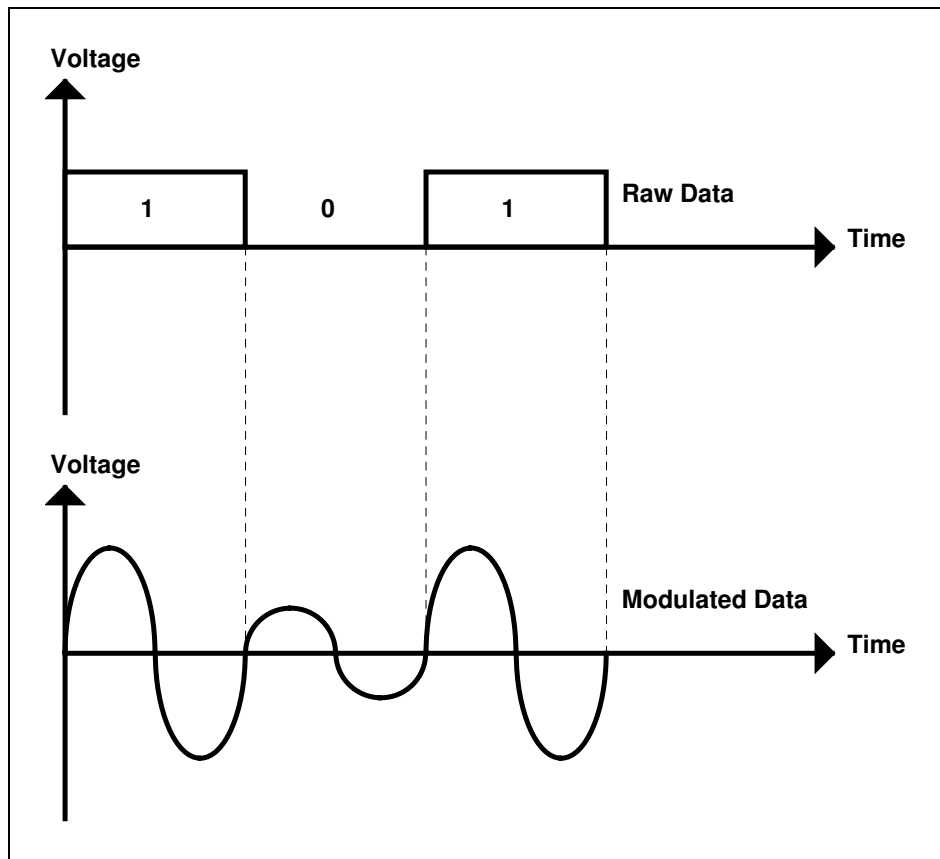
Leaving aside the mathematics of the situation, we now ask a question. Why can there not be an infinite number of channels on any one communications cable? As with any physical piece of apparatus there are always practical limits to operation. The higher up in frequency that we move signals (through modulation on a communication cable) the more the output signal is attenuated by that cable.

The limits or the "total bandwidth of the cable", are entirely dependent upon the physical make-up of the cable that is being used to convey the signal. When the entire cable bandwidth is used to transmit only one channel, then we refer to this as a "baseband" transmission. When the cable is divided into a number of channels then we refer to this as "broadband" transmission.

The earlier example of modulation, or signal channelling, is only one of many techniques used to achieve the same result. DSB modulation is not a common technique used for digital communications. Common modulation techniques used in conjunction with serial communications links include:

- (i) Amplitude Modulation (AM)
- (ii) Frequency-Shift Keying (FSK)
- (iii) Phase-Shift Keying (PSK)

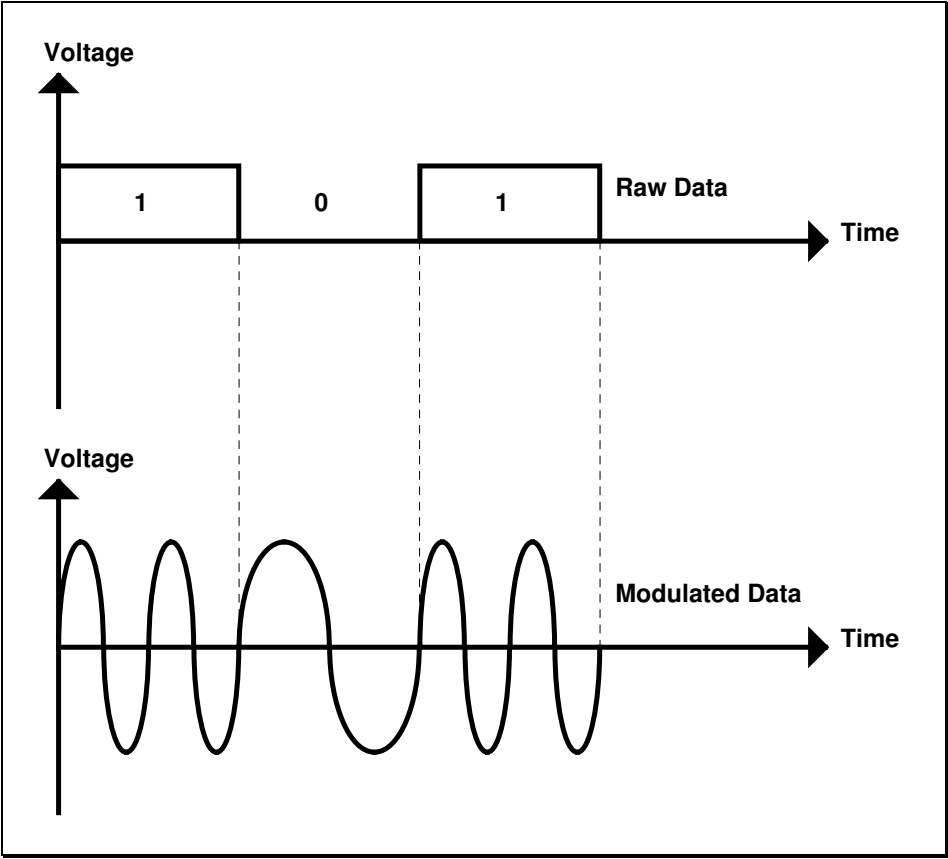
In an Amplitude Modulated (AM) system, a constant frequency, sinusoidal waveform is used to represent binary data. This waveform is referred to as the "carrier". The amplitude of the carrier varies depending on whether a binary "1" or "0" is transmitted. This is shown in Figure 4.25.



*Figure 4.25 - Amplitude Modulated Binary Signal*

If we select different carrier frequencies on the same communications link, then it is possible to have a number of different channels, each transmitting different binary information. In practice this form of modulation is not widely used, primarily because all the information is contained within the amplitude of the carrier. Line attenuation therefore affects make it difficult to maintain bit accuracy in complex communications systems.

In a Frequency-Shift Keying (FSK) system, a constant amplitude sinusoidal waveform is used to represent binary data. However, the frequency of the waveform varies depending on whether a binary "1" or "0" is transmitted. This is shown in Figure 4.26.

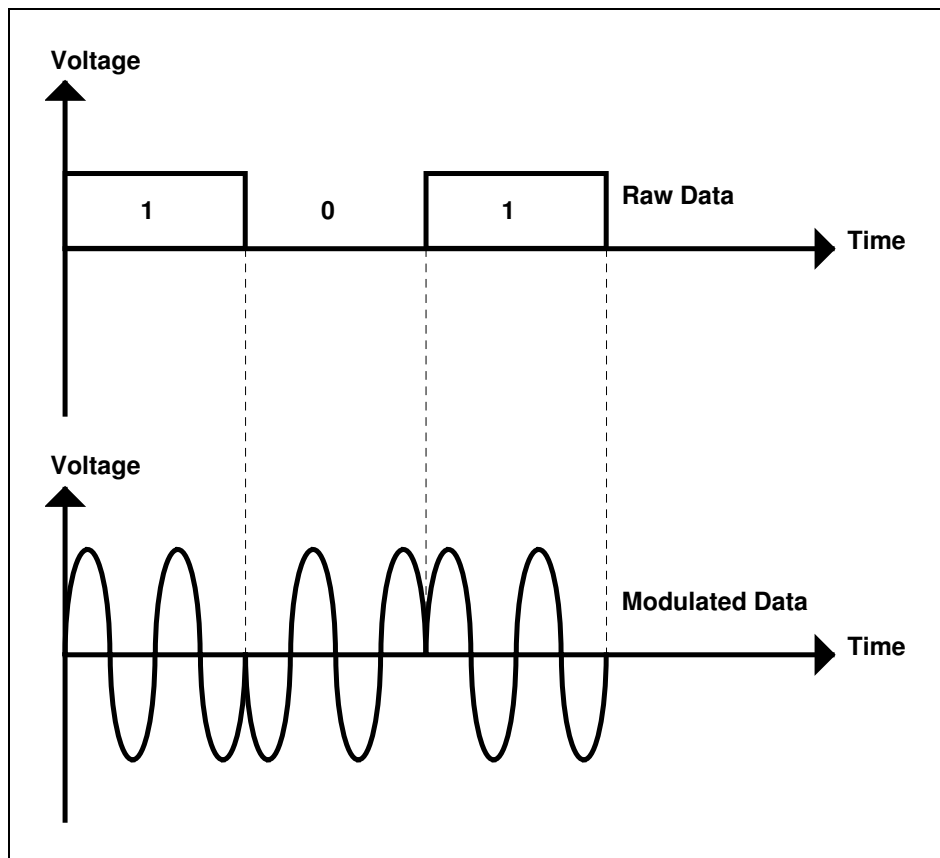


*Figure 4.26 - FSK Modulated Binary Signal*

This type of modulation is very commonly used for transmission of binary data at relatively low bit rates (300 - 1200 bits/second). Multiple channels can be achieved by representing binary numbers on one channel with one pair of sinusoidal frequencies, and on another channel by a different pair of sinusoidal frequencies.

The Phase-Shift Keying (PSK) system of modulation is based on the transmission of a constant amplitude, constant frequency sinusoidal waveform. The phase of the waveform varies, depending on whether a binary "1" or "0" is transmitted. This is shown in Figure 4.27, where there is a  $180^\circ$  phase change in the carrier between a binary 1 and 0. As with other modulation techniques, multiple channels can be supported on the same cable by using different pairs of carrier frequencies for each channel.

The form of PSK shown in Figure 4.27 requires the receiver to maintain a reference carrier, with which to compare the incoming signal and is sometimes referred to as "Phase Coherent" PSK. This coherent form of PSK modulation is difficult to decode and therefore an alternative form, referred to as "Differential" PSK is often used. In differential PSK, the carrier signal shifts by  $270^\circ$  in phase to indicate a binary 1 and by  $90^\circ$  to indicate a binary 0.



*Figure 4.27 - PSK Modulated Binary Signal*

Note that regardless of the form of modulation, if we have two devices connected together, we can have a full duplex communication link over a single cable. We achieve this by having two channels for transmission. One device represents binary numbers through the use of one pair of frequencies and the other device uses another pair of frequencies. We call this a "forward" and "reverse" channel arrangement

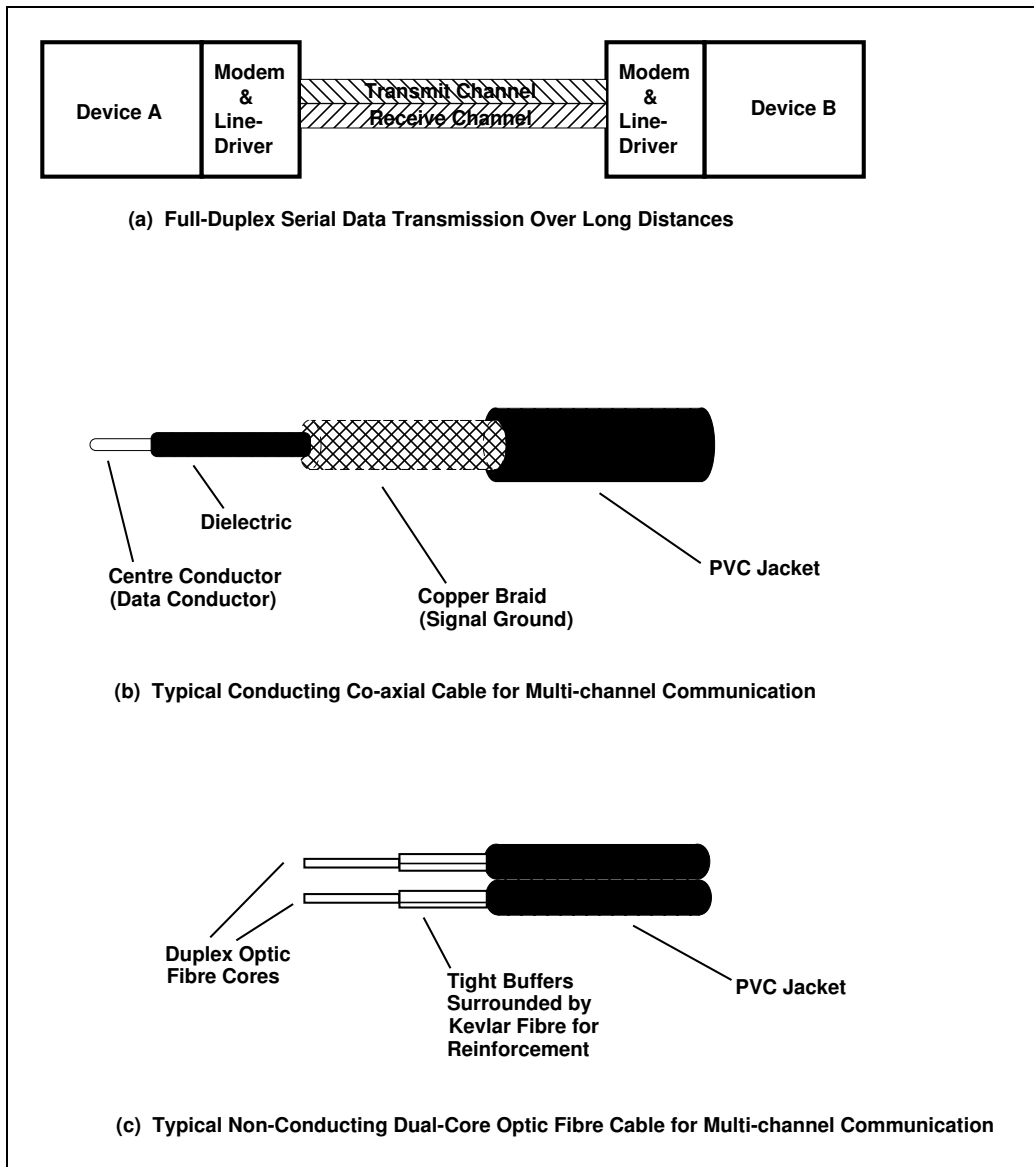
The electronic circuits that are used to decode modulated signals are referred to as "demodulators". Demodulation circuits are designed to isolate specific components from the frequency spectrum of an incoming signal (as shown simplistically in Figure 4.20 for DSB). The specific mechanism by which demodulation is achieved depends entirely on the type of modulation used and the design principles of these circuits are outside the scope of this book.

Modulation can be used in two-way serial communications (half-duplex and full-duplex) and therefore intelligent devices need both a modulator and a demodulator circuit for transmission. The combined unit is referred to as a "Modem".

Referring back to our earlier, point to point serial link (Figure 4.5), where we observed that it was necessary to have two wires for simultaneous bi-directional transmission (full-duplex), we now see that with modulation only a single wire (and reference or ground line), with forward and reverse is required. In order to enhance the transmitted signal, a line-driver system, which amplifies the modulated signal is also commonly used. In very long links, line drivers are placed at strategic points along the link to ensure that the signal does not attenuate unacceptably along any one section. The complete system for long distance transmission is shown schematically in Figure 4.28 (a).

Once we have modulated a signal so that we can realise a multi-channel communications system for transmission and reception of data, then we can work with a range of different media - such as air, optic fibre or co-axial cable. The most prevalent data communications media are co-axial and optic fibre cables, illustrated in Figures 4.28 (b) and (c) respectively.

A Co-axial cable is composed of a signal-carrying, central conductor, which is enclosed in an outer conducting shell. The outer conductor forms the other half of the wire pair (signal return path). The two conductors are isolated by a dielectric material and the complete structure is housed in a non-conducting jacket. The physics of the co-axial cable design is such that it is highly immune from electromagnetic interference.



**Figure 4.28 - Serial Data Transmission Using Long Distance Cables**

Optic fibres are fine glass filaments sheathed in a cable. A light impulse at one end of a filament is reflected along the sides of that filament to the target end. The properties of the filament are such that light impulses can be transmitted over very long distances without significant attenuation. Electrical signals can be converted to light impulses through the use of specialised semi-conductor devices such as Light Emitting Diodes (LEDs). Conversion from light impulses back to electrical signals is also possible.

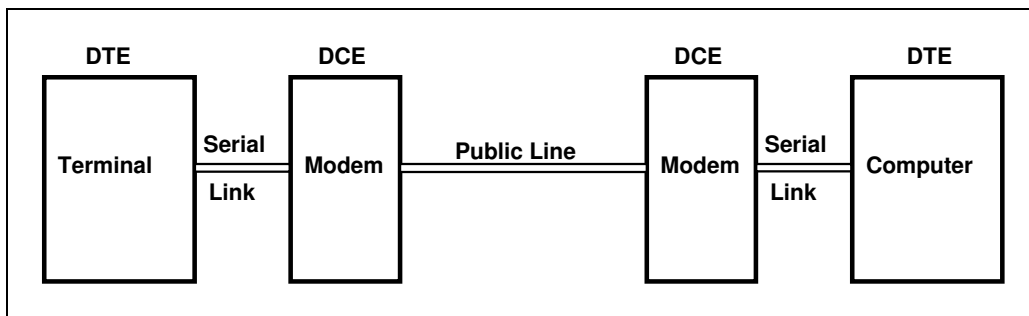
The beauty of optic fibre cable is that, unlike all conducting cables, it is not subject to electro-magnetic interference. This characteristic alone is of prime importance in the industrial environment. Optic fibre cable also has a very high bandwidth, thus allowing for a large number of communications channels. For these reasons optic fibre technology is gaining widespread acceptance. The only significant disadvantages of optic fibre are that connectors and terminators are more complex, and naturally, there must also be (electrical ↔ optical) interface circuitry at either end of a communication link. These factors obviously contribute to the overall cost of establishing an optic fibre link.

To conclude our discussion on modulation, we should note that its channelling properties enable a single cable to provide not only multiple digital data channels, but also voice and video channels. This is a very important, because it means that a single trunk cable, running through a large organisation can handle an enormous range of different communications functions, thus amortising the high installation costs.

## 4.8 DCE and DTE

Modern microprocessor based devices are often categorised by names or functionalities that are now either redundant or not directly applicable. A prime example of this is in the way such devices are configured for serial communications purposes.

In the early years of computing, communications between a dumb-terminal or teletype machine and a distant mainframe computer occurred through public communications lines and modems. Although it is now possible for modems to connect directly to a computer bus, some modems are connected to their local computer, terminal or teletype through a short, asynchronous serial link. This is shown in Figure 4.29.



*Figure 4.29 - Long Distance Connection of Terminals to Computers*

On top of each device in Figure 4.29 is a label, specifying that a device is either Data Communications Equipment (DCE) or Data Terminal Equipment (DTE). It seems a relatively logical definition. The two acronyms DCE and DTE are said to define the "gender" of the device. At one stage they also defined the gender of plug connectors, the pin configuration of communications plugs and so on. DTE devices and plugs were designed to connect directly to DCE devices and plugs.

Unfortunately the definitions shown above have, through changing technology and applications, been corrupted since their initial inception. Nowadays, one often comes across computers that are cited by their manufacturers as being DCE rather than DTE devices. Some computers are configurable to be either DCE or DTE. If it were not for the problems of plug connections and plug configurations then the nomenclature itself would not be of relevance.

As it happens, there is often a need to plug two devices together and therefore it is essential that we know the correct gender of each device, in order for plugs and wiring to be connected accordingly. For example, two devices of the same gender need specially reversed wiring in order to be interconnected.

Ultimately, we need to determine the gender of devices before we attempt to join them with a communication link. Sometimes this is achieved through reading a manufacturer's specification (if we believe that we can trust their understanding of DCE and DTE) and sometimes gender must be determined by electrical testing of plugs and connectors. The electrical testing can only be done after consulting the appropriate communications standard. As you may have guessed, each standard defines specific electrical properties for plugs which are of DCE and DTE genders, and it is these properties we need to test.

## 4.9 UARTS and USRTS

Now that we have examined some of the broader aspects of serial communications, we can return briefly to two important integrated circuits that are the basis of parallel to serial conversion within a computer. The Universal Asynchronous Receiver Transmitter (UART) and the Universal Synchronous Receiver Transmitter (USRT) chip are the basic, internal hardware units of asynchronous and synchronous computer communication. These circuits are generally used for character-oriented transmission schemes, since they divide bit streams into blocks of 7 or 8 data bits, encapsulated in stop and start bits.

The schematic diagram for the UART is shown in Figure 4.30, together with its links to the computer data bus. The computer address bus has been omitted to preserve some clarity.

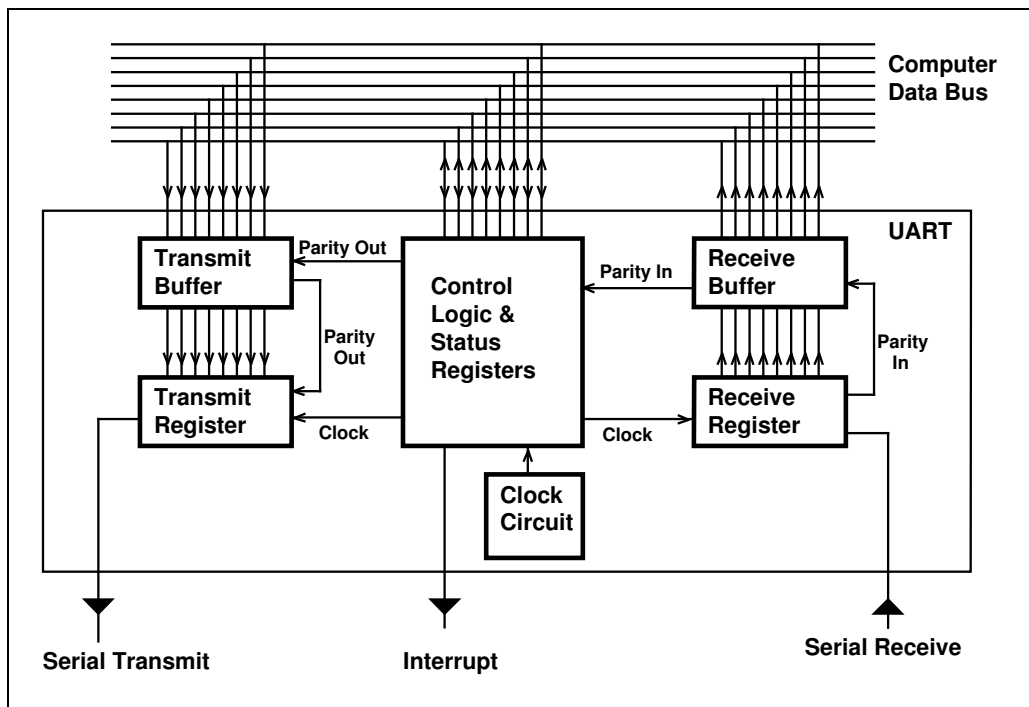


Figure 4.30 - Schematic of UART

The control logic block within the UART is responsible for adding parity bits, stop and start bits to outgoing characters. The control block also strips incoming frames of stop and start bits, checks incoming parity bits and the frames themselves, for validity. It is capable of detecting errors in:

- parity
- overrun
- framing.

Overrun errors occur when incoming characters enter the buffer faster than they can be cleared. This often happens when the bit rate (clock speed) on the transmitting device does not match that of the receiver. Framing errors occur when a stop bit is not received after the control logic has counted the correct number of data bits within a frame.

The control logic circuitry of the UART has a number of internal registers that contain all the relevant status information for the device. These registers are generally mapped onto the computer's memory and can be accessed from the main CPU. UART overrun, parity and framing errors do not halt the operation of the device. These errors are instead flagged within the various status registers of the UART and it is the responsibility of the CPU to act upon them.

Most UARTs are designed to "assert" an interrupt line when data has been received. This enables the CPU to run an Interrupt Service Routine (ISR) that removes data from the volatile registers and buffers in the UART and places it into a more stable area of general purpose computer memory.

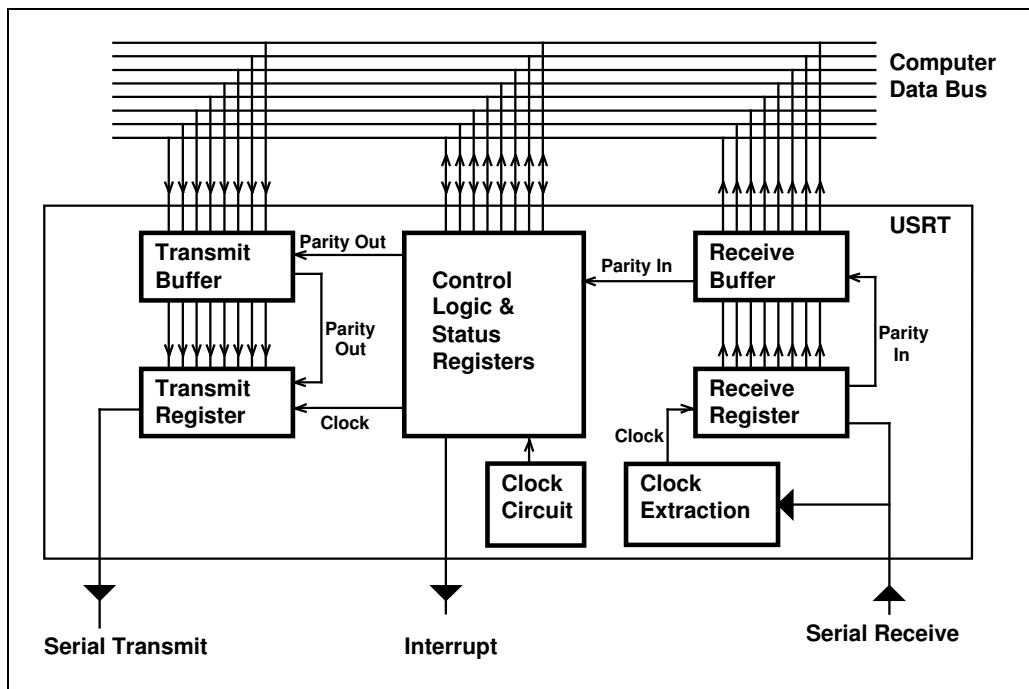
Another important feature of the control circuitry within a UART is the mode control register. By transferring an appropriate bit pattern from the computer's CPU into this register, the UART can be programmed for a number of different parity schemes, including:

- odd parity
- even parity
- mark parity (always 1)
- space parity (always 0).

The number of stop bits (1 or 2) and the number of data bits within a frame (5,6,7 or 8) can also be programmed through the status register. Bit rates in the order of 110 bps to 19200 bps can be provided by the control logic, which scales the clock signal supplied by the external clock circuit.

For synchronous transmission schemes the USRT is used for conversion between serial and parallel data formats. Its schematic is shown in Figure 4.31. In principle, the USRT is similar to the UART, except that the incoming data is clocked into the receive register with a clock signal that is derived (extracted) from the data itself.

From a user point of view, the key point to note about both the UART and the USRT devices is that they perform hardware checking of incoming data. Some of the hardware checking cannot normally be by-passed. This is of importance because it means that if these devices are not "programmed" (set-up) correctly by the CPU (ie: user-program), through use of the UART/USRT mode control registers, then they will continually flag and report hardware errors back to the CPU. These hardware errors and subsequent mismatches in framing, cause incoming data to be interpreted incorrectly by user software.



*Figure 4.31 - Schematic of USRT*

The "programming" of a USRT or UART involves matching its parameters to those of the device at the other end of the transmission line. These parameters are entered as data in special control registers in each device. This fundamental step of setting and matching:

- data bits within a frame
- bit rate
- parity
- stop bits within a frame (UART only)

must always be carried out as a first priority.

Without a matching set of these key parameters on transmitting and receiving devices, there can be no sensible interpretation of the data transferred on a serial link.

