

Serial Communication Standards

This chapter discusses the main physical interface standards associated with data communications for instrumentation and control systems. It includes information on balanced and unbalanced transmission lines, current loops, and serial interface converters.

Objectives

When you have completed studying this chapter you will be able to:

- List and explain the function of the important standards organizations
- Describe and compare the serial data communications interface standards:
 - RS-232
 - RS-449
 - RS-423
 - RS-422
 - RS-485
 - RS/TIA-530A
 - RS/TIA-562
- Explain troubleshooting in serial data communication circuits
- Describe commonly used serial interface techniques:
 - 20 mA current loop
 - Serial interface converters
 - Interface to serial printers
- Describe the most important parallel data communication interface standards:
 - General purpose interface bus
 - Centronics

3.1 Standards organizations

There are seven major organizations worldwide involved in drawing up standards or recommendations, which affect data communications. These are:

- ISO: International Standards Organization
- ITU-T: International Telecommunications Union (ITU formerly CCITT)
- IEEE: Institute of Electrical and Electronic Engineers
- IEC: International Electrotechnical Commission
- RS: Electronic Industries Association
- ANSI: American National Standards Institute
- TIA: Telecommunication Industries Association

ANSI is the principal standards body in the USA and is that country's member body to the ISO. ANSI is a non-profit, non-governmental body supported by over 1000 trade organizations, professional societies, and companies.

The International Telecommunications Union (ITU) is a specialist agency of the United Nations Organization (UNO). It consists of representatives from the Postal, Telephony, and Telegraphy organizations (PTTs), common carriers and manufacturers of telecommunications equipment. In Europe, administrations tend to follow the ITU defined recommendations closely. Although the US manufacturers did not recognize them in the past, they are increasingly conforming to ITU recommendations.

The ITU defines a complete range of standards for interconnecting telecommunications equipment. The standards for data communications equipment are generally defined by the ITU-T 'V' series recommendations.

The two ITU-T physical interface standards are:

- V.24: equivalent to RS-232 for low speed asynchronous serial circuits
- V.35: equivalent to RS-449 for wide bandwidth circuits

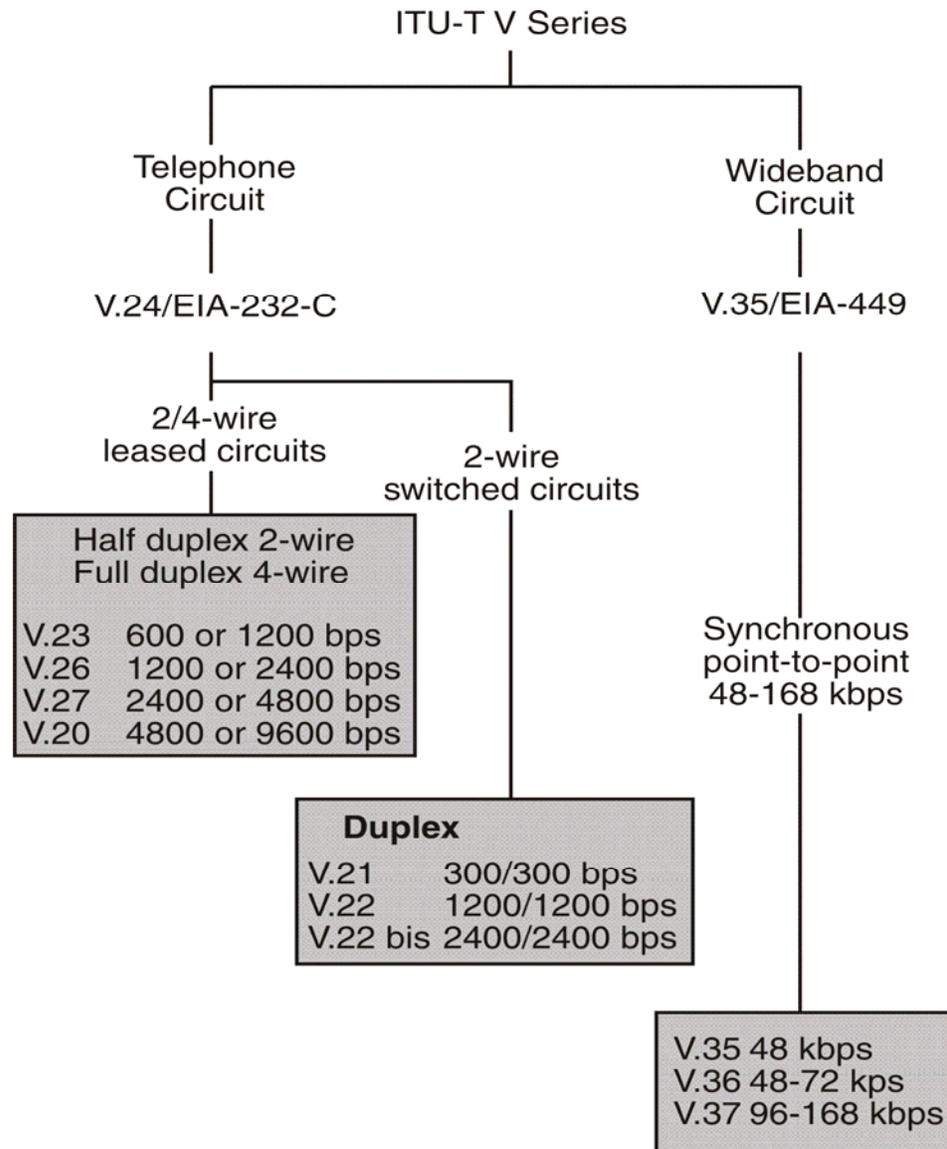


Figure 3.1
ITU-T V series

The RS is a voluntary standards organization in the USA, specializing in the electrical and functional characteristics of interface equipment. It mainly represents the manufacturers of electronic equipment. Since the RS and the TIA merger in 1988, the TIA represents the telecommunications sector of the RS and its initials appear on certain RS standard documents.

The IEC is an international standards body, affiliated to ISO. It concentrates on electrical standards. The IEC developed in Europe and is used by most Western countries, except the USA or those countries closely affiliated with the USA.

The IEEE is a professional society for electrical engineers in the USA and issues its own standards and codes of practice. The IEEE is a member of ANSI and ISO.

The ISO draws members from all countries of the world and concentrates on coordination of standards internationally.

3.2 Serial data communications interface standards

An interface standard defines the electrical and mechanical details that allow equipment from different manufacturers to be connected and able to communicate.

The RS have produced several well known data interface standards, which will be discussed in this chapter. They are:

- RS-232 and revisions
- RS-449
- RS-423
- RS-422
- RS-485
- RS/TIA-530A
- RS/TIA-562

Specific interfacing techniques discussed here also include:

- The 20 mA current loop
- Serial interface converters
- Interface to serial printers

3.3 Balanced and unbalanced transmission lines

The choice between unbalanced and balanced transmission lines is an important consideration when selecting a data communications system.

Unbalanced transmission

In an unbalanced system, the signal common reference conductor is simultaneously shared by many signals and other electronic circuitry. Only one wire carries the signal voltage, which is referenced to a signal common wire, sometimes called the signal ground. The transmitted signal is the voltage between the signal conductor and the common reference conductor.

Theoretically, unbalanced transmission should work well if the signal currents are small and the common conductor has very low impedance. In practice, unbalanced systems only work over short communication links. The signal common conductor has characteristics similar to other conductors (resistance, inductance and capacitance) and is not a perfect reference point. For long communication distances, the common conductor does not have the same zero voltage at all points along its length or at its ends. The common conductor can also pick up noise and have other voltages superimposed on it. Sometimes the shield conductor is used as the common reference wire. This practice can introduce excessively high noise-levels and should be avoided. Unbalanced transmission is used in the RS-232 and RS-423 interfaces.

The fact that the common reference conductor may carry superimposed interference voltages means that the voltages V_1 , V_2 , and V_3 measured at the receiver will be affected (Figure 3.2).

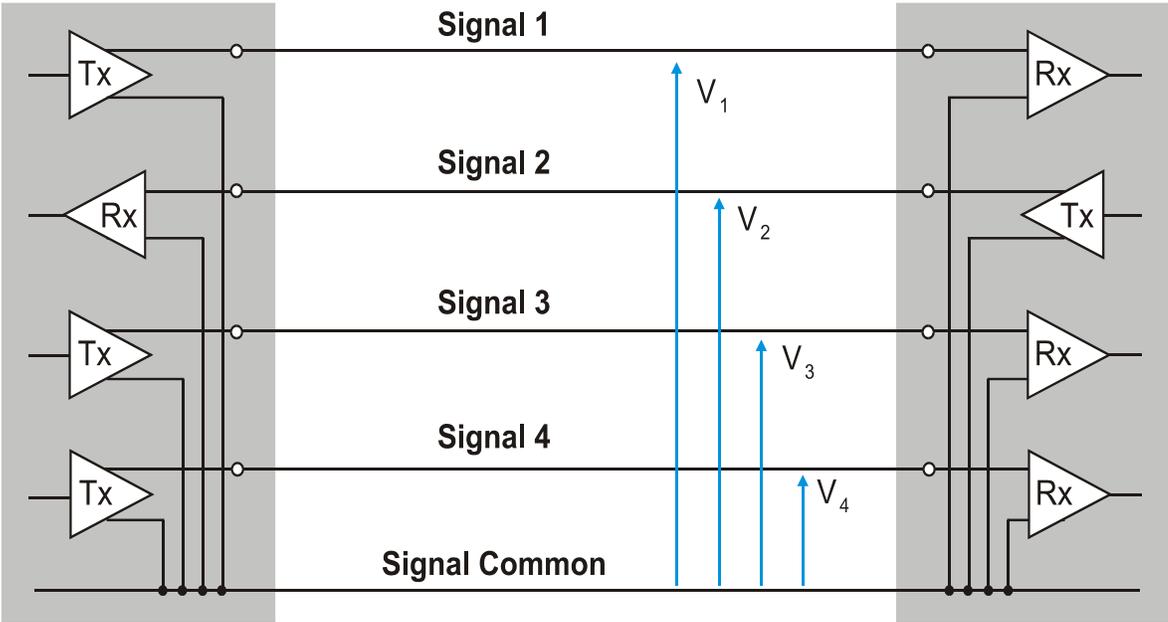


Figure 3.2
Data communication with unbalanced interfaces

Balanced transmission

Balanced communication interfaces require two conductors to transmit each signal. The voltage at the receiving end is measured as the voltage difference between these two wires. This is known as a balanced or differential system. This eliminates many of the interference problems associated with the common reference wire.

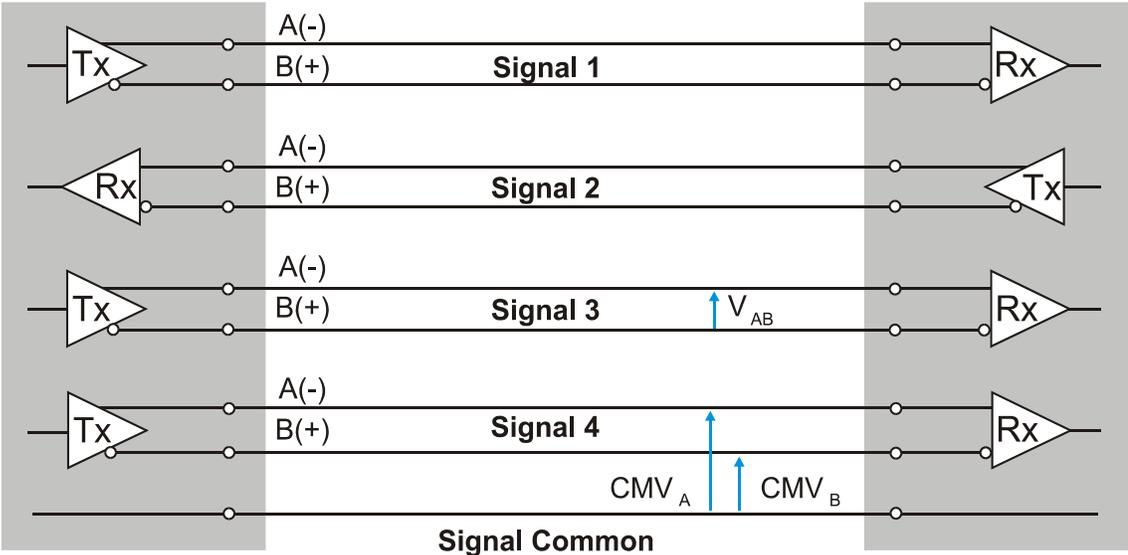


Figure 3.3
Data communications with balanced interfaces

The balanced transmission line permits a higher rate of data transfer over longer distances. The differential method of data transfer is preferable in industrial applications where noise can be a major problem. The disadvantage is that a balanced system requires two conductors for every signal.

The successful transfer of voltage signals across two conductors in the presence of, say noise or voltage drops is based on the assumption that the conductors have similar characteristics and will be affected equally. It does not mean that noise does not exist in the balanced differential system. The voltages on both conductors should rise and fall together, and the differential voltage should remain the same. The voltage between the signal conductor and the common reference conductor is called the common mode voltage (CMV). The CMV is an indication of the induced voltage or noise on the communication link. Ideally, the CMV on the two wires will cancel out completely. However, the greater the CMV, the higher the likelihood of output voltage distortion and damage to the device.

The receiver circuitry of a 2-wire differential system is designed to ignore or reject the CMV, using a technique called common mode rejection (CMR). The effect of noise on the signal is measured as the ratio of the voltage after passing through the receiver to the CMV. The success of the receiver in rejecting the noise is measured as the common mode rejection ratio (CMRR).

$$\text{CMRR(dB)} = 20 \log \left[\frac{V_{\text{OUT}}}{V_{\text{CM}}} \right]$$

Balanced transmission is used in most of the fast interfaces such as RS-422 and RS-485.

3.4 EIA/TIA-232 (RS-232) interface standard (CCITT V.24 interface standard)

The RS-232 interface standard was developed for the single purpose of the interface between data terminal equipment (DTE) and data circuit terminating equipment (DCE) employing serial binary data interchange. In particular, RS-232 was developed for interfacing data terminals to modems.

The RS-232-C interface standard was issued in the USA in 1969. The prefix 'RS' (Recommended Standard), was superseded by 'EIA/TIA' in 1988. The current revision is EIA/TIA-232E (1991), which brings it into line with the international standards ITU V.24, ITU V.28, and ISO-2110. The common convention is to call all revisions of the EIA/TIA 232 standards as EIA-232, as they are effectively functionally equivalent. Only where the differences between specific versions are being discussed, will the version letters be added.

Poor interpretation of RS-232 has been responsible for many problems in interfacing equipment from different manufacturers. This had led some users to dispute whether it is a 'standard'. It should be emphasized that RS-232 and other related RS standards define the electrical and mechanical details of the interface and do not define a protocol.

The RS-232 interface standard specifies the method of connection of two devices – the DTE and DCE.

- DTE:** Data terminal equipment, for example, a computer or a printer. A DTE device communicates with a DCE device. A DTE device transmits data on pin 2 and receives data on pin 3.
- DCE:** Data communications equipment, for example a modem, now also called data circuit-terminating equipment in RS-232E. A DCE device receives data from the DTE and retransmits via another data communications link, such as the telephone system. A DCE device transmits data on pin 3 and receives data on pin 2.

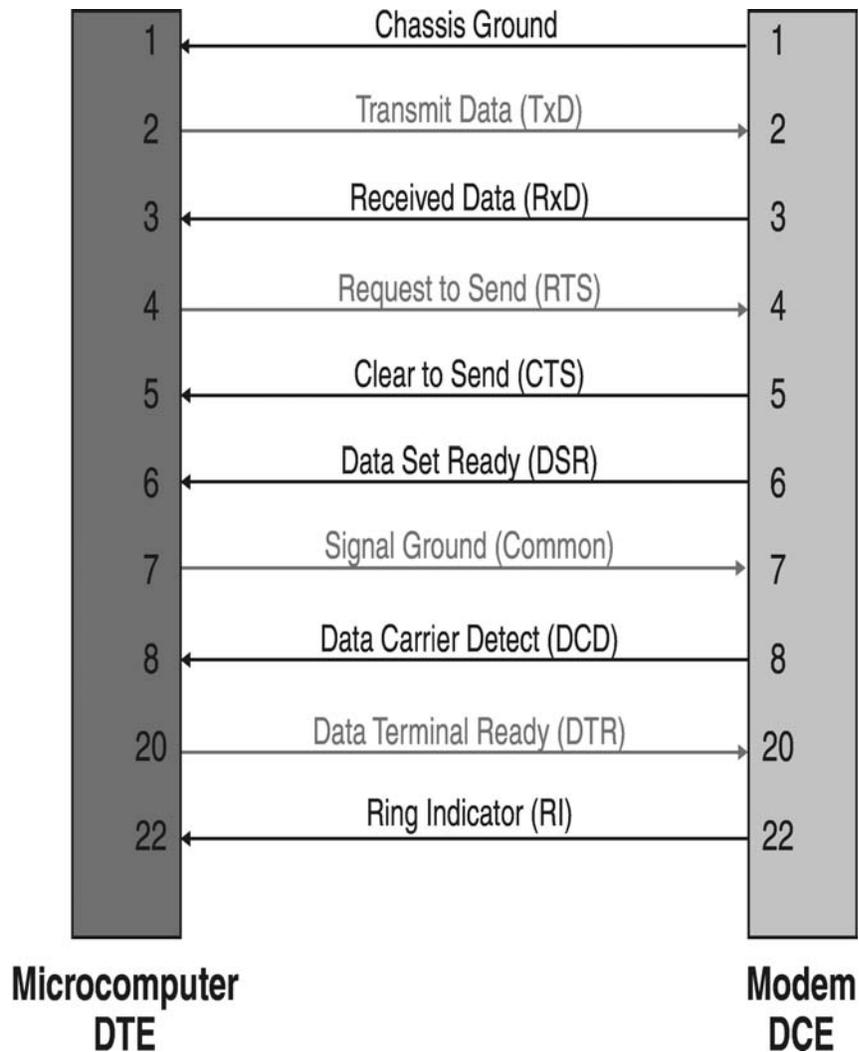


Figure 3.4
Connections between the DTE and the DCE

The major elements of RS-232

The RS-232 standard consists of three major parts, which define:

- Electrical signal characteristics
- Mechanical characteristics of the interface
- Functional description of the interchange circuits

Electrical signal characteristics

RS-232 defines electrical signal characteristics such as the voltage levels and grounding characteristics of the interchange signals and associated circuitry for an unbalanced system.

The RS-232 transmitter is required to produce voltages in the range:

- Logic 1: -5 V to -25 V
- Logic 0: $+5\text{ V}$ to $+25\text{ V}$
- Undefined logic level: $+5\text{ V}$ to -5 V

At the RS-232 receiver the following voltage levels are defined:

- Logic 1: -3 V to -25 V
- Logic 0: $+3\text{ V}$ to $+25\text{ V}$
- Undefined logic level: -3 V to $+3\text{ V}$

Note: The RS-232 transmitter requires the slightly higher voltage to overcome voltage drop along the line.

The voltage levels associated with a microprocessor are 0 V to $+5\text{ V}$ for transistor-transistor Logic (TTL). A line driver is required at the transmitting end to adjust the voltage to the correct level for the communications link. Similarly, at the receiving end a line receiver is required to translate the voltage on the communications link to the correct voltages for interfacing to the microprocessor.

Modern PC power supplies usually have a standard $+12\text{ V}$ output that could be used for the line driver.

The control, or 'handshaking', lines have the same range of voltages as transmission of logic 0 and logic 1, except that they are of opposite polarity. This means that:

- A control line asserted or made active by the transmitting device has a voltage range of $+5\text{ V}$ to $+25\text{ V}$. The receiving device connected to this control line is allowed a voltage range of $+3\text{ V}$ to $+25\text{ V}$.
- A control line inhibited or made inactive by the transmitting device has a voltage range of -5 V to -25 V . The receiving device of this control line is allowed a voltage range of -3 V to -25 V .

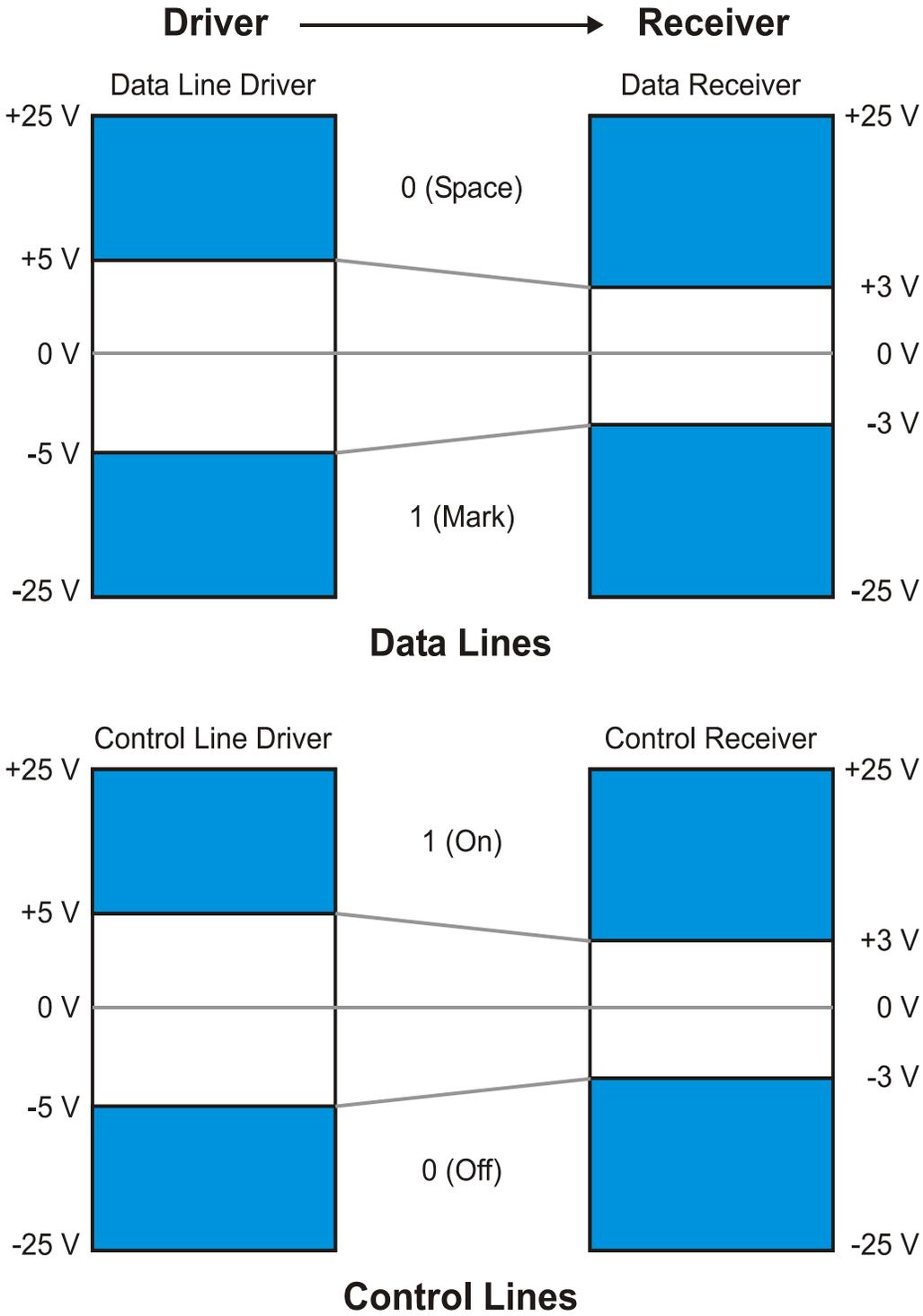


Figure 3.5
Voltage levels for RS-232

At the receiving end, a line receiver is necessary in each data and control line to convert the line voltage levels back to the 0 V and +5 V logic levels required by the internal electronics.

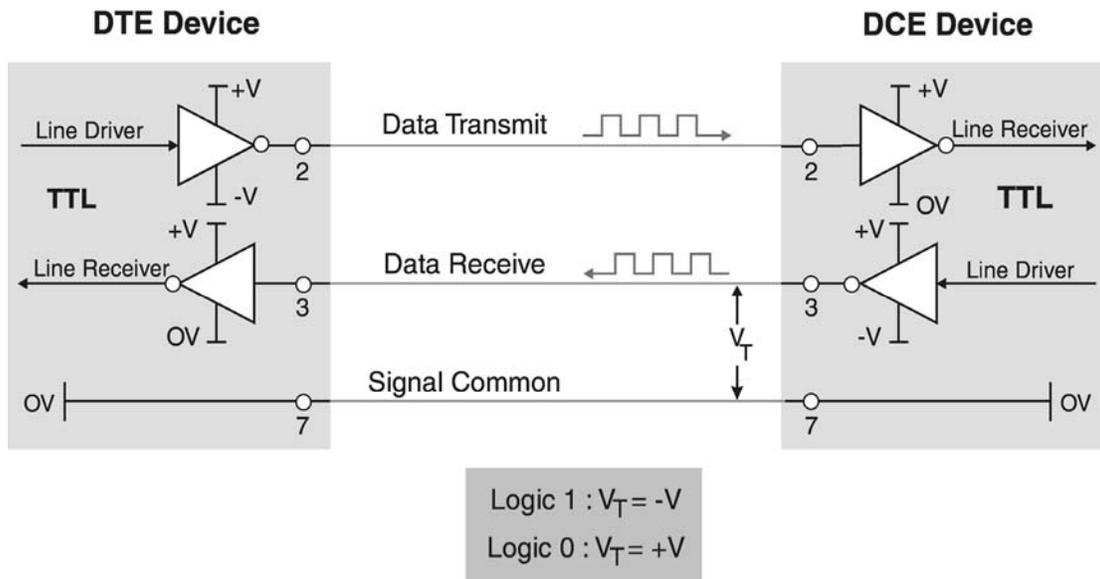


Figure 3.6
RS-232 transmitters and receivers

The RS-232 standard defines 25 electrical connections. The electrical connections are divided into four groups:

- Data lines
- Control lines
- Timing lines
- Special secondary functions

Data lines are used for the transfer of data. Data flow is designated from the perspective of the DTE interface. The *transmit line*, on which the DTE transmits and the DCE receives, is associated with pin 2 at the DTE end and pin 2 at the DCE end. The receive line, on which the DTE receives, and the DCE transmits, is associated with pin 3 at the DTE end and pin 3 at the DCE end. Pin 7 is the common return line for the transmit and receive data lines. The allocations are illustrated in Table 3.2.

Control lines are used for interactive device control, which is commonly known as hardware *handshaking*. They regulate the way in which data flows across the interface. The four most commonly used control lines are:

- RTS: request to send
- CTS: clear to send
- DCE Ready: or data set ready (DSR in RS-232-C)
- DTE Ready: or data terminal ready (DTR in RS-232-C)

It is important to remember that with the handshaking lines, the enabled state means a positive voltage and the disabled state means a negative voltage.

Hardware handshaking is the cause of most interfacing problems. Manufacturers sometimes omit control lines from their RS-232 equipment or assign unusual applications to them. Consequently, many applications do not use hardware handshaking but instead use only the three data lines (transmit, receive and signal common ground) with some form of software handshaking. The control of data flow is then part of the

application program. Most of the systems encountered in data communications for instrumentation and control use some sort of software-based protocol in preference to hardware handshaking. These protocols are discussed in Chapter 8.

There is a relationship between the allowable speed of data transmission and the length of the cable connecting the two devices on the RS-232 interface. As the speed of data transmission increases, the quality of the signal transition from one voltage level to another, for example from -25 V to $+25\text{ V}$, becomes increasingly dependent on the capacitance and inductance of the cable.

The rate at which voltage can 'slew' from one logic level to another depends mainly on the cable capacitance, and the capacitance increases with cable length. The length of the cable is limited by the number of data errors acceptable during transmission. The RS-232 D&E standard specifies the limit of total cable capacitance to be 2500 pF. With typical cable capacitance having improved from around 160 pF/m to only 50 pF/m, the maximum cable length has extended from around 15 meters (50 feet) to about 50 meters (166 feet).

The common data transmission rates used with RS-232 are 110, 300, 600, 1200, 2400, 4800, 9600 and 19 200 bps. Based on field tests, Table 3.1 shows the practical relationship between selected Baud rates and maximum allowable cable length, indicating that much longer cable lengths are possible at lower Baud rates. Note that the achievable speed depends on the transmitter voltages, cable capacitance (as discussed above) as well as the noise environment.

Baud Rate	Cable Length (metres)
110	850
300	800
600	700
1200	500
2400	200
4800	100
9600	70
19 200	50
115 K	20

Table 3.1
Demonstrated maximum cable lengths with RS-232 interface

Mechanical characteristics of the interface

RS-232 defines the mechanical characteristics of the interface between the DTE and the DCE. This section dictates that the interface must consist of a plug and socket and that the socket will normally be on the DCE. The familiar DB-25 connector is specified together with a smaller 26 pin alternative connector.

Although not specified by RS-232C, the DB-25 connector (25 pin, D-type) is closely associated with RS-232 and became the *de facto* standard with revision D. Revision E formally specifies a new connector in the 26-pin alternative connector (known as the ALT A connector). This connector supports all 25 signals associated with RS-232. ALT A is

physically smaller than the DB-25 and satisfies the demand for a smaller connector suitable for modern computers. Pin 26 is not currently used. On most RS-232 compatible equipment, where little or no handshaking is required, the DB-9 connector (9 pin, D-type) is common. This practice originated when IBM decided to make a combined serial/parallel adapter for the AT personal computer. A small connector format was needed to allow both interfaces to fit onto the back of a standard ISA interface card. Subsequently, the DB-9 connector has also become an industry standard to reduce the wastage of pins. The pin allocations commonly used with the DB-9 and DB-25 connectors for the RS-232 interface are shown in Table 3.2. The pin allocation for the DB-9 connector is not the same as the DB-25 and often traps the unwary.

The data pins of a DB-9 IBM connector are usually allocated as follows:

- Data transmit pin 3
- Data receive pin 2
- Signal common pin 5

Pin No. DTE	DB-9 Connector IBM Assignment	DB-25 Connector EIA-232 Pin Assignment	DB-25 Connector EIA-530 Pin Assignment
1	Received Line Signal	Shield	Shield
2	Received Data	Transmitted Data	Transmitted Data (A)
3	Transmitted Data	Received Data	Received Data (A)
4	DTE Ready	Request to Send	Request to Send (A)
5	Signal/Common Ground	Clear to Send	Clear to Send (A)
6	DCE Ready	DCE Ready	DCE Ready (A)
7	Request to Send	Signal/Common Ground	Signal/Common Ground
8	Clear to Send	Received Line Signal	Received Line Signal (A)
9	Ring Indicator	+Voltage (testing)	Receiver Signal DCE Element Timing (B)
10		-Voltage (testing)	Received Line (B)
11		Unassigned	Transmitter Signal DTE Element Timing (B)
12		Sec Received Line Signal Detector/Data Signal	Transmitter Signal DCE Element Timing
13		Sec Clear to Send	Clear to Send (B)
14		Sec Transmitted Data	Transmitted Data (B)
15		Transmitter Signal DCE Element Timing	Transmitter Signal DCE Element Timing (A)
16		Sec Received Data	Received Data (B)
17		Receiver Signal DCE Element Timing	Receiver Signal DCE Element Timing (A)
18		Local Loopback	Local Loopback
19		Sec Request to Send	Request to Send (B)
20		DTE Ready	DTE Ready (A)
21		Remote Loopback/Signal Quality Detector	Remote Loopback
22		Ring Indicator	DCE Ready (B)
23		Data Signal Rate	DTE Ready (B)
24		Transmit Signal DTE Element Timing	Transmitter Signal DTE Element Timing (A)
25		Test Mode	Test Mode

Table 3.2
Common DB-9 and DB-25 pin assignments for RS-232 and EIA/TIA-530 (often used for RS-422 and RS-485)

Functional description of the interchange circuits

RS-232 defines the function of the data, timing, and control signals used at the interface of the DTE and DCE. However, very few of the definitions in this section are relevant to applications for data communications for instrumentation and control.

The circuit functions are defined with reference to the DTE as follows:

- **Protective ground (Shield)**
The protective ground ensures that the DTE and DCE chassis are at equal potentials. (Remember that this protective ground could cause problems with circulating earth currents.)
- **Transmitted data (TxD)**
This line carries serial data from the DTE to the corresponding pin on the DCE. The line is held at a negative voltage during periods of line idle.
- **Received data (RXD)**
This line carries serial data from the DCE to the corresponding pin on the DTE.
- **Request to send (RTS)**
(RTS) is the request to send hardware control line. This line is placed active (+V) when the DTE requests permission to send data. The DCE then activates (+V) the CTS (clear to send) for hardware data flow control.
- **Clear to send (CTS)**
When a half duplex modem is receiving, the DTE keeps RTS inhibited. When it is the DTE's turn to transmit, it advises the modem by asserting the RTS pin. When the modem asserts the CTS, it informs the DTE that it is now safe to send data.
- **DCE ready**
Formerly called data set ready (DSR) – the DTE Ready line is an indication from the DCE to the DTE that the modem is ready.
- **Signal ground (Common)**
This is the common return line for all the data transmit and receive signals and all other circuits in the interface. The connection between the two ends is always made.
- **Data carrier detect (DCD)**
This is also called the received line signal detector. It is asserted by the modem when it receives a remote carrier and remains asserted for the duration of the link.
- **DTE ready (data terminal ready)**
Formerly called data terminal ready (DTR) – DTE ready enables, but does not cause, the modem to switch onto the line. In originate mode, DTE ready must be asserted in order to auto dial. In answer mode, DTE ready must be asserted to auto answer.
- **Ring indicator**
This pin is asserted during a ring voltage on the line.
- **Data signal rate selector (DSRS)**
When two data rates are possible, the higher is selected by asserting DSRS, however, this line is not used much these days.

Pin No.	CCITT No.	Circuit	Description	Circuit Direction
1	-	-	Shield	To DCE
2	103	BA	Transmitted Data	To DCE
3	104	BB	Received Data	From DCE
4	105/133	CA/CJ	Request to Send/Ready for Receiving	To DCE
5	106	CB	Clear to Send	From DCE
6	107	CC	DCE Ready	From DCE
7	102	AB	Signal Common	-
8	109	CF	Received Line Signal Detector	From DCE
9	-	-	Reserved for Testing	-
10	-	-	Reserved for Testing	-
11	126	See Note	Unassigned	-
12	122/112	SCF/CI	Secondary Received Line Signal Detector/ Data Signal Rate Selector	From DCE
13	121	SCB	Secondary Clear to Send	From DCE
14	118	SBA	Secondary Transmitted Data	To DCE
15	114	DB	Transmitter Signal Element Timing (DTE Source)	From DCE
16	119	SBB	Secondary Received Data	From DCE
18	141	LL	Local Loopback	To DCE
19	120	SCA	Secondary Request to Send	To DCE
20	108/112	CD	DTE Ready	To DCE
21	140/110	RL/CG	Remote Loopback/Signal Quality Detector	T/F DCE
22	125	CE	Ring Indicator	From DCE
23	111/112	CH/CI	Data Signal Rate Selector (DTE/DCE Source)	T/F DCE
24	113	DA	Transmit Signal Element Timing (DTE Source)	To DCE
25	142	TM	Test Mode	From DCE
26		None	(Alt A Connector) No Connection at this Time	

Table 3.3
ITU-T V24 pin assignment (ISO 2110)

The half duplex operation of the RS-232 interface

The following description of one particular operation of the RS-232 interface is based on a half duplex data interchange. The description encompasses the more generally used full duplex operation.

Figure 3.7 shows the operation with the initiating user terminal, DTE, and its associated modem, DCE, on the left of the diagram, the remote computer, and its modem on the right.

The following sequence of steps occurs when a user sends information over a telephone link to a remote modem and computer.

- The initiating user manually dials the number of the remote computer.
- The receiving modem asserts the ring indicator line (RI) in a pulsed ON/OFF fashion reflecting the ringing tone. The remote computer already has its data terminal ready (DTR) line asserted to indicate that it is ready to receive calls. Alternatively, the remote computer may assert the DTR line after a few rings. The remote computer then sets its request to send (RTS) line to ON.
- The receiving modem answers the phone and transmits a carrier signal to the initiating end. It asserts the DCE Ready line after a few seconds.
- The initiating modem asserts the data carrier detect (DCD) line. The initiating terminal asserts its DTR, if it is not already high. The modem responds by asserting its DTE ready line.
- The receiving modem asserts its clear to send (CTS) line, which permits the transfer of data from the remote computer to the initiating side.
- Data is transferred from the receiving DTE (transmitted data) to the receiving modem. The receiving remote computer then transmits a short message to indicate to the originating terminal that it can proceed with the data transfer. The originating modem transmits the data to the originating terminal.
- The receiving terminal sets its request to send (RTS) line to OFF. The receiving modem then sets its clear to send (CTS) line to OFF.
- The receiving modem switches its carrier signal OFF.
- The originating terminal detects that the data carrier detect (DCD) signal has been switched OFF on the originating modem and switches its RTS line to the ON state. The originating modem indicates that transmission can proceed by setting its CTS line to ON.
- Transmission of data proceeds from the originating terminal to the remote computer.
- When the interchange is complete, both carriers are switched OFF and in many cases the DTR is set to OFF. This means that the CTS, RTS, and DCE Ready lines are set to OFF.

Full duplex operation requires that transmission and reception occur simultaneously. In this case, there is no RTS/CTS interaction at either end. The RTS line and CTS line are left ON with a carrier to the remote computer.

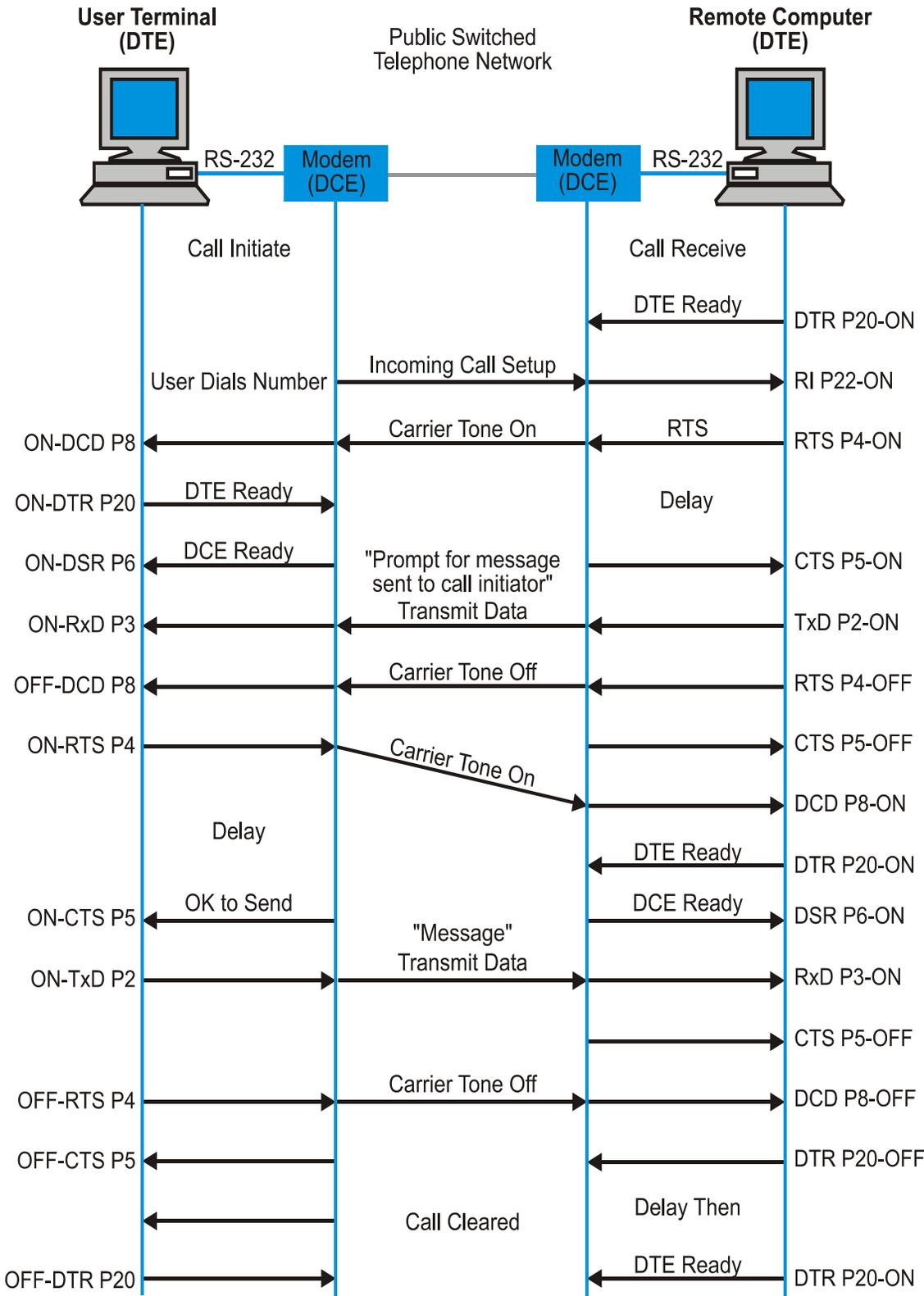


Figure 3.7
Half duplex operational sequence of RS-232

Summary of the RS/TIA-232 revisions

A summary of the main differences between RS-232 revisions, C, D, and E are discussed below.

Revision D (RS-232D)

The 25-pin D type connector was formally specified. In revision C, reference was made to the D type connector in the appendices and a disclaimer was included revealing that it was not intended to be part of the standard, however it was treated as the *de-facto* standard.

The voltage ranges for the control and data signals were extended to a maximum limit of 25 volts from the previously specified 15 volts in revision C.

The 15 meter (50 foot) distance constraint, implicitly imposed to comply with circuit capacitance, was replaced by 'circuit capacitance shall not exceed 2500 pF'. (Standard RS-232 cable has a capacitance of 50 pF/ft.)

Revision E (RS-232E)

Revision E formally specifies the new 26-pin alternative connector, the ALT A connector. This connector supports all 25 signals associated with RS-232, unlike the 9-pin connector, which has become associated with RS-232 in recent years. Pin 26 is currently not used. The technical changes implemented by RS-232E do not present compatibility problems with equipment conforming to previous versions of RS-232.

This revision brings the RS-232 standard into line with international standards CCITT V.24, V.28, and ISO 2110.

Limitations

In spite of its popularity and extensive use, it should be remembered that the RS-232 interface standard was originally developed for interfacing data terminals to modems. In the context of modern requirements, RS-232 has several weaknesses. Most have arisen as a result of the increased requirements for interfacing other devices such as PCs, digital instrumentation, digital variable speed drives, power system monitors and other peripheral devices in industrial plants.

The main limitations of RS-232 when used for the communications of instrumentation and control data in an industrial environment are:

- The point-to-point restriction is a severe limitation when several 'smart' instruments are used
- The distance limitation of 15 meters (50 feet) end to end is too short for most control systems
- The 20 kbps rate is too slow for many applications
- The -3 to -25 V and $+3$ to $+25$ V signal levels are not directly compatible with modern standard power supplies.

Consequently, a number of other interface standards have been developed by the RS, which overcome some of these limitations. The RS-422 and RS-485 interface standards are increasingly being used for instrumentation and control systems.

3.5 Troubleshooting serial data communication circuits

When troubleshooting a serial data communications interface, you need to adopt a logical approach in order to avoid frustration and wasting many hours. A procedure similar to that outlined below is recommended:

- Check the basic parameters. Are the baud rate, stop/start bits, and parity set identically for both devices? These are sometimes set on DIP switches in the device. However, the trend is towards using software, configured from a terminal, to set these basic parameters.
- Identify which is DTE or DCE. Examine the documentation to establish what actually happens at pins 2 and 3 of each device. On the 25-pin DTE device, pin 2 is used for transmission of data and should have a negative voltage (mark) in idle state, whilst pin 3 is used for the receipt of data (passive) and should be at approximately 0 volts. Conversely, at the DCE device, pin 3 should have a negative voltage, whilst pin 2 should be around 0 Volts. If no voltage can be detected on either pin 2 or 3, then the device is probably not RS-232 compatible and could be connected according to another interface standard, such as RS-422, RS-485, etc.

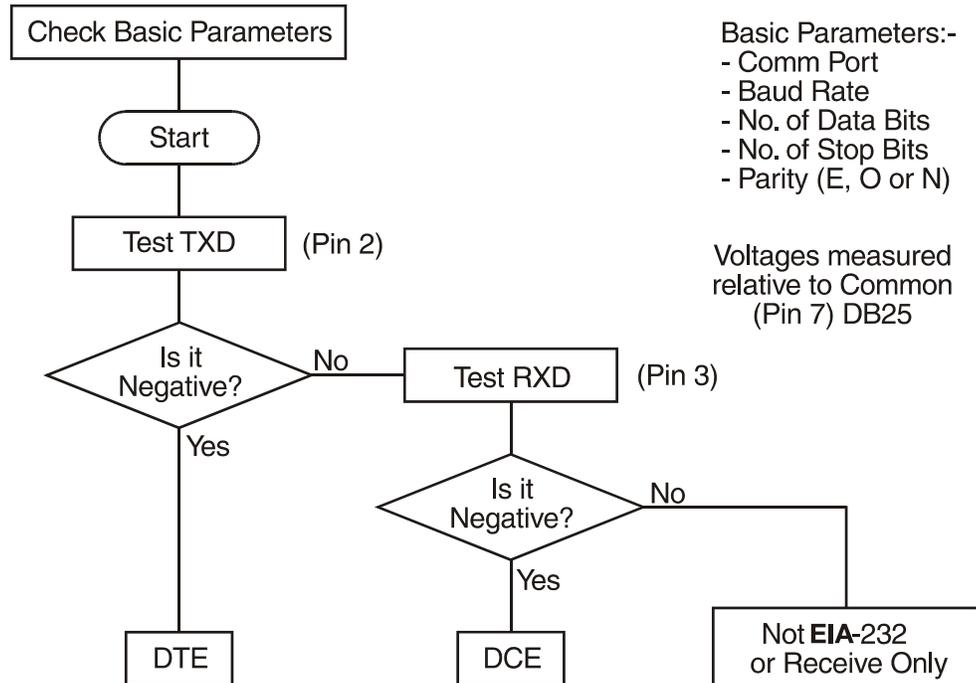


Figure 3.8
Flowchart to identify an RS-232 device as either a DTE or DCE

- Clarify the needs of the hardware handshaking when used. Hardware handshaking can cause the greatest difficulties and the documentation should be carefully studied to yield some clues about the handshaking sequence. Ensure all the required wires are correctly terminated in the cables.

- Check the actual protocol used. This is seldom a problem but, when the above three points do not yield an answer; it is possible that there are irregularities in the protocol structure between the DCE and DTE devices.
- Alternatively, if software handshaking is utilized, make sure both have compatible application software. In particular, check that the same ASCII character is used for XON and XOFF.

3.6 Test equipment

From a testing point of view, section 2.1.2 in the RS-232-E interface standard states that:

‘...The generator on the interchange circuit shall be designed to withstand an open circuit, a short circuit between the conductor carrying that interchange circuit in the interconnecting cable and any other conductor in that cable... including signal ground, without sustaining damage to itself or its associated equipment...’

In other words, any pin may be connected to any other pin, or even earth, without damage and, theoretically, one cannot blow anything up! This does not mean that the RS-232 interface cannot be damaged. The incorrect connection of incompatible external voltages can damage the interface, as can static charges.

If a data communication link is inoperable, the following devices may be useful when analyzing the problem:

- A digital multimeter
Any cable breakage can be detected by measuring the continuity of the cable for each line. The voltages at the pins in active and inactive states can also be ascertained by the multimeter to verify its compatibility to the respective standards.
- An LED
The use of LED is to determine which the asserted lines are or whether the interface conforms to a particular standard. This is laborious and accurate pin descriptions should be available.
- A breakout box
- PC-based protocol analyzer (including software)
- Dedicated protocol analyzer (e.g. Hewlett Packard)

The breakout box

The breakout box is an inexpensive tool that provides most of the information necessary to identify and fix problems on data communications circuits, such as the serial RS-232, RS-422, RS-423, RS-485, etc., interfaces and also on parallel interfaces.

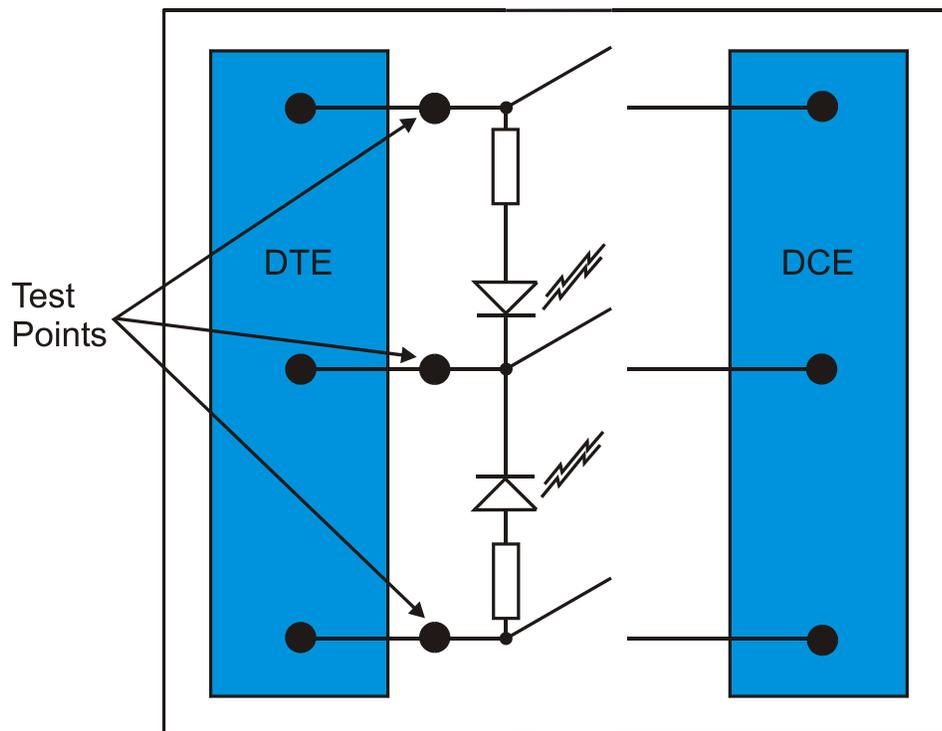


Figure 3.9
Breakout box showing test points

A breakout box is connected into the data cable, to bring out all 25 (or 9, 37, 50, etc.) conductors in the cable to accessible test points. Many versions of this equipment are available on the market, from the 'homemade' using a back-to-back pair of male and female DB-25 sockets, to fairly sophisticated test units with built in LEDs, switches and test points.

Breakout boxes usually have a male and a female socket and by using 2 standard serial cables the box can be connected in series with the communication link. The 25 test points can be monitored by LEDs, a simple digital multimeter, an oscilloscope, or a protocol analyzer. In addition, a switch in each line can be opened or closed while trying to identify where the problem is.

The major weakness of the breakout box is that, while one can interrupt any of the data lines, it does not help much with the interpretation of the flow of bits on the data communication lines. A protocol analyzer is required for this purpose.

Null modem

Null modems look like DB-25 'through' connectors and are used when interfacing two devices of the same gender (e.g. DTE-DTE, DCE-DCE) or devices from different manufacturers with different handshaking requirements. A null modem has appropriate internal connections between handshaking pins that 'trick' the terminal into believing conditions are correct for passing data. A similar result can be achieved by soldering extra loops inside the DB-25 Plug. Null modems generally cause more problems than they cure and should be used with extreme caution and preferably avoided.

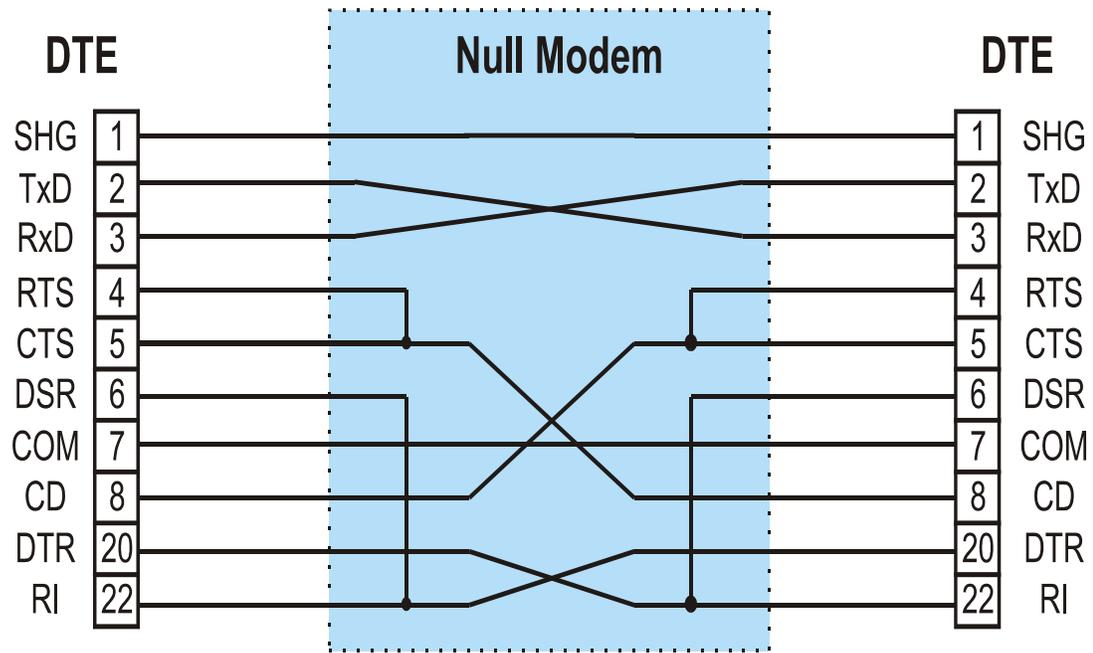


Figure 3.10
Null modem connections

Note that if the null modem connects pin 1 (end-to-end) together, thus providing a combined shield; as in Fig 3.10 – you may end up with a ground loop problem.

Loop back plug

This is a hardware plug, which loops back the transmit data pin to receive data pin and similarly for the hardware handshaking lines. This is another quick way of verifying the operation of the serial interface without connecting to another system.

Protocol analyzer

A protocol analyzer is used to display the actual bits on the data line, as well as the special control codes, such as STX, DLE, LF, CR, etc. The protocol analyzer can be used to monitor the data bits as they are sent down the line and compared with what should be on the line. This helps to confirm that the transmitting terminal is sending the correct data and that the receiving device is receiving it. The protocol analyzer is useful in identifying incorrect setting of baud rate, parity, stop bit, noise or incorrect wiring and connection. It also makes it possible to analyze the format of the message and look for protocol errors.

When the problem has been shown not to be due to the connections, baud rate, bits, or parity, then the content of the message will have to be analyzed for errors or inconsistencies. Protocol analyzers can quickly identify these problems.

Purpose built protocol analyzers are expensive devices and it is often difficult to justify the cost when it is unlikely that the unit will be used very often. Fortunately, software has been developed that enables a normal PC to be used as a protocol analyzer. The use of a PC as a test device for many applications is a growing field, and one way of connecting a PC as a protocol analyzer is shown in Figure 3.11.

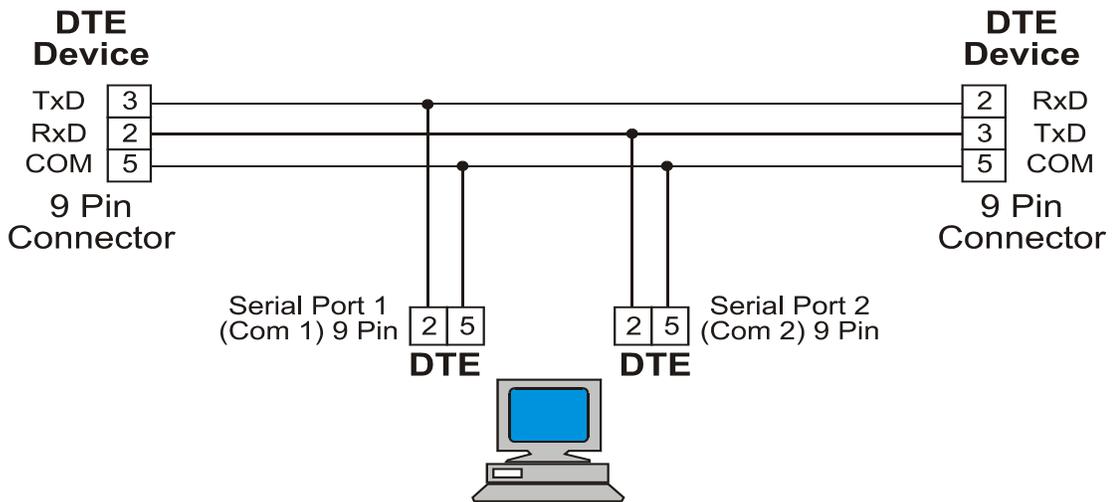


Figure 3.11
Protocol analyzer connection

Typical RS-232 problems

Below is a list of typical RS-232 problems, which can arise because of inadequate interfacing. These problems could equally apply to two PCs connected to each other or to a PC connected to a printer.

Problem	Probable Cause of Problem
Garbled or lost data	Baud rates of both connecting ports may be different
	Connecting cables could be defective
	Data Formats may be inconsistent (Stop Bit/Parity/No of data bits)
	Flow control may be inadequate
	High error rate due to electrical interference
	Buffer size of receiver is inadequate
First characters garbled	The receiving port may not be able to respond quickly enough. Precede the first few characters with the ASCII (DEL) code to ensure frame synchronization.
No data communications	Power for both devices may not be ON
	Transmit and receive lines of cabling may be incorrect
	Handshaking lines of cabling may be incorrectly connected
	Baud rates for both ports may not match
	Data format may be inconsistent
	Earth loop may have formed for EIA-232 line
	Extremely high error rate due to electrical interference for transmitter and receiver
	Protocols may be inconsistent Intermittent communications
	Intermittent interference on cable
ASCII data has incorrect spacing	There is a mismatch between 'LF' and 'CR' characters generated by transmitting device and for receiving device.

Table 3.4
A list of typical RS-232 problems

3.7 RS-449 interface standard (November 1977)

RS-449 was the intended successor to the functional portion of RS-232. It defined a mechanical specification for plugs and sockets based on a 37-pin and 9-pin assembly. Apart from its improved speed and distance specification, it also offered a number of enhancements such as the provision for automatic modem testing, new grounding arrangements and a number of new signals. Little support was given to RS-449 because of the popularity of RS-232. One problem was that the 37- and 9-pin connectors were not commonly used in the RS-232 domain. In recent years, RS-449 has gained support from the users of RS-422 and RS-485 whose standards do not specify any mechanical connectors. (Refer to Table 3.5 for pinouts). RS-449 has now been superseded by RS/TIA 530-A.

3.8 RS-423 interface standard

The RS-423 standard defines an unbalanced data communications interface similar to RS-232, but with some improvements. It allows an increase in cable length between devices, improved data transmission rates and multiple receivers on a line. RS-423 permits reliable communication for:

- Distances of up to 1200 meters (4000 feet)
- Data rates of up to 100 kbps
- Only one line driver on a line, but the driver current rating has been increased to permit multiple receivers
- Up to 10 line receivers, with lower current requirements, to be driven by the line driver

The improvements in performance have mainly been achieved by reducing the voltages to

- Logic 1: -3.6 V to -6 V
- Logic 0: $+3.6\text{ V}$ to $+6\text{ V}$

Compared to RS-232, the total voltage slew is reduced by a factor of 4, with 12 V swings compared to 50 V swings. The effect of the line capacitance is reduced which allows faster data rates. Like the RS-232 interface, the data link is unbalanced and requires 3 wires for a full duplex signal path. Figure 3.12 illustrates the transmit connections.

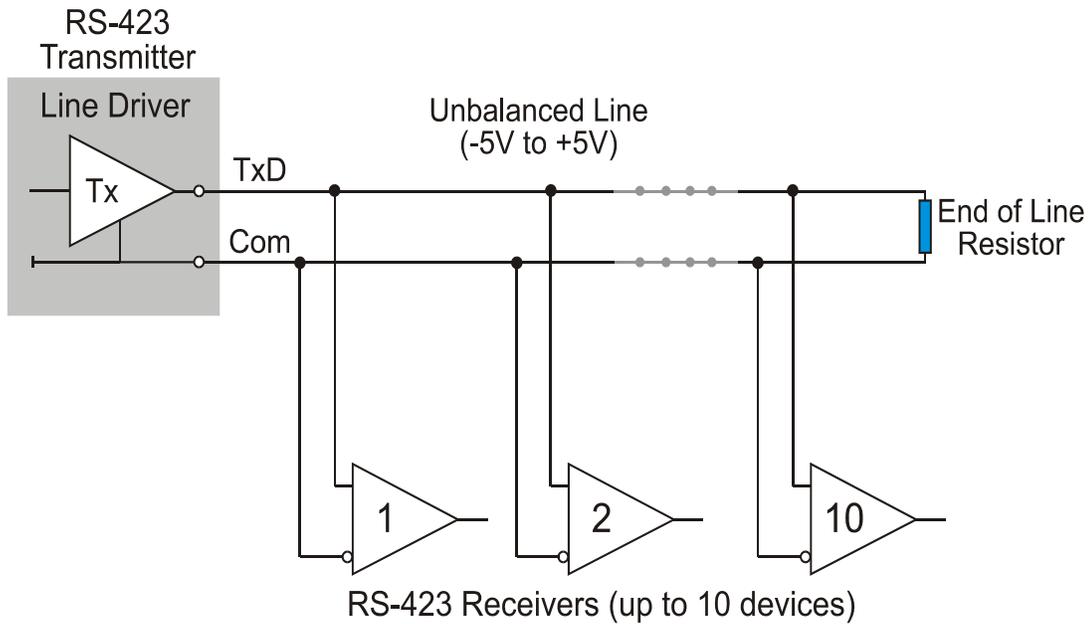


Figure 3.12
The RS-423 unbalanced line driver connection

The ability to support up to 10 receivers is achieved by increasing the current capacity of the line drivers, reducing the current drain of the line receivers, and connecting the receivers in a differential mode.

The RS-423 standard does not specify the mechanical connections or assign pin numbers. It is common to use the DB-25 connector with pin allocations as specified by RS-232.

3.9 The RS-422 interface standard

The RS-422 standard introduced in the early '70s defines a balanced, or differential, data communications interface using two separate wires for each signal. This permits very high data rates and minimizes problems with varying ground potentials because the ground is not used as a voltage reference, as in RS-232 and RS-423. RS-422 is an improvement on RS-423 and allows:

- Data to be communicated at distances of up to 1200 m (4000 feet, similar to RS-423)
- Data rates of up to 10 Mbps (increase of 100 times)
- Only one line driver on a line
- Up to 10 line receivers to be driven by one line driver

The differential voltages between the A and B lines are specified as:

- $-2\text{ V to }-6\text{ V}$ with respect to the B line for a binary 1 (MARK or OFF) state
- $+2\text{ V to }+6\text{ V}$ with respect to the B line for a binary 0 (SPACE or ON) state

The specification refers to the lines as A and B, but there are also called A(-) and B(+), or TX+ and TX-.

The line driver for the RS-422 interface produces a 5 V differential voltage on two wires. These voltage levels allows the transmitters and drivers to be supplied by the 5 V supply, common in today's computers.

Figure 3.13 illustrates the connection of RS-422 devices.

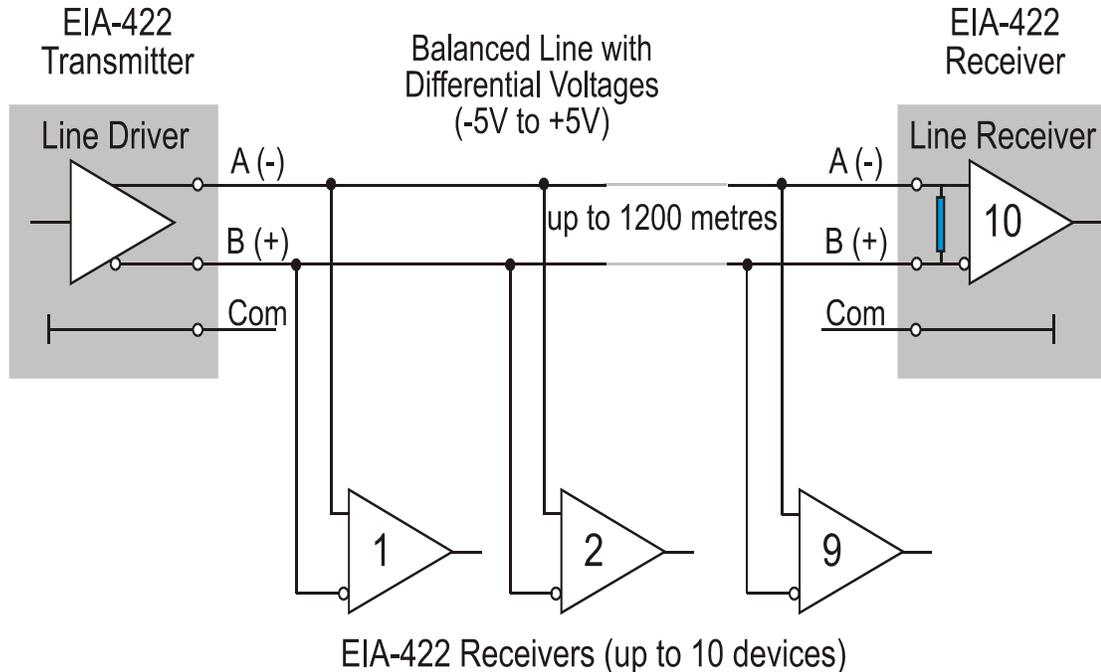


Figure 3.13
The RS-422 balanced line driver connections

As the differential receiver is only sensitive to the difference between two signals on its inputs, common noise signals picked up in both wires will have little effect on the operation of the receiver. Differential receivers are therefore said to have good common mode rejection (CMR) characteristics.

The major feature of the RS-422 standard is the differential voltage signal, which allows an increase in speed and provides higher noise immunity. Each signal is transferred on one pair of wires and is the voltage difference between them. The penalty is that two wires are required for each signal, compared to one wire for RS-232 and RS-423. A common ground wire is preferred to aid noise rejection. Consequently, 3 wires are required for a half duplex, and 5 wires for a full duplex system.

The balanced line driver can also have an input signal called an *enable* signal. The purpose of this signal is to connect the driver to its output terminals, A & B. If the enable signal is off, one can consider the driver as disconnected from the transmission line or in a *high impedance* state. (This *tri-state* approach is discussed under RS-485.)

The differential lines of the RS-422 are normally terminated with a resistor equal to the characteristic impedance (Z_0) of the line. This will prevent signal distortion due to reflections from the end of line. A typical value of Z_0 would be in the order of 120Ω , for a twisted pair line.

The RS-422 standard does not specify mechanical connections or assign pin numbers. It is common practice to use the pin assignment of RS-449 for a DB-37 connector (see Table 3.5).

Pin No	DB-9 Connector Common for EIA-422 & EIA-485	DB-37 Connector EIA-449 Pin Assignment
1	Shield	Shield
2	Send Data	B+
3	Receive Data	B+
4	Request to Send	B+
5	Clear to Send	B+
6	Send Data	A-
7	Receive Data	A-
8	Request to Send	A-
9	Clear to Send	A-
10		Local Loopback
11		Data Mode
12		Terminal Ready
13		Receiver Ready
14		Remote Loopback
15		Incoming Call
16		Select Frequency
17		Terminal Timing
18		Test Mode
19		Signal Ground
20		Receive Common
21		
22		Send Data
23		Send Timing
24		Receive Data
25		Request to Send
26		Receive Timing
27		Clear to Send
28		Terminal in Service
29		Data Mode
30		Terminal Ready
31		Receiver Standby
32		Select Standby
33		Signal Quality
34		New Signal
35		Terminal Timing
36		Standby/Indicator
37		Send Common

Table 3.5

Common DB-9 pin assignments for RS-422 and RS-485 and DB-37 pin assignments specified according to RS-449

3.10 The RS-485 interface standard

The RS-485 standard is the most versatile of the four RS interface standards discussed in this chapter. It is an extension of RS-422 and allows the same distance and data speed but increases the number of transmitters and receivers permitted on the line.

RS-485 permits a 'multidrop' network connection on 2 wires and allows reliable serial data communication for:

- Distances of up to 1200 m (4000 feet, same as RS-422)
- Data rates of up to 10 Mbps (same as RS-422)
- Up to 32 line drivers on the same line
- Up to 32 line receivers on the same line

Note: You can have 32 transceivers on a RS-485 network. If you require more than 32 devices, you would have to use repeaters (which is not defined in the RS-485 standards).

The differential voltages between the A and B lines are specified as:

- -1.5 V to -6 V with respect to the B line for a binary 1 (MARK or OFF) state
- $+1.5\text{ V}$ to $+6\text{ V}$ with respect to the B line for a binary 0 (SPACE or ON) state

The specification refers to the lines as A and B, but there are also called A(-) and B(+), or TX+ and TX-.

As with RS-422, the line driver for the RS-485 interface produces a $\pm 5\text{ V}$ differential voltage on two wires.

The major enhancement of RS-485 is that a line driver can operate in three states called tri-state operation:

- Logic 1
- Logic 0
- High-impedance

In the high impedance state, the line driver draws virtually no current and appears not to be present on the line. This is known as the 'disabled' state and can be initiated by a signal on a control pin on the line driver integrated circuit. Tri-state operation allows a multidrop network connection and up to 32 transmitters can be connected on the same line, although only one can be active at any one time. Each terminal in a multidrop system must be allocated a unique address to avoid conflicting with other devices on the system. RS-485 includes current limiting in cases where contention occurs.

The RS-485 interface standard is very useful for systems where several instruments or controllers may be connected on the same line. Special care must be taken with the software to coordinate which devices on the network can become active. In most cases, a master terminal, such as a PC or computer, controls which transmitter/receiver will be active at any one time.

The 2-wire data transmission line does not normally require special termination unless required by the manufacturer. On long lines, the leading and trailing edges of data pulses will be much sharper if terminating resistors approximately equal to the characteristic impedance (Z_0) of the line are fitted at the extreme ends. This is indicated in Figure 3.14. For twisted pair systems, the resistor used is typically $120\ \Omega$.

Figure 3.14 shows a typical two wire multidrop network. Note that the transmission line is terminated on both ends of the line but not at drop points in the middle of the line. The

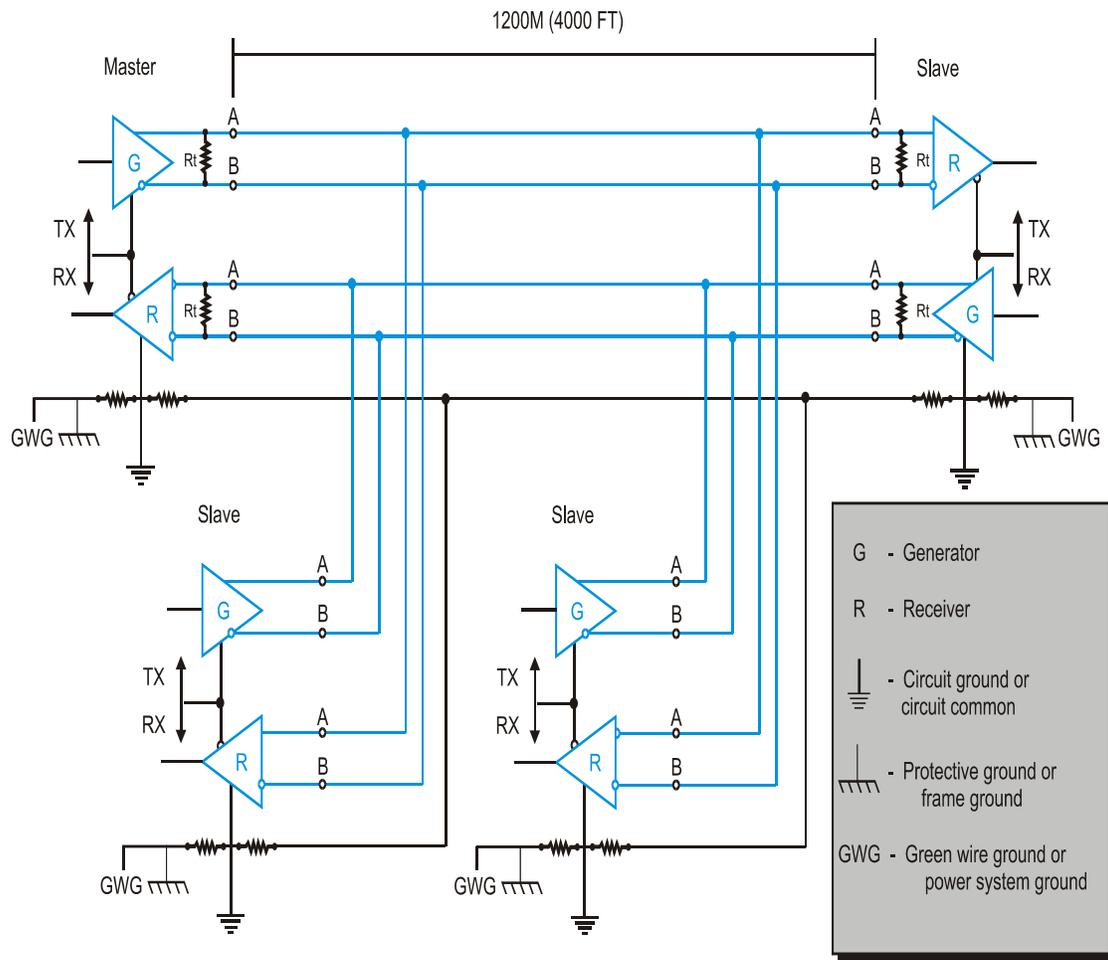


Figure 3.15
Four wire network configuration

During normal operation, there are periods when all RS-485 drivers are off, and the communications lines are in the idle, high impedance state. In this condition the lines are susceptible to noise pick up, which can be interpreted as random characters on the communications line. If a specific RS-485 system has this problem, it should incorporate 10 k Ω bias resistors as indicated in Figure 3.16. These resistors will maintain the data lines in a mark condition (idle) when the system is in the high impedance state.

The ground resistors shown in figure 3.15 are recommended in the specification and should be 100 ohms $\frac{1}{2}$ watt. Their purpose is to reduce any loop currents if the earth potentials are significantly different.

The bias resistors are chosen in such a way that the B line will be kept at least 200 mV HIGHER than the A line with no input signal (i.e. all transmitters in the high impedance state). For the purpose of the calculation, remember that the two 120 Ω terminating resistors appear in parallel for this purpose. This particular example uses bias resistors on only one node.

Some systems employ bias resistors on all nodes, in which case the values of the bias resistors will be significantly higher since they appear in parallel.

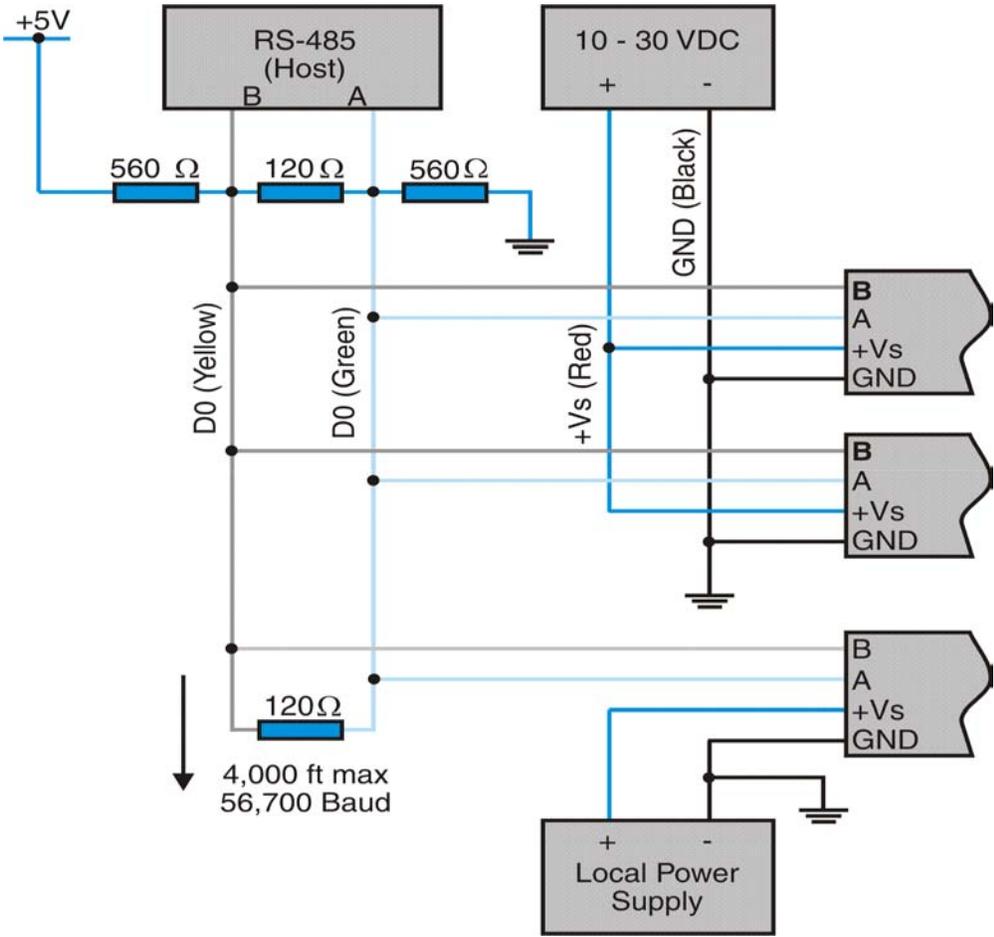


Figure 3.16
Suggested installation of resistors to minimize noise

RS-485 line drivers are designed to handle 32 nodes. This limitation can be overcome by employing an RS-485 repeater connected to the network. When data occurs in either side of the repeater, it is transmitted to the other side. The RS-485 repeater transmits at full voltage levels, consequently another 31 nodes can be connected to the network. A diagram for the use of RS-485 with a bi-directional repeater is given in Figure 3.17.

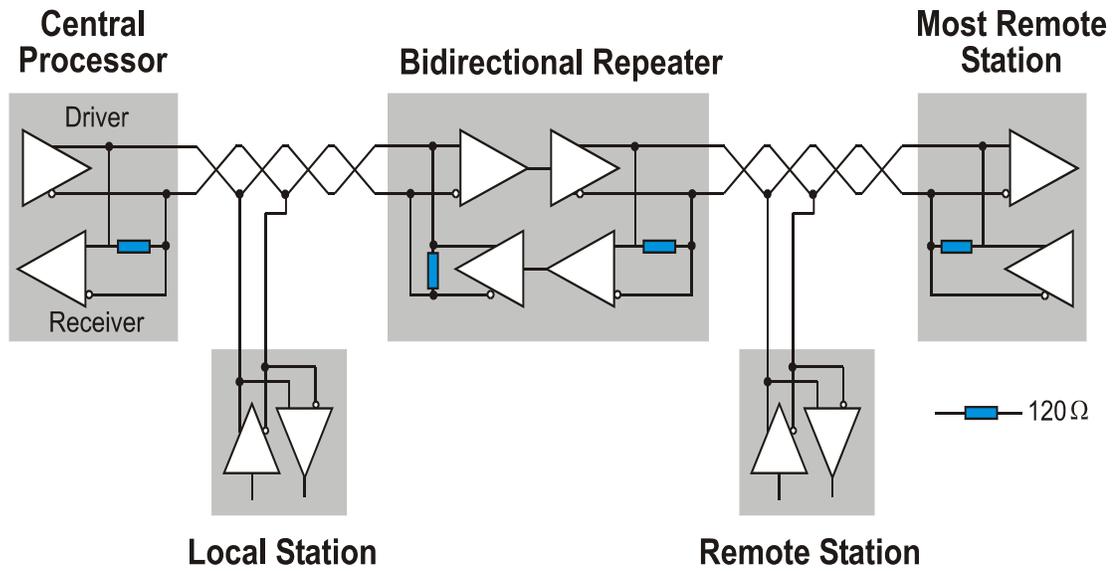


Figure 3.17
RS-485 used with repeaters

The ‘decision threshold’ of the RS-485 receiver is identical to that of both RS-422 & RS-423 receivers at $\pm 200\text{ mV}$ (0.2 V), as indicated in Figure 3.18.

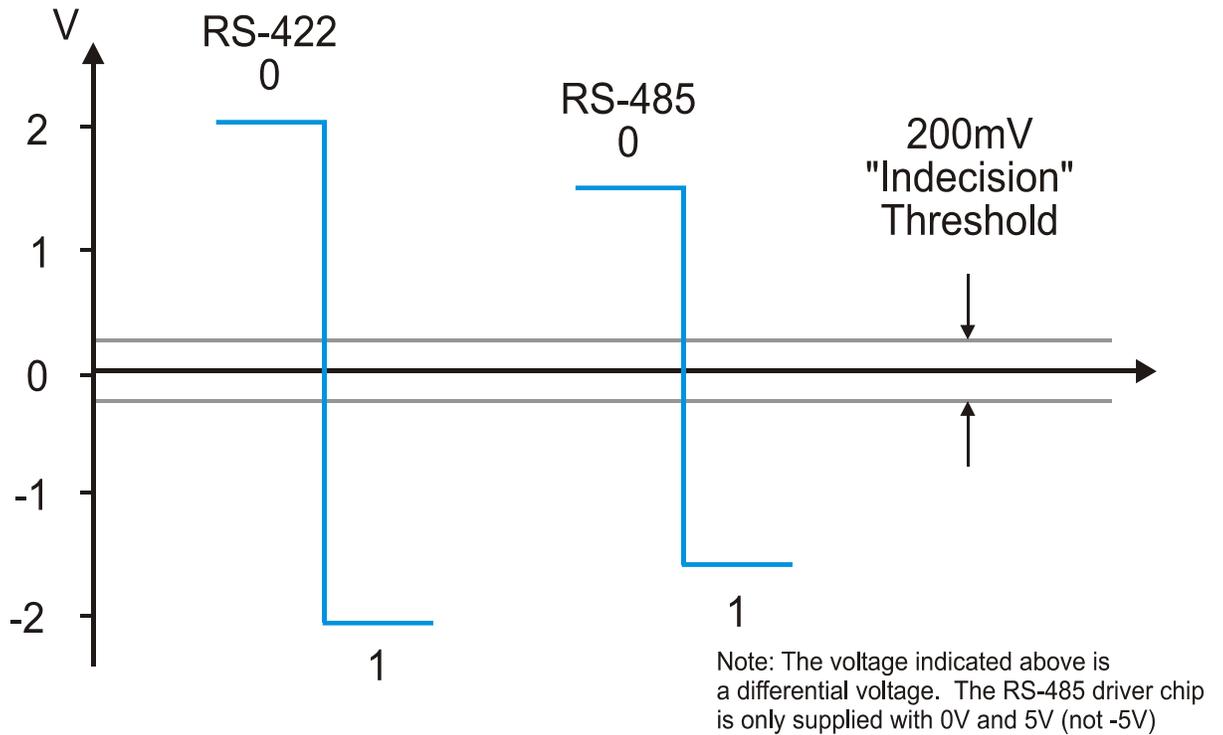


Figure 3.18
RS-485/422 & 423 receiver sensitivities

3.11 Troubleshooting and testing with RS-485

A few suggestions and testing procedures for RS-485 systems are outlined below. Both hardware and software testing will be discussed. There are also some examples of real problems and their solutions.

Hardware test procedure

- Check cabling connections for proper and complete connections. This should be done off-line (no power to the circuit). It maybe helpful to do a continuity check at this point, while the power is off.
- Check the polarity of the connections between each device on the RS-485 network. They should be A to A and B to B. Usually the black wire is B and the white wire is A, but this can be different for the various systems on the market. In most systems, the red wire of the voltmeter is to go on A and the black lead goes on the B. Check for your system by following the manufacturer's instructions.
- Check the voltage of the RS-485 system when the system is in an idle state. The idle state on most RS-485 systems is when the transmitters are not sending any data. Most RS-485 systems bias their transmitters to a minus voltage output (a mark or '1') using the voltage divider network indicated in an earlier discussion. However, this is not always the case. Some systems bias the lines with a positive voltage (a space or a '0') via the software while others turn the transmitters off (and have no voltage divider network connected externally) with a resultant zero voltage. Remember the true output of the RS-485 driver chip is usually an open collector output. Biasing keeps the lines from floating.
- Check that the shield (if there is one) is connected to ground at only one point. Ensure that there are no other connections to earth or ground. This is of course more important if the RS-485 devices are far away from each other. It would be wise if the devices were far away (100 meters/300 feet or more), to visually check the shield at each of the locations. The shields of different pairs must be connected.
- Check that the single shield ground is properly connected to an independent, clean earth ground. A clean ground is an earth ground that is as free as possible from noise.
- Check for common mode voltage problems. This is a difficult problem to detect, as it can be intermittent and sporadic. Usual symptoms may include intermittent failures and/or unusual events such as every other unit failing to communicate. The common mode voltage can be measured from each line to ground using a hand-held voltmeter. These voltages should not approach the maximum voltage of the driver chip.
- Ensure that the lines are terminated with the correct resistance terminators (if needed or required). The manufacturer's requirements must be followed.

Solution

The manufacturer's recommended termination resistors were added, one at each end. The reflections disappeared and the system worked correctly.

Note 1: Termination resistors on an RS-485 system are *not necessarily* required for operation but are preferable to reduce reflections (especially above cable lengths longer than 100 m and baud rates exceeding 40 kbaud).

Note 2: Reducing the baud rate (if possible) is another possible solution.

Note 3: Adding the terminating resistors effectively loads down the RS-485 line drivers. This can cause problems of its own. Do not put a resistor in any location except at the ends of the line. *Never* install a lower resistance than the manufacturer's specifications.

Example 2

A two wire master/slave RS-485 system had errors in the responses. The situation was:

- Master dispatched request to slave
- The slave responded with the information. Unfortunately, the master had given up and sent out another request.
- The problem was the delay in the slave responding.
- The master then recorded this as a communication error

Solution

The timeout for the master, before sending out another request to the slave for information, was extended.

Alternative solution

An increase in the baud rate may have improved the situation slightly (minor fix).

Example 3

A two wire master/slave RS-485 system had errors in the responses. The situation was:

- Master dispatched request to slave.
- Slave responded very quickly.
- Master transmitter lines had not gone into the high impedance state.
- Hence, the response from the slave was 'flattened' by being transmitted into low impedance.
- The master then recorded this as a communication error.

Solution

The master was put into the high impedance state quicker by disabling the RTS line. Obviously, ensure that this does not worsen the situation by disabling the transmitter before it has completed transmitting the signal.

Alternative solution

Slow down the response from the slave so that the master has a chance to go into the high impedance state.

3.12 EIA/TIA-530A interface standard (May 1992)

This standard supersedes RS-449. It is intended that the standard be used for applications where a balanced system is required such as RS-422 and RS-485.

3.13 EIA/TIA-562 interface standard (June 1992)

EIA/TIA-562 supports the new 3.3 V technology, which enables systems to have higher clock speeds, faster data communication rates, lower energy consumption, and to be smaller and lighter. The EIA/TIA-562 standard allows 64 kbps operation compared with the RS-232 maximum limit of 20 kbps.

Typical features of the EIA-562 standard are:

- 64 kbps operation (maximum)

- Stringent wave form specifications (ripple no larger than 5% of voltage swing)
- Maximum slew rate of 30 V/microsecond
- Ability to interface to RS-232

One of the disadvantages of the EIA-562 standard, compared to RS-232 is the reduction of the noise margin from 2 V to 0.7 V.

3.14 Comparison of the EIA interface standards

The main features of the four most common EIA interface standards are compared below:

Transmitter		EIA-232	EIA-423	EIA-422	EIA-485
Mode of operation		Unbalanced	Unbalanced	Differential	Differential
Max No. of Drivers & Receivers on line		1 Driver 1 Receiver	1 Driver 10 Receivers	1 Driver 10 Receivers	32 Drivers 32 Receivers
Recommended cable length		75 m	1,200 m	1,200 m	1,200 m
Maximum Data Rate		20 kbps	100 kbps	10 Mbps	10 Mbps
Maximum Common Mode Voltage		± 25 V	± 6 V	± 6 V to -0.25 V	$+12$ V to -7 V
Driver Output Signal		± 5.0 V min ± 25 V max	± 3.6 V min ± 6.0 V max	± 2.0 V min ± 6.0 V max	± 1.5 V min ± 6.0 V max
Driver Load		> 3 ohm	> 450 ohm	100 ohm	60 ohm
Driver Output Resistance	Power On	n/a	n/a	n/a	100 μ A -7 V $\leq V_{cm}$ ≤ 12 V
(high-Z state)	Power Off	300 ohm	100 μ A @ ± 6 V	100 μ A -0.25 V $\leq V_{cm}$ ≤ 6 V	100 μ A -7 V $\leq V_{cm} \leq$ 12 V
Receiver input resistance		3 kohm to 7 kohm	> 4 kohm	> 4 kohm	> 12 kohm
Receiver sensitivity		± 3.0 V	± 200 mV	± 200 mV -7 Vcm 7 V	± 200 mV -12 V $\leq V_{cm}$ ≤ 12 V

Table 3.6
Comparison of main features of RS-232, RS-423, RS-422, and RS-485

The data signaling rate versus cable length for balanced interface using 24 AWG twisted pair cable is shown in Figure 3.19.

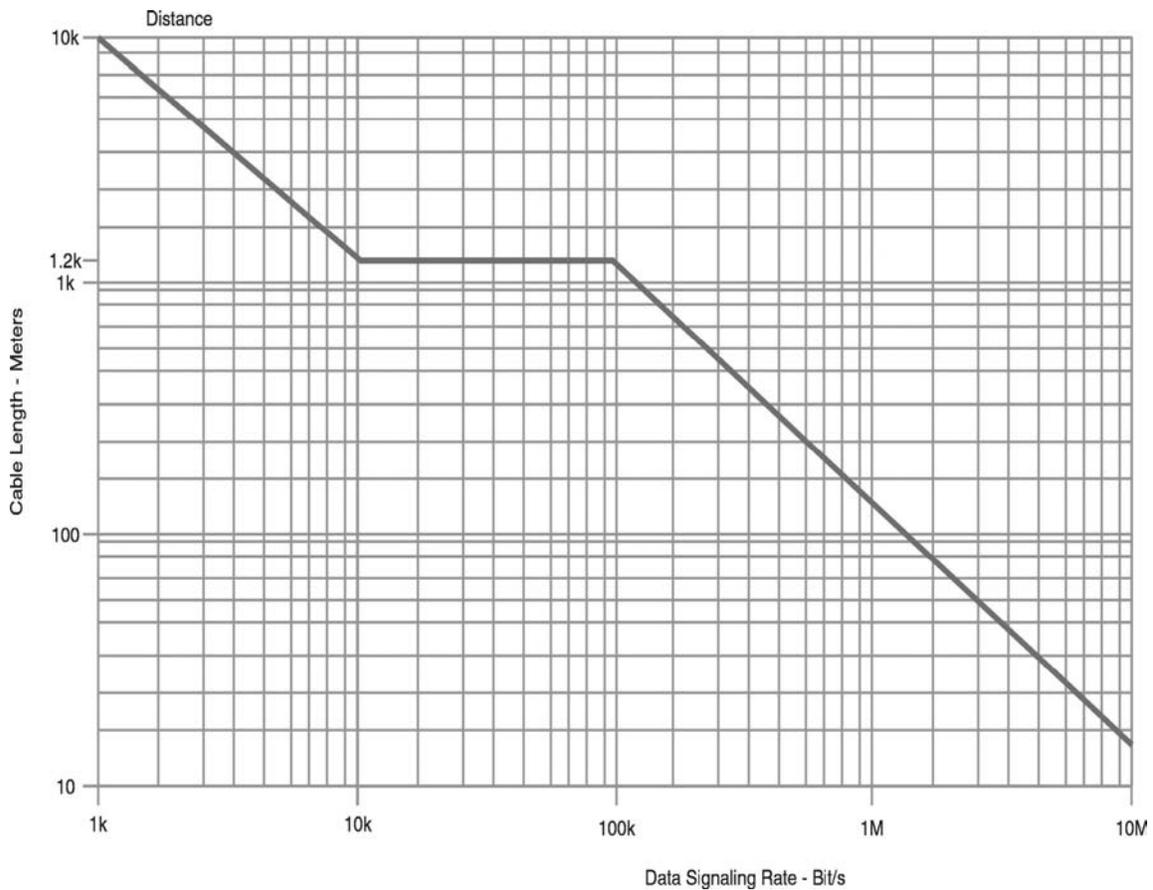


Figure 3.19
Data signaling rate vs. cable length

From a practical point of view, many RS-422/485 systems run up to 5000 meters (16 000 feet) at 1200 bps without any problems.

3.15 The 20 mA current loop

Another commonly used interface technique is the *current loop*. This uses a current signal rather than a voltage signal, employing a separate pair of wires for the transmitter current loop and receiver current loop.

A current level of 20 mA, or up to 60 mA, is used to indicate logic 1 and 0 mA logic 0. The use of a constant current signal enables a far greater separation distance to be achieved than with a standard RS-232 voltage connection. This is due to the higher noise immunity of the 20 mA current loop which can drive long lines of up to 1 km, but at reasonably slow bit rates. Current loops are mainly used between printers and terminals in the industrial environment. Figure 3.20 illustrates the current loop interface.

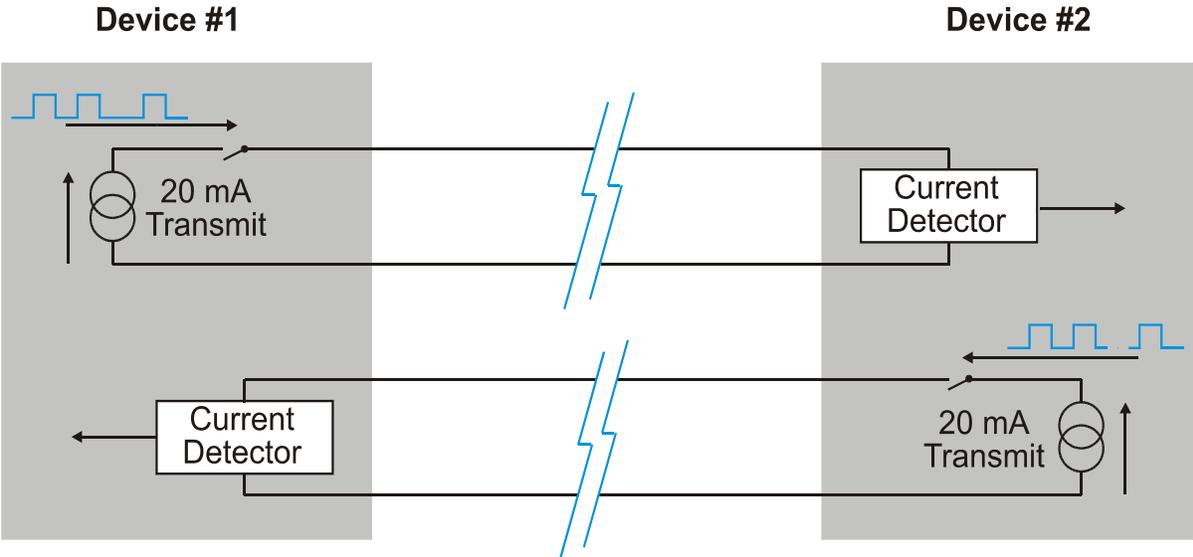


Figure 3.20
The 20 mA current loop interface

3.16 Serial interface converters

Serial interface converters are becoming increasingly important with the move away from RS-232 to industrial standards such as RS-422 and RS-485. Since many industrial devices still use RS-232 ports, it is necessary to use converters to interface a device to other physical interface standards. Interface converters can also be used to increase the effective distance between two RS-232 devices.

The most common converters are:

- RS-232/422
- RS-232/485
- RS-232/current loop

Figure 3.21 is a block diagram of an RS-232/RS-485 converter. Figure 3.22 shows a circuit wiring diagram.

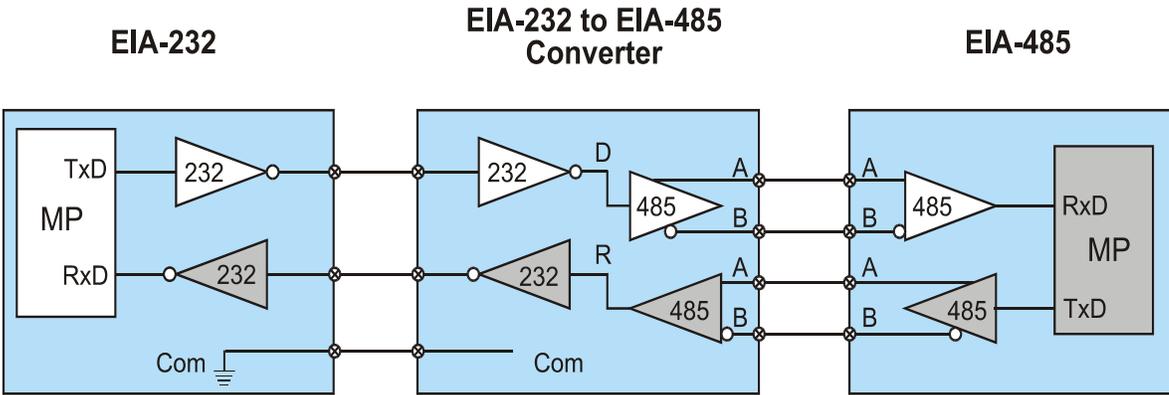


Figure 3.21
Block structure of RS-232/RS-485 converter

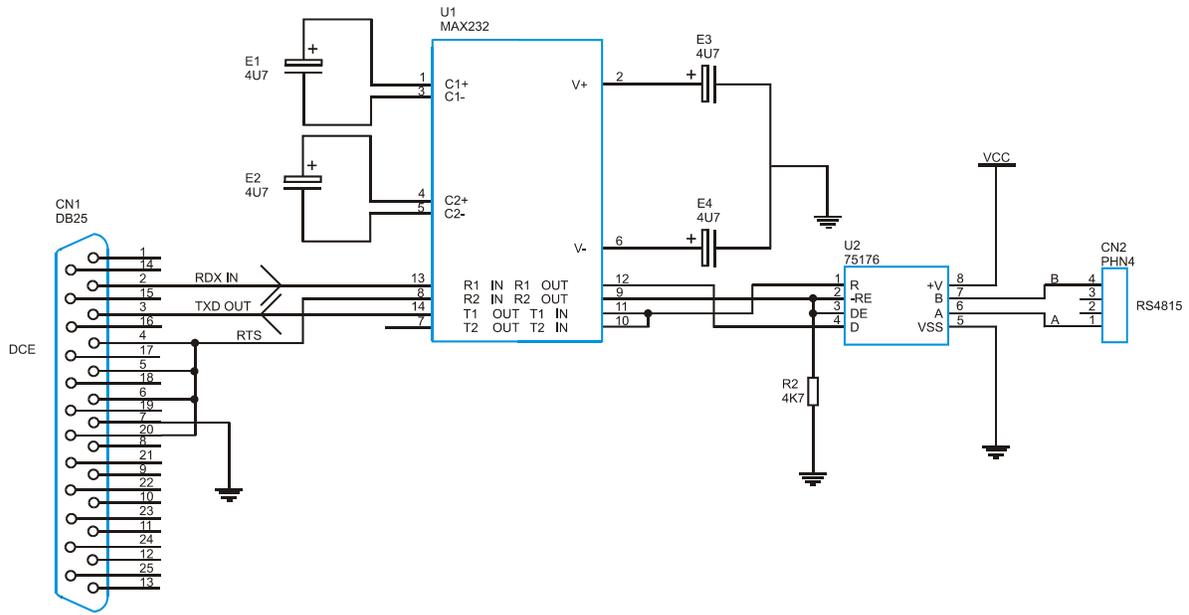


Figure 3.22
RS-232/485 converter

The RS-232/422 and RS-232/485 interface converters are very similar and provide bi-directional full-duplex conversion for synchronous or asynchronous transfer between RS-232 and RS-485 ports. These converters may be powered from an external AC source, such as 240 V, or smaller units can be powered at 12 V DC from pins 9 and 10 of the RS-232 port. For industrial applications, externally powered units are recommended. The RS-232 standard was designed for communications, not as a power supply unit!

The connections for a typical, externally powered RS-232/485 converter (Black Box Corporation) are shown below. Black Box does not recommend operating both ports on the converter at both DCE and DTE. LEDs are provided to show the status of incoming signals from both EIA-232 and EIA-485.

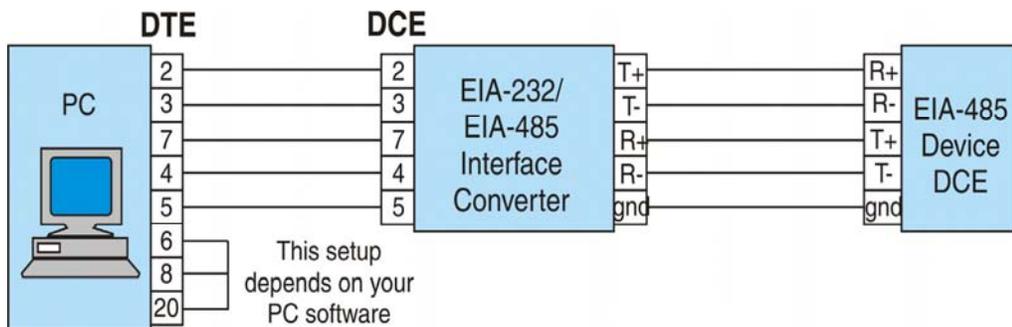


Figure 3.23
Wiring diagram for EIA-232/485 converter

When operating over long distances, a useful feature of interface converters is optical isolation. This is especially valuable in areas prone to lightning. Even if the equipment is

not directly hit by lightning, the earth potential rise (EPR) in the surrounding area may be sufficient to damage the communications equipment at both ends of the link. Some specifications quote over 10 kV isolation, but these figures are seldom backed up with engineering data and should be treated with some caution.

Typical specifications for the RS-232/422 or RS-232/485 converters are:

- Data transfer rate of up to 1 Mbps
- DCE/DTE switch selectable
- Converts all data and control signals
- LEDs for status of data and control signals
- Powered from AC source
- Optically isolated (optional)
- DB-25 connector (male or female)
- DB-37 connector (male or female)

Typical specifications for the RS-232/current loop converters are:

- 20 mA or 60 mA operation
- DCE/DTE, full/half-duplex selectable
- Active or passive loops supported
- Optically isolated (optional)
- Powered from AC source
- Data rates of up to 19 200 kbps over 3 km (10 000 feet)
- DB-25 connector (male or female)
- Current loop connector – 5 screw

3.17 Interface to serial printers

It is important that the serial interface for serial printers is set up correctly. The following points should provide a guide:

- Select the correct terminator for the end of each block of characters (or bytes) sent down from the PC to the printer: This can either be a CR or LF (although CR is more common).
- Check what sort of flow control is being supported between the PC and the printers. If XON/XOFF is selected on the printer (as opposed to DTR), the printer cable must have the transmit and receive data pins connected and the PC software must support the XON/XOFF flow control. The PC's hardware and firmware itself does not directly support XON/XOFF. Ensure that the ASCII characters used to represent XON and XOFF are the same for both the printer and the PC.
- Select the appropriate settings for the buffer full (XOFF) and buffer empty (XON) codes on the printer. Most PCs have a maximum line buffer length of 256 bytes. The printer should send an XOFF when the buffer has 256 free bytes of storage remaining and XON when the buffer has emptied to less than 256 bytes. Typical printers have at least 2 K bytes of buffer memory.
- Check that the option of using pin 20 or pin 11 for the indication of DTR or printer ready line has been set correctly.
- Check that the correct selection has been made for either parallel or serial connections to the PC if this is optional on the printer.

- Ensure that the baud rate is not set too high, which do not employ any form of handshaking or flow control at all, as there could be loss of data.



Figure 3.24
PC to printer serial connection

3.18 Parallel data communications interface standards

The two most important parallel data communication interface standards are:

- The general purpose interface bus (GPIB) or IEEE-488, used mainly for scientific purposes and automatic testing/measurement
- The Centronics standard – used mainly for cabling between PCs and printers.

3.19 General purpose interface bus (GPIB) or IEEE-488 or IEC-625

The GPIB was originally developed for automatic testing and for use with scientific equipment in laboratories, manufacturing, and other industrial and power system applications. In the early days of automatic testing, it became clear to manufacturers of digital test and scientific equipment that a universally accepted data communications interface was required between the computer controlling the testing sequence, the test equipment and recording devices, such as plotters and printers.

The standard used today was originally defined by Hewlett-Packard in 1965 as a digital data interface standard for the interconnection of engineering test instruments and was initially called the Hewlett-Packard Interface Bus (HPIB). This standard was adopted by other manufacturers and was published in 1975 as the IEEE-488 standard. IEEE-488 was updated in 1978 and issued internationally as IEC-625. There have been further revisions to the standard since 1978. IEC-625 is the common designation for GPIB in Europe.

The GPIB is an interface design that allows the simultaneous connection of up to 15 devices or instruments on a common parallel data communications bus. This allows instruments to be controlled or data to be transferred to a controller, printer, or plotter. It defines methods for the orderly transfer of data, addressing of individual units, standard bus management commands and defines the physical details of the interface.

Physical connection configurations

The devices on the GPIB can be connected in one of two ways:

- A star configuration
- A chain (linear) configuration

A star configuration is one in which each instrument is connected directly to the controller by means of a separate GPIB cable. The connectors are all connected to the same port as the controller, as shown in Figure 3.25. A drawback to this simple arrangement is that all devices on the same bus must be relatively close to the controller because of the length limitation of each cable.

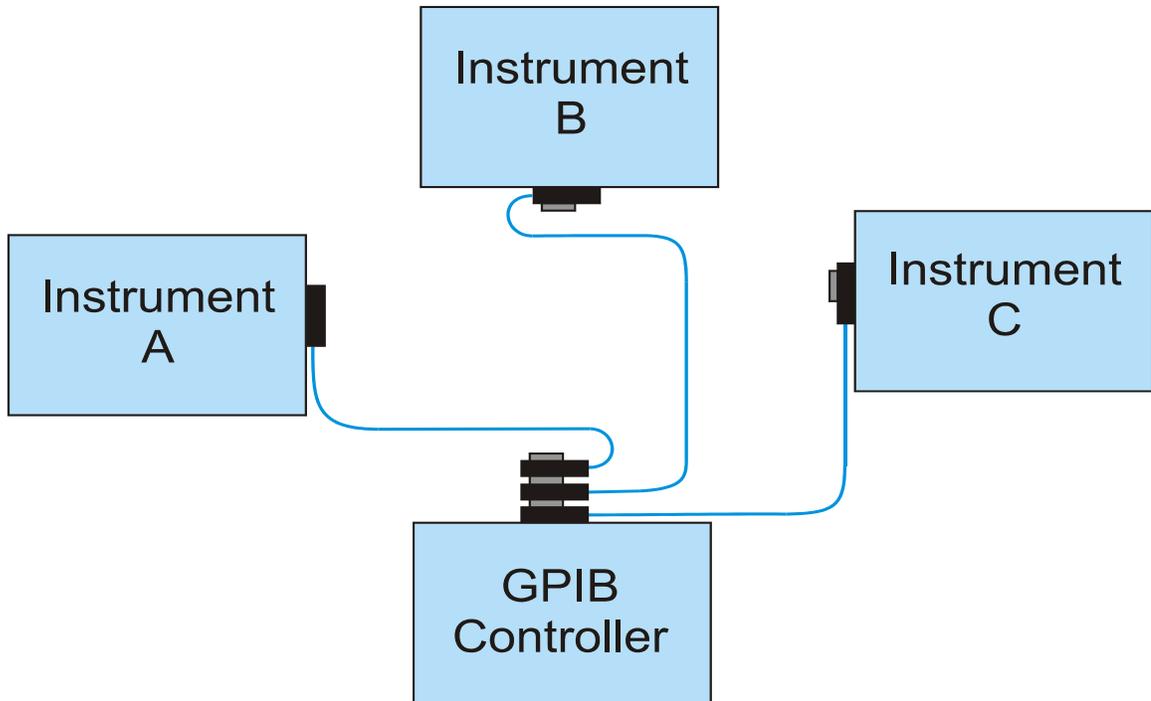


Figure 3.25
The GPIB star connection

The chain configuration, each device, including the controller, is connected to the next in a chain. The controller does not have to be the first or last device in the chain, but can be linked in anywhere as shown in Figure 3.26. A controller is a controller in the sense that it coordinates the events on the bus. Physically and electrically, it is similar to any other device connected to the GPIB. This configuration is usually the most convenient way to connect equipment.

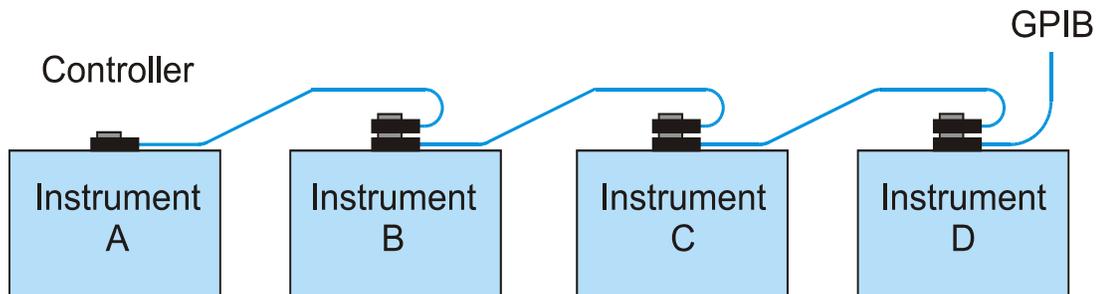


Figure 3.26
The GPIB chain (linear) configuration

Although the star and chain configurations are suggested for GPIB, connections can be made in any other way, if the following rules are observed:

- All devices are connected to the bus
- No more than 15 devices, including the controller, may be on the bus with no less than two thirds powered on
- Cable length between any two devices may not exceed 4 meters (13.33 feet) with an average separation of 2 meters (6.66 feet) over the entire bus
- Total cable length may not exceed 20 meters (66.66 feet)

A single device on the GPIB can transfer data to up to 14 other devices. The GPIB uses asynchronous handshaking, so the actual data transfer rate is determined by the devices themselves. Generally, the hardware limits the maximum data rate to about 250 kbps.

Device types

There are many thousands of different types of GPIB-compatible devices available for various applications. There are 4 different groups of devices:

- Talkers
- Listeners
- Talkers/listeners
- Controllers

A talker is a one-way communicating device that can only send data to other devices. It does not receive data. The talker waits for a signal from the controller and then places its data on the bus. Common examples are simple digital voltmeters (DVMs) and some A/D converters.

A listener is a one-way communicating device that can only receive data from another device. It does not send data. It receives data when the controller signals it to read the bus. Common examples are printers, plotters, and recorders.

A talker/listener has the combined characteristics of both talkers and listeners, with the limitation that it is never a talker and a listener at the same time. A common example is a programmable one, which is a listener while its range is being set by the controller, and a talker while it sends the results back to the controller. Most modern digital instruments are talker/listeners as this is the most flexible configuration.

A controller manages and controls everything that happens on the GPIB. It is usually an intelligent or programmable device, such as a PC or a microprocessor controlled device. The controller determines which devices will send and which will receive data and when. To avoid confusion in any GPIB application, there can only be one active controller, called the controller in charge (CIC). There can be several controllers, but to avoid confusion, only one can be active at any time. A controller also has the features of a talker/listener. In some cases, when several PCs are simultaneously connected on a GPIB, one is usually configured as the controller and the others configure as talkers/listeners. The controller needs to be involved in every transfer of data. It needs to address a talker and a listener before the talker can send its message to a listener. After the message is sent, the controller unaddresses both units.

Some GPIB configurations do not require a controller, for example, when only one talker is connected to one or more listener. A controller is necessary when the active or addressed talker or listener must be changed.

Electrical or mechanical characteristics

The GPIB bus is carried inside a shielded 24-wire cable with standard connectors at each end. The standard connector used is the ‘Amphenol’ as shown in Figure 3.27. Adding a new device to the bus is done by connecting a new cable in a star or chain configuration. Screws hold one connector securely to the next. Since the 24-pin connectors are usually stackable, it is easy to connect or disconnect devices to the bus.

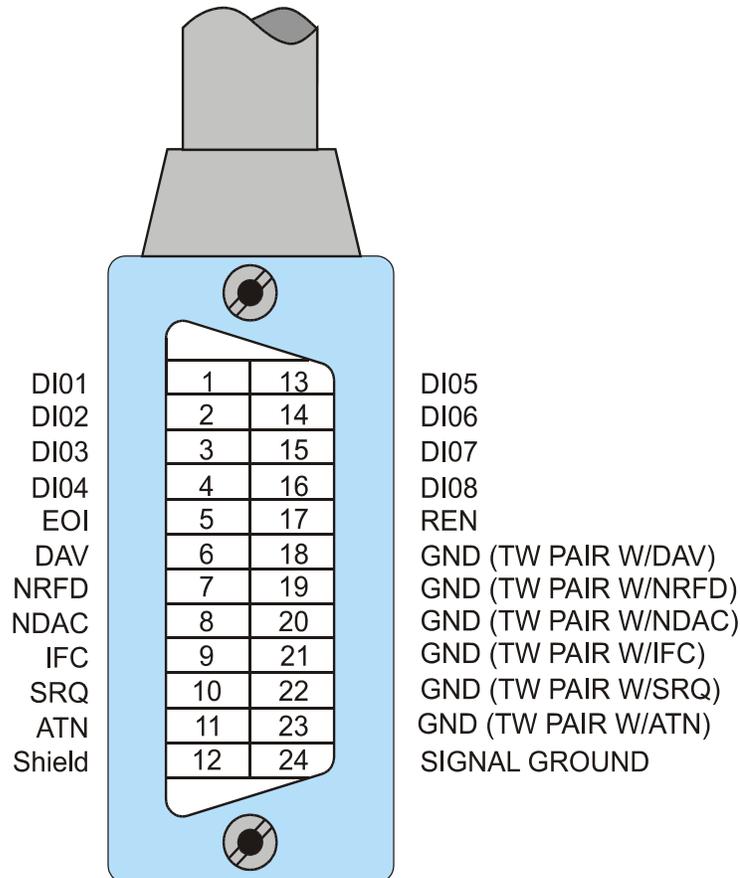


Figure 3.27
GPIB connector and pin assignment

The 24 lines in each cable consist of 8 data lines and 8 pairs (16) of control and bus management lines. The data lines are used exclusively to carry data, in a parallel configuration (byte by byte), along the bus. The control and bus management lines are used for various bus management tasks that synchronize the flow of data. When data or commands are sent down the bus, the bus management lines distinguish between the two. Detailed knowledge of how these management lines interact is useful but not necessary to effectively use the GPIB.

In the RS-232, the UART is used to coordinate the ‘housekeeping’ activities associated with the serial interface. The full capacity of the microprocessor can then be directed to other duties. In a similar way, the coordination of the GPIB parallel interface is controlled by a GPIB IC. The most common GPIB ICs used for this purpose are the Texas Instruments TMS9914A and the NEC-7210.

GPIB bus structure

The GPIB consists of 8 data lines (DI01–DI08) and 8-pairs of control lines. Three of the eight control line pairs are the handshaking lines, which coordinate the transfer of data (DAV, NRFD and NDAC). The other five pairs are for bus control and management (ATN, REN, IFC, SRQ and EOI). The 8 ‘ground’ wires provide electronic shielding and prevent bus control signals from interfering with one another or from being influenced by external signals.

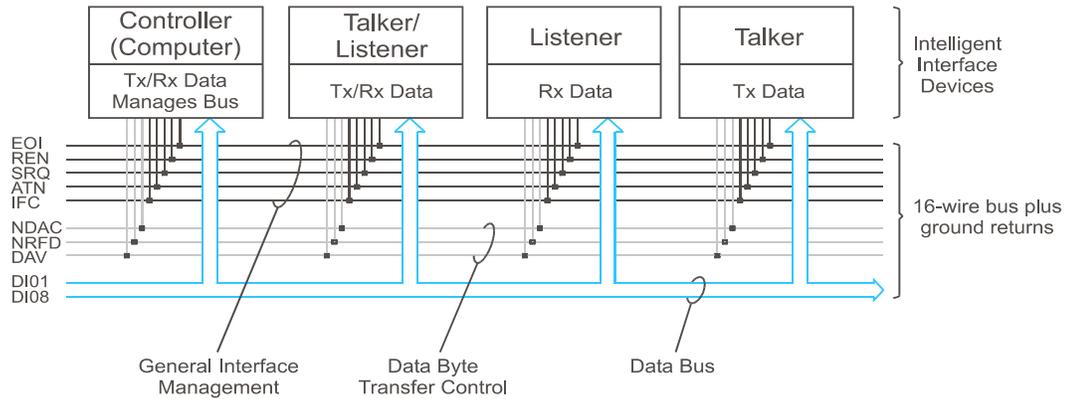


Figure 3.28
The GPIB bus structure

The signal lines can be separated into three groups:

Data Bus Lines	D101	D108 Data Bus Lines
Handshaking Lines	DAV	Data Available
	NRFD	Not Ready for Data
	NDAC	Not Data Accepted
General Interface Management Lines	ATN	Attention
	IFC	Interface Clear
	SRQ	Service Request
	REN	Remote Enable
	EOI	End or Identify

Table 3.7
Three groups of signal lines

The eight data lines DI01 to DI08 carry both data and command messages. All commands and most data use the 7-bit ASCII code, in which case the eighth bit, DI08, is either unused or used for parity.

The GPIB uses binary signals to represent the information that is carried on the lines of the bus. It uses the symbols ‘true’ and ‘false’ to represent the two states of voltages on the lines. The GPIB uses the logic convention called ‘low-true’ or negative logic, where the lower voltage state is ‘true’, and the higher voltage states are ‘false’. Standard TTL voltage levels are used. For example, when DAV is ‘true’, the TTL voltage level is low (± 0.8 V), when DAV is false, the TTL level is high (± 2.0 V). Despite low being ‘true’, no line can be high (i.e. ‘false’), unless all devices on that line allow it to go high. This is

convenient when there are several listeners accepting data. The ‘not data accepted’ (NDAC) line cannot go to the ‘false’ state, indicating data accepted, until the last listener has accepted the data. Consequently, the handshaking process waits for the slowest listener on the bus.

Each device connected to the GPIB has a unique device address and must be designed with enough intelligence to identify whether the data or command sent down the data line is meant for it or for another device. Device addresses are arbitrary and are set by the user, usually on a DPI switch on the back of the device, or by programming the device software. Each connected device is identified in the software of the controllers program. The only limitation in choosing a device address is that it must be an integer number in the range 1 to 30.

GPIB handshaking

Data is transmitted asynchronously on the GPIB parallel interface one byte at a time. The transfer of data is coordinated by the handshake voltage signals on the 3 bus control lines (DAV, NDAC and NRFD), called a three-wire interlocked handshake. Handshaking ensures that a talker will put a data byte on the bus only when all listeners are ready and will keep the data on the bus until it has been read by all listeners. It also ensures that listeners will accept data only when a valid byte is available on the bus.

The talker must wait for the NRFD line to go high (false) before any data can be put onto the bus. The NRFD line is controlled by the listeners. Only when NRFD voltage is high (false) are all listeners ready to receive data. The talker then asserts DAV ‘true’ (voltage low) and when the listeners detect the low level on DAV, they read the byte on the data lines. As each listener accepts the data, it releases NDAC. After the last listener has accepted the data, the NDAC line voltage goes high (false) and this signals the talker that the data has been accepted. Only when the data byte has been accepted by all the listeners can the talker allow DAV voltage to go high (false) and remove its data from the bus. Figure 3.29 illustrates this handshaking sequence.

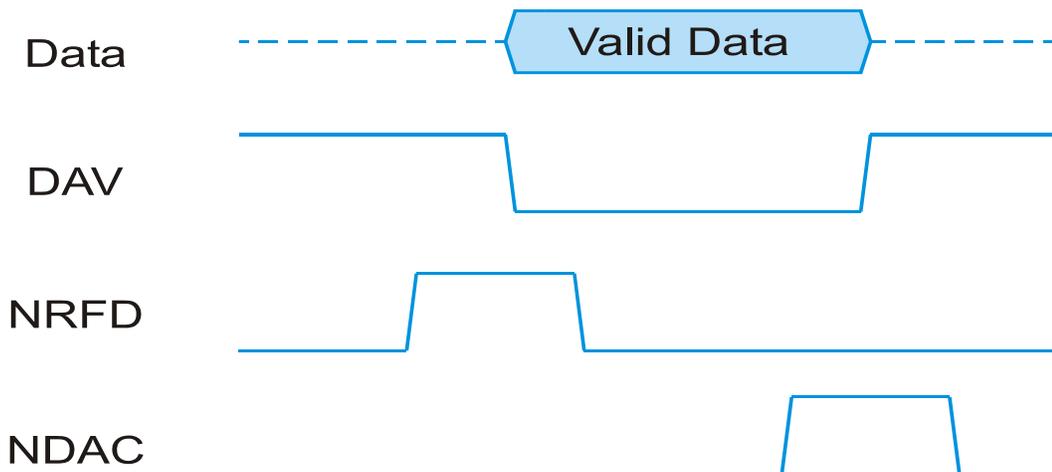


Figure 3.29
The GPIB handshaking timing diagram

Interface management lines

The other five lines manage the flow of information across the interface:

- ATN
- REN
- IFC
- SRQ
- EOI

Data transfers

Device addresses are any integer in the range of 1 to 30. There are 2 kinds of addresses for a device:

- Talk address (maximum of 15)
- Listen address (maximum of 15)

If a device sees its talk address on the bus, it knows it will act as a talker and will be required to send data. Conversely, when it sees its listen address on the bus, it will be required to act as a listener and receive data.

Both command/response with polling and interrupt driven protocols are allowed under the IEEE-488 specification.

3.20 The Centronics interface standard

The parallel printer or Centronics interface standard is used primarily to interface printers to computers or other intelligent devices and includes a 36-pin connector. This interface has a limited distance capability because of its low level +5 V signals. Full signal definitions are given in Table 3.8.

Signal Name	Signal Pin	Return Pin	Signal Definition
*DSTB	1	19	Low level pulse of 0.5 microseconds or more, used to strobe the DATA signals into the printer. The printer reads the data at the Low Level of this signal. Ensure an Acknowledge has been returned before using the next Data Strobe. Data Strobe is ignored if the BUSY is high.
*DATA 1-8	2-9	20-27	8 data lines from the host. High level represents binary 1, Low level represents binary 0. DATA 8 is the most significant bit. Signal must be High at least 0.5 microseconds before the falling edge of the Data Strobe signal and held at least 0.5 microseconds after the rising edge.
Acknowledge	10	28	Low level pulse of 2 to 6 microseconds indicates input of a character into the print data buffer, or the end of an operation.
BUSY	11	29	High level indicates the printer cannot receive data. Typical conditions that cause a High BUSY level are buffer full or ERROR condition.
PE(Paper Empty)	12		High Level indicates that the printer is out of paper.
SLCT (Select)	13		High Level indicates that the printer is ON LINE.
*AUTO FEED	14		Low Level indicates LF (Line Feed) occurs after each

XT		CT (carriage return) code.
No Connection	15	Reserved Signal Line.
Signal Ground	16	Logic/Signal Ground Level (0 V)
Frame Ground	17	Printer Cabinet/Frame Ground line
No Connection		Reserved Signal Line
Signal Ground	19-30	Twisted Pair cable return lines.
*INIT Initialize	31	Low Level pulse of 50 microseconds or more, resets the buffer and initializes the printer.
ERROR	32	Low Level indicates the printer is OFF LINE, has a PAPER OUT or has sensed an ERROR condition.
Signal Ground	33	Logic/Signal Ground Level (0 V)
No Connection	34	Not used.
+5 V Regulate	35	Connected to the +5 V source through a 3.n k Ohm Resistor.
*SLCT IN (-Select In)	36	Low level indicates the printer is placed ON LINE (Selected) when the power is turned ON.

Table 3.8
Centronics pin assignment

* Indicates that the signal is generated by the host system, for example, the PC.771.

3.21 The universal serial bus (USB)

In September 1998, Microsoft, Intel, Compaq, and NEC developed revision 1.1 of the universal serial bus. The objective was to standardize the input/output connections on the IBM PC for devices like printers, mice, keyboards, and speakers. Data acquisition (DAQ) devices were not envisioned to be connected to the USB system. However, that does not mean that the USB cannot be used for DAQ. In many ways, the USB is well suited for DAQ systems in the laboratory or other small-scale systems.

Small-scale DAQ systems have traditionally suffered from the need for an easy to use and standardized bus system for connecting smart DAQ devices. The nearest thing was the IEEE 488 GPIB system. The GPIB system can be expensive and is not supported on every PC without purchasing additional hardware. There is a need for an easy to operate, inexpensive, and standardized bus system to connect small-scale DAQ devices. The USB can fill those needs. With its plug-and-play ability, it is extremely easy to implement and use. In addition, it is now standard on all IBM compatible PCs. Although it is not in any way as cheap as say an RS-232 connection, it is affordable.

The USB is limited by its very nature for its application to DAQ systems. The biggest problem is the maximum cable distance. The low-speed version is limited to 3 meters (10 feet) and the high-speed version is limited to 5 meters (16 feet) in total cable length. This requirement reduces the ability of USB to be used in the large factory or plant environment. Typically, the DAQ systems in these industries need to cover distances of up to 1 km (0.6 miles). Due to the timing requirements of the USB, the length of the cable **cannot** be increased with repeaters. This limits the use of the USB to the laboratory or bench top systems.

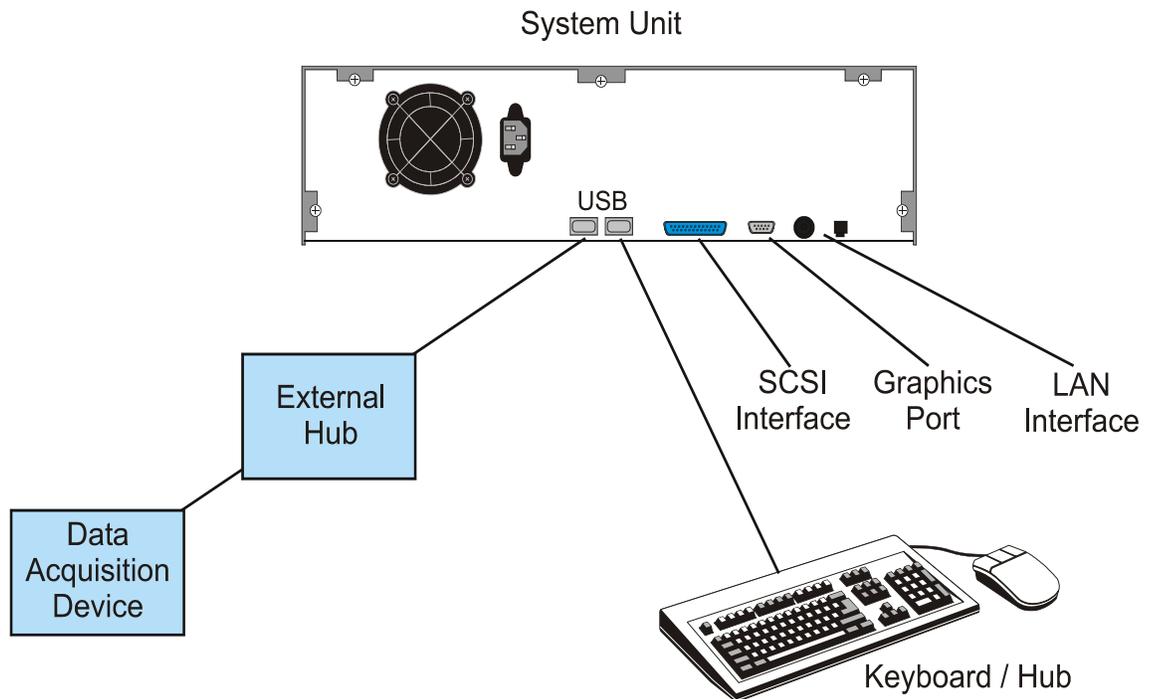


Figure 3.30
Connecting the USB

USB – overall structure

The USB is a master/slave, half-duplex, timed communication bus system designed to connect close peripherals and hubs to an IBM compatible PC. It runs at either 1.5 Mbps (low-speed) or 12 Mbps (high-speed). The PC's software program (using device drivers) creates packets of information that are going to be sent to devices connected on the USB bus. The USB drivers in the computer allocate a certain time within a frame for the information. The packet is then placed in this 1 ms frame that can contain many packets. One frame might contain information for many devices or it may contain information for only one device. The frame is then sent to the physical layer via the USB drivers, and then on to the bus.

The device receives its part of the packet and if necessary formulates a response. It places this response on the bus. The USB drivers in the PC detect the response on the bus and verify that the frame is correct using a CRC checking method. If the CRC indicates that the frame is correct, the software in the PC accepts the response.

The devices connected to the USB bus can also be powered off the bus cable. Devices can use no more than 500 mA. This works well for small scale DAQ devices, larger DAQ devices usually use external power supplies. Both power and communications are on the same cable and connector.

There are many parts in the USB system that make the communication possible. These include:

- Host hubs
- External hubs
- Type A connector
- Type B connector
- Low-speed cables

- High-speed cables
- USB devices
- Host hub controller hardware and driver
- USB software driver
- Device drivers

Topology

The USB uses a pyramid-shaped topology with everything starting at the host hub. The host hub usually consists of two USB ports on the back of the PC. These ports are basically in parallel with each other.

Each port is a four-pin socket with two pins reserved for power and two for communications. The cables from external hubs or USB devices are plugged into the host hub ports. One or both of the ports can be used. It does not matter which one is used if only one connection is being made. If the external device or hub has a removable cable then a 'type A or type B' cable is used to make the connection. The 'A' plug goes into the back of the PC (host hub) and the 'B' plug goes into the device or external hub. If the external hub or device has an in-built cable then the 'A' plug is plugged into the host hub port. The socket on the host hub is keyed so the plug will only go in one way. 'B' plugs will not go into 'A' sockets and vice-versa.

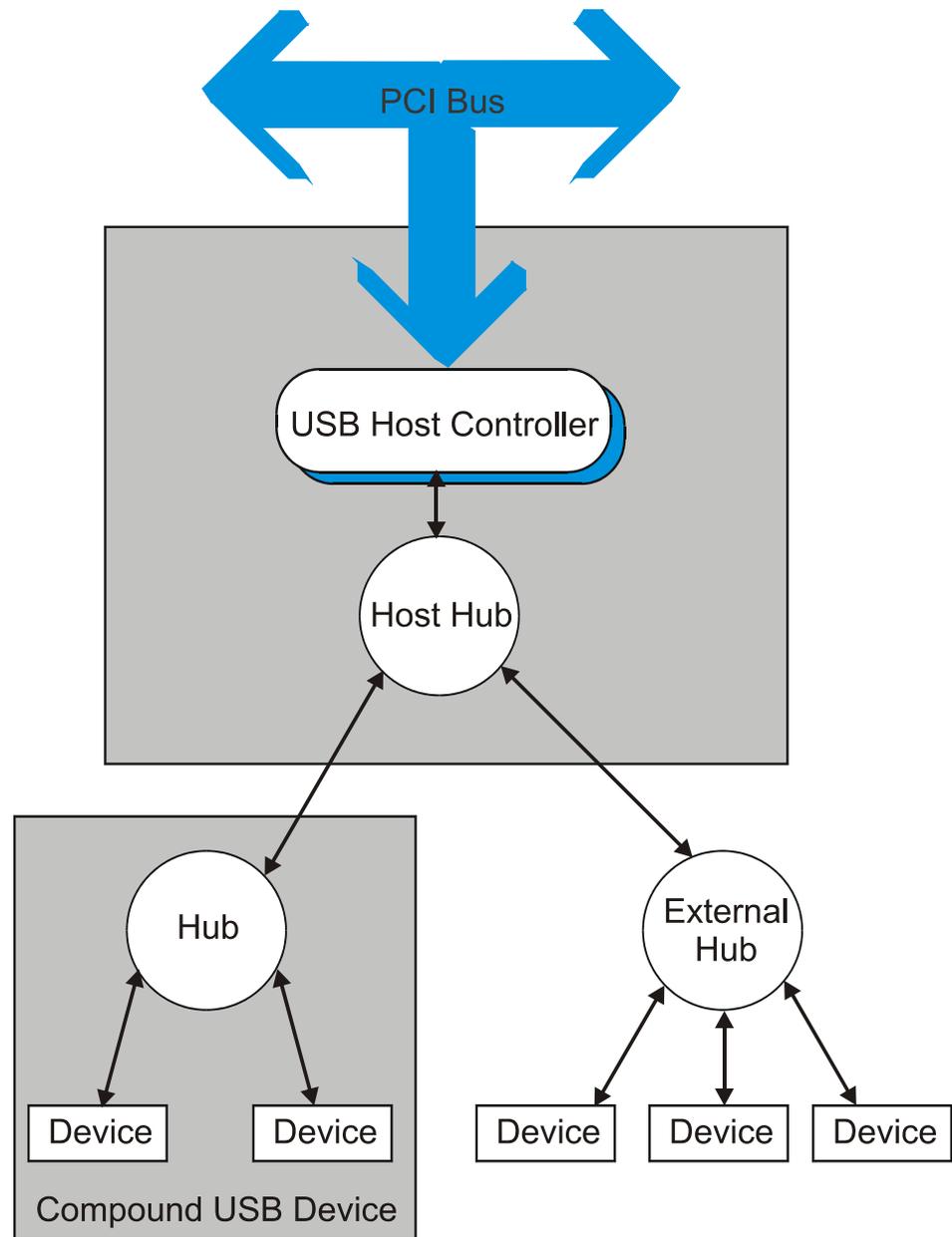


Figure 3.31
USB topology

The limitations on cable lengths are very important for the USB system. All cables, even if they come out of repeater hubs, must be counted in the total length of the cables.

Host hubs

The controller chips for the host hub usually reside on the motherboard inside the PC, although the hub could be a PCB in a PCI slot. The host controller does the parallel to serial and serial to parallel conversion from the PCI bus to the USB connectors. Sometimes a pre-processor is used to improve efficiency of the USB system. This host controller and connector combination is called the root hub or host hub. The host hub's

function is to pass the information to and from the PCI bus to the data lines (+D and -D) on the USB socket. The host controller can control the speed at which the USB operates. It also connects power lines (+5 V and ground) to a USB device via the USB cable. The external USB device may be another USB Hub or a USB type device like a printer

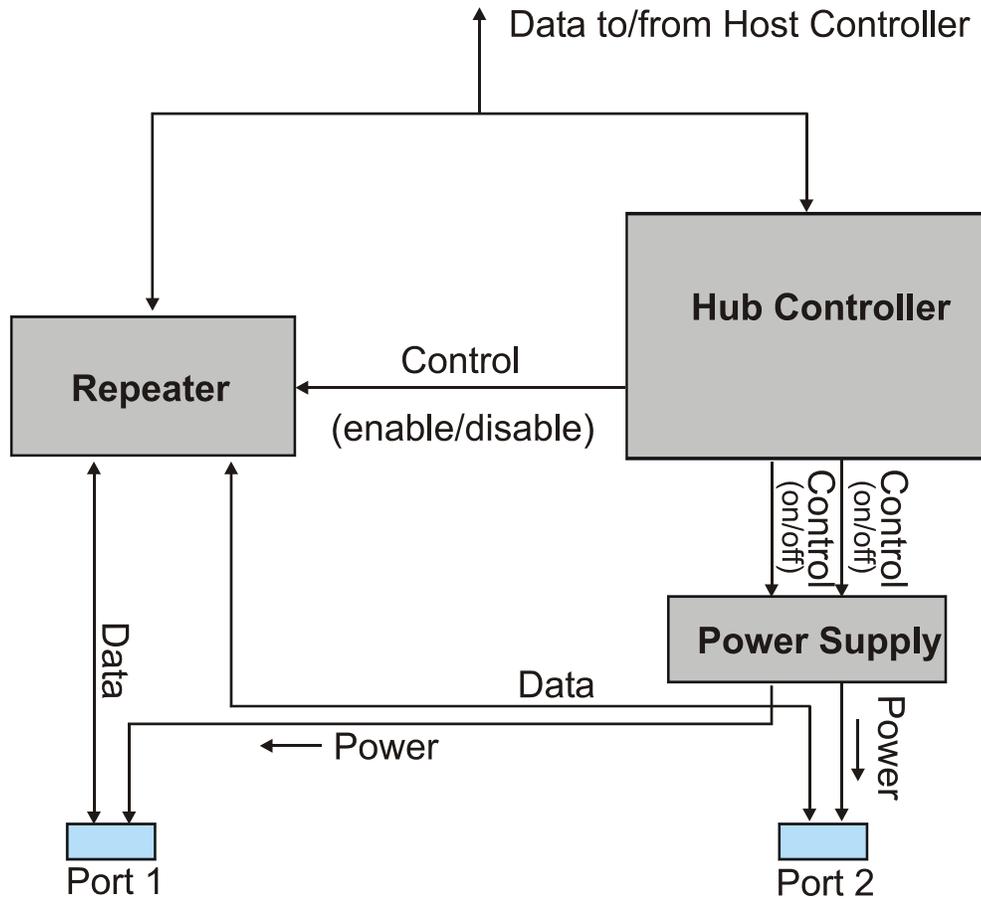


Figure 3.32
Host hub block diagram

The host hub has complete control over the USB ports. This control would include:

- Initialization and configuration
- Enabling and disabling the ports
- Recognizing the speed of devices
- Recognizing that a device has been connected
- Getting information from the application software
- Creating a packet and then frame
- Sending the information on to the bus
- Waiting and recognizing a response
- Error correction
- Recognizing that a device has been disconnected
- Using the port as a repeater

The connectors (Type A and B)

There are two types of connectors, type A and type B. The reason there are two types is that some devices have built in cables while others have removable cables. If the cables were the same, it would be possible to connect a host hub port to another host hub port. Because of the polarity of the connectors, the +5 V would be connected to ground. To keep this from happening the hub's output ports use type A connectors and the device input ports are type B. This means that it is impossible to connect one hub port to another hub port. On an external hub, the input to the hub is a type B connector unless the cable on the hub is permanently connected (no connector).

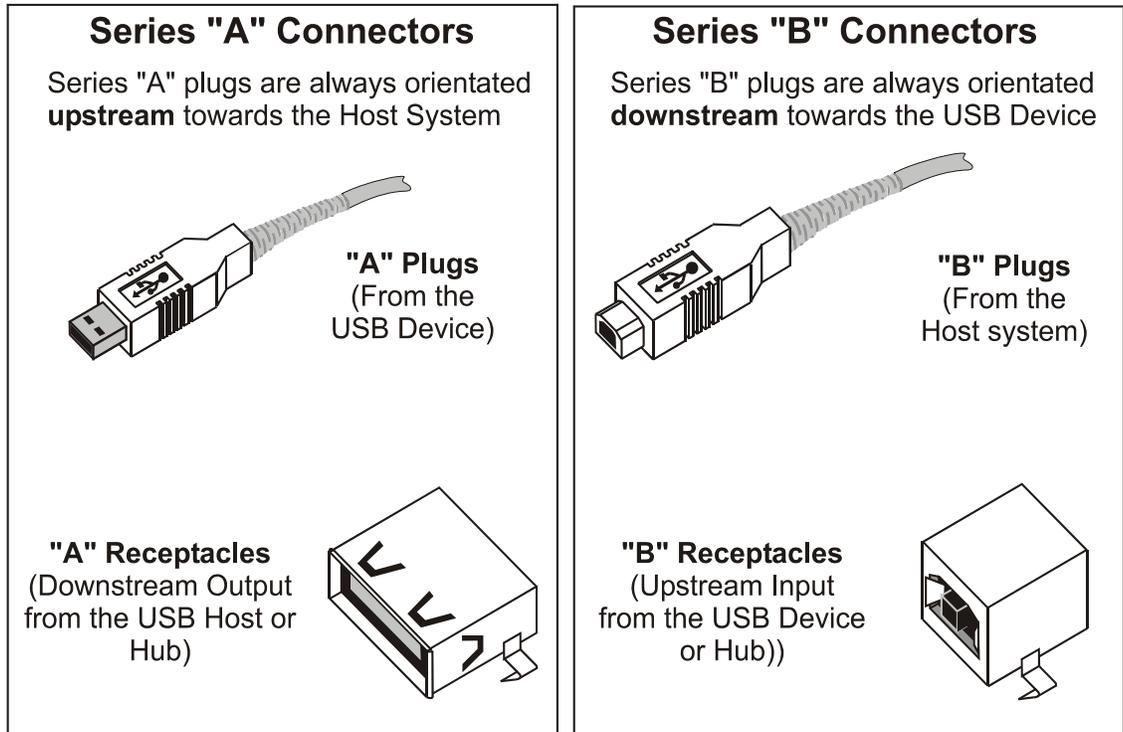


Figure 3.33
USB connectors

Low-speed cables and high-speed cables

The USB standard states that the USB will run at either 1.5 Mbps (slow-speed) or 12 Mbps (fast speed). The USB must have low-speed cables and high-speed cables. This is due to the impedance difference caused by the different frequencies of data transfer. The low-speed cables use untwisted unshielded cable. The data pair is 28 AWG and the power pair is 20–28 AWG. The low-speed cable is used on devices like keyboards and mice. The maximum distance for low-speed cabling is 3 meters (10 feet). The high-speed cables use twisted shielded cable pairs. The data pair is 28 AWG and the power pair is 20–28 AWG. The maximum propagation delay must be less than 30 ns. The maximum distance for high-speed USB is 5 meters (16 feet).

External hubs

The external hubs are used to increase the amount of devices connected to the system. Usually they have four USB output ports and either one type B input connector or a dedicated cable. This cable has a type A plug. It is usually connected to a host hub, but could be connected to the output socket (type A) of another external hub. Even though the external hub is a repeater, it cannot extend the overall length of the system. This is because of the timing requirements of the USB standard.

The external hub is an intelligent device that can control the communication lines and power lines on its USB ports. It is a bi-directional repeater for information coming from the host hub and from USB devices. It talks to and even acts like an external USB device to the host hub. It plays an integral part in the configuration of devices at start up. There is no physical limit to the number of hubs.

USB devices

The USB system supports every existing peripheral that can be connected to a PC. It also can and has been adapted to devices that are not usually considered peripherals. This would include data acquisition devices such as digital I/O modules and analogue input/output modules. All USB devices must be intelligent devices. Smart devices obviously cost more than 'dumb' RS-232 and RS-485 connected devices. With this increased cost, the user gets more functions, ease of use and the ability to connect more devices to the PC. With the old non-USB system, the computer was limited to a few devices. The USB system allows 127 devices to be connected to the PC at the same time.

There are two types of USB device:

- Low-speed, and
- High-speed

The low-speed devices are not only limited in their speed but also in features. These devices include keyboards, mice, and digital joysticks. Since these devices put out small amounts of information, they are polled less frequently and are slower than other devices. When high-speed devices access the USB bus, the low-speed device communication is disabled. Turning off the low-speed device ports at the root or external hubs disables the low-speed devices. The hubs re-enable the low-speed ports after receiving a special preamble packet.

High-speed devices like printers, CD-ROMs, and speakers need the speed of the 12 Mbps bus to transfer the large amount of data required for these devices. All high-speed devices see all traffic on the bus. They are never disabled like the low-speed devices. When a device like a microphone is 'connected' to the speakers most of the traffic and therefore packets will be used by the audio system. Other traffic like keyboard and mouse functions will have to wait. The host hub controller driver decides who has to wait and how long.

Host hub controller hardware and driver

The host hub controller hardware and software driver controls all transactions. The host hub controller hardware does the physical connections from the PCI bus to the USB connectors. It enables and initializes the host ports one at a time. It determines the speed and direction of data transfer on both host ports. The host controller in conjunction with the host hub software driver determines the frame contents, prioritization of the devices and how many frames are needed for a particular transfer.

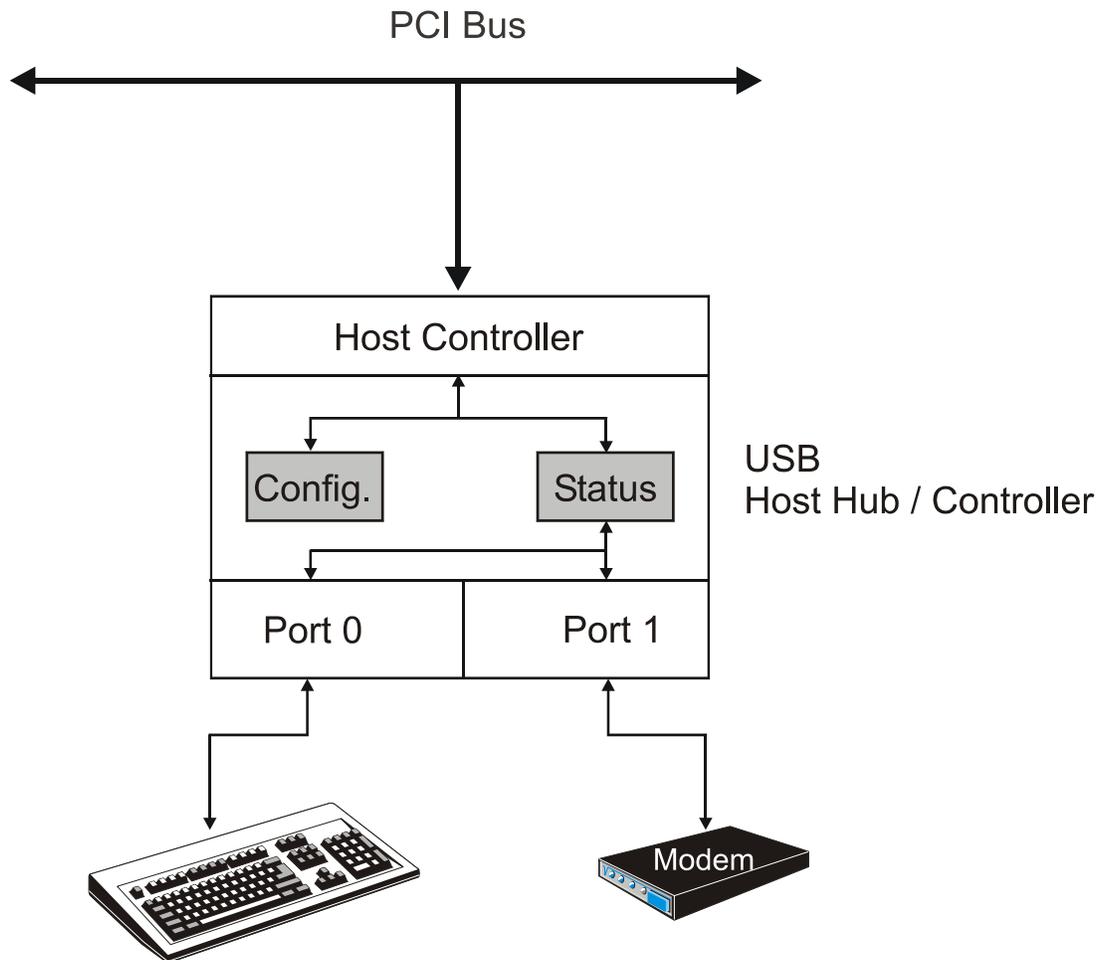


Figure 3.34
Host hub controller diagram

USB software driver

The USB software driver handles the interface between the USB devices, the device drivers, and the host hub driver. When it receives a request from a device driver in the PC to access a certain device, it coordinates the request with other device requests from the application software in the PC. It works with the host hub controller driver to prioritize packets before they are loaded into a frame. The USB software driver gets information from the USB devices during device configuration. It uses this information to tell the host hub controller how to communicate to the device.

Device drivers

For each USB device, a device driver must be loaded into the PC. This device driver is a software interface between the external USB device and the application software, the USB software driver and the host hub controller driver. It has information for the other drivers about that particular device's needs. This information is used to determine things like the type, speed (although that information can be determined physically by the hub ports), priority, and function of the device, as well as the size of packet needed for the transfer of data.

Communication flow

As mentioned before, the USB system is a master/slave, half duplex, timed communication bus system designed to connect peripherals and external hubs. This means that the peripherals cannot initiate a communication on the USB bus. The master (or host) hub has complete control over the transaction. It initiates all communications with hubs and devices. The USB is timed because all frames are sent within a 1 ms time slot. More than one device can place a packet of information inside that 1 ms frame. The host hub driver, in conjunction with the USB software driver determines the size of the packet and how much time each device is allocated in one frame.

If the applications software wants to send or receive some information from a device, it initiates a transfer via the device driver. Either the manufacturer of the device supplies this device driver or it comes with the operating system. The USB driver software then takes the request and places it in a memory location with other requests from other device drivers. Working together the USB driver, the host hub driver and the host hub controller place the request, data, and packets from the device drivers into a 1 ms wide frame. The host controller then transfers the data serially to the host hub ports. Since all the devices are in parallel on the USB bus, all devices ‘hear’ the information (except low-speed devices, unless it is a low-speed transfer. Low-speed devices are turned off when they are not being polled.) If necessary, the host waits for a response. The remote USB device then responds with an appropriate packet of information. If a device does not see any bus activity for 3 ms, it will go into the suspend mode.

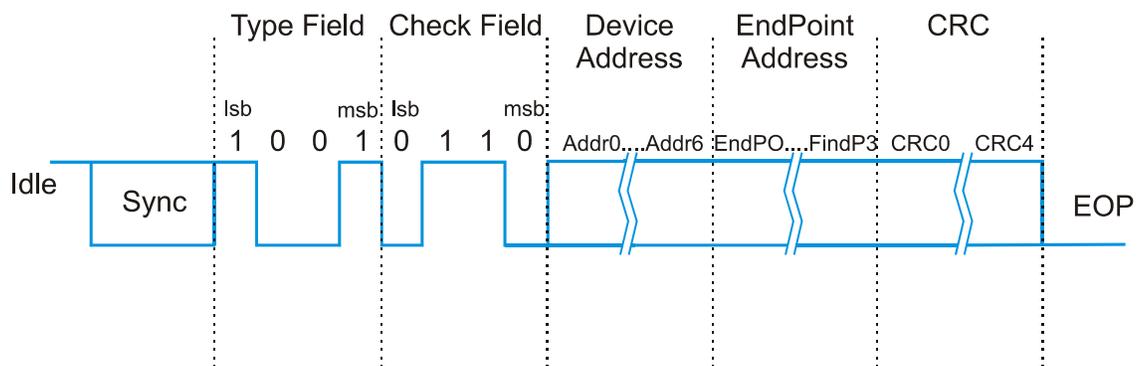


Figure 3.35
Example of an IN packet

There are four types of IN packets (reading information from a device) and three types of OUT packets (sending information out to a device).

Certain devices like mice and keyboards need to be polled (IN packets), but not too often. The USB software driver knows about these devices and schedules a regular poll for them. Included in the response are three levels of error correction. This type of transfer is very reliable. The peripherals are usually low-speed devices and therefore need a distinct low-speed packet to enable them. This packet is called a preamble packet. The preamble packet is sent out before the poll. The low-speed devices are disabled until they receive this preamble packet. Once they are enabled, they hear the poll and respond. Only one device can be polled at a time and therefore only one device will respond. USB has no provisions for multiple responses from devices.

On the other hand, there are devices that need constant attention but polling is not possible. These would be devices like microphones (IN packets), speakers (OUT packets), and CD-ROMs (both types of packets IN and OUT). The transfer rate is very important to these devices. Obviously, they would use the high-speed transfer rate and they would use a large portion of the frame (up to 90%). The receiving device does **not** respond to the data transfer. This transfer is a one-way data transfer (simplex). This means that error correction is effectively turned off for these types of transfers.

The physical layer

The physical layer of the Universal Serial Bus is based on a differential ± 3 V dc communication system. It is in some ways very similar to the RS-485 voltage standard. Unfortunately, it does not have the range of RS-485. This is not because of the type of wire used or because of the USB voltage standard itself, but because of the timing requirements of the USB protocol. In order to fit in everything the peripherals do on a USB bus, it was necessary to put very strict time requirements on the USB.

The USB physical standard has many benefits to the user. It is fast – 12 MHz – is very resistant to noise and is very reliable as long as the cabling rules are followed. With standardized cables and connectors, it is very hard for the user to get things wrong when cabling the USB system.

Contact Number	Signal Name	Cable Color
1	VCC	Red
2	-Data	White
3	+Data	Green
4	Ground	Black

Table 3.9
USB connector pins

The story goes that one day Bill Gates was watching his new computer being installed. When he saw the number of wires coming out of the back of the computer, he called the CEO at Intel and said, ‘We have to get rid of this mess of cables and connectors’. And as they say, the rest is history.

Connectors

The plugs and sockets on the USB have two wires for data communication and two wires for power. Using bus-powered devices is optional. The pins on the plug are not the same lengths. The power pins are 7.41 mm long and the communication pins are 6.41 mm long. This means that if a cable is plugged in ‘hot’ the power will be applied to the device before the communications lines. More importantly, it also means that when a cable is unplugged the data communications lines will be disconnected before the power. This reduces the possibility of back EMF voltage damaging the equipment. There are two types of connectors for the USB, type A and type B.

Type A is a flat semi-rectangular keyed connector that is used on the host ports, external hubs, and devices. The type B keyed connector is half-round and smaller than the

type A connector is. Note that both type A and B plugs have the USB symbol on the **top** of the connector. This is for orientation purposes.

The hubs and devices all have female sockets, while the cables have a type A male plug on one end and a type B on the other end. This is because if there were a type A on both ends it would be possible to connect two host hub sockets or external hub sockets. Cables that are not removable from the device or external hub only have a type A plug on one end.

Cables

The cables for the USB are specified as either low- or high-speed cables. Both the low- and high-speed cables can use type A connectors, but only a high-speed device can use type B connectors. Detachable cables are therefore always high-speed cables.

Due to that fact that the impedance of a cable is determined in part by the frequency of the signal, the two speeds need two different cables. External hubs are always high-speed units, but they accept low- and high-speed cables. Low-speed devices like keyboards only connect to other low-speed devices using low-speed cables. The ports on the hub can detect the speed of the device on the other end. If the D+ line is pulled high (+3.0 V dc to +3.6 V dc) then the device is considered high-speed. If the D- line is pulled high then the device is considered low-speed.

The low-speed (1.5 Mbps) cable is an unshielded, untwisted data cable. The communication pair is 28 AWG gauge but due to the lack of shielding and twisting, the overall diameter of the cable is smaller than a high-speed cable. The maximum distance for the low-speed cable is 3 meters (10 feet). This includes all host hub ports to external hub as well as the external hub to device cables. Usually on data communication systems, slower data speeds mean longer distances. In this case, the cable is unprotected against noise and because of the FCC restrictions on 1 to 16 Mbps communication the distance is severely limited.

The high-speed (12 Mbps) cable uses shielded twisted pair 28 AWG gauge wire. The maximum distance for high-speed cables is 5 meters (16 feet). Again, this includes all hub-to-hub and hub-to-device connections. The shield is internally connected to chassis ground at both ends. Usually on data communication systems, the ground is connected at only one end, but because the distances are short, this is not a problem.

Note: It is recommended to measure the chassis to chassis ground difference between both devices before making the connection.

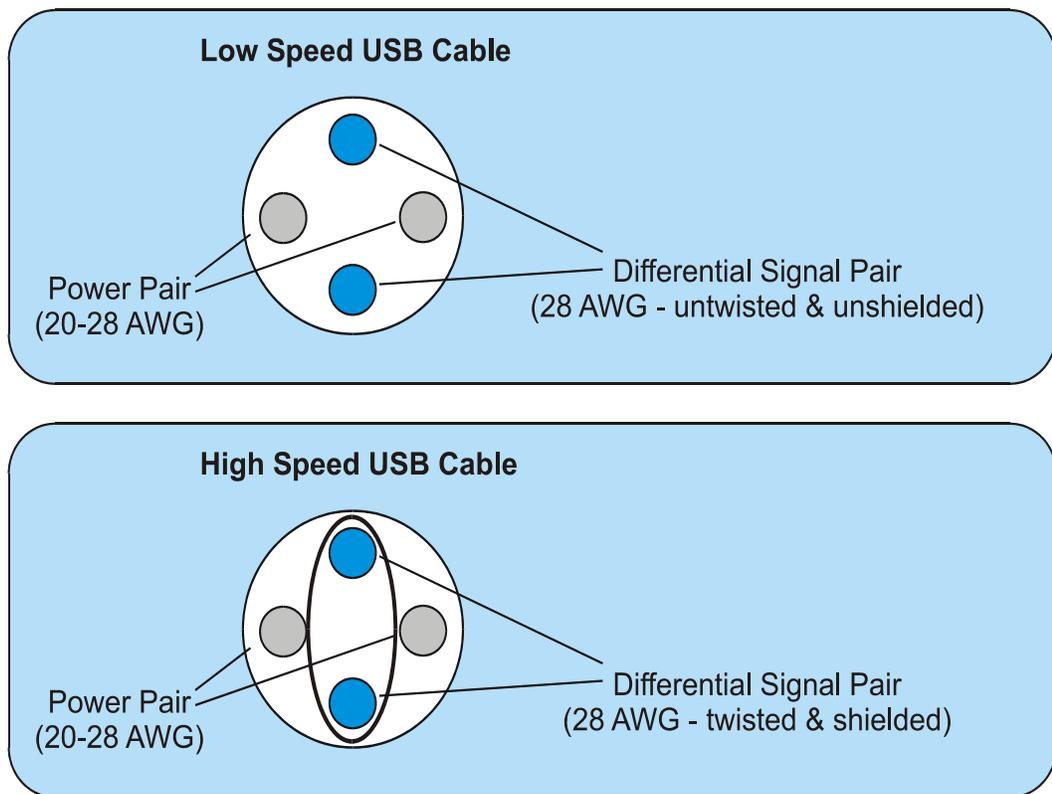


Figure 3.36
Low-speed and high-speed cables

The power pair on both low- and high-speed cables is 20 to 28 AWG gauge. The power pair supplies between 500 and 100 mA to external devices at +5 V dc. Every port on a hub provides this power to the devices if enabled by the hub. All hubs can decide if a port has power applied to the connector. If an external hub is itself powered by the bus then it divides the 500 mA up into 100 mA or so per port.

Signaling

When a device is plugged in to a hub, the port on the hub immediately determines the speed of the device. The port looks at the voltage on the D+ and D− lines. If the D+ line goes positive, the port knows that the device is a high-speed device. If the D− line goes positive, the port knows that the device is a low-speed device.

If both D+ and D− voltages fall below 0.8 V dc for more than 2.5 ms, the hub sees this as the device having been disconnected. If the voltage on either line is raised above 2 V dc for more than 2.5 ms, the port sees this as the device is plugged in.

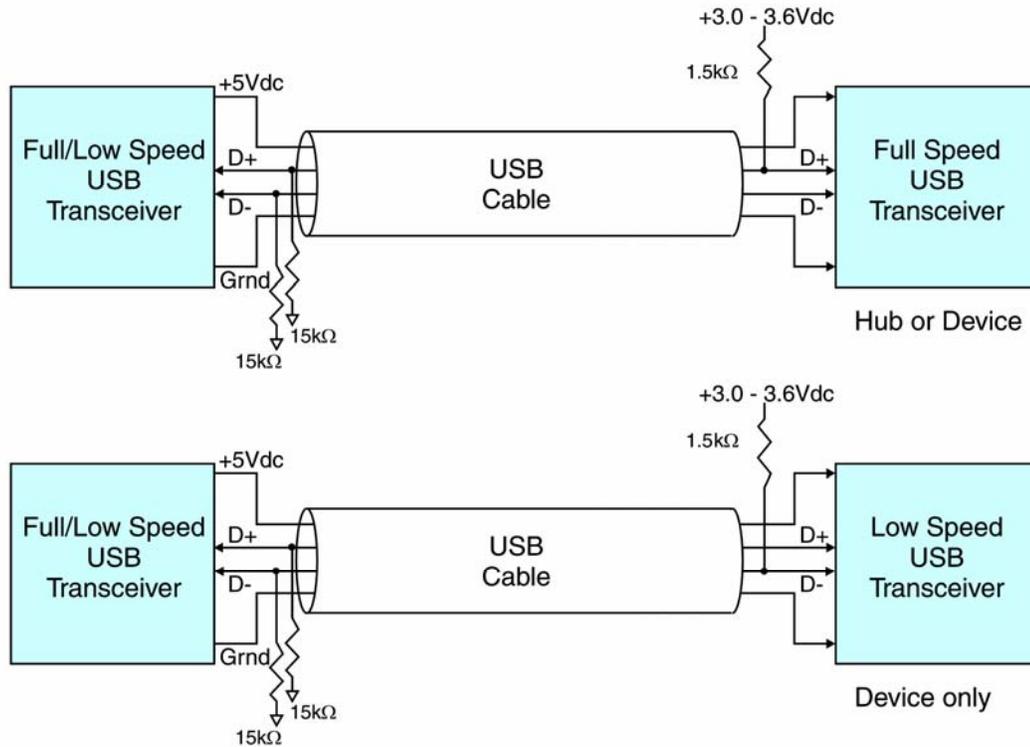


Figure 3.37
USB wiring diagram

The idle states for low and high-speed devices are opposite each other. For the low-speed device the idle state is the D+ line is a 0 V and the D- is a positive voltage. The idle state for the high-speed devices is such that the D+ is a positive voltage and the D- is 0 V at idle. In most data communications, a positive voltage indicates a zero (0) condition and a one (1) is minus voltage. In the USB system, it is not possible to say this because it uses an encoding system called NRZI.

The voltages used for the differential balanced signaling are:

- Maximum voltage transmitted +3.6 V dc
- Minimum voltage transmitted +2.8 V dc
- Minimum voltage needed to sense a transition ± 0.2 V dc
- Typical line voltage as seen from the receiver ± 3 V dc

NRZI and Bit Stuffing

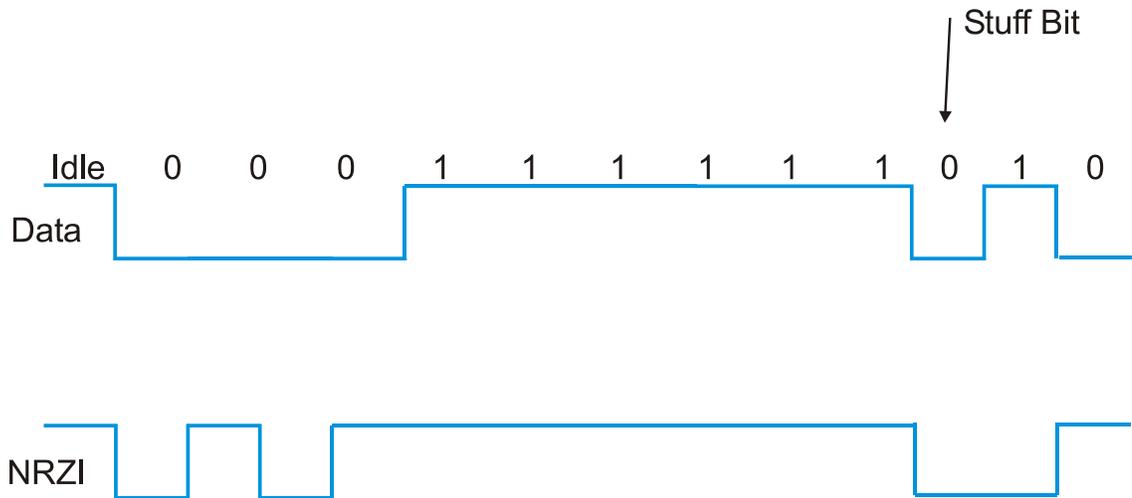


Figure 3.38
NRZI example

The USB uses the non-return to zero inverted (NRZI) encoding scheme. In NRZI a ‘1’ is defined as no change or transition of voltage whereas a “0” is a change or transition of voltage. A string of 0s would cause a clock-like data stream. The USB signaling system uses the transition from one voltage to another to synchronize the receivers. A stream of 1s therefore would mean no transitions. This would cause the receiver to lose synchronization. To overcome this problem the USB system uses a 6 of 7 bit stuffing technique. If six or more 1s are to be transmitted in a row, the transmitter stuffs in a 0 (a transition). If the receiver sees six 1s in a row, it knows that the next transition (zero) is to be ignored.

Power distribution

Devices like keyboards and mice need power to operate. This power is supplied by the USB system through the cables and hubs. External hubs can be either self-powered or powered off the bus. The voltage supplied by a USB hub is +5 V dc. The hubs must be able to supply minimum of 100 mA and maximum of 500 mA through each port. If an external hub with four ports is powered off the bus it divides the 500 mA supplied off the bus between the ports. Four times 100 mA equals 400 mA. This leaves 100 mA to run the hub. It is not possible to connect two bus-powered hubs together unless the devices connected to the last hub are self-powered. If the external hub is self-powered (i.e. mains-powered), it should be able to supply 500 mA to each of the ports.

Data link layer

The data link layer within the USB specification defines the USB as a master/slave, half duplex, timed communication bus system designed to connect close peripherals and external hubs. The hardware and software devices such as the host hub controller hardware and driver, USB software driver and device drivers all contribute to the data link layer of the USB.

As stated before, two speeds can be used in the USB system. For the most part the data link layer is the same, but there are some differences. The low-speed devices do not support bulk and isochronous transfers. The reason for this will become apparent in the following transfer descriptions.

The **interrupt transfer** is used for devices that traditionally used IRQ lines. Devices like keyboards, mice, and DAQ cards use the IRQ lines to tell the computer that they needed service. The USB does not support devices that initiate requests to the computer. To overcome this problem the USB driver initiates a poll of those devices that it knows need periodical attention. This poll must be frequent enough so that data does not get lost, but not too frequent, as not to use up much needed bandwidth. When installed, the device determines its minimum requirements for polling. Devices that need to be polled are rarely polled on every frame. The keyboard is typically polled only every 100th frame.

Isochronous transfer is used when the devices need to be written to or read from at a constant rate. This includes devices like microphones and speakers. The transfer can be done in an asynchronous, synchronous or device specific manner, depending on the device. This constant attention requires that the bulk of the bandwidth of the frame be allocated to one or two devices. If too many of these transfers take place at the same time, data could be lost. This type of transfer is not critical for data quality. There is no error correction and lost data or data that contains errors is ignored. Low-speed devices cannot use isochronous transfer because of the small amounts of data being transferred. It is not possible to move data fast enough using low-speed devices. In an isochronous transfer, the maximum amount of data that can be placed in one packet is 1023 bytes. There is no maximum number of packets that can be sent.

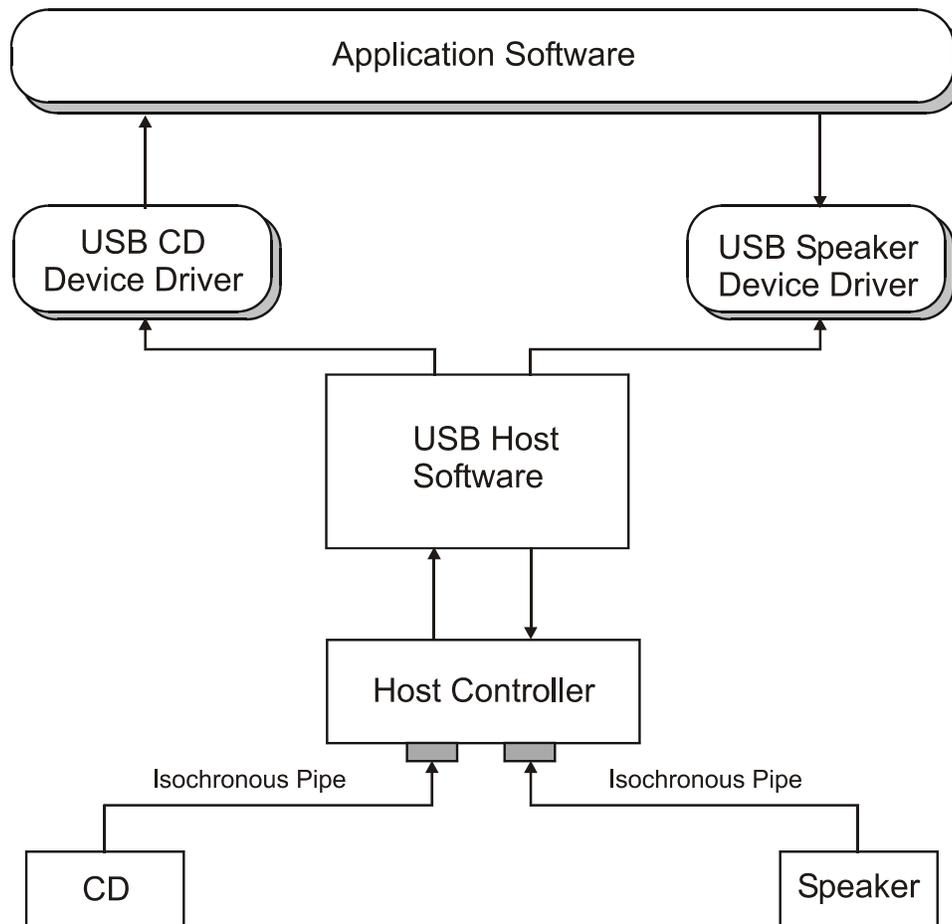


Figure 3.40
Isochronous transfer example

Control transfers are used to transfer specific requests and information to specific devices. This method is used mostly during the configuration and initialization cycles. These transfers are very data critical and require a response or acknowledgment from the device. Full error correction is in force for this type of transfer. All devices use this type of transfer at one time or another. These transfers use very little bandwidth but because the device must respond back to the host hub, the frames are dedicated to this one transfer.

Bulk transfers are used to transfer large blocks of data to devices that are not time dependent but where data quality is important. A typical device that would use the bulk transfer method would be a writeable CD or printer. These devices need large amounts of data but there is no time constraint like there is for a speaker. Whether the data get there in this 10 ms block or the next is not a problem. However, they do need correct data, so this type of transfer includes handshaking and full error correction.

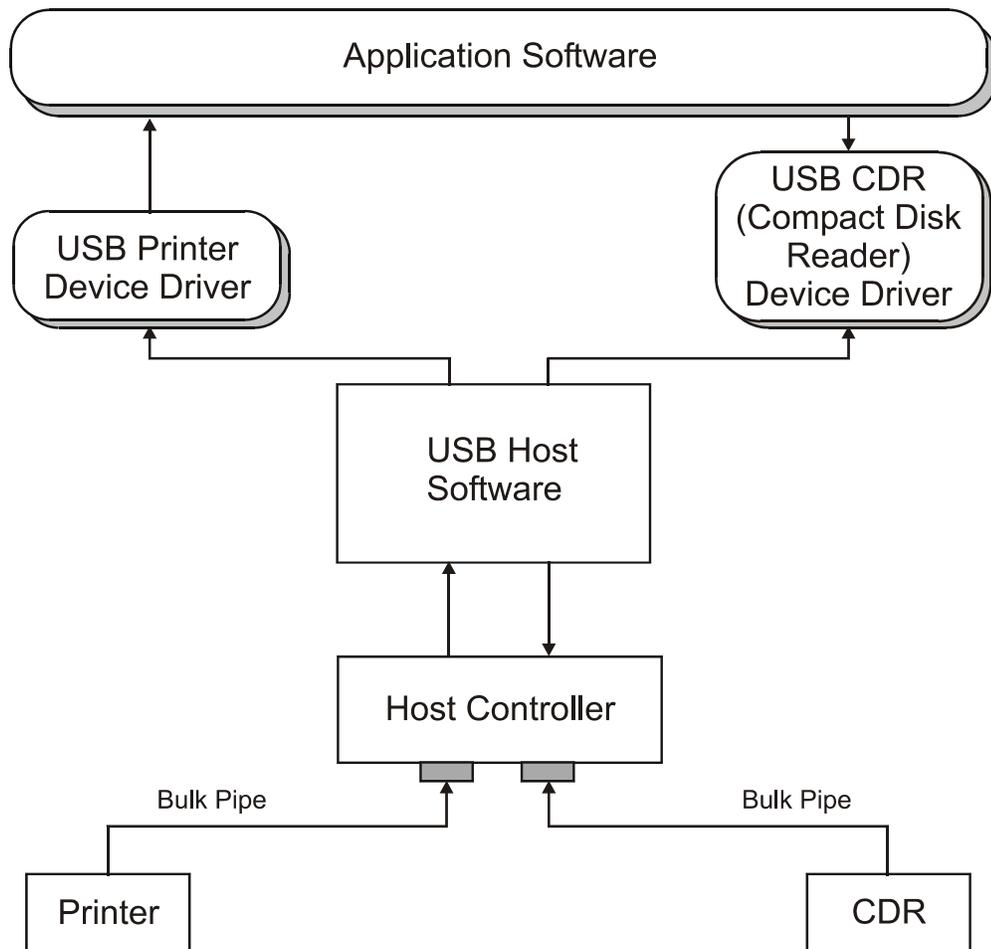


Figure 3.41
Bulk transfer example

Packets and frames

The USB protocol can and often does use a multi-packeted frame format. The USB frame is made up of up to three parts. One frame equals one transaction.

The three parts of the USB frame are:

- The token packet
- The data packet
- Handshaking

Every frame starts with a token packet. The token packet includes the other smaller packets. These include the synchronization pattern, packet type ID and token packet type.

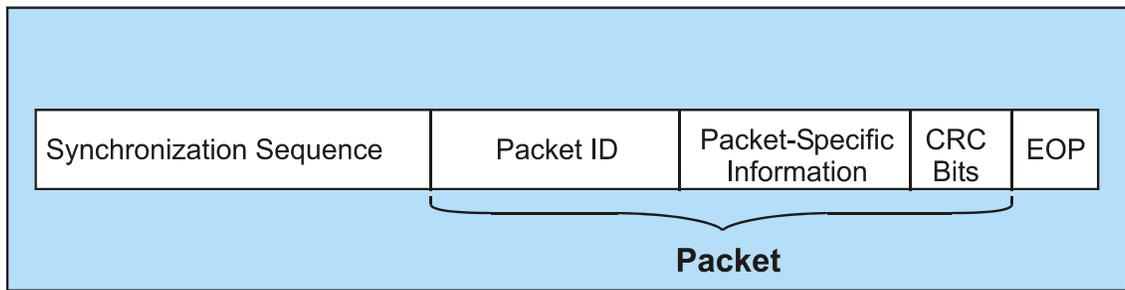


Figure 3.42
Packet format

There are four types of token packets:

- Start of frame packets
- In packets
- Out packets
- Setup packets

The start of frame token packet indicates the start of the packet. This tells the receiver that this is the beginning of the 1 ms frame. The 'in' packets transfer data in from the devices to the PC. The 'out' packets transfer data out from the PC to the device. The 'set-up' packet is used to ask the devices or hubs for startup information. They have information for the devices or hubs.

A special packet is only used on low-speed transfers. It is called the preamble packet. It is a shorter packet than the high-speed frame, only holds up to 64 bytes of data and always uses handshaking. It only has three variations, in packet, out packet and setup packet.

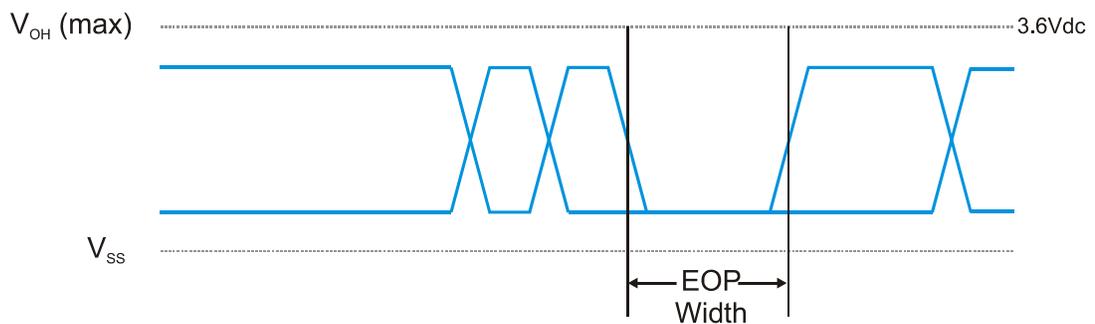


Figure 3.43
End of packet waveform

At the end of all packets, except for isochronous frames, there is an error correction packet. On high-speed frames, this is a 16 bit CRC, low-speed devices use a 5 bit CRC because of their smaller packets. If a device or host hub sees an end of frame message, it checks the CRC. If the CRC is correct, it assumes that this is the end of the message. If the CRC is not correct and the timeout limit has not been reached, the receiver waits. If the CRC is not correct and the timeout has been reached, the receiver assumes that the frame is not correct.

Application layer (user layer)

The application layer can be divided into two sub-layers, the operating system, (such as Windows 2000) and the device application software (such as a modem application program).

The application layer of the USB standard is really a user layer, because the USB standard does not define a true application layer. What it does define is a user layer that can be used (by an application programmer) to build an application layer.

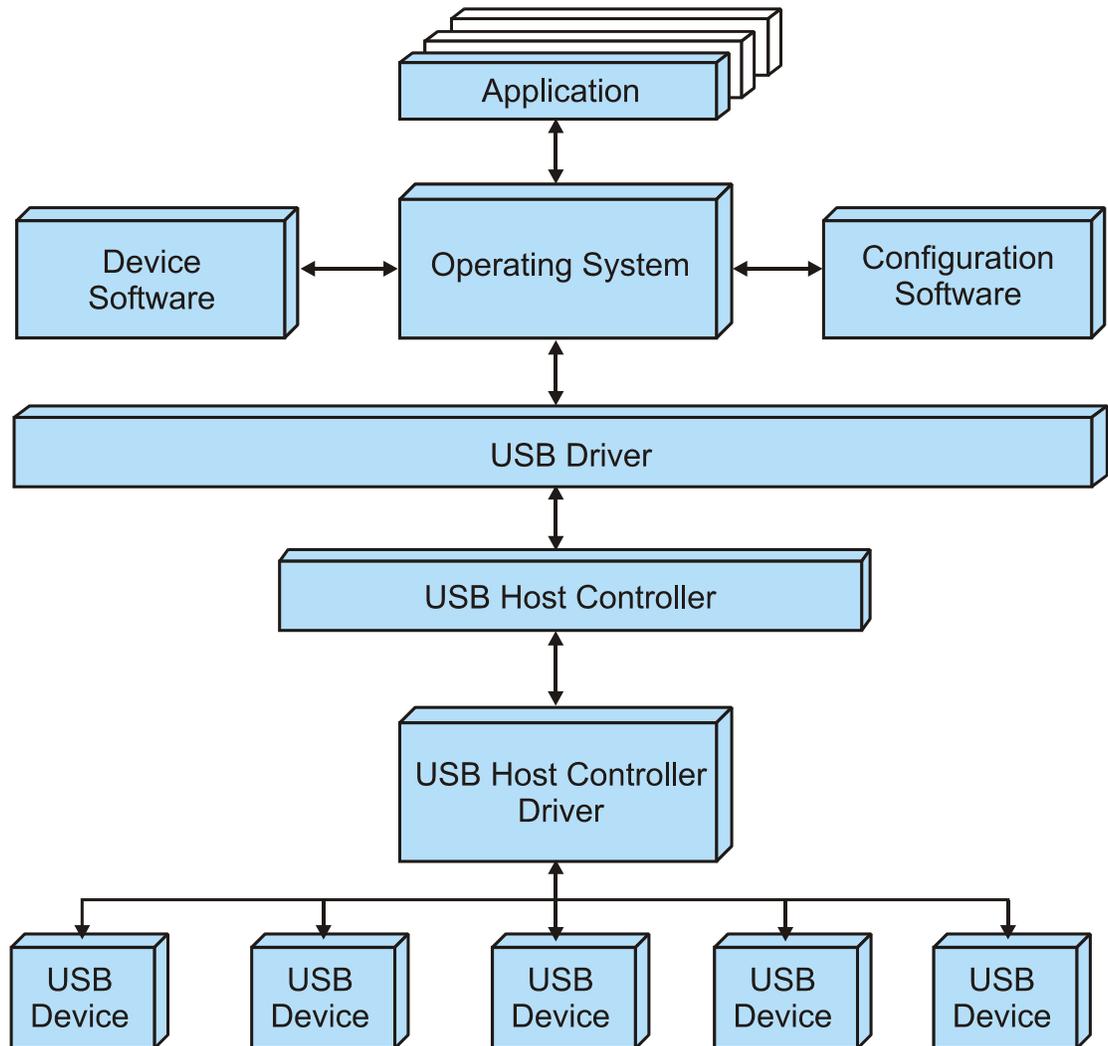


Figure 3.44
Application software diagram

The operating system user layer includes:

- Commands
- Software drivers
- Hub configuration
- Bandwidth allocation

Device applications would use:

- Commands
- Device drivers
- Device configuration

Specific user layer information can be found in the universal serial bus specifications at the USB Implementers Forum web page at <http://www.usb.org>.

Conclusion

Designed as a peripheral connection system for the PC, the USB can be adapted to be used on data acquisition systems. Now that the DAQ industry is developing increasingly intelligent data acquisition and control systems, the USB is easily adaptable to modern DAQ. The devices can be either low- or high-speed devices and very quickly and easily connected to a PC. There are many devices on the market now and it is bound to grow in the future. With the plug and play, system incorporated in USB the user does not have to spend hours or even days configuring the DAQ system. These time savings often offset the extra cost of the devices.

The target speed of USB 2.0 is 480 Mbps, as announced by the USB 2.0 promoter group, consisting of Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, and Philips. The target speed announcement coincides with the release of the USB 2.0 specification draft to industry developers.

Acknowledgments

Information from the following sources has been included in this section:

- Universal Serial Bus Specification, USB Implementers Forum web page at <http://www.usb.org>
- Universal Serial Bus System Architecture, MindShare Inc., by Don Anderson
- Intel USB Product Specifications Intel 8x930 and 8x931 USB Peripheral Controllers at: <http://www.intel.com/design/usb/prodbref/29776501.htm>
- Other Web sites of interest:
<http://www.lucent.com/micro/suite/usb.html>
<http://www-us.semiconductors.philips.com/usb/>

Appendix A

Numbering systems

A generalized number system

A number system is formed by allocating symbols to specific numerical values. Any group of symbols can be used with the total number of symbols for a number system called the **base** of the system.

The three most common bases are:

- Binary, with two symbols (0 and 1) and hence a base of 2
- Hexadecimal, with sixteen symbols (0,1,2...9,A, B...F) and hence a base of 16
- Decimal, with ten symbols (0,1,2...9) and hence a base of 10

When numbers with different bases are being used in the same descriptive text they sometimes have the subscript referring to the base being used, as in 3421.19_{10} for a decimal or base 10 number.

Numerical symbols have to be combined in a certain way to represent other combinations of numbers. The decimal numbering system has the structure laid out in Table A.1 for weighting each digit in the number 3421.19_{10} in a combination of numbers written together.

Exponential notation is used here, for example: 10^2 means 100 and 10^{-3} means 0.001.

Weight	10^4	10^3	10^2	10^1	10^0	.	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
	0	3	4	2	1	.	1	9	0	0	0

Table A.1

Decimal weighting structure

The Most Significant Digit (or MSD) in this number is 3. This refers to the left-most digit that has the greatest weight (10^3 or 1000) assigned to it.

The Least Significant Digit (or LSD) in this number is 9. This refers to the right-most digit that has the least weight (10^{-2} or 0.01) assigned to it.

This represents the number calculated below:

$$\dots 0 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 2 \times 10^1 + 1 \times 10^0 + 1 \times 10^{-1} + 9 \times 10^{-2} + 0 \times 10^{-3} + \dots$$

Binary numbers

Binary numbers are commonly used with computers and data communications because they represent two states – either ON or OFF. For example, the RS-232 standard has two voltages assigned for indicating ‘1’ (say, –5 Volts,) or ‘0’ (say, +5 Volts). Any other voltages outside a narrow band around these voltages are undefined.

The word **bit**, referred to often in the literature, is a contraction of the words **binary digit**.

The same principles for representing a binary number apply as in section 1 above. For example, the number 1011.1₂ means the following using Table A.2.

Weight	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	.	2 ⁻¹	2 ⁻²	2 ⁻³	2 ⁻⁴	2 ⁻⁵
	0	1	0	1	1	.	1	0	0	0	0

Table A.2
Binary weighting system

This translates into the following number:

$$\dots 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + \dots$$

The Most Significant Bit (MSB) in the above number is the left-most bit and is 1 with a weighting of 2³. The right-most bit is the Least Significant Bit (LSB) and is valued at 1 with a weighting of 2⁻¹.

Conversion between decimal and binary numbers

Table A.3 gives the conversion between decimal and binary numbers. Note that the binary equivalent of decimal 15 is written in binary form as 1111 (using 4 bits). This-4 bit binary grouping will have significance in hexadecimal arithmetic later. As expected, binary 0 is equivalent to decimal 0.

Decimal number	Binary equivalent
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

10	1010 contd...
contd... 11	1011
12	1100
13	1101
14	1110
15	1111

Table A.3
Equivalent binary and decimal numbers

The procedure to convert from a binary number to a decimal number is straightforward. For example, to convert 1101.01_2 to decimal; use the weighting factors for each bit to make the conversion.

$$1101.01_2 = 1 \times (2^3) + 1 \times (2^2) + 0 \times (2^1) + 1 \times (2^0) + 0(2^{-1}) + 1 \times (2^{-2})$$

This is equivalent to:

$$1101.01_2 = 1 \times (8) + 1 \times (4) + 0 \times (2) + 1 \times (1) + 0 \times (\frac{1}{2}) + 1 \times (\frac{1}{4})$$

This then works out to:

$$1101.01_2 = 8 + 4 + 0 + 1 + 0.25$$

$$1101.01_2 = 13.25$$

The conversion process from a decimal number to a binary number is slightly more complex. The procedure here is to repeatedly divide the decimal number by 2 until the quotient (the result of the division) is equal to zero. Each of the remainders forms the individual bits of the binary number.

For example, to convert decimal number 43_{10} to binary form:

2	43 remainder 1 (LSB)
2	21 remainder 1
2	10 remainder 0
2	5 remainder 1
2	2 remainder 0
2	1 remainder 1 (MSB)
	0

Table A.4
Illustration of decimal to binary conversion

This translates a number 43_{10} to 101011_2 .

Hexadecimal numbers

Most of the work done with computers and data communications systems is based on the Hexadecimal number system, with the base of 16 and uses the sequence of symbols:

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Hence, the number of $FA9.02_{16}$ (also written as $0xFA9.02$) would be represented as below in Table A.5

Weight	16^4	16^3	16^2	16^1	16^0	.	16^{-1}	16^{-2}	16^{-3}	16^{-4}	16^{-5}
	0	0	F	A	9	.	0	2	0	0	0

Table A.5
Hexadecimal weighting structure

This translates into the following number:

$$\dots 0 \times 16^4 + 0 \times 16^3 + F \times 16^2 + A \times 16^1 + 9 \times 16^0 + 0 \times 16^{-1} + 2 \times 16^{-2} + \dots$$

The MSD in the above number is the left-most symbol and is F with weighting of 16^2 . The right-most symbol is the LSD and is valued at 2 with a weighting of 16^{-2} .

Conversion between binary and hexadecimal

The conversion between binary and hexadecimal is effected by modifying Table A.6 to Table A.6 below:

Decimal number	Hexadecimal equivalent	Binary equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Table A.6
Relationship between decimal, binary and hexadecimal numbers

In converting the binary number 1000010011110111_2 to its hexadecimal equivalent the following procedure should be adopted. First, break up the binary number into groups of

four commencing from the least significant bit. Then equate the equivalent hex symbol to it (derived from Table A.6 above).

1000010011110111 becomes:

1000	...	0100	...	1111	...	0111 ₂
8	...	4	...	F	...	7 ₁₆

or 84F7₁₆.

In order to convert a hexadecimal number back to binary the procedure used above must be reversed.

For example, in converting from C9A4 to binary this becomes:

C	...	9	...	A	...	4 ₁₆
1100	...	1001	...	1010	...	0100 ₂

or 1100100110100100₂.

Binary arithmetic

Addition

Knowledge of binary addition is useful although it can be cumbersome. It is based on the following four combinations of adding binary numbers:

0	0	1	1
<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>
0	1	1	0 and carry 1

The carry 1 (or bit) is the only difficult part of the process. This addition of the individual bits of the number should be done sequentially from the LSB to the MSB (as in normal decimal arithmetic).

An example of addition is given below:

$$\begin{array}{r} 1010001001_2 \\ 0011101010_2 \\ \hline 1101110011_2 \end{array}$$

Subtraction

The most commonly used method of binary subtraction is to use 2's complement. This means that instead of subtracting two binary numbers (with the attendant problems such as having 'carry out' bits); the addition process is applied.

For example, take two numbers and subtract the one from the other as follows:

12	which is equivalent to:	1100
<u>-4</u>	Subtrahend	<u>-0100</u>
8	Result	1000

The two's complement is found by first complementing all the bits in the subtrahend and then adding 1 to the least significant bit.

Complementing the number results in 0100 becoming: 1011.

Add 1 to the least significant bit gives a two's complement number of: 1100.

Add 1100_2 to 1100_2 as follows:

1100

1100

1000 carry 1

(This is the same result as above).

Exclusive-OR (XOR)

Exclusive-OR is a procedure very commonly used with binary numbers in the error detection sequences of data communications. The result of an XOR operation on any two binary digits is the same as the **addition** of two digits **without the carry bit**. Consequently, this operation is sometimes also called the Modulo-2 adder. The truth table for XOR is shown below:

Bit 1	Bit 2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Table A.7
Exclusive-OR truth table

Hardware/firmware and software

The hardware refers to the physical components of a device, such as a computer, sensor, controller or data communications system. These are the physical items that one can see.

The software refers to the programs that are written by a user to control the actions of a microprocessor or a computer. These may be written in one of many different programming languages and may be changed by the user from time to time.

The firmware refers to the 'microprograms', usually residing in a read-only memory (ROM) and which normally cannot be changed by the user. The firmware usually controls the sequencing of a microprocessor. Consequently, it is a combination of hardware and software.

A port is the place of access to a device or a network used for the input or output of digital data signals.

Appendix B

Practical Exercises

PART 1: Serial communications

Exercise 1: Setting up

1.1 Overview

This section deals with setting up the software used for the USB-based serial communications exercises.

1.2 Software/ documentation required

- Listen32.zip (Win-Tech software)
- Quick Installation guide for Moxa UPort 1110/1130 USB-to-Serial adaptors
- Driver/documentation CD for Moxa UPort 1110/1130 USB-to-Serial adaptors
- Driver CD for PicoScope
- Virtual Serial Port Driver (Eltima)

NB The same driver disk is used for both the RS-232 and RS-485 converters (just install once). However, please ensure that you use only version 2.2 or later (and not 2.0) as the older version seems to cause hardware conflicts. The version number can be seen in the 'Readme' file on the CD. Also, do not use the old driver CD that was shipped with the PicoScope. Ensure that you have a CD with the latest version downloaded from the Internet.

1.3 Implementation

1.3.1 Moxa UPort adaptors

The following steps assume that the adaptor driver has not been installed yet. Please check before proceeding.

The installation procedure is very well documented in the Quick Installation Guide as well as the PDF version of the manual on the CD, hence it will not be repeated here. Basically just follow the prompts and ignore messages indicating that the software has not been tested to run on XP.

With the converter UNPLUGGED from the computer, run the installation program. This is located on the CD; go to *UPort 1100 ->Software->[Your operating system]*. Follow the prompts as per the Quick Installation Guide.

After installing the driver, plug in the UPort 1110 converter. Windows will detect the new hardware and proceed with the installation. Follow the prompts as per the Quick Installation Guide.

Run the Device Manager (*Start->Settings->Control Panel->Hardware*) and confirm that there are no conflicts (yellow circles with exclamation marks). This might occur if you use an older version of the driver disk.

The actual configuration of the converter will be covered in the next exercise.

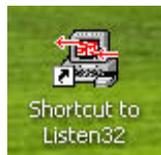
1.3.2 Listen32

The following steps assume that Listen32 has not been installed yet. Please check before proceeding.

For the sake of consistency (and to help the next person using the machine), please create a folder *C:\Listen32* and then unzip Listen32.zip into this folder. You will notice the following components:

LISTEN32.EXE

This is the executable. Right-click and create a shortcut. Move the shortcut to the desktop.



LSTN32.DLL

This is a hooking DLL for tracing COMM API calls from third-party applications. It should be copied to the Windows search path (*C:\Windows* or *C:\Windows\System*).

LISTEN32.HLP

This represents the on-line help menu support for the application and should remain in the same directory as Listen32.exe.

LISTEN32.CFG

Represents configurable parameters modified during execution of the application. It will automatically be created in the working directory upon exiting the application.

At this point Listen32 will run, but in demo mode only. Please type in the registration info to be found in a Word file on the distribution CD.

1.3.3 PicoScope

With the Scope unplugged from the computer, run the PicoFull setup program from CD. This will be on a CD supplied by IDC. Please do not use the CD that came with the device, as that is out of date. The following shortcut will appear on the desktop.



Plug the Scope into the USB port. Windows will detect the new hardware TWICE (once for each channel).

1.3.4 Virtual Serial Port Driver

Run the installation program ('vspd xp') from the distribution CD. The program will be fully functional, but for 14 days only. Please type in the registration info to be found in a Word file on the distribution CD. The following **shortcut** will appear on the desktop.



This concludes the software setup.

Exercise 2: RS-232 basics

2.1 Overview

The objective of this exercise is to give delegates an understanding of how asynchronous data serial data communications operate, using a breakout box on the COM port of a laptop computer. Since most laptops nowadays do not have physical COM ports, we will use a USB-to-Serial converter to accomplish this. The MOXA adapter used actually has an on-board UART, so the UART has simply been relocated from the motherboard to the USB adapter. The output of the serial port will be monitored by means of an oscilloscope and a voltmeter.

2.2 Hardware required

- 1x Laptop with at least two USB ports. If it has only one port, you may have to run the USB oscilloscope on a different machine
- 1x MOXA NPort U1110 (a.k.a UPort 1110) USB-to-serial (RS-232) adapter
- 1x 9-pin D-type female to 25-pin D-type male adapter plug ('straight') (DB-9F/DB-25M)
- 1x breakout box
- 1x Voltmeter
- 1x PicoScope 2202 USB oscilloscope, or similar, with probes

2.3 Software required

It is assumed that all software and drivers have been installed as per Exercise 1.

2.5 Implementation

2.5.1 Basic port setup

The keystrokes described here are for Windows XP.

Turn on the computer

Plug the RS-232 adapter into the USB port and visually confirm that the red LED on the converter lights up.

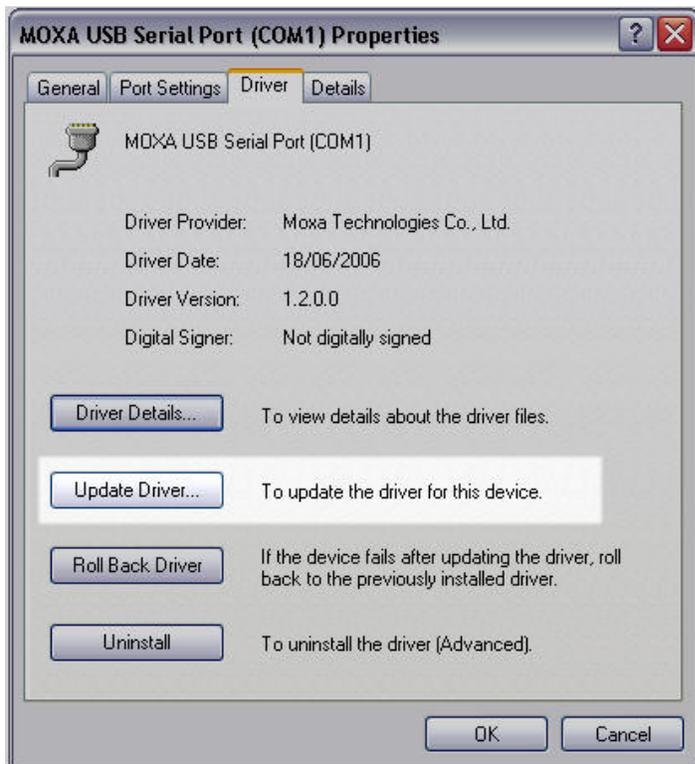


The purpose of the three LEDs are as follows (refer to the operating manual on CD)

Name	Description
Active	This LED indicates normal operation. If the driver is installed correctly and the adaptor is plugged into a functioning USB port, the Active LED will light up and remain on. If the LED does not light up, there may be a problem with the adaptor, the driver installation, or PC configuration.
TxD	This LED blinks when the adaptor transmits data through its COM port.
RxD	This LED blinks when the adaptor receives data through its COM port.



If the red LED on the converter did not light up earlier, or a yellow exclamation mark shows up against the MOXA entries in the Device Manager, it could be that the driver settings have been altered because someone has previously used the MOXA UPort 1130 USB-to-RS-485 converter on this machine.

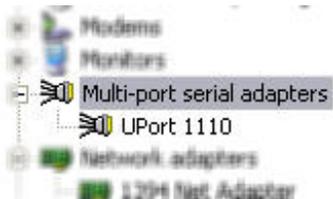


In this case, simply select the Drivers tab and click on 'Update Driver'.

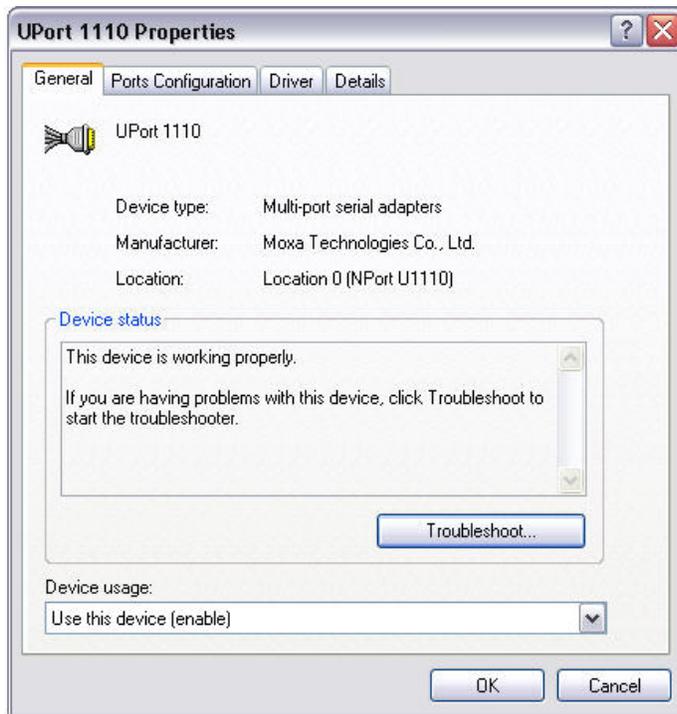
Let's assume that the LED lights up.

Click: Start->Control Panel ->System->Hardware->Device Manager.

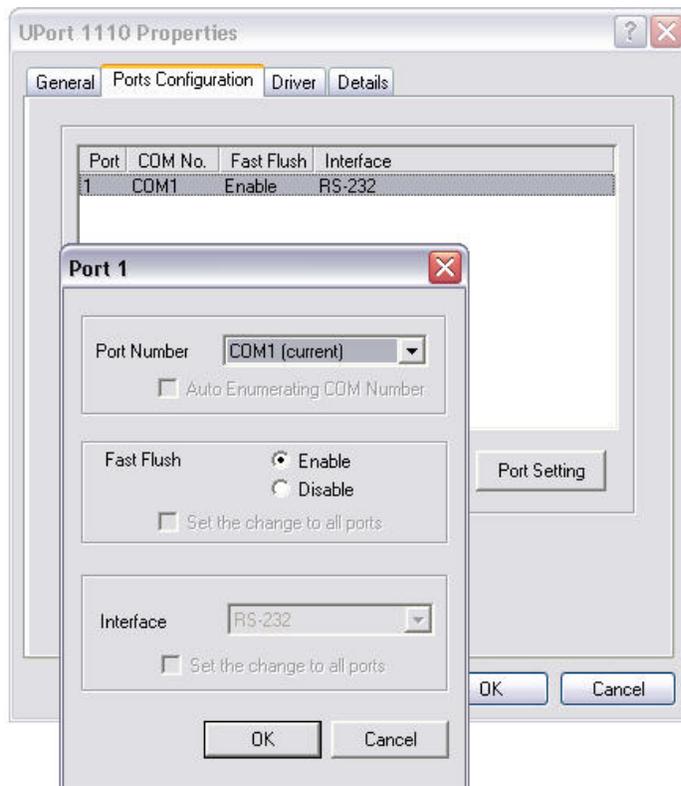
Locate the entry that reads: 'Multi-port serial adapters' and expand it to show the UPort 1110.



Double-click (or right-click->Properties) on 'UPort 1110' to obtain its properties window.

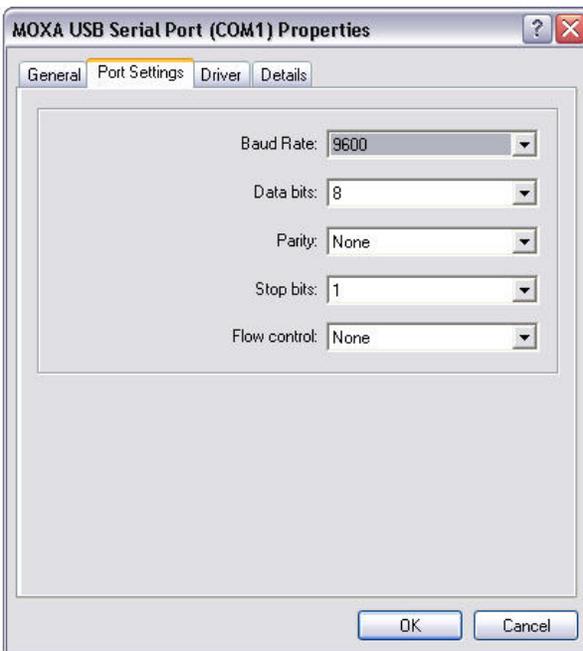
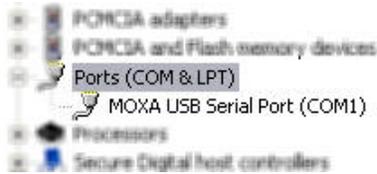


Select the 'Ports Configuration' tab and confirm that the correct COM port has been allocated to the UPort 1110.



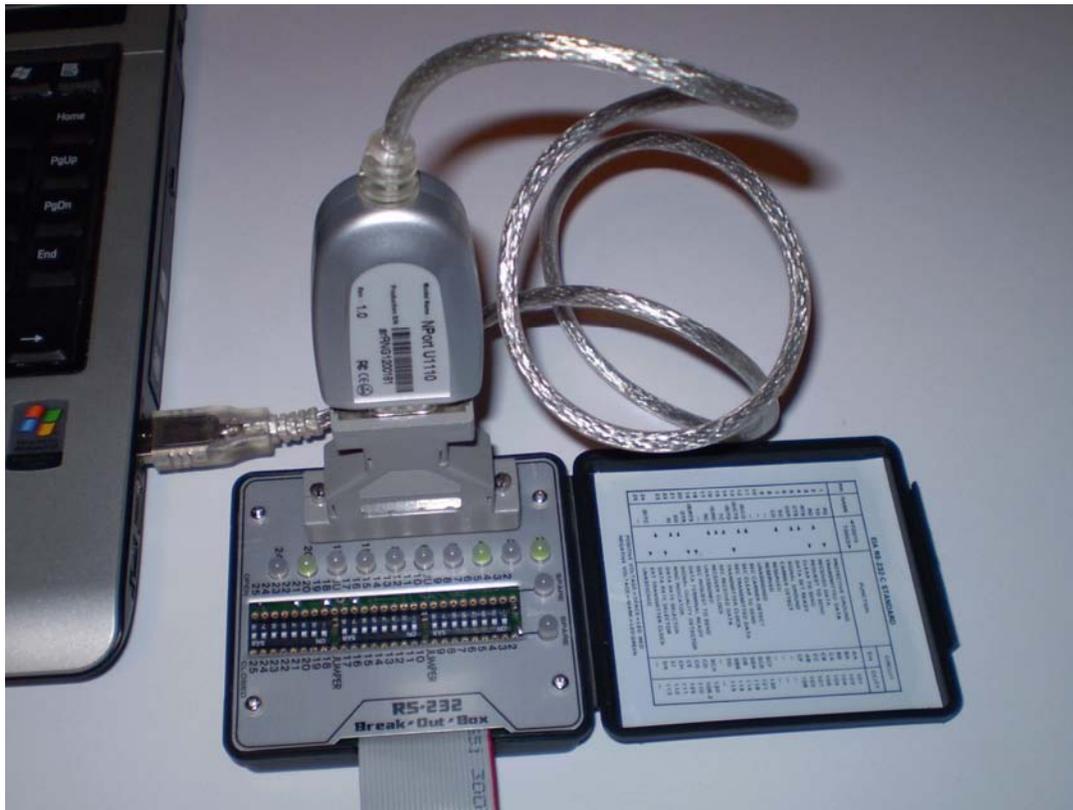
If the port number is incorrect, click on 'Port Setting' and select an appropriate one. On a machine without any COM ports any one could be selected, but on machines with existing COM ports you have to select a different one to avoid conflicts. On this particular machine there are no physical COM ports, so we selected COM1.

As an additional check, go back to the Device Manager and locate 'MOXA USB Serial Port (COM1)' (this is the COM port number we selected earlier). Double-click (or right-click->Properties) on the MOXA entry to obtain the MOXA serial port properties. Select the 'Port Settings' tab and check the Universal Asynchronous Receiver Transmitter (UART) settings. These settings can, however, be overwritten by application programs such as Listen32.



2.5.2 Confirmation of idle voltages

Connect the USB converter to the breakout box by means of the DB-9/DB-25 plug. You may have to remove the two hexagonal lock nuts on the 9 pin connector as they interfere with those on the adaptor.

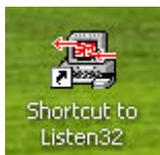


Verify the following:

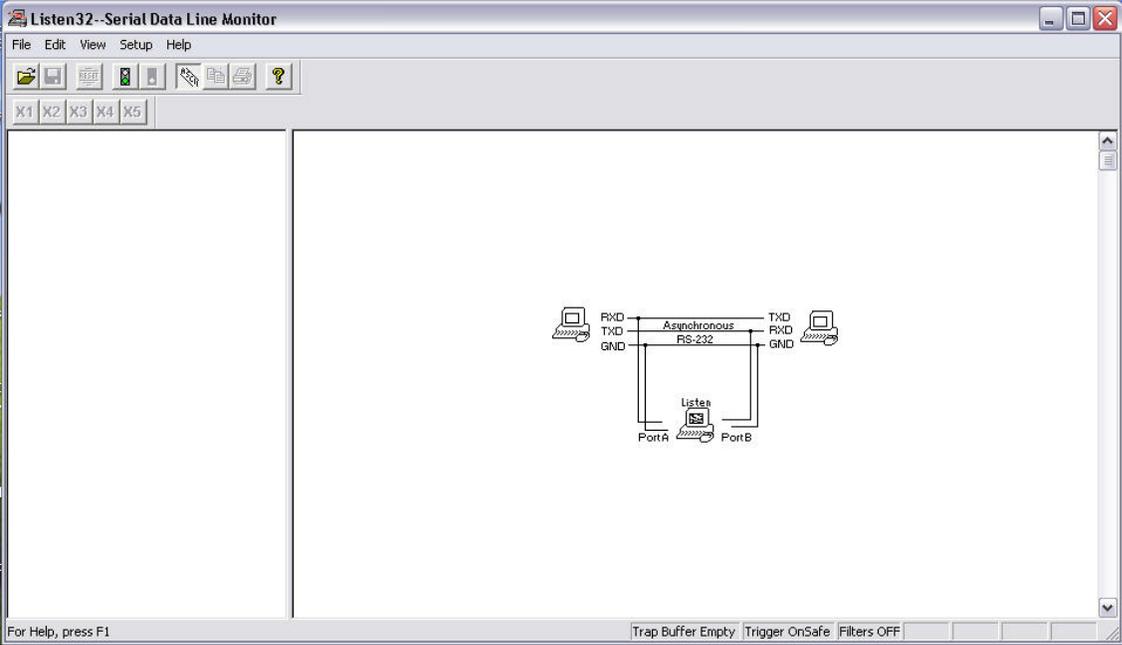
- TxD is a '1' i.e. between -5V and -25V with respect to ground (note value) and corresponding LED is green
- RxD is zero (not connected to DCE)
- RTS is '0' i.e. between -5V and -25V with respect to ground and corresponding LED is green
- CTS is zero (not connected to DCE)
- DTR is '0' i.e. between -5V and -25V with respect to ground and corresponding LED is green
- DSR is zero (not connected)

2.5.3 Transmitting a single character

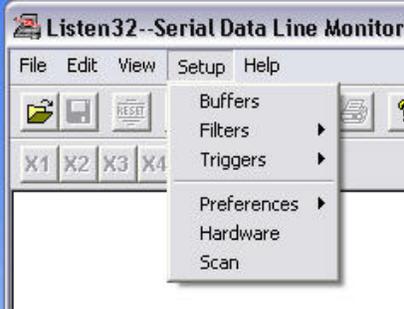
Invoke Listen32 by clicking on the desktop icon.



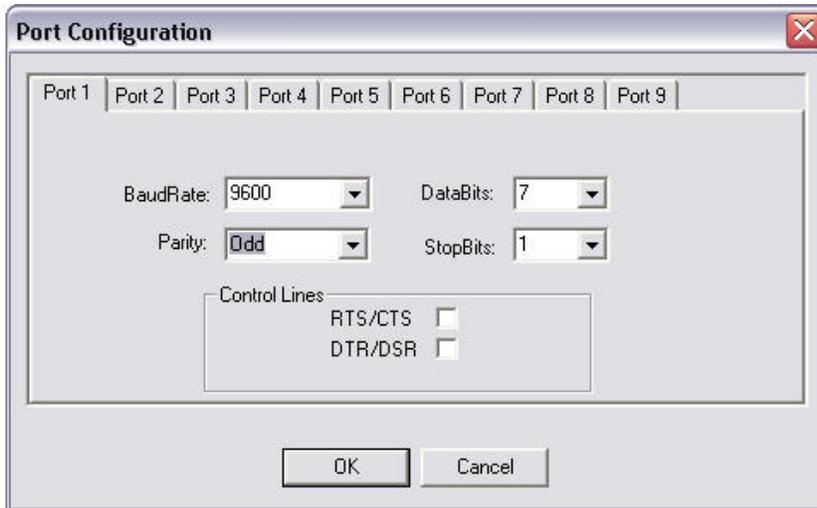
Listen32 will open up:



Click Setup->Hardware:



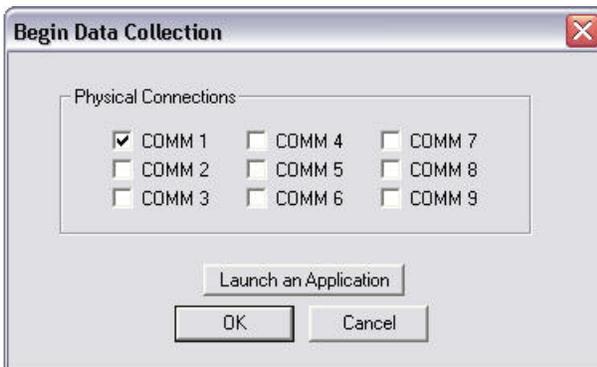
Then set the UART parameters for the appropriate COM port (COM1 in this case). Set the parameters exactly as per the following figure: 9600, 7, O, 1.



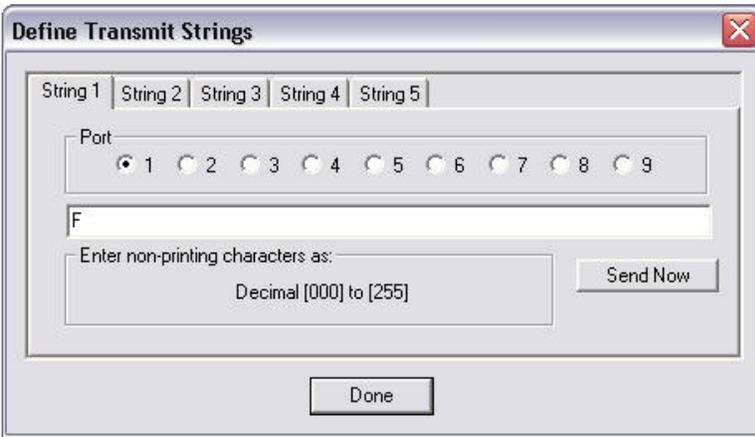
Next, start the port monitoring process by clicking on the green traffic light (click red to stop).



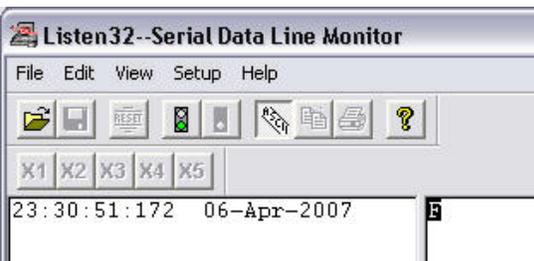
The 'Begin Data Collection' window will appear. Select the appropriate port (COM1 in this case).



Then click Edit->Transmit string. The 'Define Transmit Strings' window will appear.



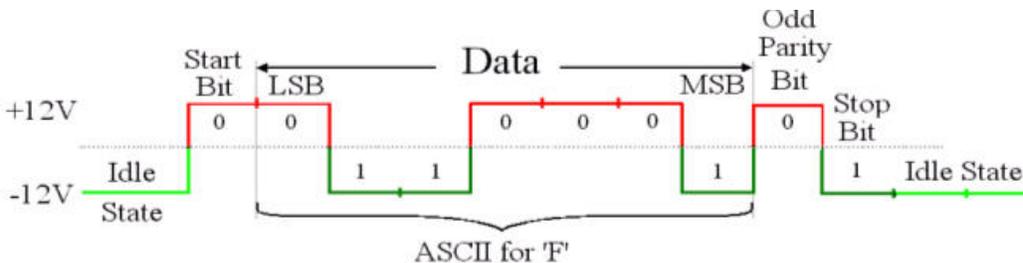
Select String 1 for the appropriate, and enter upper case F. Click 'Send Now' to transmit. The transmission will be logged. Note: At 9600 baud the flickering of the TxD LED on the converter due to a single character will not be noticed. However, sending several Fs (say, 20 to 30) will allow you to visually confirm that transmission is actually taking place.



Send a single 'F' again, but this time monitor the TxD output (against ground) with the PicoScope. If the machine running Listen32 has only one USB port, run the scope on an adjacent machine. Set the PicoScope as follows:

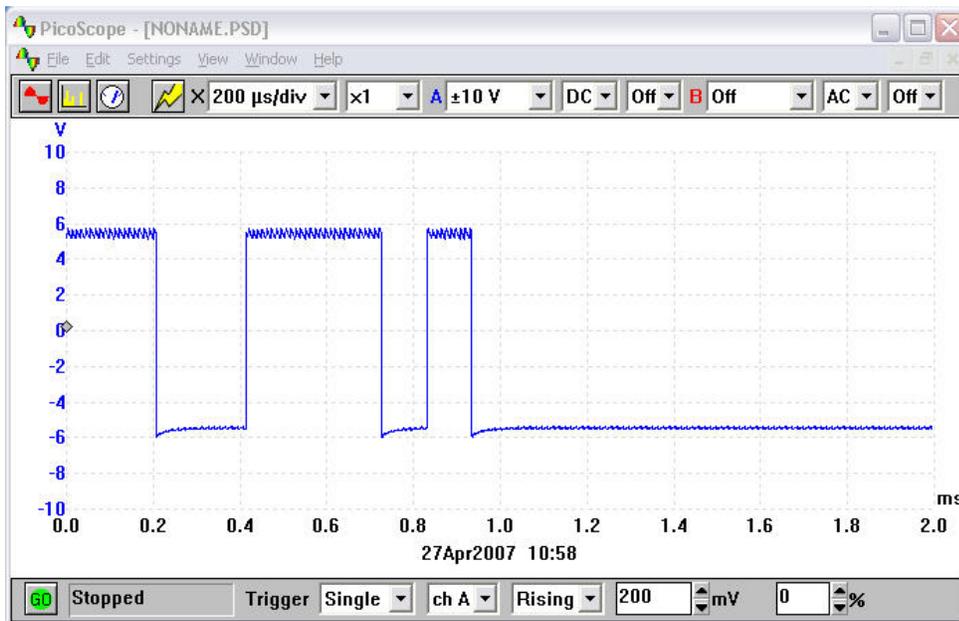
- Trigger: Single trigger, Channel A , rising edge
- X-axis: 200 microseconds per division
- Magnification: X1
- Y-axis full scale: +/-10V
- Coupling: DC

Click 'Go' in the bottom left hand corner and then send 'F' again. In theory, the waveform should look like this.



The actual waveform is shown in the following figure. The voltage levels are around -5.5V, +5.5V due to the specific machine (Toshiba Satellite) to which the converter is connected. For laptops, these voltages seem to vary from around +/- 5.5V to +/-9V and they are often not symmetrical with respect to ground.

Do not worry too much about attaching the ground clip on the probe, as the PicoScope and the USB converter are connected to the same ground via USB.



2.5.4 Other exercises involving single character transmission

Choose an arbitrary number from the ASCII table, draw its waveform on a piece of paper, then press the corresponding key and corroborate the waveform. You may change the parity setting. Click on View->ASCII table (in Listen32) if you do not have a copy of the ASCII table handy.

Next, select a character (change the port settings if necessary) that will produce a square wave with a 50% duty cycle.

2.5.5 Loop-back test

Perform a loop-back test by inserting a jumper between pins 2 and 3 of the breakout box. What do you see on the screen when you type a character? What do you see on the breakout box in terms of pins 2 and 3?

Exercise 3: RS-232 point-to-point communication

3.1 Overview

The objective of this exercise is to set up communications at 9600 Baud between two laptops (DTEs) by means of a null modem cable

3.2 Hardware required

- 2 x Laptop with MOXA UPort 1110 already set up.
- 1x null modem cable

3.3 Software required

It is assumed that Listen32 and the USB-to-serial adapter driver have already been installed

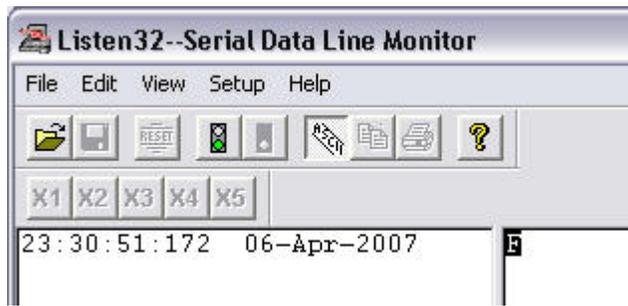
3.4 Implementation:

Interconnect the two COM ports with a null modem cable.



Follow the steps outlined in the previous exercise and set the ports up to 9600,7,N,1

Type any characters and check if the communication is working. The transmitted character(s) should show up on both machines.



Now click on Setup->Preferences->Num base, and alternate the display between decimal, hex and binary; each time verifying the letter(s) that you have typed against the ASCII table.

Exercise 4: RS-232 via virtual null modem

4.1 Overview

The objective of this exercise is to demonstrate the use of a virtual null modem to check communications between two applications on the same PC.

4.2 Hardware required

- 1 x Laptop

4.3 Software required

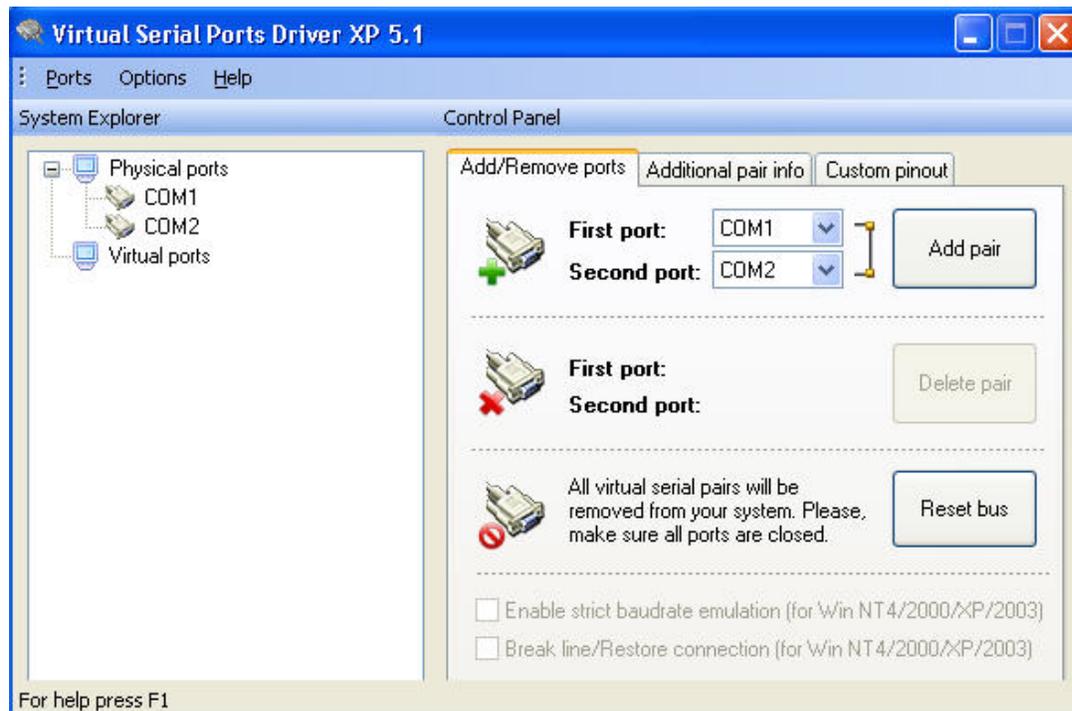
It is assumed that Listen32 and the Eltima software have already been installed in Exercise 1.

4.4 Implementation:

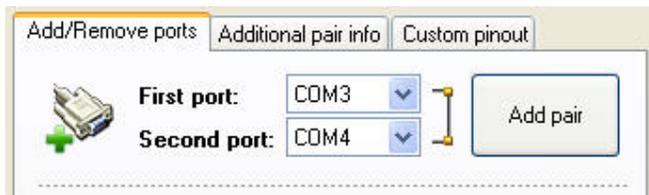
4.4.1 Setting up the virtual null modem



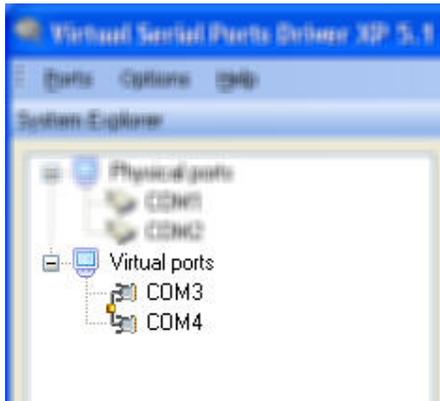
Run VSPD CP 5 from the desktop by clicking on the icon.



Notice the two physical ports (COM1 and COM2) detected on the computer, and shown under 'Physical Ports' in the System Explorer box. Now select any two non-conflicting ports for the 'First port' and 'Second port' in the top right-hand corner. In this case we have selected COM3 and COM4.

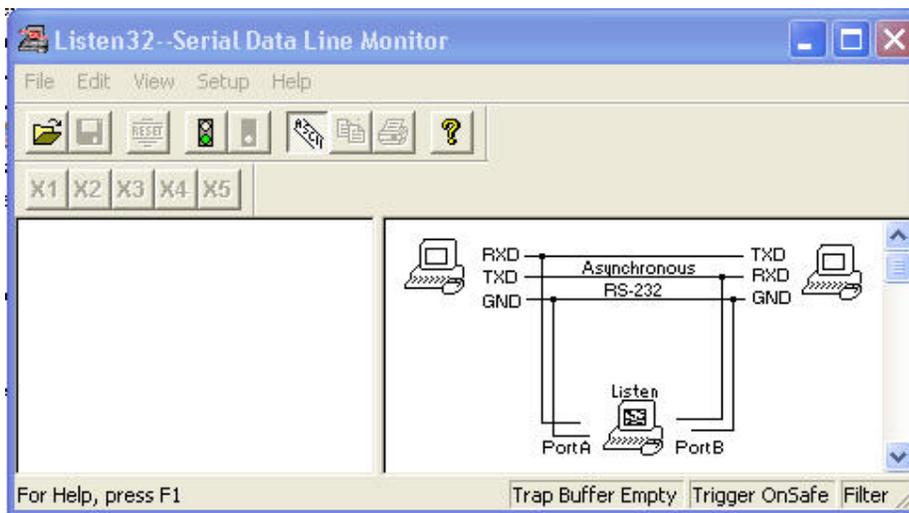


Click 'Add pair'. COM3 and COM4 will now appear as 'Virtual Ports' in the System Explorer box, with a null modem connection between them.

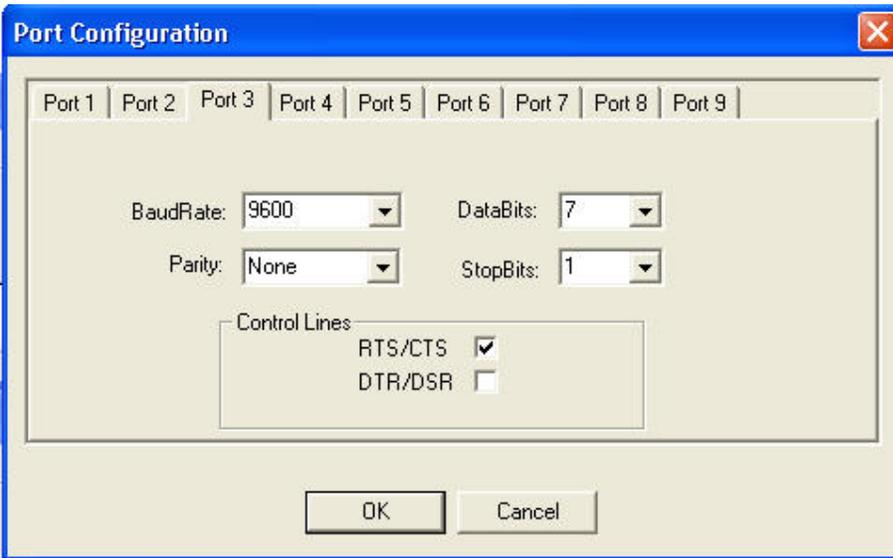


4.4.2 Running two applications across the virtual null modem

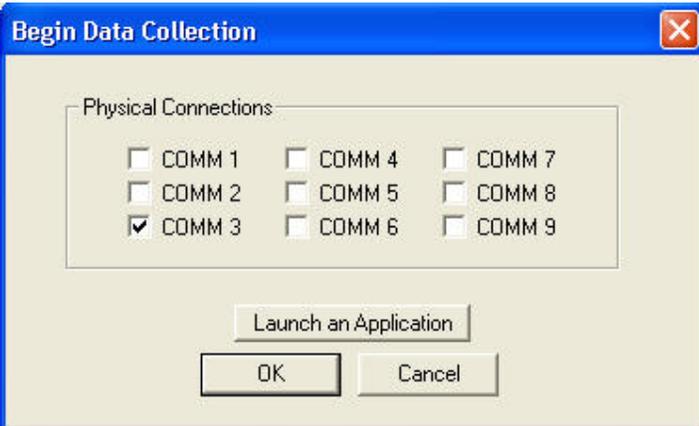
Run Listen32 by clicking on the desktop icon.



Do this again to create a second instance of the program, and position them side by side. Set them up in the same way, but associate them with COM3 and COM4 respectively. The next figure shows the port configuration for one instance. Note the checked RTS/CTS box to enable hardware handshaking.



Also start the data collection on both instances, using COM3 and COM4.



Finally, send data from one instance to the other, and try it in both directions.

Exercise 5: RS-485 basics

5.1 Overview

This exercise will build upon the previous RS-232 exercise, and will introduce the use of RS-485

5.2 Hardware required

- 2x laptops with USB ports
- 2x MOXA UPort 1130
- Voltmeter
- Oscilloscope (if available)
- 2 meters Cat5 twisted pair wire
- Screwdriver, wire stripper

5.3 Software required

It is assumed that the Listen32, PicoScope software and USB-to-serial adapter drivers have been installed.

5.4 Implementation

5.4.1 Basic setup

With the exception of its outputs, the UPort 1130 is identical to the UPort 1110.



Plug the UPort 1130 into the USB port in place of the U1110. Check that the red LED lights up. Attach the screw terminal to the DB-9 connector. If the red LED does not light up, update the driver as described in paragraph 2.5.1 of Exercise 2.

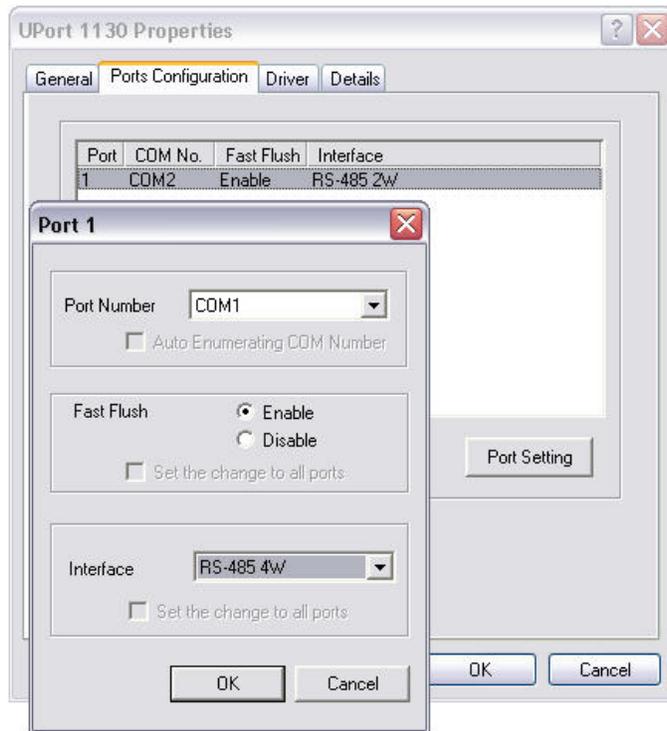


The first step is to confirm the device configuration.

As in the case of the UPort 1110, go to the Device manager and locate the 'Multi-port adapters'



entry. Open it up to show the UPort 1130, then double-click (or right-click->Properties) on the 1130 entry to open the UPort 1130 Properties window.



There are three transmission options possible, viz.:

- RS-485 4-wire
- RS-485 2-wire
- RS-422

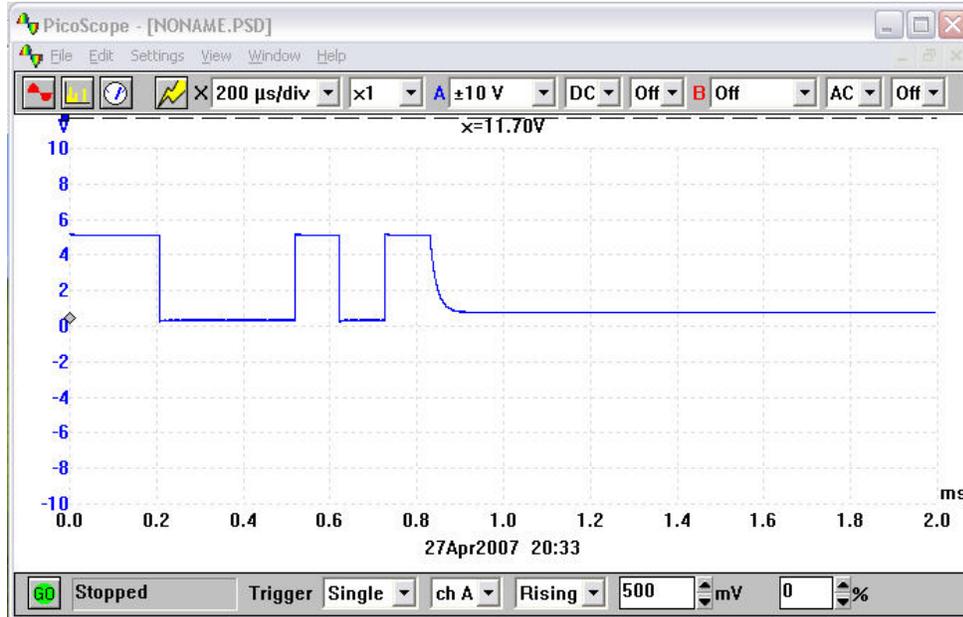
5.4.2 RS-485 4-wire

Check that '4W' mode has been selected (refer to figure above). Interconnect two machines via pins 1, 2, 3 and 4, using 2 pairs of Cat5. Observe the colors so that pin 1 or 'TxD-(A)' (a.k.a. 'A-wire out', shown as 'T+' on the connector- very confusing) on the one machine goes to pin 4 or 'RxD-(A)' (a.k.a. 'A wire in', shown as R+ on the connector) on the other machine. In similar fashion, connect pin 2 or 'TxD+(B)' (a.k.a. 'B-wire out', shown as 'T-' on the connector) on the one machine to 'RxD+(B)' (a.k.a. 'B wire in', shown as R- on the connector) on the other machine. In plain English, connect T+ and T- on the one terminal strip to the corresponding R+ and R- on the other terminal strip. This, of course, has to be done in both directions.

DB9 (male)	Pin	UPort 1110	UPort 1130	
		RS-232	RS-422 4-wire RS-485	2-wire RS-485
	1	DCD (in)	TxD-(A)	-
	2	RxD (in)	TxD+(B)	-
	3	TxD (out)	RxD+(B)	Data+(B)
	4	DTR (out)	RxD-(A)	Data-(A)
	5	GND	GND	GND
	6	DSR (in)	-	-
	7	RTS (out)	-	-
	8	CTS (in)	-	-

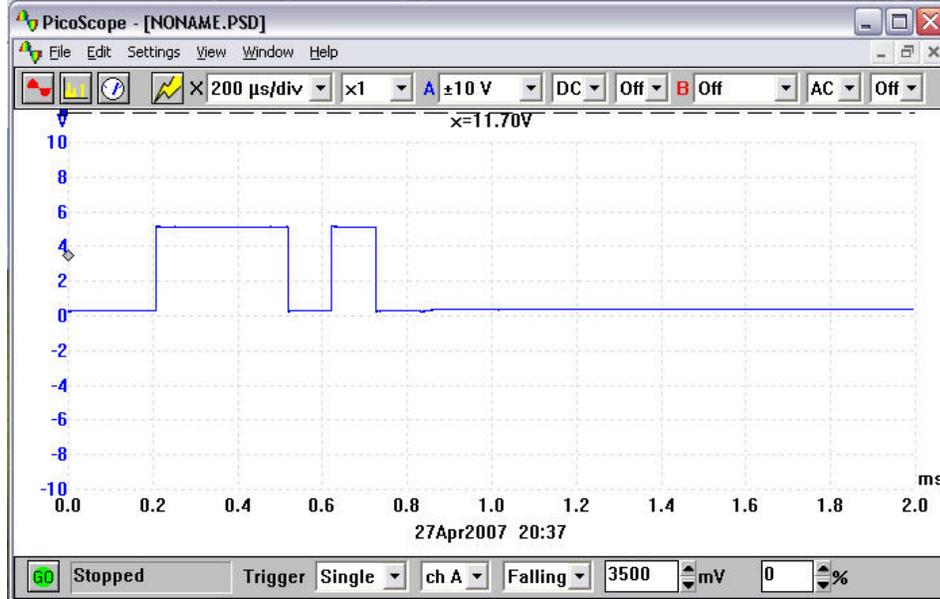
Use Listen32 as in the previous exercises to send characters from one laptop to the other. Observe the results on the display.

Observe the voltage on T+ (i.e. the 'A' wire) on the one machine by means of the PicoScope and send an 'F'. Use the scope settings as shown below for a baud rate of 9600. The trigger level needs to be higher than 200mV as the RS-485 signal does not go right down to 0V (in this case we used 500mV). Remember to click the GO button before sending the 'F'. The transmission starts off with a low-to-high transition that is not evident from this picture, not shown because the actual triggering took place on that transition. Notice how the output at the end of the last bit (at approximately 8.5 mS) is not driven back to 0V but, instead, just decays to a residual level because the driver has gone into a high-impedance state at that point.



Now observe T- (i.e. the 'B' wire) when sending the same character. In this case the transmission begins with a high-to-low transition, on which the triggering takes place.

These scope traces can be obtained simultaneously with two probes (traces showing up in red and blue). We have done them separately here simply because the two colors would not show up in B/W print.



Notice the slight 'step' where the output goes into the high-impedance state.

5.4.3 RS-485 2-wire

Change the adapter configuration from '4W' to '2W'. This time we are not cross-connecting the outputs and inputs. Instead, all outputs are connected in parallel, since they are connected to a bus. Data-(A) is connected to Data-(A) (pin 3) and Data+(B) is connected to Data+(B) (pin 4); regardless of the number of laptops used.

Repeat the previous exercise by sending characters across the link.

PART 2: Network communications

Exercise 6: Setting up Ethernet network

6.1 Overview

This exercise illustrates the setup of a very simple 10/100 Mbps Ethernet LAN. This will serve as the basis for our subsequent exercises.

6.2 Hardware required

The hardware required for this practical is as follows:

- Two or more laptops
- 1x Ethernet network interface card (NIC) per computer, unless the Ethernet interface is already built-in
- 1x Cat5 UTP fly-lead per computer
- 1 x hub. Note: the kit contains two small 5 port hubs (SureCom) that can be wired together if necessary.

6.3 Software required

The software required is as follows

- All the computers loaded with Windows 2000/XP
- Driver software for the NICs as provided by the manufacturer.

6.4 Implementation

To set up a basic network insert the NICs, if required, into the appropriate slots. Then connect each NIC to the hub with a flylead.



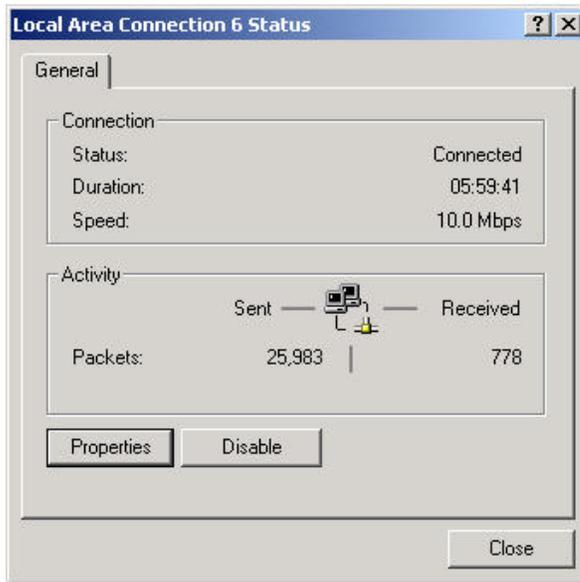
Hub connections

When the computer prompts you to log on, use the user name and password supplied by the instructor. You HAVE to log on if you require access to the LAN.

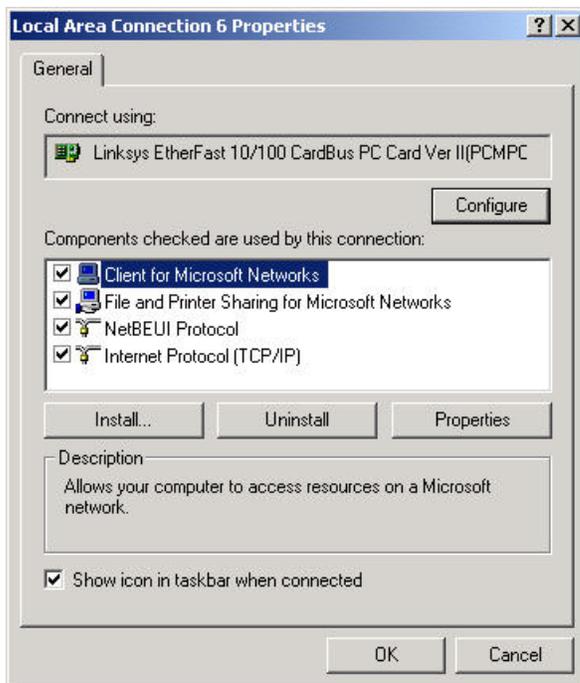
Now click *Start -> Settings -> Control panel -> Network and Dial-up Connections*. The red 'X' indicates that the NIC representing 'local area connection 3' is not currently connected to a hub.



Double-click the local area connection you wish to configure (e.g. local area connection 6 in this example). The following will appear:



Click on *Properties*.



Notice the following:

- The *client* (Client for Microsoft Networks) enables your computer to connect to other computers and access their resources
- ‘Connect using.....’ refers to the hardware device that physically connects your computer to the network, i.e. the Ethernet NIC in this case. The presence of the ‘adapter’ icon indicates that the driver for the card has been installed.
- The *protocols* occupying OSI layer 3 and 4 are, in our case, NetBEUI and TCP/IP. All hosts must use the same protocols in order to communicate with each other. Although TCP/IP would have been

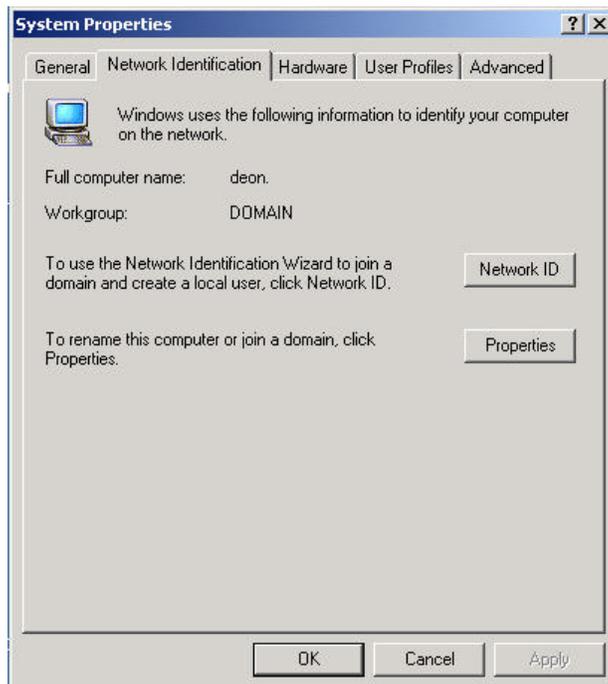
sufficient, NetBEUI had been added to facilitate file sharing between computers using NetBIOS names such as 'computer1'. It will also allow you to access resources on the other machines in case TCP/IP is not working e.g. because of incorrect IP addresses.

- The *service* (File and Printer Sharing for Microsoft Networks) enables your computer to share its resources such as files, printers and other services with other computers on the network.

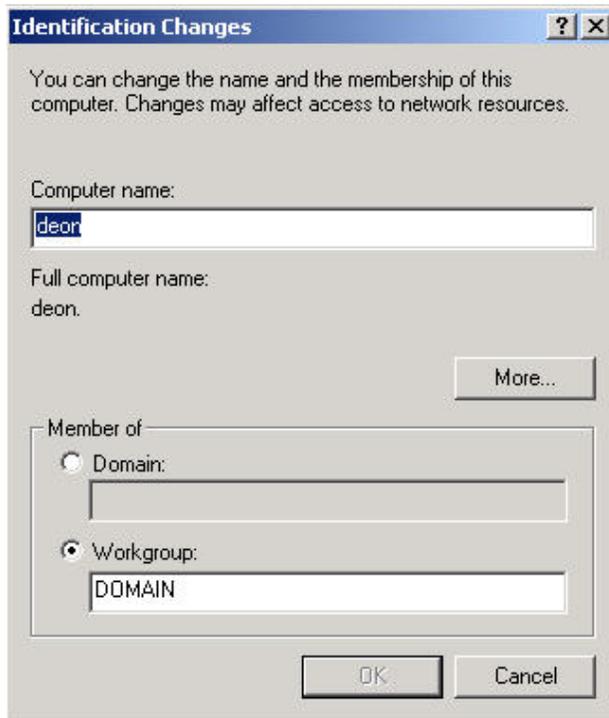
As soon as the NIC is inserted into the computer for the first time, the operating system prompts the user to provide the necessary device drivers. Thus the Client, Service, Adapter and Protocol components are automatically installed. TCP/IP is normally installed by default. If NetBEUI is not installed, you may have a problem seeing all the interconnected machines under 'My Network Places'. You can add it with the 'Install' button just below the list of checked components.

Click *OK* and *Close*.

Now **right** click on *My Computer*, and then click on *Properties* and select *Network Identification*.



Now click on *Properties* and check/edit the computer name as well as the workgroup or domain name.



Computer name is a unique identity for your host computer on the network e.g. 'Computer1' (check with the instructor). This is also referred to as the NetBIOS name.

Workgroup is the name of the workgroup in which the host computer will appear e.g. 'idc' (check with the instructor). In the snag above, the workgroup name is shown as 'domain' but only because that was the name of the workgroup at that given moment in time; it does not refer to an actual domain.

You will be prompted for a user name and a password after rebooting. ALWAYS log on otherwise you will not get access to the network.

The computer should now be suitably configured for networking, and you can explore some troubleshooting of the wiring and hubs.

You will notice that the NIC and associated port on the hub both have their link integrity lights illuminated. Unplug the UTP cable and observe both lights extinguish. The link integrity lights are illuminated only when both devices are operational and correctly wired.

Unplug the UTP cable at both the NIC and hub and replace it with the crossover cable. Note that both sets of lights are extinguished because the wiring is now incorrect. If the hub has a crossover (Uplink) port then plug the crossover cable into this and observe the link integrity lights illuminate as the wiring is once again correct.

Plug the crossover cable directly into another NIC and observe both sets of lights illuminate. This is a useful way of connecting two computers without a hub, for file transfers etc.

If the link integrity light at the hub is not illuminated simply swap that cable to another working port on the hub to check that the port itself is not faulty.

Whenever data is being sent on the network the network activity lights on ALL ports and NICs will flicker since all messages on an Ethernet network are sent to all users. As other computers are being booted up onto the network activity should be noted. (NB: This is true for a hub, but not for a switch).

Some of the newer devices such as access points we will be using have auto MDIX on all or some ports, meaning that you can use either straight or crossover cable and the port will automatically configure itself.

Invoke Windows (e.g. right click on *Start*, then click *Explore*).

Select the *C drive* and then using the right mouse button right click on the same and select *Sharing*.



Select *Share this folder* and provide a *share name* for this resource. Click on *Permissions* to set access permissions. Click on *Apply* followed by *OK*.

Notice the small hand symbol under the C drive icon indicating that this drive is now shared.

Other resources may similarly be shared.

This concludes the basic network setup of host computers. To verify whether the network setup is OK, double click on *My network places* and *Computers near me* to view the other computers on the network. To see the shared resources available on a specific computer, just click its icon.

You may now browse the network neighborhood and the other computers' shared resources. You may also transfer files back and forth across the network, view a remote file, execute a remote program, etc., all from the comfort of the local host computer.

Notes:

During boot-up, there may be errors displayed stating that there is already another computer on the network with the same name, and all networking services would be disabled. Follow the procedure described above and choose another computer name and restart

Some users may not be able to browse the network neighborhood ('My network places') since they have not logged in with an appropriate username and password at the time of boot-up. Log-off and logon as a valid user. Others may not be able to either see their own host computer or one or more of the other host computers. This may be a result of not sharing any resources. Follow the procedure described above and share some resource, such as a folder.

Exercise 7: Configuring IP

7.1 Overview

This exercise will review the basic steps in configuring the IP addressing details, and introduces some important troubleshooting utilities.

7.2 Hardware Required

Wired LAN as set up in Exercise 6

Software Required

7.3 Windows-based ping/trace utilities such as TJPingPro and AngryIP

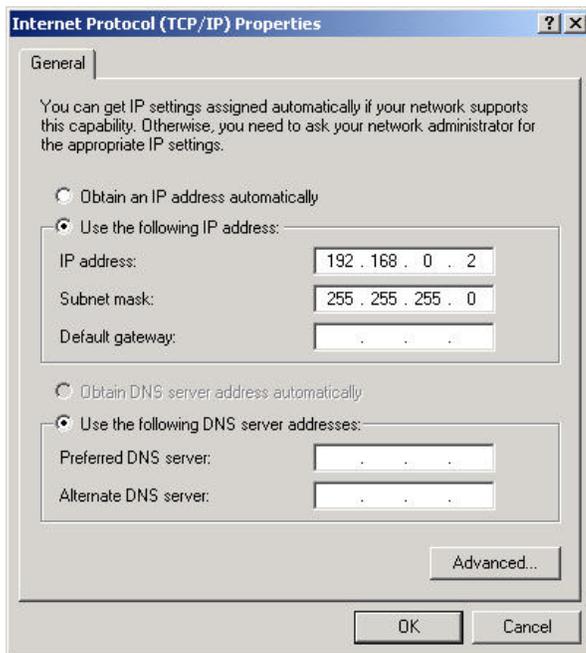
7.4 Implementation

Call up the *Local Area Connection Status* box (as in the previous exercise) and check if TCP/IP is present. In the very unlikely situation of it not being there, install now (Add ->protocol).

Select TCP/IP and click on the *Properties* button.

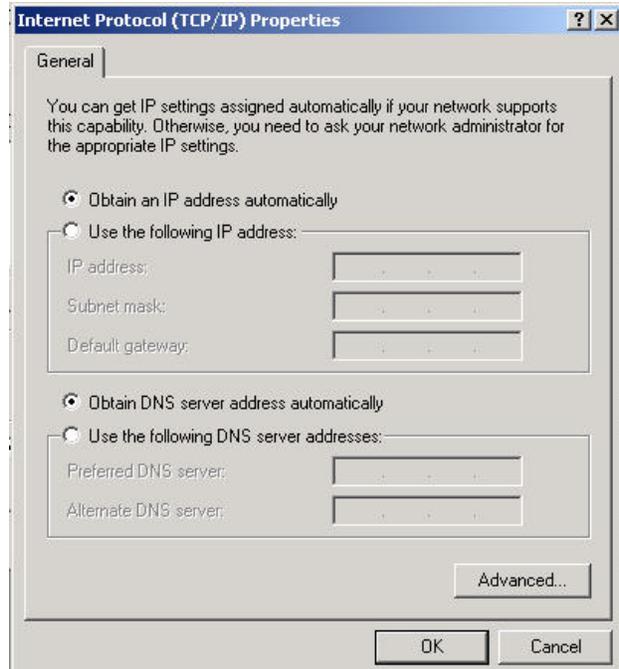
Choose '*Use the following IP address*' and then supply an IP address of the form: W.X.Y.Z (for example 192.168.0.x) where the first three dotted decimals (W.X.Y) represent the network identity (NetID), and the last dotted decimal (Z) represents the host computer identity (HostID) for your machine. Thus we could have IP addresses in the range 192.168.0.1 to 192.168.0.254 inclusive.

Enter the subnet mask of 255.255.255.0 to indicate a default 'class C' network. Click *Apply* ->*OK* -> *Close*.



All other configuration information remains unchanged from the previous exercise. With Windows98 you have to restart at this point, with 2000 and XP it is not necessary.

An alternative is to let a DHCP server allocate an IP address. In this case, select *Obtain an IP address automatically*. The address will be issued when you restart your computer, alternatively you can type `ipconfig /renew` at the DOS prompt.



To verify whether all configuration information has been entered correctly, you can run `wntipcfg.exe`. This can be invoked by clicking on *Start*, selecting *Run*, typing `wntipcfg` in the dialog box, and then clicking *OK*. This utility is not part of 2000/XP but can be downloaded from www.microsoft.com. Alternatively, type `ipconfig /all` at the DOS prompt.

The IP configuration will be displayed along with the Ethernet adapter information. Note the ‘Physical Address’ (MAC address).

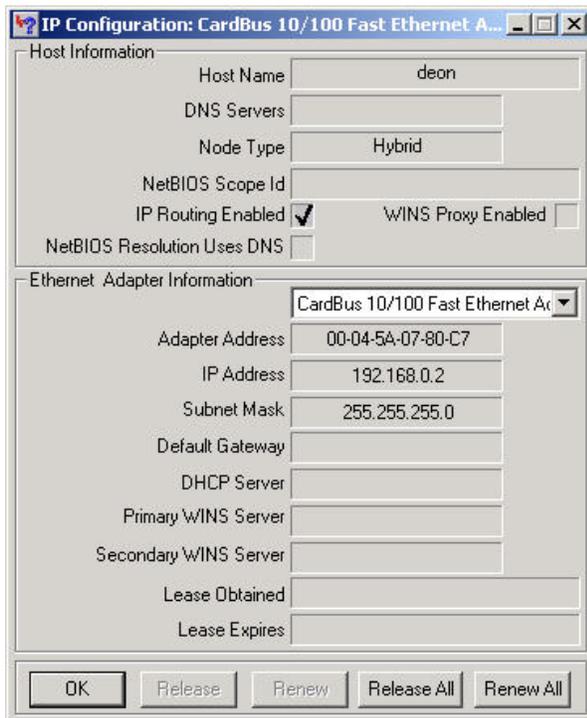
The following is the result of `ipconfig /all`.

```

Ethernet adapter Local Area Connection 6:

    Connection-specific DNS Suffix  . : 
    Description . . . . . : Linksys EtherFast 10/100 CardBus PC
    Card Ver II (PCMP200 vII) #2
    Physical Address. . . . . : 00-04-5A-07-80-C7
    DHCP Enabled. . . . . : No
    IP Address. . . . . : 192.168.0.2
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 
    DNS Servers . . . . . : 
  
```

A similar result is obtained with `wntipcfg`.



To verify whether this entire configuration is working correctly within the local host computer, type `ping localhost` or `ping 127.0.0.1` in DOS.

If there is a response received from 127.0.0.1 (a universal local loop-back IP address, common to all computers), then it may be assumed that TCP/IP is now correctly configured. Note that the host need not be connected to the network for this command to function.

To verify communication between the local host and remote computers, ping the remote computer’s IP address and/or NetBIOS name. If a reply is received then it is indication that the remote host computer is powered, connected to the network, and the TCP/IP configuration on that remote host computer is correct.

```

C:\>ping 192.168.0.1
Pinging 192.168.0.1 with 32 bytes of data:
Reply from 192.168.0.1: bytes=32 time<10ms TTL=127

Ping statistics for 192.168.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>_

```

Notes:

During boot-up, there may be an error message stating that there is already another computer on the network with the same IP address, and all networking services will be disabled. Follow the procedure described above, choose another IP address and restart.

Some users may not be able to browse the network neighborhood because they have not logged in with an appropriate username and password. They need to log off and log on again valid users. Others may not be able to either see their own computer or other computers on the network. This is a result of not sharing any resources. Follow the procedure described in exercise 1 and share a resource.

Even though we enter only one ping command, we receive four replies. This is due to the default programmed within the ping.exe executable that sends out 4 sequential ICMP *echo requests*. The echo request message includes 32 bytes of ICMP data and therefore we observe *bytes=32*. The *time=1ms* denotes the total round trip time taken for the ping message to be transmitted and a reply to be received, in milliseconds. *TTL=128* indicates that the replying host is permitting this reply to traverse through a total of 128 routers. Why do we need this?

At the DOS prompt, type '*ping*' (without any options) and it will display a list of the various switches that are available to customize the ping message.

Also type in *arp -a* after pinging, in order to see the arp cache.

If time permits, use third party ping/trace utilities such as AngryIP and TJPingPro. With TJPingPro select *options*, then set a range of IP addresses (e.g.192.168.0.1 to 192.168.0.10) in the top right hand corner. Click OK, followed by 'Sequential Scan' (*Seqscan*).

Exercise 8: Protocol analysis

8.1 Overview

This exercise is a brief introduction to protocol analysis, using a high-quality freeware analyzer.

8.2 Hardware required

Existing network infrastructure

8.3 Software required

Latest version of 'Ethereal' (know 'Wireshark) plus the capture engine 'winpcap' version 4 or later.

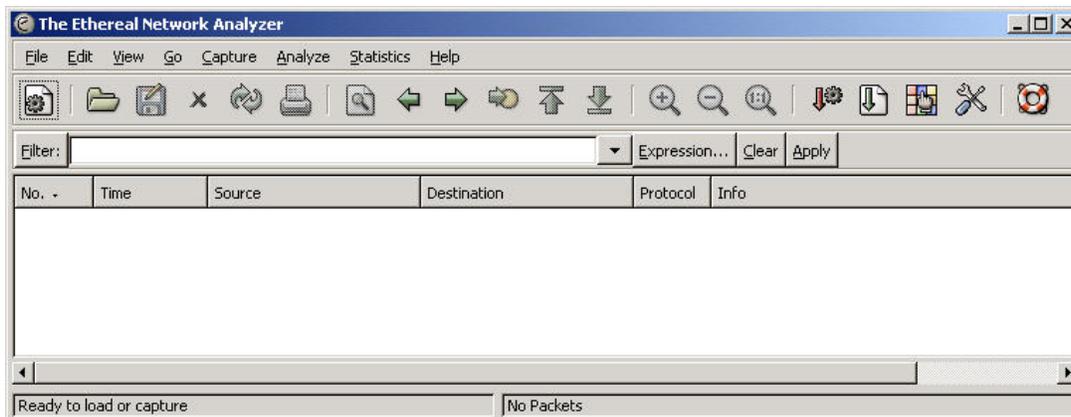
Implementation

First install winpcap, then install Ethereal. Now set up your machine to ping another machine repetitively, e.g. `ping 192.168.0.1 -t`.

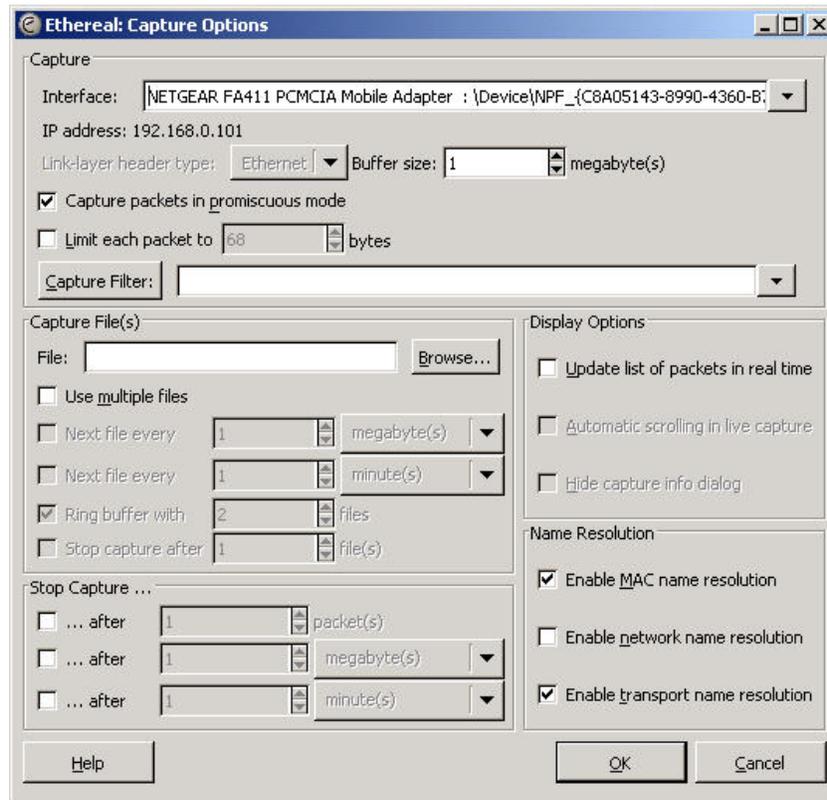
Click on the Ethereal GUI, or click start->programs->Ethereal



The Ethereal control panel will appear.

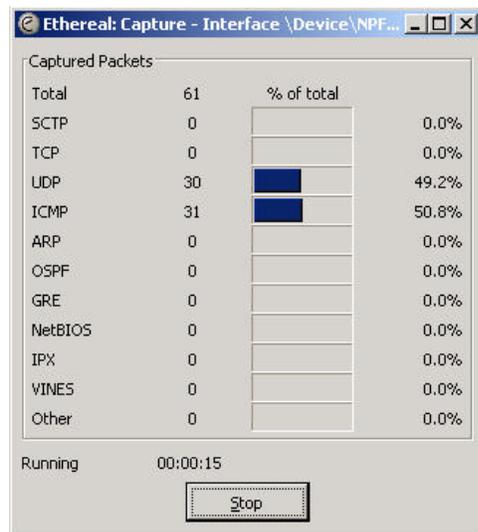


Click *capture->start* and select the appropriate network interface (in this case a Netgear 411 card).



You may experiment with the other default settings, but for now leave them as they are. It is possible to set up rather complex filters to screen out unwanted packets, but in this exercise we will set up the display filters afterwards.

Click *OK*. Capturing will commence and you will be able to observe some statistics regarding the packets being captured.



When you have captured a few ICMP packets, click *stop* and the captured packets will be displayed.

The screenshot shows the Ethereal (Wireshark) interface with a list of captured packets. The first packet is selected, and its details are shown below the list. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.101	192.168.0.110	ICMP	Echo (ping) request
2	0.001591	192.168.0.110	192.168.0.101	ICMP	Echo (ping) reply
3	0.187934	192.168.0.101	192.168.0.255	NBNS	Name query NB VXIFRAME.BIZ<00>
4	0.938948	192.168.0.101	192.168.0.255	NBNS	Name query NB VXIFRAME.BIZ<00>
5	1.001395	192.168.0.101	192.168.0.110	ICMP	Echo (ping) request
6	1.002966	192.168.0.110	192.168.0.101	ICMP	Echo (ping) reply
7	1.730888	192.168.0.101	192.168.0.255	NBNS	Name query NB VXIFRAME.BIZ<00>
8	2.002770	192.168.0.101	192.168.0.110	ICMP	Echo (ping) request
9	2.004344	192.168.0.110	192.168.0.101	ICMP	Echo (ping) reply
10	2.481165	192.168.0.101	192.168.0.255	NBNS	Name query NB VXIFRAME.BIZ<00>
11	3.004370	192.168.0.101	192.168.0.110	ICMP	Echo (ping) request
12	3.005952	192.168.0.110	192.168.0.101	ICMP	Echo (ping) reply
13	3.232248	192.168.0.101	192.168.0.255	NBNS	Name query NB VXIFRAME.BIZ<00>
14	4.007105	192.168.0.101	192.168.0.110	ICMP	Echo (ping) request

The details pane for the selected packet (Frame 1) shows:

- Frame 1 (74 bytes on wire, 74 bytes captured)
- Ethernet II, Src: 00:09:5b:0b:2a:f1, Dst: 00:0d:88:80:35:6b
- Internet Protocol, Src Addr: 192.168.0.101 (192.168.0.101), Dst Addr: 192.168.0.110 (192.168.0.110)
- Internet Control Message Protocol

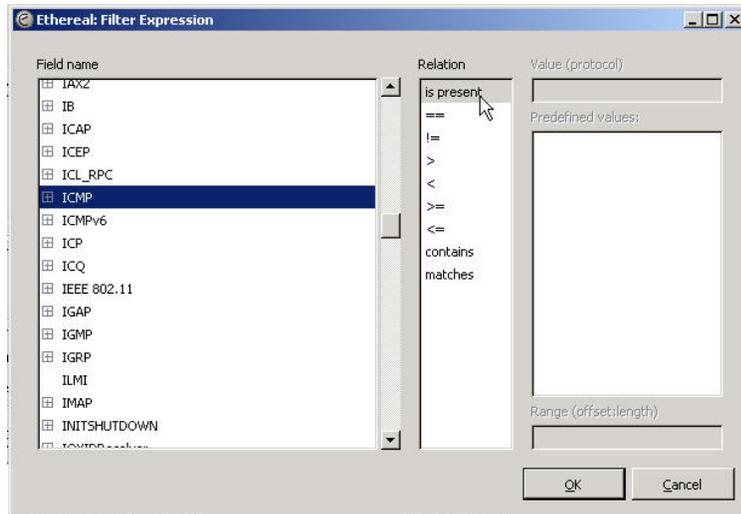
The raw packet data is displayed in hexadecimal and ASCII:

```

0000  00 0d 88 80 35 6b 00 09 5b 0b 2a f1 08 00 45 00  ...5k... [.*...E.
0010  00 3c 7d 76 00 00 80 01 3b 27 c0 a8 00 65 c0 a8  .<}\v... ;...E.
0020  00 6e 08 00 81 59 02 00 ca 02 61 62 63 64 65 66  .n...Y... .abcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76  ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67 68 69                    wabcdefg hi
    
```

Now click *analyze->display filters->expression...*

The screenshot shows the 'Ethereal: Display Filter' dialog box. It has two tabs: 'Edit' and 'Filter'. The 'Filter' tab is active, showing a large empty text area for entering filter expressions. On the left side, there are 'New' and 'Delete' buttons. Below the text area, there are 'Properties' fields for 'Filter name:' and 'Filter string:', with an 'Expression...' button next to the latter. At the bottom, there are buttons for 'Help', 'OK', 'Apply', 'Save', and 'Close'.



Select '+ICMP' and 'is present', then click *OK*. Only the ping packets (ICMP) will now be displayed.

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.0.101	192.168.0.110	ICMP	Echo (ping) request
2	0.001591	192.168.0.110	192.168.0.101	ICMP	Echo (ping) reply
5	1.001395	192.168.0.101	192.168.0.110	ICMP	Echo (ping) request
6	1.002966	192.168.0.110	192.168.0.101	ICMP	Echo (ping) reply
8	2.002770	192.168.0.101	192.168.0.110	ICMP	Echo (ping) request
9	2.004344	192.168.0.110	192.168.0.101	ICMP	Echo (ping) reply

Highlight one of the echo request packets, then expand the Ethernet header (click on the '+' next to 'Ethernet') in the middle screen.

```

Ethernet II, Src: 00:09:5b:0b:2a:f1, Dst: 00:0d:88:80:35:6b
  Destination: 00:0d:88:80:35:6b (D-Link_80:35:6b)
  Source: 00:09:5b:0b:2a:f1 (Netgear_0b:2a:f1)
  Type: IP (0x0800)

```

Note the source and destination MAC addresses, with the first 6 characters decoded to show the manufacturer. In this case the destination is the D-Link DI-624 access point. Now expand the IP header.

```

Internet Protocol, Src Addr: 192.168.0.101 (192.168.0.101), Dst Addr: 192.168.0.110 (192.168.0.110)
  Version: 4
  Header length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
  Total Length: 60
  Identification: 0x7d76 (32118)
  Flags: 0x00
    0... = Reserved bit: Not set
    .0.. = Don't fragment: Not set
    ..0. = More fragments: Not set
  Fragment offset: 0
  Time to live: 128
  Protocol: ICMP (0x01)
  Header checksum: 0x3b27 (correct)
  Source: 192.168.0.101 (192.168.0.101)
  Destination: 192.168.0.110 (192.168.0.110)

```

Then the ICMP header.

```

Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x8159 (correct)
  Identifier: 0x0200
  Sequence number: 0xca02
  Data (32 bytes)

```

In the bottom section of the screen you can observe the ICMP data (a, b, c, d, f, g, etc) embedded in the message.

```

0020 00 6e 08 00 81 59 02 00 ca 02 61 62 63 64 65 66 .n...Y.. ..abcdef
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmn opqrstuv
0040 77 61 62 63 64 65 66 67 68 69 wabcdefghijkl hi

```

Exercise 9: RTU MOde

6.1 Overview

In this exercise we will run Modbus Serial in RTU mode over an RS-232 link.

6.2 Hardware Required

- 2x laptops
 - 1x null modem cable with DB-9 connectors
- NB: If a second machine or a null modem cable is not available, then both Modbus Poll and Modbus Slave can be run on the same machine via a virtual null modem.

6.3 Software Required

- Modbus Poll v3.60
- Modbus Slave v3.10

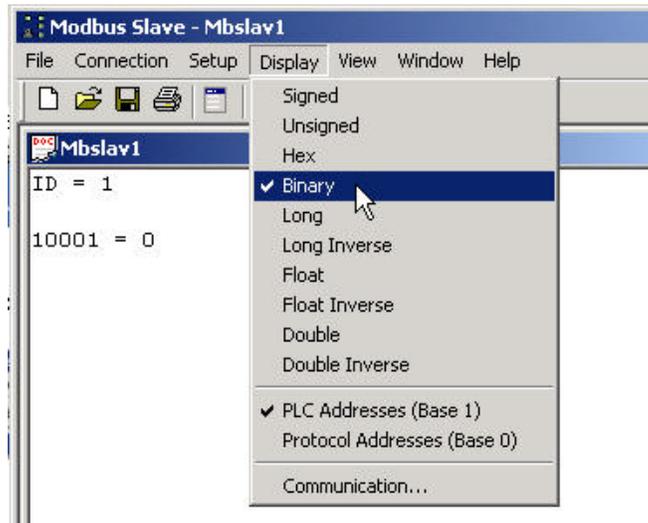
6.4 Implementation:

Run Modbus Slave on the remote machine by clicking on the desktop icon.

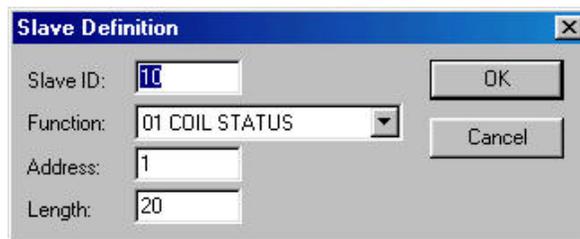


The control panel will open up. Assume that the current setup is as shown.

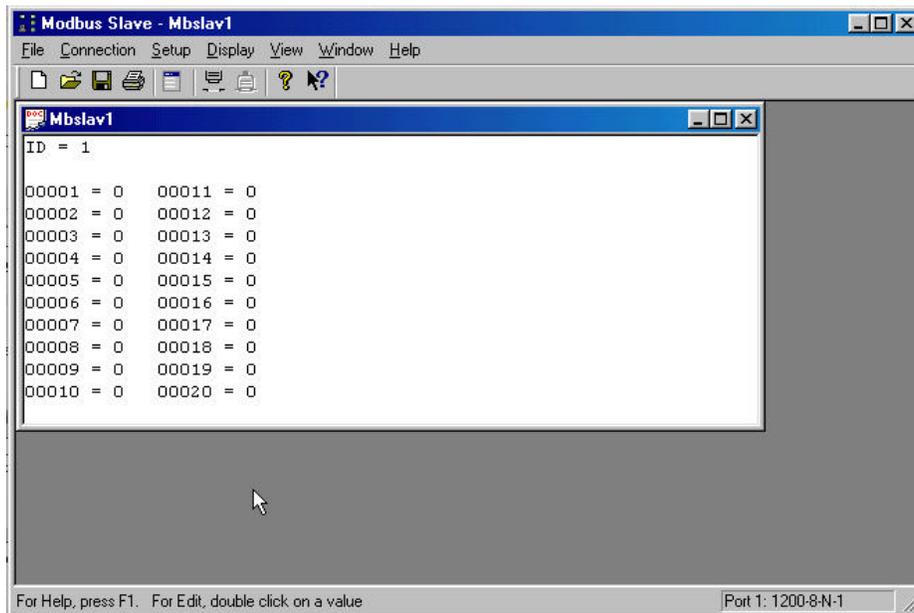
Click *display*; select *binary* and *PLC addresses (Base 1)*. In this mode we will use the absolute addresses of the registers as opposed to their relative addresses.



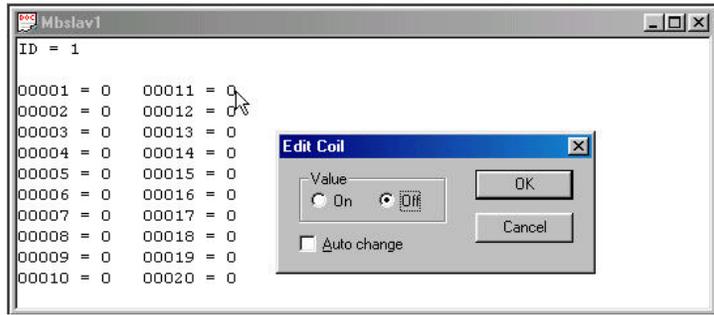
Now hit F2 or click *Setup-> Slave definition*. Set the slave up as follows.



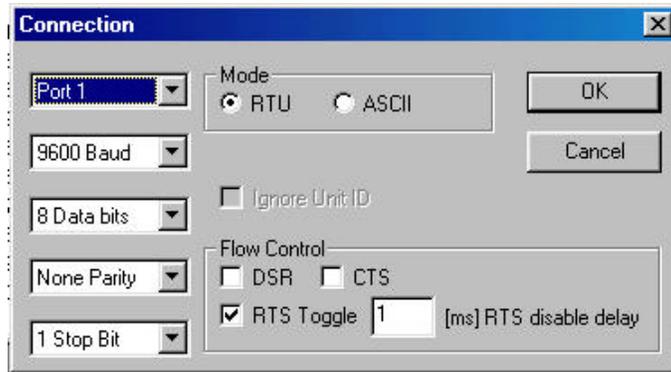
The slave address in this case is 10 (decimal). Function = 01 (read coil status). Address = 1 refers to physical PLC address 00001 (protocol address = offset = 0), and length = 20 means that there are 20 consecutive coils. These will now show up as follows.



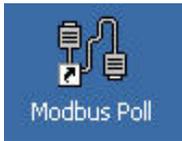
To edit any coil, just double-click on it and toggle between on and off.



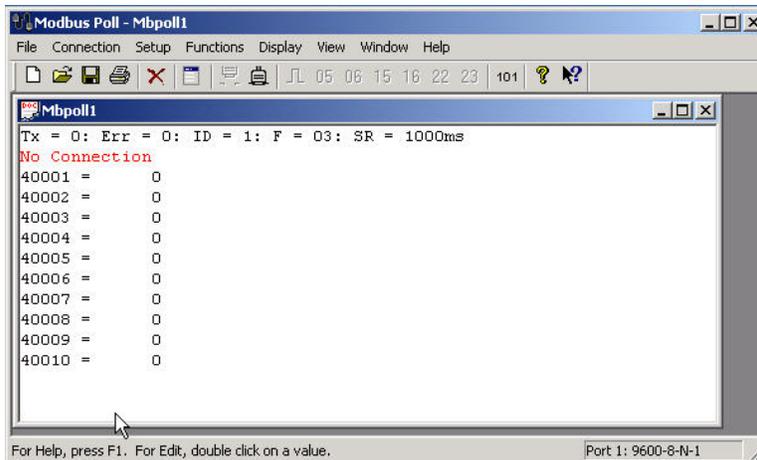
Click *Connection*->*connect* and set up the serial communications parameters as shown. Ensure that RTU mode is selected. Then click OK.



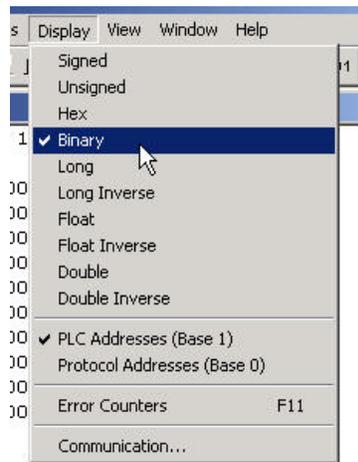
With the slave setup complete, invoke Modbus Poll on the other machine by clicking on the desktop icon.



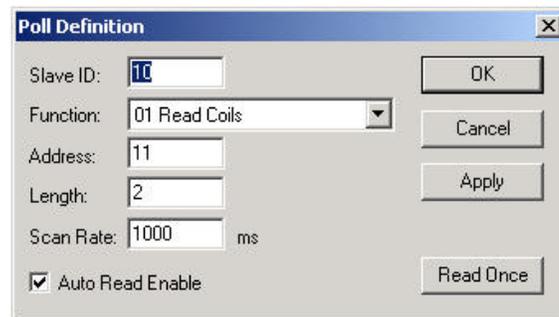
The Modbus Poll control panel will appear.



Click *display* and set it up as follows (the same as for the slave).

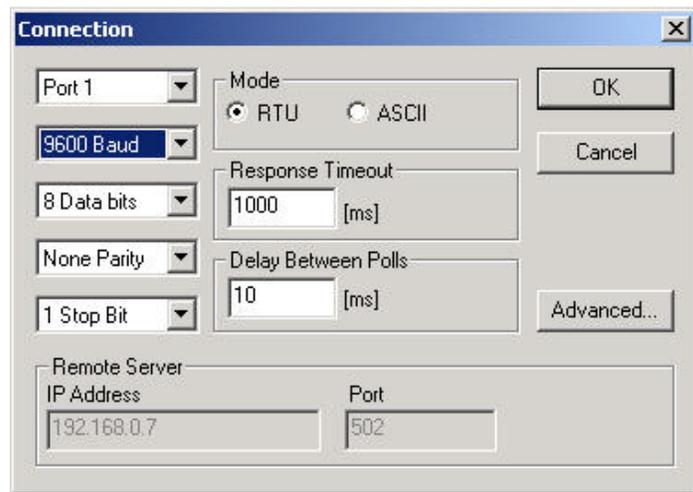


Now hit F2 or click *Setup-> Poll definition*.



In the poll definition above, coils 11 and 12 on slave 10 will be read once every second. Click *OK*.

Hit F3 or click *Connection->connect*. Select COM1 and 9600, 8, N, 1. Ensure that RTU mode is selected.

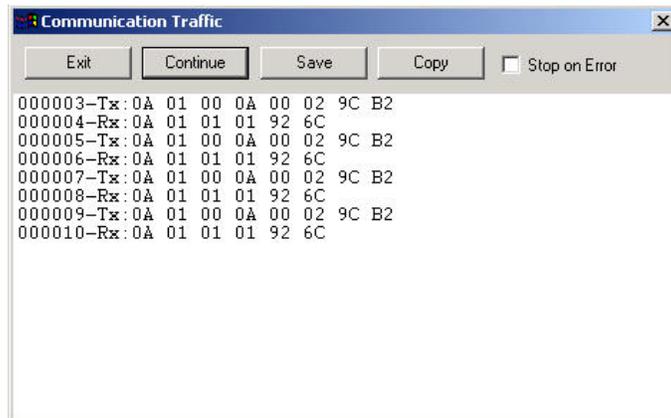


If all goes according to plan, connection will be established. If a red 'timeout' message appears, do the following.

- Click *Disconnect* on both sides
- Check if the poll definitions match (slave addresses and function codes)
- Check that the coils read by the Poll program are a SUBSET of the coils defined by the Slave program, and not the other way around
- Check that the communications parameters are the same for both sides

- Reconnect on both sides

Click *Display->communications* and observe the traffic between master and slave. Remember that it is as seen from the master's perspective; the display on the slave will be the other way around.



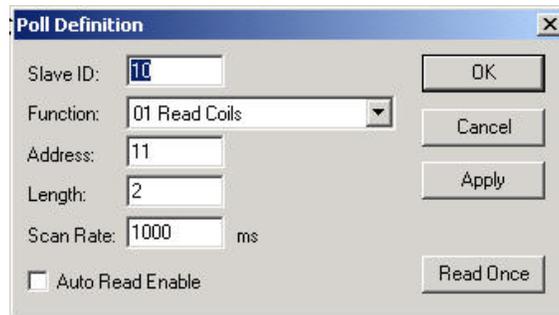
Tx refers to the Modbus request. 0x means Hex.

- Slave = 0x0A (i.e. 10 decimal)
- Function code = 0x01 (1 = Read coil status)
- Initial coil address = 0x000A (i.e. decimal 10 relative or decimal 11 absolute)
- Number of coils = 0x0002
- CRC = 0x9CB2

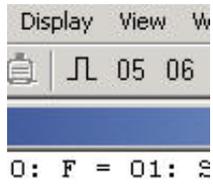
Rx refers to the Modbus Rx response:

- Slave = 0x0A
- Function = 0x01
- Byte count = 0x01
- Coil status = 0x01 = 00000001 Binary, i.e. coil A is set and coil B is cleared.
- CRC = 0x926C

Disconnect on both sides. Terminate the Modbus Slave program and run Listen32 on the slave machine. Set up Listen32 to display data in Hex, and start monitoring the COM port. Set up Modbus Poll to transmit once only (Auto Read Enable un-checked)



Execute the function once by clocking on the 'pulse' sign.



Now stop logging and observe the transmitted Modbus request

Listen32 should display the Modbus ADU, viz. <0A><01><00><0A><00><02><9C><B2>.

Appendix C

Cyclic redundancy check (CRC) program listing

```
#include <malloc.h>
#include <stdlib.h>
#include <stdio.h>

#ifndef DAT // to avoid multiple definitions due to order
#define DAT // of #includes
struct dat // this is a data structure used for passing between objects
{
    int    addr, // address of device
          fcn, // number of function
          dcount, // number of elements in data
          crc; // crc check code
    char  *data; // pointer to data
};
#endif

static unsigned CRC16=0xA001; // Polynomial used for CRC-16 checksum

union doub { // union for CRC check
    unsigned i; // as an unsigned word
    char c[2]; // as two characters
    struct bits{ // as a bitfield
```

```

        unsigned msb:1;           // most significant bit
        unsigned:14;
        unsigned lsb:1;          // least significant bit
    } b;
};

void whatcrc(struct dat *d, int mode) // calculate a CRC given a dat structure
{
    char *msg;                    // buffer to message
    int i,j,len;                  // counters and length of message
    union doub sck,byt;

    len=(mode?2:0)+2+(d->dcount); // calculate length (data is the only field
    without fixed length)

    msg = (char *)malloc(len); // allocate space for message buffer
    if (!msg)                    // didn't happen? Say so and quit.

        {
            printf("Sorry, but I couldn't allocate memory\n");
            exit(1);
        }

    // Load the msg buffer
    msg[0]=d->addr;                // load the addr field as byte 1
    msg[1]=d->fcn;                 // load the fcn field as byte 2

    for (i=0;i<d->dcount;++i)
        msg[i+2]=d->data[i];

    if (mode)
        {
            msg[i+2]=(d->crc&0xFF00)>>8;
            msg[i+3]=d->crc&0x00FF;
        }

    // CRC check algorithm live!

    sck.i=0xFFFF;                // set initial remainder
    for (i=0;i<len;++i)           // for each byte in buffer

```

```

{
  byt.c[0]=msg[i];    // put the current character at end of working byt
  byt.c[1]=0;        // set start of byt to 0
  sck.i^=byt.i;      // set sck to sck XOR byt (MOD-2 maths)
  for (j=0;j<8;++j) // for each bit
    {
      if (sck.b.msb)
        {
          sck.i>>=1; // shift the remainder right 1 bit
                      // (divide by 2)
          sck.b.lsb=0; // and set the MSB to 0
          sck.i^=CRC16; // set remainder = sck XOR the CRC16
                       // polynomial
        }
      else
        {
          sck.i>>=1; // shift the remainder right 1 bit
                      // (divide by 2)
          sck.b.lsb=0; // and set the MSB to 0
        }
    }
  d->crc = (sck.i)<<8; // update the CRC in the data structure
  d->crc |= (sck.i&0xFF00)>>8;
  free (msg);        // free the temporary storage area
}

```

Appendix D

Serial link design

C.1 Strategies in writing protocol software

Bearing in mind the complexity of implementing a protocol program, the anecdotal stories of cost over-runs in writing protocols are not surprising. This section will examine the software implementation of a protocol.

Protocol software has to support bi-directional communications between two devices. This requires the appropriate data (of interest to the user) being packaged in a 'system level envelope' by the transmitter, and decoded by the receiver. The system level is usually of fixed length and describes fully how the protocol works. The data level, on the other hand, can often be of a variable length.

The process of writing a protocol involves various levels of sophistication and the following factors have to be considered in implementing an appropriate protocol:

Cost	The budget size has to be carefully assessed against the required level of protocol implementation.
Level of performance	If you only require a low level of performance there is no point in implementing the full protocol at an increased cost.
Future requirements	Future requirements may encourage the programmer to include additional features now, which may only be used in the future.
Risk and security	Where possible failure of the serial link has the potential for catastrophic results, it may be prudent to put significant effort into protocol development.
Access to information, technical support	Many vendors are very reluctant to release all the information about their particular protocol for fear of compromising their market position, or do not have the local organization to provide adequate technical support.

The three levels of writing protocol software are:

- Simple one-way asynchronous
- Simple one-way synchronous
- Bi-directional asynchronous

Simple one-way asynchronous

This allows the programmer to drive the software development with a protocol that constructs a message and transmits it from device A to device B. The response from device B would then be crudely received and displayed by device A. Any messages generated independently by device B (without initiation from device A) would not be read by device A.

This would apply to both read and write type messages from device A. In the case of read messages, the user can verify that the correct data status was received in the return string. In the case of write messages, the user can confirm that the address specified in the request message has been updated correctly by the response message.

Simple one-way synchronous

This implementation would be to extend the first option to make the program in device A respond to synchronous messages from device B. By synchronous we mean that the program in device A would enter a mode where it would wait for a command to be issued from device B and respond accordingly. During this time device A would not be able to send a command to device B.

Bi-directional asynchronous

This option would provide asynchronous bi-directional communications between the two devices. That is, device A could send commands to device B, and simultaneously service any requests received from device A.

This option would require an interrupt service routine to initiate the response to a device 'A' command, as it arrives. This is important because the speed of response is also a constraint imposed by the protocol. A response that is too slow may have to be ignored.

The disadvantages of this option are the complexity of the protocol software required. In addition, the device used for such a task would have to be sufficiently fast to allow the interrupts to be serviced, and to handle its own processing requirements.

C.2 A typical program structure

A block diagram of a typical program structure is given in Figure C.1. The structure has been used successfully in implementing a number of different protocol structures, but is included here only as a guide.

Typically, most industrial protocols are involved in the following operations:

- Read digital data
- Read analog data
- Write digital data
- Write analog data

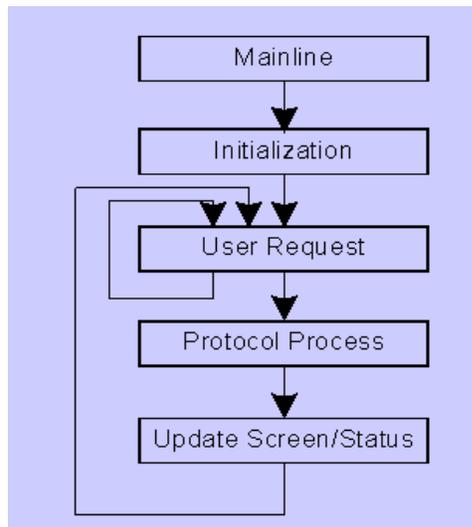


Figure C.1
Block diagram of typical protocol program structure

The following describes each part of the typical protocol program structure.

Initialization

This routine runs at the start of the program only and comprises a number of routines (or tasks).

Variable initialization

The program should declare adequate storage for the data area, to be used for transfer of the data points, across the serial link by the protocol. The data areas used in each communicating device do not necessarily have to be identical in size or even structure. The easiest way to use storage is to define an array of variables representing a block of addresses. All data should be initialized before the other program steps are commenced.

Device status and display initialization

The display and status variables should be initialized. This includes putting the display into the correct video mode and writing all static text to the screen. The communication port should also be initialized.

User request

This function processes incoming commands from the user. The use of interrupts here would be ideal.

C.3 Protocol process

This is the main body of the program; the steps are as follows:

- Get any input required from the user and write to or read from the other device.
- Compute any parts of the protocol that require calculation, e.g. CRC check word, byte count etc.

- Send a command message and wait for the response.
- Update the display showing the response message, system and data fields.
- Read any input data from the other device and decode as per the protocol structure.
- Identify, define any error messages that occur and pass onto the next step.

Update device status

This is where any clean-up operations are performed. This includes such items as communications port status, or a message indicating the success or failure of the preceding operation. The 'protocol process' routine could return an error code to the main part of the program.

C.4 Typical problems encountered in protocol software

Some typical practical problems, which a programmer may encounter when implementing protocol software, are listed below. A lot of these may sound like commonsense but it is surprising how many times they are ignored.

Typical problems are:

- The list of data points transferred over the communications link is normally dynamic. Appropriate data structures to handle the variation in the points transferred (and their addresses) are not implemented.
- Speed of transfer of data across the communications link is too slow because of the quantity of points transferred across. A prioritized or exception reporting scheme of transfer of data points can solve this problem.
- Buffer size of receiver is inadequate. This causes loss of data or delays in transferring information across the link.
- Receiving station, CPU cannot react fast enough to service data coming in, because of overload generated by other activities.
- Physical disruption to communications channel (by a link breakage) causes a catastrophic collapse in re-establishing transfer of the latest data because of inadequate error handling and the build up of old messages.
- The use of interrupt handling of messages is avoided because of the complexity associated with this feature. This results in a significant degradation in performance because of the overload of the CPU.
- Scaling of data at transmitter and receiver is often not done correctly.

C.5 Fragments of protocol programs written in BASIC

A program is included at the end of this chapter, which demonstrates the use of a protocol with a PC (refer to section C.11). This is to illustrate the basics of constructing a protocol and the various functions used.

Microsoft QuickBASIC has been chosen for illustrating the basics of protocol construction, for the following reasons:

- It is a simple and straightforward computer language to use.
- It is relatively powerful and efficient (especially in the later versions).
- Many people are familiar with it.
- It can be compiled (instead of merely interpreted).
- It is low cost and freely available.

It should be emphasized that QuickBASIC is not the most efficient computer language for this sort of work. Most programmers elect to use C language because of its portability, power and efficiency. C is, however, a fairly cryptic language, difficult to remember, and learning it is time-consuming. Hence, it has not been used for this example.

Microsoft QuickBASIC is a programming environment that includes all the tools needed for writing, editing, running, and de-bugging programs. A full help facility is available online to assist in writing the programs. The software has to run on an IBM PC or IBM PC compatible, which uses MS-DOS.

BASIC program implementation

A few elementary 'housekeeping' rules are necessary when using QuickBASIC language. Comments have been added to all QuickBASIC statements, used in the example program, to help you understand the programming process.

The following points should be remembered when writing a program. Although obviously directly related to QuickBASIC, the concepts will be applicable to implementations in other languages.

In this discussion, COM1 and COM2 refer to the two serial ports on the IBM or compatible PC. The following opening statement, which makes BASIC as tolerant as possible of hardware-related problems, should always be used when uncertain of the hardware and software configuration:

```
OPEN 'COM1:300,N,8,1,BIN,CD0,CS0,DS0,OP0,RS,TB2048,
RB2048' AS #1
```

(This OPEN is FOR RANDOM access). The following is an explanation of each recommended parameter used in the OPEN statement:

- The higher the baud rate, the greater the chances of problems; thus, 300 baud is unlikely to give any problems. 56 K baud is the highest speed possible over most telephone lines, due to their limited high-frequency capability. 19 200 baud, which requires a direct wire connection, is most likely to cause problems. (Possible baud rates for QuickBASIC are: 75, 110, 150, 300, 600, 1200, 1800, 2400, 4800, 9600 and 19 200).
- Parity usually does not help significantly. Because of this, no parity (*N*) is recommended. As discussed in an earlier chapter, parity error detection is not a very efficient way of identifying errors.

For those devices that require parity, the parity enable (PE) option should be used in the OPEN COM statement, which is required to turn on parity checking. When the PE option turns on parity checking, a 'Device I/O error' occurs if the two communication programs have two different parities (parity can be Even, Odd, None, Space or Mark). For example, a 'Device I/O error' occurs when two programs try to talk to each other across a serial line using the following two different OPEN COM statements.

```
OPEN 'COM,1200,O,7,2,PE'FOR RANDOM AS #1
```

and

```
OPEN "COM2:1200,E,7,2,PE"FOR RANDOM AS#2
```

If the PE option is removed from the OPEN COM statements above, no error message is displayed.

- The above example uses 8 data bits and 1 stop bit. 8 data bits requires no parity (N), because of the size limit for BASIC's communications data frame (10 bits).
- The BIN (binary mode) is the default. Note: The ASC option does NOT support XON/XOFF protocol, and the XON and XOFF characters are passed without special handling.
- Ignoring hardware handshaking often corrects many problems. Thus, if the application does not require handshaking, try turning off the following hardware line-checking:
 - CD0 Turns off time out for data carrier detect (DCD) line
 - CS0 Turns off time out for clear to send (CTS) line
 - DS0 Turns off time out for data set ready (DSR) line
 - OP0 Turns off time out for a successful OPEN
- RS suppresses detection of request to send (RTS).
- For buffer-related problems, increase the transmit and receiver buffer sizes above the 512-byte default:
 - TB2048 = Increases the transmit buffer size to 2048 bytes
 - RB2048 = Increases the receive buffer size to 2048 bytes

A large receive buffer can work around BASIC delays caused by statements, like the graphics function, PAINT, which use the processor intensively.

The following are additional important hints for troubleshooting communications problems:

- Use the INPUT\$(x) function in conjunction with the LOC (n) function to receive all input from the communications device (where 'x' is the number of characters returned by LOC(n)), which is the number of characters in the input queue waiting to be read. 'n' is the file number that is OPENed for 'COM1:' or 'COM2:'.
- Avoid using the INPUT#n statement to input from the communications port because INPUT#n waits for a carriage return (ASCII 13) character.
- Avoid using the GET#n statement for communications because GET#n waits for the buffer to fill (and buffer overrun could then occur).
- Avoid using the PUT#n statement for communications and use the PRINT#n statement instead. For example, in QuickBASIC 4.00b and 4.50, in BASIC Compiler 6.00 and 6.00b, and in BASIC PDS 7.00 and 7.10 using the PUT#n,,x\$ syntax for sending a variable length string variable as the third argument of the PUT#n statement sends an extra 2 bytes containing the string length before the actual string. These 2 length bytes sent to the communications port may confuse the receiving program, if it is not designed to handle them. No length bytes are sent with PUT#n,,x\$ in QuickBASIC 4.00 (QuickBASIC versions earlier than 4.00 don't offer the feature to use a variable as the third argument of the PUT#n statement).
- Many communications problems can only be shown on certain hardware configurations and are difficult to resolve or duplicate on other computers. Experimenting with a direct connection (with a short null modem cable) is recommended instead of with a phone/modem link, between sender and receiver, to isolate problems on a given configuration.

- The wiring scheme for cables varies widely. Check the pin wiring on the cable connectors. For direct cable connections, a long or high-resistance cable is more likely to give problems than a short, low-resistance one.
- If both 'COM1:' and 'COM 2:' are open, 'COM2:' will be serviced first. At high baud rates, 'COM1:' can lose characters when competing for processor time with 'COM2:'.
- Using the ON COM GOSUB statement, instead of polling the LOC(n) function to detect communications input, can sometimes work around timing or buffering problems caused by delays in BASIC. Delays in BASIC can be caused by string space garbage collection, PAINT statements, or other operations that heavily use the processor.

C.6 Management of data points over the serial link

Although possibly considered a trivial subject by most engineers who are more concerned with the development and commissioning of the data communications system, the management of the data coming over the link can be a challenging issue.

The main reasons why this requires attention are:

- Data can vary from a few points to a few thousand.
- Data points coming across a serial link can vary in terms of addressing as the design proceeds.
- Update times of serial data points can vary from point to point.
- Tag names of serial points can vary as the design and implementation proceeds.

Typical parameters that need to be recorded (preferably in a database program such as dBase to allow easy manipulation of the data) are:

- Tag name of the data point
- Description of the data point
- Address of point (e.g. Modbus address)
- Update time for point
- Scaling factor
- Maximum and minimum and ranges (including the possibility of the data becoming negative)
- Data format (maximum number of digits)
- Type of data for point (e.g. ASCII, integer, floating point)
- Destination of point
- Source of point

An area, which always causes problems, is scaling of the data on the link. For example, if the protocol restricts the range of values over the link to 0 to 4095 (i.e. a 12 bit quantity) and the actual engineering (or 'real world' quantities) are -10 kPa to 20 000 kPa, some delicate footwork has to be done. This is to ensure that there are no problems with the scaling at the transmitting end, transfer of the data across the link and rescaling at the receiving end. In addition, there should be a careful analysis of the loss in resolution caused by scaling.

C.7 Suggested testing philosophy for a communications system

Data communication is a strategic part of a control system. Failure of a communications link could be the cause of information loss from thousands of data points. It is imperative, therefore, that a communications system is thoroughly tested within the framework of a rigorous standard.

There are numerous reasons for the test requirements to be more demanding than those for a standard control system; some of the main reasons are:

- Information losses resulting from a failure of a communications link can be catastrophic.
- Loading factors of a data communications system can vary considerably and may lead to failure if the system is pushed to the limit.
- The communications interface can be complex consisting of hardware/firmware and software components.
- The communication link normally serves as an interface between two dissimilar systems (sometimes consisting of different design personnel, different hardware), which raises the level of technical risk and the need for common understanding of the overall requirements of the system.

A good framework in which to do the testing is ANSI/IEEE standard 829-1983 for software test documentation. While some engineers may be less than enthusiastic about formal testing procedures of this nature, the investment in time and effort is worthwhile in creating a high quality final product, engineered with proven standards of performance. The authors can testify from bitter experience that this approach pays off.

A typical test procedure (or master test plan) for the link between a PC and the Modbus port of a new PLC is sketched out below.

C.8 Typical test procedure example

The test specification procedures and recording practices have been prepared in accordance with ANSI/IEEE Std 829-1983. All software written or modified for this installation will be tested according to these guidelines.

This test is to confirm that the link from the PC to the Modbus port of the new PLC operates correctly as per the specifications.

Features to be tested

The functions to be tested will be derived from the serial link requirements specification PC-MOD1 and functional specification PC-MOD2. These functions will be grouped under the following headings:

- Hardware/firmware checks
- Interface protocol
- Interface I/O points (and addresses)
- Control strategy
- Program structure

Test environment

A PC/AT (the monitor) will interface via a second PC/AT (or protocol analyzer) to the Modbus port of the serial hardware being tested. Serial port (COM 1) of the second

PC/AT will connect to the PC/AT monitor. Serial port (COM 2) will connect to the Modbus port of the serial hardware being tested.

The relevant 'C' language compiled software modules for the serial link will be downloaded into the monitor PC/AT.

The appropriate EPROM (Revision C, 10 Nov.'91) for the PLC will be inserted into the PLC communication board. Test data will be downloaded into the monitor PC controller and the proprietary unit control system.

Testing approach

There are no risks and contingencies at the initial phase of testing, as this is performed offline and merely tests the serial interface system. The second stage of testing is envisaged to directly interface with the operational hardware, but will be done in a manual mode at the specific construction yard. The third phase of testing will be the commissioning phase and will be carried out at the plant.

The second and third phases of the testing which do have risks will not form part of this test procedure, but will be incorporated into an overall test program covering all aspects of testing.

General strategy

The test hardware will be connected to the proprietary hardware under test. The appropriate version of compiled C code and data will be downloaded to the monitor PC. The vendor will download certain specified data structures in the unit controller. Each item of data will be transferred between the two nodes of the link in the appropriate form.

Specifically the following characteristics will be checked:

- Accuracy of data transfer
- Average speed of data transfer
- Loopback and diagnostic tests where applicable
- Loading of link with high traffic loads
- Interaction of multiple nodes on link (e.g. multiplexers, dual controllers)
- Transmission of incorrect function requests
- Interface between 'C' code and the EPROM on the proprietary hardware
- Interface 'C' code and the user software in PLC
- Interface between 'C' code and the serial port hardware/firmware
- Handling of failure of:
 - Serial link
 - Proprietary hardware/firmware/software
- Monitor hardware/firmware/software (with reference to fallback strategies, recovery times, diagnostics, error messages)
- High levels of electrostatic/electromagnetic noise on the link
- Adequacy of earthing systems for each mode of line (including earth potential rise)
- Performance of error checking features of the link (e.g. using CRC-16)

Note: A PC-based protocol analyzer will be used to confirm that the data structures being transmitted down the link and the appropriate responses are correct. This will be in addition to the diagnostic messages generated by both the monitor PC and the PLC serial hardware being tested.

Acceptance tests on the various portions of the system will occur at different stages of the testing. Tests will be jointly performed by the client and the contractor. No modified software will be available for use by the client until it has been fully tested and accepted by the client.

Associated test documentation

Full test documentation should be filled out correctly and stored in a central safe location.

C.9 An example serial data communication link

This section contains an example of a serial data communication link for the control of a variable speed drive using the EIA-485 interface and ANSI-X3.28 protocol.

‘Smart’ instrumentation and other digital sensors and actuators are increasingly being used in factory automation and industrial process control systems. A ‘sensor’ is a general term that refers to instruments, monitors, etc. that measure field variables such as temperature, pressure, levels, flow and power in a process control system. An ‘actuator’ is a general term that refers to devices located in the field, such as valves, variable speed drives, positioners, servos, etc., that implement instructions from the control system.

Making effective use of these devices depends on their ability to transfer data reliably and quickly to and from other controlling devices, such as computers, PLCs, DCSs, etc. via a common data communications network. Data communications at this level is usually referred to as the ‘field level’ communications and the type of networks used are often called the ‘field bus’.

Data communications at the field level is usually reliable when all the equipment comes from one manufacturer. When several different types of equipment from various manufacturers are required to communicate on the same network, difficulties always seem to appear. One major reason is that no clear and universally acceptable data communication network standard has yet emerged for systems for the field level.

The process of developing and implementing acceptable international standards is a painfully slow process and a solution to this problem is still a long way off. In the meantime, manufacturers have created their own standards or have used a combination of available standards that may have been developed for other similar applications.

Therefore, the practical problem of controlling a field actuator device, such as a variable speed drive (VSD), from an intelligent control device, such as a PLC or a PC, needs to be addressed on an application by application basis. This section describes the process of designing and implementing a simple data communications system for transferring data between an IBM compatible PC and an AC VSD to achieve the following:

- To read data from the list of parameter registers in the VSD, transfer them to the PC and display variables such as speed, current etc., on the PC monitor.
- To write data to the VSD parameter registers for starting, stopping, changing settings (e.g. speed reference), or adjusting any other variable in the VSD, as would be required in an industrial application.

Most modern VSDs have some form of communications capability, usually based on a well known physical standard, such as EIA-232 or EIA-485. The transfer of data can be controlled by a suitable program (written by the user) based on one of the ASCII character protocols. The program should be able to address multiple VSDs on the network without having a problem with data collision on the network. Typically, the programs use the poll/response method with one ‘master’ (PC or PLC) in control of the

network and several 'slaves' (VSDs). The slaves respond only when they are polled by the master. Although this approach works quite well, it has some limitations that affect the overall performance of the system:

- This type of system is slow because it uses low baud rates, with inefficient ASCII coding. The master also needs to address each slave individually. This type of data communications solution is not suitable for controlling several drives in applications with fast speed and torque dynamics.
- There is no available software standardization, so a special program has to be written for each application.

The physical interface

The physical connections between the PC and VSDs are according to the EIA-232 and EIA-485 interface standards, both of which are covered in detail in Chapter 3. The standard PC is fitted with an EIA-232 port. The standard VSD port is EIA-485, suitable for multidropping up to 32 units.

From Chapter 3, it is clear that EIA-232 and EIA-485 are not directly compatible and the two devices cannot be directly connected and expected to work.

This apparent mismatch at the physical level can be overcome by one of the following methods:

- Interface Converter – an EIA-232/EIA-485 interface converter can be connected between the two devices to convert the voltage levels and the connection configuration from one to the other (unbalanced to differential). An interface converter should be physically located close to the EIA-232 port (i.e. at the PC end) to take advantage of the better performance characteristics of the EIA-485 interface for the longer distance to the VSD in the field. The interface between the PC and the interface converter is one-to-one, while the EIA-485 side may have several drives (up to 32) connected in a multidrop configuration. The internal connection details, of an EIA-232/EIA-485 converter, are shown in figure below.

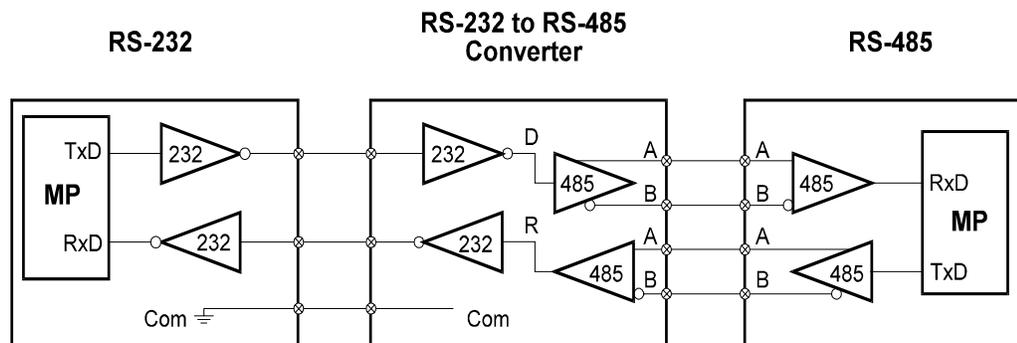


Figure C.2
Block diagram of an EIA-232/EIA-485 converter

- EIA-485 PC interface card – plug-in EIA-485 interface cards are available for IBM compatible PCs for mounting directly onto the motherboard. Similar cards are also available for PLCs. This card must be configured as a separate port in the PC. The PC can then be connected directly to the EIA-485 network.

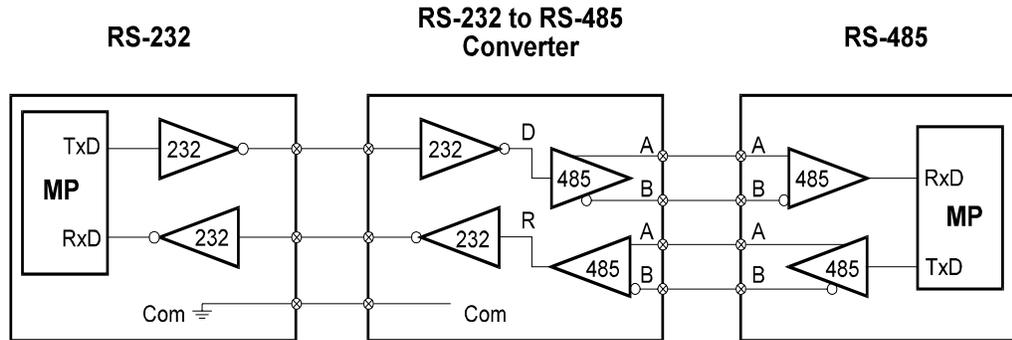


Figure C.3
Block diagram of an EIA-232/EIA-485 converter

The software interface

Once the physical interface problems have been solved, the flow of data between the PC and the VSD must be controlled by software located in the master device. In our example, the program is based on ANSI-X3.28-2.5-A4, which is an ASCII based protocol that defines the format, order, and syntax of the characters. There is no standard format, or content, for this type of program and it is usually written by the user to suit the application. The program below is an example of a simple program written in QuickBASIC for demonstration purposes only.

In accordance with ANSI-X3.28, the 10 bit character format is as follows:

- 1 Start Bit : Logic 0
- 7 Data Bits : ASCII Code for each character
- 1 Parity Bit : Even or None
- 1 Stop Bit : Logic 1

There are two styles of message order and syntax:

- Read message
- Write message

The read message comprises of a maximum 9 characters in the order shown in the following flow chart. The read message is used to transfer data from the VSDs to the master. This data is usually the field data, such as speed, current, etc. or the VSD's setting parameters.

The block checksum character (BCC) is a single character generated from all the data in the message and is used to detect errors in the transmitted data.

The write message comprises of a maximum 17 characters in the order shown in the following flowchart. The write message is used to transfer data from the master to the VSDs. This data is used to issue commands to the field device or change parameters (e.g. start, stop, change speed).

The baud rate can be set to any one of the 'standard' values between 300 to 19,200 bps.

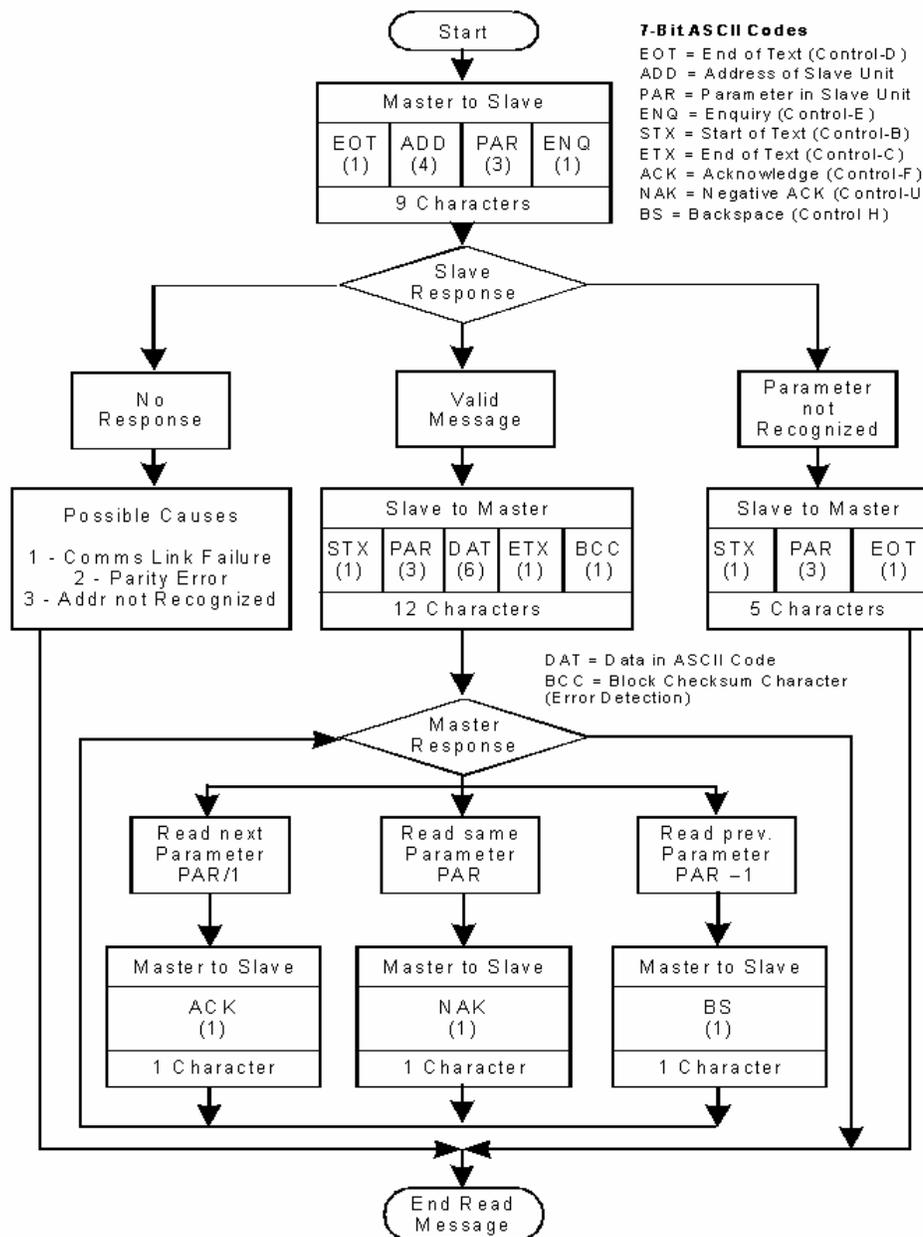


Figure C.4
 Flow chart showing the order and syntax of the read message

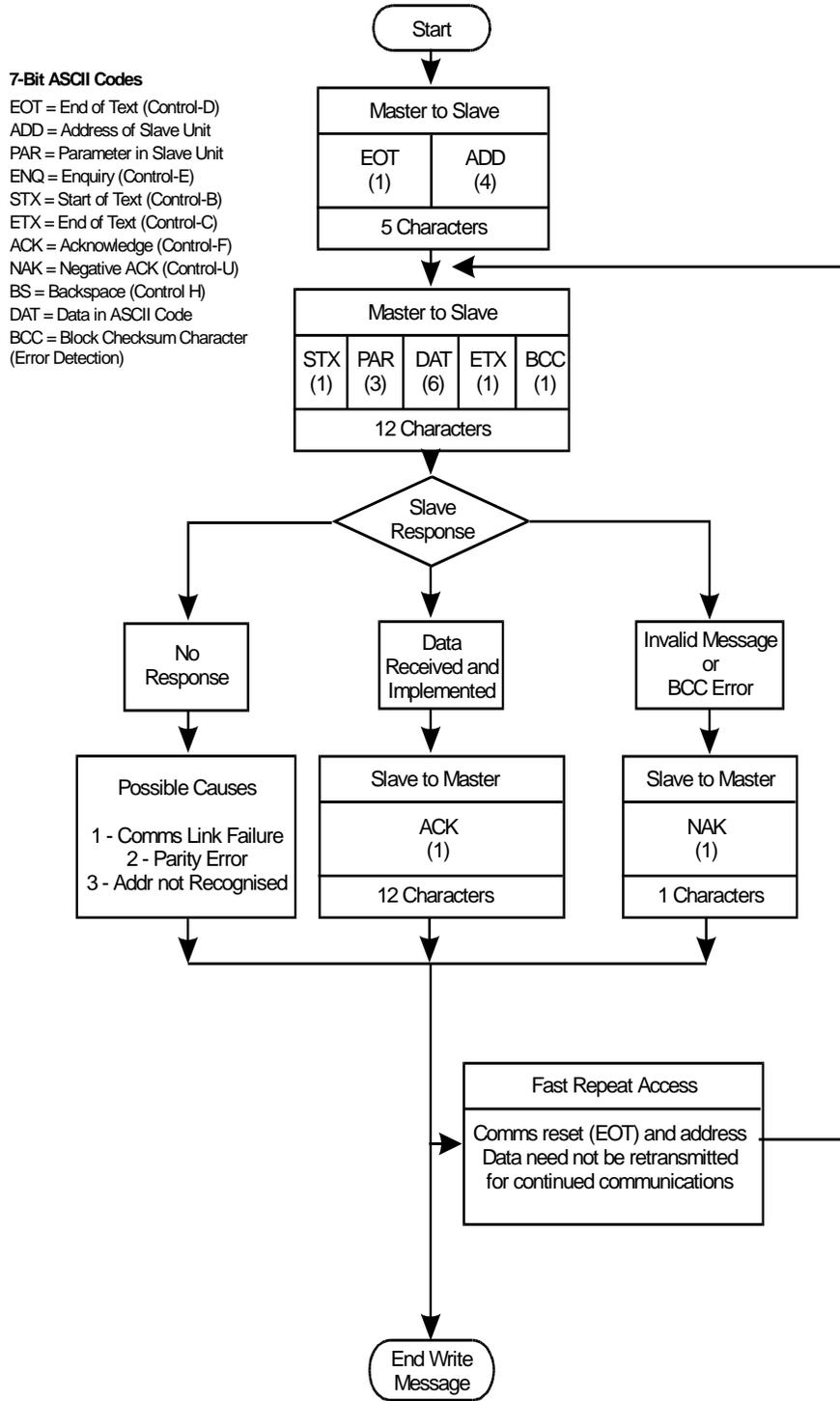


Figure C.5
 Flow chart showing the order and syntax for the write message

C.10 Typical list of VSD parameters

Table C.1 shows a shortened list of typical parameters for a Control Techniques 'Vector' drive. The parameter registers in the range 00 to 99 contain analog (numerical) values, while those from 100 to 199 contain binary digital values. The read command can be used to transfer data from the VSD parameter registers to the PC over the serial data communications link.

The write command can be used to transfer data to the VSD parameter registers from the PC.

Parameter no.	Analog/ bits	Description	Default Value
00	A	Digital speed reference, run	100
01	A	Digital speed reference, inch	- 100
02	A	Analog speed reference offset	0
03	A	Minimum speed limit	- 1500
04	A	Maximum speed limit	+ 1500
05	A	Analog reference input filter	32
06	A	Torque limit - motoring	150
07	A	Torque limit - generating	150
08	A	Internal torque reference	0
09	A	Forward acceleration limit	0.01
10	A	Reverse acceleration limit	0.01
11	A	Forward deceleration limit	0.01
12	A	Reverse deceleration limit	0.01
13	A	Speed loop proportional gain	1.5
14	A	Speed loop internal gain	1.5
15	A	Speed loop derivative gain	0
16	A	Analog output scaling	1.67
17	A	Speed reference select	3
18	A	Analog speed input scaling	600
22	A	Serial link - drive address	01
23	A	Serial link - baud rate	9600
25	A	Security key	0
40	A	Drive model number	n/a
41	A	Motor full load current	n/a
42	A	Motor magnetizing current	n/a
43	A	Motor base frequency	n/a
70	A	Motor speed	n/a
71	A	Motor frequency	n/a
75	A	Active current	n/a
78	A	DC bus voltage	n/a
82	A	Motor line current	n/a
100	B	Security key enable	n/a
101	B	Miscellaneous trip	1
102	B	Drive enable	1
103	B	Drive reset	n/a
104	B	Zero torque demand	1
105	B	Torque/speed control mode	1

106	B	Torque reference select	1
120	B	Serial link – parity enable	1
121	B	Serial link – block checksum enable	1
133	B	Drive healthy	n/a

Table C.1
Example list of VSD parameters

C.11 Example

```

'* INSTRUMENT DATA COMMUNICATIONS
'*
'* Example of PC program written in Microsoft QuickBASIC Version 4.5
'* for IBM compatible PC to a Control Techniques CD Variable Speed Drive.
'*
'* Creation date : 02-12-91
'*
'*
'* SERIAL INTERFACE CONNECTION
'* This program provides for communication to a Commander CD Variable
'* Speed Drive through one of the PC's EIA-232 serial interface ports.
'* An EIA-485 interface would be preferable because of its noise immunity.
'*
'*
'* THE PROTOCOL
'* The protocol implemented is ANSI-X3.28-2.5-A4. It defines the
'* format and order of characters and the syntax of the commands.
'* A character consists 10 bits, comprising 1 start bit (logic-0),
'* followed by 7 data bits (ASCII Code), 1 parity bit (even or none)
'* and the final bit is 1 stop bit (logic-1).
'*
'* There are two styles of commands.
'* The READ command allows reading of all the drives parameters.
'* The WRITE command allows the read/write to be changed.
ON ERROR GOTO FAULT 'Activate error trapping
CLS 'Clears the screen

OPEN "COM1:9600,E,7,1,CD0,CS0,DS0,OP0,RS" FOR RANDOM AS #1
'* Opens the PC's serial port 1 for serial bit-by-bit communication
'* with the peripheral device. (The OPEN is FOR RANDOM access)
'* An explanation of each parameter used in the OPEN statement follows:

'* 1. 9600 specifies the Baud Rate (bits transmitted each second). The
'* baud rate setting for the drive and the host (PC) must be set
'* equal to allow communication to take place.

'* 2. E sets the parity of the drive to EVEN. With even parity selected,
'* the parity bit is set to logic 1 when the data segment of the
'* character consists of an odd number of logic 1's.

```

- * 3. 7 bits in each byte of data transmitted or received constitute actual data.
- * 4. Since this application does not require handshaking, setting CD, CS, DS and OP to 0 tells the hardware to ignore handshaking
 - * CD0 = Turns off time-out for Data Carrier Detect (DCD) line
 - * CS0 = Turns off time-out for Clear To Send (CTS) line
 - * DS0 = Turns off time-out for Data Set Ready (DSR) line
 - * OP0 = Turns off time-out for a successful OPEN
- * 5. RS suppresses detection of Request To Send (RTS)

```
MENU: VIEW PRINT 1 TO 24 'set text viewport
CLS
LOCATE 4, 5: PRINT "VECTOR Drive communications program"
LOCATE 5, 5: PRINT "~~~~~"
LOCATE 7, 5: PRINT " 1. List Parameter Values"
LOCATE 8, 5: PRINT " 2. Get a specific particular parameter from VECTOR"
LOCATE 9, 5: PRINT " 3. Write data to VECTOR"
LOCATE 10, 5: PRINT " 4. Quit"
LOCATE 12, 5: PRINT " Select an option (1-4)"
```

```
OPT: '* Wait for key to be pressed
      KY$ = INKEY$: IF KY$ = "" THEN GOTO OPT
      A = VAL(KY$)
      IF A >= 4 THEN GOSUB QUIT
      ON A GOSUB PARAMVALUES, GETPARAM, WRITEDATA
      GOTO MENU
```

```
QUIT: CLOSE #1 'end communication with the Flux Vector Drive
      END 'end the basic program
```

```
*****
```

```
' Read all the drive parameters
' ~~~~~
' The message format is: <EOT>  ADD  PAR  <ENQ>
'                        1 char  4 chars  3 chars  1 char

' When the ASCII control character <EOT> is sent it initializes the drive
' connected to the serial link. (<EOT> - ASCII code 04 HEX, CONTROL D)

' ADD represents the drive address. The drive address identifies which
' device connected to the serial link is to be communicated with. For
' data integrity the digits are sent twice. For example if addressing
' drive 31 send 3311.

' PAR - this is the parameter which is required to be read. It is a maximum
' of 3 characters. e.g. For the Minimum Speed Limit, Pr-3 : just the number
' 003 is sent

' The message is terminated by the ASCII control character <ENQ>.
```

```
' (<ENQ> - ASCII CODE 05 HEX, CONTROL E)
'*****

' Response from the drive: <STX>  PAR  DATA  <ETX>  BCC
' ~~~~~ 1 char 3 chars 6 chars 1 char 1 char

' <STX> is an ASCII control character. This indicates to the host the start
' of the reply.          (<STX> - ASCII code 02 HEX, CONTROL B)

' PAR - The drive parameter. (See above for description)

' DATA - This symbol represents a parameter's numerical value. It is 1-6
' numeric characters in length, with optional decimal point and sign
' character, and is accurate to 1 decimal point.
' <ETX> is an ASCII control character. It indicates to the host that the
' data is finished.
' BCC, Block checksum - The final character is generated by the drive to
' allow the host which is receiving data to perform an error check on the
' data received. This character is not sent if the drive is set for BCC
' disabled, bit parameter b-21. BCC is the exclusive-OR of all the characters
' after the <STX> character up to and including the <ETX> character.
' If the BCC is disabled the ASCII control character <CR> is sent.
' NOTE: The same parameter can be re-read simply by sending the ASCII
' control character <NAK> (ASCII code 15 HEX, CONTROL U).
' Alternatively it is possible to step forward or backwards through the
' parameters sequentially by sending the ASCII control characters <ACK>
' (ASCII code 6 HEX, CONTROL F) OR <BS> (ASCII code 8 HEX, CONTROL H)
'*****
```

PARAMVALUES:

CLS

```
txd$ = CHR$(4) + "0011000" + CHR$(5)      'String transmitted
                                     '<EOT> - CHR$(4)
                                     '<ENQ> - CHR$(5)
```

rxid\$ = ""

LOCATE 1, 5: PRINT "STRING TRANSMITTED IS :- "; txd\$

LOCATE 3, 1: PRINT "PARAMETER"

LOCATE 3, 14: PRINT "VALUE"

LOCATE 3, 40: PRINT "PARAMETER"

LOCATE 3, 54: PRINT "VALUE"

VIEW PRINT 4 TO 25 'set text viewport

x = 4 'x-coord cursor position

i = 0 'variable to keep track of y-coord position

parameter\$ = "00"

PRINT #1, txd\$ 'Writes data to Coms port 1

GOSUB delay

DO UNTIL parameter\$ = "83" 'display up to parameter 83

rxid\$ = INPUT\$(LOC(1), #1) 'receive input from the comms device

STXloc = INSTR(rxid\$, CHR\$(2)) 'find position of control character

```

'<STX> to obtain start of message

rxid$ = MID$(rxid$, STXloc, 11) 'remove unwanted noise from response OR
'obtain required info from response.
'Ignore BCC - ie BCC disabled
parameter$ = MID$(rxid$, 2, 3)
IF VAL(parameter$) > 83 THEN RETURN
parameter$ = MID$(rxid$, 3, 2) 'extract parameter from response
dat$ = MID$(rxid$, 5, 6) 'extract data from driver's response
GOSUB CURSORPOSI
LOCATE y, x
PRINT parameter$; " "; dat$
IF VAL(parameter$) >= 83 THEN GOSUB MENURETURN
prevrxid$ = rxid$ 'keep record of the previous received
'message from the drive in case of error
PRINT #1, CHR$(6) 'Send <ACK> to the drive to step forward
'through the parameters sequentially

GOSUB delay
LOOP
RETURN
GETPARAM: '*****
'Get a specific parameter from VECTOR
CLS
LOCATE 3, 3: PRINT "1. Minimum Speed Limit"
LOCATE 4, 3: PRINT "2. Maximum Speed Limit"
LOCATE 5, 3: PRINT "3. Drive Address"
LOCATE 6, 3: PRINT "4. Baud Rate"
LOCATE 7, 3: PRINT "5. Motor Speed"
LOCATE 8, 3: PRINT "6. Motor Frequency"
LOCATE 9, 3: PRINT "7. Active Current"
LOCATE 10, 3: PRINT "8. Drive Status"
LOCATE 11, 3: PRINT "9. Parity Status"
LOCATE 12, 3: PRINT "10. Block CheckSum Status "
LOCATE 13, 3: PRINT "11. Hardware Status"
LOCATE 14, 3: PRINT "12. Return to main menu"
LOCATE 16, 1: PRINT "Choose a parameter and press 'return':"
INPUT PR$
IF LEN(PR$) = 0 OR LEN(PR$) > 2 OR PR$ = "12" THEN RETURN
IF PR$ = "1" THEN GOSUB GP1
IF PR$ = "2" THEN GOSUB GP2
IF PR$ = "3" THEN GOSUB GP3
IF PR$ = "4" THEN GOSUB GP4
IF PR$ = "5" THEN GOSUB GP5
IF PR$ = "6" THEN GOSUB GP6
IF PR$ = "7" THEN GOSUB GP7
IF PR$ = "8" THEN GOSUB GP8
IF PR$ = "9" THEN GOSUB GP9
IF PR$ = "10" THEN GOSUB GP10
IF PR$ = "11" THEN GOSUB GP11

```

```

TRANSMIT: txd$ = CHR$(4) + "0011" + DP$ + CHR$(5)
          rxd$ = ""
          PRINT #1, txd$
          GOSUB delay

          rxd$ = INPUT$(LOC(1), #1)
          IF rxd$ = "" THEN GOTO TRANSMIT
          STXloc = INSTR(rxd$, CHR$(2)) 'Filter out any noise if it exists
          rxd$ = MID$(rxd$, STXloc, 11)
          IF MID$(rxd$, 2, 3) <> DP$ THEN GOTO TRANSMIT
          dat$ = MID$(rxd$, 5, 6)
          LOCATE 18, 4: PRINT MSG$; " "; dat$; " "; UNIT$
          LOCATE 20, 1: PRINT "Would you like to read another parameter (Y/N)"

```

```

RAGAIN: KY$ = INKEY$: IF KY$ = "" THEN GOTO RAGAIN
        IF KY$ = "y" OR KY$ = "Y" THEN GOTO GETPARAM
        RETURN

```

```

GP1: DP$ = "003"
     MSG$ = "The minimum speed limit is: "
     UNIT$ = "r.p.m."
     RETURN
GP2: DP$ = "004"
     MSG$ = "The maximum speed limit is: "
     UNIT$ = "r.p.m."
     RETURN
GP3: DP$ = "022"
     MSG$ = "The drive address is:"
     UNIT$ = ""
     RETURN
GP4: DP$ = "023"
     MSG$ = "The baud rate is:"
     UNIT$ = "baud"
     RETURN
GP5: DP$ = "070"
     MSG$ = "The motor speed is:"
     UNIT$ = "r.p.m."
     RETURN
GP6: DP$ = "071"
     MSG$ = "The motor frequency is:"
     UNIT$ = "Hz"
     RETURN
GP7: DP$ = "075"
     MSG$ = "The active current is:"
     UNIT$ = "%"
     RETURN
GP8: DP$ = "102"
     MSG$ = "The drive is:"
     UNIT$ = " 0-drive enabled  1-drive disabled"
     RETURN

```

```

GP9: DP$ = "120"
    MSG$ = "The parity is:"
    UNIT$ = " 0-parity disabled  1-parity enabled"
    RETURN
GP10: DP$ = "121"
    MSG$ = "The block checksum is:"
    UNIT$ = " 0-BCC disabled  1-BCC enabled"
    RETURN
GP11: DP$ = "190"
    MSG$ = "The hardware status is:"
    UNIT$ = " 0-Inactive  1-Active"
    RETURN
*****
*****
'The Write Command. The message sent to the drive is :-

' <EOT>  ADD  <STX>  PAR  DATA  <ETX>  BCC
'  1 char 4 chars  1 char  3 chars  6 chars  1 char  1 char

' If the drive parameter, data or the BCC is in error then the control
' character <NAK> is sent.
' If it is required to write further data to the drive, it is not necessary
' to re-send the initialization character or the drive address. The drive
' parameter characters are the first characters sent.
*****
*****
WRITEDATA: '* Send data to VECTOR
    CLS
    COUNT = 0
    LOCATE 3, 3: PRINT "1.  Minimum Speed Limit"
    LOCATE 4, 3: PRINT "2.  Maximum Speed Limit"
    LOCATE 5, 3: PRINT "3.  Drive Status"
    LOCATE 6, 3: PRINT "4.  Return to main menu"
    LOCATE 8, 1: PRINT "Choose a parameter: (1,2,3 or 4)";

WDMENU: KY$ = INKEY$: IF KY$ = "" THEN GOTO WDMENU
    A = VAL(KY$)
    IF A >= 4 THEN GOSUB MENU
    ON A GOSUB WD1, WD2, WD3
    COUNT = COUNT + 1

WDSEND: IF COUNT <= 1 THEN
    txd$ = CHR$(4) + "0011" + CHR$(2) + DP$ + DATA$ + CHR$(3) + " "
    ELSEIF COUNT > 1 THEN
    txd$ = DP$ + DATA$ + CHR$(3) + " "
    END IF
    rxd$ = ""
    PRINT #1, txd$
    FOR j = 1 TO 3
        GOSUB delay

```

```

NEXT
rxd$ = INPUT$(LOC(1), #1)
IF rxd$ = "" THEN GOTO WDSEND
'***** First locate the beginning of the received string
FOR k = 1 TO LEN(rxd$)
  ch$ = MID$(rxd$, k, 1)
  IF ch$ = CHR$(6) OR ch$ = CHR$(15) THEN messageposi = k
NEXT

rxd$ = MID$(rxd$, messageposi, 1)

IF rxd$ = CHR$(15) THEN
  PRINT " I/O error"; rxd$
ELSEIF rxd$ = CHR$(6) THEN
  PRINT "successful write"; rxd$
ELSEIF rxd$ <> CHR$(15) OR rxd$ <> CHR$(6) THEN
  GOTO WDSEND
END IF
LOCATE 20, 1: PRINT "Would you like to write to another parameter (Y/N)"

WAGAIN: KY$ = INKEY$: IF KY$ = "" THEN GOTO WAGAIN
IF KY$ = "y" OR KY$ = "Y" THEN GOTO WRITEDATA
RETURN

WD1: DP$ = "003"
  GOSUB INPUTDATA
  RETURN
WD2: DP$ = "004"
  GOSUB INPUTDATA
  RETURN
WD3: DP$ = "102"
  LOCATE 15, 1: INPUT "Do you wish to (E)nable or (D)isable the drive"; ABLE$
  IF ABLE$ = "E" THEN
    DATA$ = " 0000."
  ELSEIF ABLE$ = "D" THEN
    DATA$ = " 0001."
  ELSEIF ABLE$ <> "E" OR ABLE$ <> "D" THEN
    GOTO WD3
  END IF
  RETURN

'*****
'*****
'Error handling routine

' This is the first statement the program branches to after an error.
' The error handler is placed where it cannot be executed during the normal
' flow of program execution.
'*****
'*****

```

```

FAULT: LOCATE 23, 1
      PRINT "ERL="; ERL, "ERR="; ERR;
      GOSUB PAUSE
      rxd$ = prevrxd$
      RESUME NEXT

```

```

PAUSE: LOCATE 24, 1: PRINT "Press any key to continue ... ";
KEYPRESSED: KY$ = INKEY$
      IF KY$ = "" THEN GOTO KEYPRESSED
      LOCATE 23, 1: PRINT "          "
      LOCATE 24, 1: PRINT "          "
      RETURN

```

```

NEXTSCR: LOCATE 24, 1: PRINT "Press any key for next screen of Parameters ... ";
KYPRESSED: KY$ = INKEY$
      IF KY$ = "" THEN GOTO KYPRESSED
      RETURN

```

```

MENURETURN: LOCATE 24, 1: PRINT "Press any key to return to the main menu
..."
KYPRESSED: KY$ = INKEY$
      IF KY$ = "" THEN GOTO KYPRESSED
      RETURN

```

```

CURSORPOSI:
  i = i + 1
  IF i <= 19 THEN
    y = i + 3
    x = 4
  ELSEIF i > 19 AND i <= 38 THEN
    y = i - 16
    x = 44
  ELSEIF i > 38 AND i <= 57 THEN
    y = i - 35
    x = 4
  ELSEIF i > 57 THEN
    y = i - 54
    x = 44
  END IF
  IF i = 39 THEN GOSUB NEXTSCR: CLS
  RETURN

```

```

delay: 'Delay
  FOR k = 1 TO 10
    z = SIN(z * 3.4) + COS(z * 3.45)
  NEXT k
  RETURN

```

```

INPUTDATA:
  DECPOSI = 0

```

```

LOCATE 12, 1: INPUT "Enter data to be sent (-1500 to 1500) (No decimals) ",
dat$
IF ABS(VAL(dat$)) > 1500 OR LEN(dat$) > 5 THEN GOTO WD1

FOR CNT = 1 TO LEN(dat$)
  DUM$ = MID$(dat$, CNT, 1)
  IF DUM$ = "." THEN DECPOSI = CNT
NEXT

IF LEFT$(dat$, 1) = "-" THEN
  IF DECPOSI = 0 THEN 'No decimal
    IF LEN(dat$) = 3 THEN dat$ = "-" + "00" + RIGHT$(dat$, 2) + "."
    IF LEN(dat$) = 4 THEN dat$ = "-" + "0" + RIGHT$(dat$, 3) + "."
    IF LEN(dat$) = 5 THEN dat$ = dat$ + "."
  ELSEIF DECPOSI <> 0 THEN
    LOCATE 15, 1: INPUT "Please enter a whole number:"; b
    GOTO WD1
  END IF
ELSEIF LEFT$(dat$, 1) <> "-" THEN
  IF DECPOSI = 0 THEN 'No decimal
    IF LEN(dat$) = 2 THEN dat$ = " " + "00" + RIGHT$(dat$, 2) + "."
    IF LEN(dat$) = 3 THEN dat$ = " " + "0" + RIGHT$(dat$, 3) + "."
    IF LEN(dat$) = 4 THEN dat$ = " " + RIGHT$(dat$, 4) + "."
    IF LEN(dat$) = 5 THEN dat$ = dat$ + "."
  ELSEIF DECPOSI <> 0 THEN
    LOCATE 15, 1: INPUT "Please enter a whole number:"; b
    GOTO WD1
  END IF
END IF
DATA$ = dat$
RETURN

```

Appendix E

Glossary

A

ABM	Asynchronous balanced mode
ACE	Association control element
ACE	Asynchronous communications element. Similar to UART
ACK	Acknowledge (ASCII - control F)
Active filter	Active circuit devices (usually amplifiers), with passive circuit elements (resistors and capacitors) and which have characteristics that more closely match ideal filters than do passive filters.
Active passive device	Device capable of supplying the current for the loop (active) or one that must draw its power from connected equipment (passive).
ADCCP	Advanced data communication control procedure
ADDR	Address field
Address	A normally unique designator for location of data or the identity of a peripheral device that allows each device on a single communications line to respond to its own message.
Algorithm	Normally used as a basis for writing a computer program. This is a set of rules with a finite number of steps for solving a problem.
Alias frequency	A false lower frequency component that appears in data reconstructed from original data acquired at an insufficient sampling rate (which is less than two (2) times the maximum frequency of the original data).
ALU	Arithmetic logic unit

Amplitude flatness	A measure of how close to constant the gain of a circuit remains over a range of frequencies.
Amplitude modulation	A modulation technique (also referred to as AM or ASK) used to allow data to be transmitted across an analog network, such as a switched telephone network. The amplitude of a single (carrier) frequency is varied or modulated between two levels – one for binary 0 and one for binary 1.
Analog	A continuous real time phenomena where the information values are represented in a variable and continuous waveform.
ANSI	American National Standards Institute – the principal standards development body in the USA.
APM	Alternating pulse modulation
Appletalk	A proprietary computer networking standard initiated by the Apple Computer for use in connecting the Macintosh range of computers and peripherals (including laser writer printers). This standard operates at 230 kbps.
Application layer	The highest layer of the seven layer ISO/OSI reference model structure, which contains all user or application programs.
Arithmetic logic unit	The element(s) in a processing system that perform(s) the mathematical functions such as addition, subtraction, multiplication, division, inversion, AND, OR, NAND and NOR.
ARP	Address resolution protocol A transmission control protocol/ Internet protocol (TCP/IP) process that maps an IP address to Ethernet address, required by TCP/IP for use with Ethernet.
ARQ	Automatic request for transmission A request by the receiver for the transmitter to retransmit a block or frame because of errors detected in the originally received message.
AS	Australian standard
ASCII	American standard code for information interchange. A universal standard for encoding alphanumeric characters into 7 or 8 binary bits. Drawn up by ANSI to ensure compatibility between different computer systems.
AS-i	Actuator sensor interface
ASIC	Application specific integrated circuit
ASK	Amplitude shift keying – see Amplitude modulation

ASN.1	Abstract syntax notation 1 – an abstract syntax used to define the structure of the protocol data units associated with a particular protocol entity.
Asynchronous	Communications where characters can be transmitted at an arbitrary unsynchronized point in time and where the time intervals between transmitted characters may be of varying lengths. Communication is controlled by start and stop bits at the beginning and end of each character.
Attenuation	The decrease in the magnitude of strength (or power) of a signal. In cables, generally expressed in dB per unit length.
AWG	American wire gauge

B

Balanced circuit	A circuit so arranged that the impressed voltages on each conductor of the pair are equal in magnitude but opposite in polarity with respect to ground.
Bandpass filter	A filter that allows only a fixed range of frequencies to pass through. All other frequencies outside this range (or band) are sharply reduced in magnitude.
Bandwidth	The range of frequencies available expressed as the difference between the highest and lowest frequencies is expressed in Hertz (or cycles per second).
Base address	A memory address that serves as the reference point. All other points are located by offsetting in relation to the base address.
Baseband	Baseband operation is the direct transmission of data over a transmission medium without the prior modulation on a high frequency carrier band.
Baud	Unit of signaling speed derived from the number of events per second (normally bits per second). However if each event has more than one bit associated with it the baud rate and bits per second are not equal.
Baudot	Data transmission code in which five bits represent one character. 64 alphanumeric characters can be represented. This code is used in many teleprinter systems with one start bit and 1.42 stop bits added.
BCC	Block check calculation
BCC	Block check character – error checking scheme with one check character; a good example being block sum check.
BCD	Binary coded decimal A code used for representing decimal digits in a binary code.
BEL	Bell (ASCII for control-G)
Bell 212	An AT&T specification of full duplex, asynchronous or synchronous 1200 baud data transmission for use on the public telephone networks.
BER	Bit error rate

BERT/BLERT	Bit error rate/block error rate testing – an error checking technique that compares a received data pattern with a known transmitted data pattern to determine transmission line quality.
BIN	Binary digits
BIOS	Basic input/output system
Bipolar	A signal range that includes both positive and negative values.
BISYNC	Binary synchronous communications protocol
Bit	Derived from “ BI nary Digi T”, a one or zero condition in the binary system.
Bit stuffing with zero bit insertion	A technique used to allow pure binary data to be transmitted on a synchronous transmission line. Each message block (frame) is encapsulated between two flags that are special bit sequences. Then if the message data contains a possibly similar sequence, an additional (zero) bit is inserted into the data stream by the sender, and is subsequently removed by the receiving device. The transmission method is then said to be data transparent.
Bits per second (bps)	Unit of data transmission rate.
Block sum check	This is used for the detection of errors when data is being transmitted. It comprises a set of binary digits (bits) which are the modulo 2 sum of the individual characters or octets in a frame (block) or message.
Bridge	A device to connect similar subnetworks without its own network address. Used mostly to reduce the network load.
Broadband	A communications channel that has greater bandwidth than a voice grade line and is potentially capable of greater transmission rates. Opposite of baseband. In wideband operation the data to be transmitted are first modulated on a high frequency carrier signal. They can then be simultaneously transmitted with other data modulated on a different carrier signal on the same transmission medium.
Broadcast	A message on a bus intended for all devices that requires no reply.
BS	Backspace (ASCII Control-H)
BS	British standard
BSC	Bisynchronous transmission A byte or character oriented communication protocol that has become the industry standard (created by IBM). It uses a defined set of control characters for synchronized transmission of binary coded data between stations in a data communications system.
BSP	Binary synchronous protocol

Buffer	An intermediate temporary storage device used to compensate for a difference in data rate and data flow between two devices (also called a spooler for interfacing a computer and a printer).
Burst mode	A high-speed data transfer in which the address of the data is sent followed by back to back data words while a physical signal is asserted.
Bus	A data path shared by many devices with one or more conductors for transmitting signals, data or power.
Byte	A term referring to eight associated bits of information; sometimes called a ‘character’.

C

CAN	Controller area network
Capacitance	Storage of electrically separated charges between two plates having different potentials. The value is proportional to the surface area of the plates and inversely proportional to the distance between them.
Capacitance (mutual)	The capacitance between two conductors with all other conductors, including shield, short-circuited to the ground.
CATV	Community Antenna Television
CCITT	See ITU
Cellular polyethylene	Expanded or ‘foam’ polyethylene consisting of individual closed cells suspended in a polyethylene medium.
Character	Letter, numeral, punctuation, control figure or any other symbol contained in a message.
Characteristic impedance	The impedance that, when connected to the output terminals of a transmission line of any length, makes the line appear infinitely long. The ratio of voltage to current at every point along a transmission line on which there are no standing waves.
CIC	Controller in charge
Clock	The source(s) of timing signals for sequencing electronic events e.g. synchronous data transfer.
CMD	Command byte
CMR	Common mode rejection
CMRR	Common mode rejection ratio
CMV	Common mode voltage
Common carrier	A private data communications utility company that furnishes communications services to the general public.

Composite link	The line or circuit connecting a pair of multiplexers or concentrators; the circuit carrying multiplexed data.
Contention	The facility provided by the dial network or a data PABX which allows multiple terminals to compete on a first come, first served basis for a smaller number of computer posts.
CPU	Central processing unit
CR	Carriage return (ASCII control-M)
CRC	Cyclic redundancy check – an error-checking mechanism using a polynomial algorithm based on the content of a message frame at the transmitter and included in a field appended to the frame. At the receiver, it is then compared with the result of the calculation that is performed by the receiver. Also referred to as CRC-16.
CRL	Communication relationship list
Cross talk	A situation where a signal from a communications channel interferes with an associated channel's signals.
Crossed planning	Wiring configuration that allows two DTE or DCE devices to communicate. Essentially it involves connecting pin 2 to pin 3 of the two devices.
Crossover	In communications, a conductor that runs through the cable and connects to a different pin number at each end.
CSMA/CD	Carrier sense multiple access/collision detection – when two senders transmit at the same time on a local area network; they both cease transmission and signal that a collision has occurred. Each then tries again after waiting for a predetermined time period.
CTS	Clear to send
Current Loop	Communication method that allows data to be transmitted over a longer distance with a higher noise immunity level than with the standard EIS-232-C voltage method. A mark (a binary 1) is represented by current of 20 mA and a space (or binary 0) is represented by the absence of current.

D

DAQ	Data acquisition
Data integrity	A performance measure based on the rate of undetected errors.
Data link layer	This corresponds to layer 2 of the ISO reference model for open systems interconnection. It is concerned with the reliable transfer of data (no residual transmission errors) across the data link being used.
Data reduction	The process of analyzing large quantities of data in order to extract some statistical summary of the underlying parameters.

Datagram	A type of service offered on a packet-switched data network. A datagram is a self contained packet of information that is sent through the network with minimum protocol overheads.
DCD	Data carrier detect
DCE	Data communications equipment or data circuit-terminating equipment – devices that provide the functions required to establish, maintain, and terminate a data transmission connection. Normally it refers to a modem.
DCS	Distributed control systems
Decibel (dB)	A logarithmic measure of the ratio of two signal levels where $\text{dB} = 20 \log_{10} V1/V2$ or where $\text{dB} = 10 \log_{10} P1/P2$ and where V refers to voltage or P refers to power. Note that it has no units of measurement.
Default	A value or setup condition assigned, which is automatically assumed for the system unless otherwise explicitly specified.
Delay distortion	Distortion of a signal caused by the frequency components making up the signal having different propagation velocities across a transmission medium.
DES	Data encryption standard
DFM	Direct frequency modulation
Dielectric constant (E)	The ratio of the capacitance using the material in question as the dielectric, to the capacitance resulting when the material is replaced by air.
Digital	A signal which has definite states (normally two).
DIN	Deutsches Institut Fur Normierung
DIP	Dual in line package, referring to integrated circuits and switches.
Direct memory access	A technique of transferring data between the computer memory and a device on the computer bus without the intervention of the microprocessor. Also abbreviated to DMA.
DISC	Disconnect
DLE	Data link escape (ASCII character)
DNA	Distributed network architecture
DPI	Dots per inch
DPLL	Digital phase locked loop
DR	Dynamic range The ratio of the full-scale range (FSR) of a data converter to the smallest difference it can resolve. $\text{DR} = 2^n$ where n is the resolution in bits.

Driver software	A program that acts as the interface between a higher-level coding structure and the lower level hardware/firmware component of a computer.
DSP	Digital signal processing
DSR	Data set ready or DCE ready in EIA-232D/E – A EIA-232 modem interface control signal which indicates that the terminal is ready for transmission.
DTE	Data terminal equipment – devices acting as data source or data sink, or both.
DTR	Data terminal ready or DTE ready in EIA-232D/E
Duplex	The ability to send and receive data simultaneously over the same communications line.

E

EBCDIC	Extended binary coded decimal interchange code An eight bit character code used primarily in IBM equipment. The code allows for 256 different bit patterns.
EDAC	Error detection and correction
EFTPOS	Electronic funds transfer at the point of sale
EIA	Electronic Industries Association – a standards organization in the USA specializing in the electrical and functional characteristics of interface equipment.
EISA	Enhanced industry standard architecture
EMI/RFI	Electromagnetic interference/radio frequency interference ‘background noise’ that could modify or destroy data transmission.
EMS	Expanded memory specification
Emulation	The imitation of a computer system performed by a combination of hardware and software that allows programs to run between incompatible systems.
ENQ	Enquiry (ASCII Control-E)
EOT	End of transmission (ASCII Control-D)
EPA	Enhanced performance architecture
EPR	Earth potential rise
EPROM	Erasable programmable read only memory – non-volatile semiconductor memory that is erasable in an ultra violet light and reprogrammable.
Error rate	The ratio of the average number of bits that will be corrupted to the total number of bits that are transmitted for a data link or system.
ESC	Escape (ASCII character)
ESD	Electrostatic discharge
ETB	End of transmission block

Ethernet	Name of a widely used LAN, based on the CSMA/CD bus access method (IEEE 802.3). Ethernet is the basis of the TOP bus topology.
ETX	End of text (ASCII control-C)
Even parity	A data verification method normally implemented in hardware in which each character must have an even number of 'ON' bits.

F

Farad	Unit of capacitance whereby a charge of one coulomb produces a one volt potential difference.
FAS	Fieldbus access sublayer
FCC	Federal communications commission
FCS	Frame check sequence – a general term given to the additional bits appended to a transmitted frame or message by the source to enable the receiver to detect possible transmission errors.
FDM	Frequency division multiplexer – a device that divides the available transmission frequency range in narrower bands, each of which is used for a separate channel.
FIB	Factory information bus
FIFO	First in, first out
Filled cable	A telephone cable construction in which the cable core is filled with a material that will prevent moisture from entering or passing along the cable.
FIP	Factory instrumentation protocol
Firmware	A computer program or software stored permanently in PROM or ROM or semi-permanently in EPROM.
Flame retardancy	The ability of a material not to propagate flame once the flame source is removed.
Flow control	The procedure for regulating the flow of data between two devices preventing the loss of data once a device's buffer has reached its capacity.
FMS	Fieldbus message specification
FNC	Function byte
Frame	The unit of information transferred across a data link. Typically, there are control frames for link management and information frames for the transfer of message data.
Frequency modulation	A modulation technique (abbreviated to FM) used to allow data to be transmitted across an analog network where the frequency is varied between two levels – one for binary '0'

	and one for binary '1'. Also known as frequency shift keying (or FSK).
Frequency	Refers to the number of cycles per second.
FRMR	Frame reject
FSK	Frequency shift keying, see frequency modulation
Full duplex	Simultaneous two-way independent transmission in both directions (4 wire). See Duplex.

G

G	Giga (metric system prefix – 10 ⁹)
Gateway	A device to connect two different networks which translates the different protocols.
GMSK	Gaussian minimum shift keying
GPIB	General purpose interface bus – an interface standard used for parallel data communication, usually used for controlling electronic instruments from a computer. Also known as IEEE 488 standard.
Ground	An electrically neutral circuit that has the same potential as the earth. A reference point for an electrical system also intended for safety purposes.

H

Half duplex	Transmissions in either direction, but not simultaneously.
Hamming Distance	A measure of the effectiveness of error checking. The higher the Hamming distance (HD) index, the safer is the data transmission.
Handshaking	Exchange of predetermined signals between two devices establishing a connection.
Hardware	Refers to the physical components of a device, such as a computer, sensor, controller or data communications system. These are the physical items that one can see.
HART	Highway addressable remote transducers
HDLC	High level data link control The international standard communication protocol defined by ISO to control the exchange of data across either a point-to-point data link or a multidrop data link.
Hertz (Hz)	A term replacing cycles per second as a unit of frequency.
Hex	Hexadecimal
HF	High frequency
Host	This is normally a computer belonging to a user that contains (hosts) the communication hardware and software necessary to connect the computer to a data communications network.

HSE High speed Ethernet

I

I/O address	A method that allows the CPU to distinguish between different boards in a system. All boards must have different addresses.
IA5	International alphabet number 5
IC	Integrated circuit
ICS	Instrumentation and control system
IDF	Intermediate distribution frame
IEC	International Electrotechnical Commission
IEE	Institution of Electrical Engineers – an American based international professional society that issues its own standards and is a member of ANSI and ISO.
IEEE	Institute of Electrical and Electronic Engineers
IFC	International Fieldbus Consortium
ILD	Injection laser diode
Impedance	The total opposition that a circuit offers to the flow of alternating current or any other varying current at a particular frequency. It is a combination of resistance R and reactance X, measured in Ohms.
Inductance	The property of a circuit or circuit element that opposes a change in current flow, thus causing current changes to lag behind voltage changes. It is measured in henrys.
Insulation	
resistance (IR)	That resistance offered by insulation to an impressed dc voltage, tending to produce a leakage current though the insulation.
Interface	A shared boundary defined by common physical interconnection characteristics, signal characteristics and measurement of interchanged signals.
Interrupt handler	The section of the program that performs the necessary operation to service an interrupt when it occurs.
Interrupt	An external event indicating that the CPU should suspend its current task to service a designated activity.
IP	Internet protocol
IRQ	Interrupt request line
ISA	Industry Standard Architecture (for IBM Personal Computers)
ISB	Intrinsically safe barrier
ISDN	Integrated services digital network – the new generation of worldwide telecommunications network that utilizes digital

	techniques for both transmission and switching. It supports both voice and data communications.
ISO	International Standards Organization
ISP	Interoperable systems project
ISR	Interrupt service routine, see interrupt handler
ITB	End of intermediate block
ITS	Interface terminal strip
ITU	International Telecommunications Union – formerly CCITT (Consultative Committee International Telegraph and Telephone). An international association that sets worldwide standards (e.g. V.21, V.22, V.22bis).

J

Jumper	A wire connecting one or more pins on the one end of a cable only.
---------------	--

K

k (kilo)	This is 2^{10} or 1024 in computer terminology, e.g. 1 kB = 1024 bytes.
-----------------	---

L

LAN	Local area network – a data communications system confined to a limited geographic area typically about 10 km with moderate to high data rates (100 kbps to 50 Mbps). Some type of switching technology is used, but common carrier circuits are not used.
LAN	Local area network. See Local area network.
LAP-M	Link access protocol modem
LAS	Link active scheduler
LCD	Liquid crystal display – a low-power display system used on many laptops and other digital equipment.
LDM	Limited distance modem – a signal converter which conditions and boosts a digital signal so that it may be transmitted further than a standard EIA-232 signal.
Leased (or Private) line	A private telephone line without inter-exchange switching arrangements.
LED	Light emitting diode A semiconductor light source that emits visible light or infra red radiation.
LF	Line feed (ASCII Control-J)
Line driver	A signal converter that conditions a signal to ensure reliable transmission over an extended distance.

Line turnaround	The reversing of transmission direction from transmitter to receiver or vice versa when a half duplex circuit is used.
Linearity	A relationship where the output is directly proportional to the input.
Link layer	Layer 2 of the ISO/OSI reference model. Also known as the data link layer.
Listener	A device on the GPIB bus that receives information from the bus.
LLC	Logical link control (IEEE 802)
LLI	Lower layer interface
Loaded line	A telephone line equipped with loading coils to add inductance in order to minimize amplitude distortion.
Loop resistance	The measured resistance of two conductors forming a circuit.
Loopback	Type of diagnostic test in which the transmitted signal is returned on the sending device after passing through all, or a portion of, a data communication link or network. A loopback test permits the comparison of a returned signal with the transmitted signal.
LRC	Longitudinal redundancy check
LSB	Least significant bits – the digits on the right hand side of the written HEX or BIN codes.
LSD	Least significant digit

M

M	Mega. Metric system prefix for 10 ⁶ .
m	Meter. Metric system unit for length.
MAC	Media Access Control (IEEE 802).
MAN	Metropolitan Area Network
Manchester encoding	Digital technique (specified for the IEEE 802.3 Ethernet baseband network standard) in which each bit period is divided into two complementary halves; a negative to positive voltage transition in the middle of the bit period designates a binary '1', whilst a positive to negative transition represents a '0'. The encoding technique also allows the receiving device to recover the transmitted clock from the incoming data stream (self clocking).
MAP 3.0	Standard profile for manufacturing developed by MAP.
MAP	Manufacturing automation protocol – a suite of network protocols originated by General Motors, which follow the seven layers of the OSI model. A reduced implementation is referred to as a mini-MAP.
Mark	This is equivalent to a binary 1.

Master/slave	Bus access method whereby the right to transmit is assigned to one device only, the master , and all the other devices, the slaves may only transmit when requested.
MDF	Main distribution frame
MIPS	Million instructions per second
MMI	Man-machine-interface
MMS	Manufacturing message services – a protocol entity forming part of the application layer. It is intended for use specifically in the manufacturing or process control industry. It enables a supervisory computer to control the operation of a distributed community of computer-based devices.
MNP	Microcom networking protocol
Modem eliminator	A device used to connect a local terminal and a computer port in lieu of the pair of modems to which they would ordinarily connect, allow DTE to DTE data and control signal connections otherwise not easily achieved by standard cables or connections.
Modem	MODulator/DEModulator – a device used to convert serial digital data from a transmitting terminal to a signal suitable for transmission over a telephone channel or to reconvert the transmitted signal to serial digital data for the receiving terminal.
MOS	Metal oxide semiconductor
MOV	Metal oxide varistor
MSB	Most significant bits – the digits on the left hand side of the written HEX or BIN codes.
MSD	Most significant digit
MTBF	Mean time between failures
MTTR	Mean time to repair
Multidrop	A single communication line or bus used to connect three or more points.
Multiplexer (MUX)	A device used for division of a communication link into two or more channels either by using frequency division or time division.

N

NAK	Negative acknowledge (ASCII Control-U)
Network architecture	A set of design principles including the organization of functions and the description of data formats and procedures used as the basis for the design and implementation of a network (ISO).

Network layer	Layer 3 in the ISO/OSI reference model, the logical network entity that services the transport layer responsible for ensuring that data passed to it from the transport layer is routed and delivered throughout the network.
Network topology	The physical and logical relationship of nodes in a network; the schematic arrangement of the links and nodes of a network typically in the form of a star, ring, tree or bus topology.
Network	An interconnected group of nodes or stations.
NMRR	Normal mode rejection ratio
Node	A point of interconnection to a network.
Noise	A name given to the extraneous electrical signals that may be generated or picked up in a transmission line. If the noise signal is large compared with the data carrying signal, the latter may be corrupted resulting in transmission errors.
NOS	Network operating system
NRM	Unbalanced normal response mode
NRZ	Non return to zero – pulses in alternating directions for successive 1 bits but no change from existing signal voltage for 0 bits.
NRZI	Non return to zero inverted
Null modem	A device that connects two DTE devices directly by emulating the physical connections of a DCE device.
Nyquist sampling theorem	In order to recover all the information about a specified signal it must be sampled at least at twice the maximum frequency component of the specified signal.

O

OD	Object dictionary
Ohm (Ω)	Unit of resistance such that a constant current of one ampere produces a potential difference of one Volt across a conductor.
Optical isolation	Two networks with no electrical continuity in their connection because an optoelectronic transmitter and receiver have been used.
OSI	Open systems interconnection

P

Packet	A group of bits (including data and call control signals) transmitted as a whole on a packet switching network. Usually smaller than a transmission block.
PAD	Packet access device – an interface between a terminal or computer and a packet switching network.

Parallel transmission	The transmission model where a number of bits is sent simultaneously over separate parallel lines. Usually unidirectional such as the Centronics interface for a printer.
Parity bit	A bit that is set to a '0' or '1' to ensure that the total number of 1 bits in the data field is even or odd.
Parity check	The addition of non-information bits that make up a transmission block to ensure that the total number of bits is always even (even parity) or odd (odd parity). Used to detect transmission errors but rapidly losing popularity because of its weakness in detecting errors.
Passive filter	A circuit using only passive electronic components such as resistors, capacitors and inductors.
PBX	Private branch exchange
PCIP	Personal computer instrument products
PDU	Protocol data unit
Peripherals	The input/output and data storage devices attached to a computer e.g. disk drives, printers, keyboards, display, communication boards, etc.
Phase modulation	The sine wave or carrier changes phase in accordance with the information to be transmitted.
Phase shift keying	A modulation technique (also referred to as PSK) used to convert binary data into an analog form comprising a single sinusoidal frequency signal whose phase varies according to the data being transmitted.
Physical layer	Layer 1 of the ISO/OSI reference model, concerned with the electrical and mechanical specifications of the network termination equipment.
PID	Proportional integral derivative – a form of closed loop control.
PLC	Programmable logic controller
Point to point	A connection between only two items of equipment.
Polling	A means of controlling devices on a multipoint line. A controller queries devices for a response.
Polyethylene	A family of insulators derived from the polymerization of ethylene gas and characterized by outstanding electrical properties, including high IR, low dielectric constant, and low dielectric loss across the frequency spectrum.
Polyvinyl chloride (PVC)	A general-purpose family of insulations whose basic constituent is polyvinyl chloride or its copolymer with vinyl acetate. Plasticizers, stabilisers, pigments and fillers are

	added to improve mechanical and/or electrical properties of this material.
Port	A place of access to a device or network, used for input/output of digital and analog signals.
Presentation layer	Layer 6 of the ISO/OSI reference model, concerned with negotiating suitable transfer syntax for use during an application. If this is different from the local syntax, the translation to/from this syntax.
Profibus	Process field bus developed by a consortium of mainly German companies with the aim of standardization.
Protocol entity	The code that controls the operation of a protocol layer.
Protocol	A formal set of conventions governing the formatting, control procedures and relative timing of message exchange between two communicating systems.
PSDN	Public switched data network Any switching data communications system, such as telex and public telephone networks, which provides circuit switching to many customers.
PSK	See Phase shift keying
PSTN	Public switched telephone network – this is the term used to describe the (analog) public telephone network.
PTT	Post, Telephone and Telecommunications Authority or: push to talk signal
PV	Primary variable

Q

QAM	Quadrature amplitude modulation
QPSK	Quadrature phase shift keying

R

R/W	Read/write
RAM	Random access memory – semiconductor read/write volatile memory. Data is lost if the power is turned off.
Reactance	The opposition offered to the flow of alternating current by inductance or capacitance of a component or circuit.
REJ	Reject
Repeater	An amplifier that regenerates the signal and thus expands the network.
Resistance	The ratio of voltage to electrical current for a given circuit measured in Ohms.
Response time	The elapsed time between the generation of the last character of a message at a terminal and the receipt of the first character of the reply. It includes terminal delay and network delay.

RF	Radio frequency
RFI	Radio frequency interference
Ring	Network topology commonly used for interconnection of communities of digital devices distributed over a localized area, e.g. a factory or office block. Each device is connected to its nearest neighbors until all the devices are connected in a closed loop or ring. Data is transmitted in one direction only. As each message circulates around the ring, it is read by each device connected in the ring.
RMS	Root mean square
RNR	Receiver not ready
ROM	Read only memory – computer memory in which data can be routinely read but written to only once using special means when the ROM is manufactured. A ROM is used for storing data or programs on a permanent basis.
Router	A linking device between network segments which may differ in layers 1, 2a and 2b of the ISO/OSI reference model.
RR	Receiver ready
RS	Recommended standard (e.g. RS-232C) – newer designations use the prefix EIA (e.g. EIA-RS-232C or just EIA-232C).
RS-232-C	Interface between DTE and DCE, employing serial binary data exchange. Typical maximum specifications are 15 m (50 feet) at 19200 Baud.
RS-422	Interface between DTE and DCE employing the electrical characteristics of balanced voltage interface circuits.
RS-423	Interface between DTE and DCE, employing the electrical characteristics of unbalanced voltage digital interface circuits.
RS-449	General purpose 37 pin and 9 pin interface for DCE and DTE employing serial binary interchange.
RS-485	The recommended standard of the EIA that specifies the electrical characteristics of drivers and receivers for use in balanced digital multipoint systems.
RSSI	Receiver signal strength indicator
RTS	Request to send
RTU	Remote terminal unit – terminal unit situated remotely from the main control system.
RxRDY	Receiver ready

S

S/N	Signal to noise (ratio)
SAA	Standards Association of Australia
SAP	Service access point

SDLC	Synchronous data link control – IBM standard protocol superseding the bisynchronous standard.
SDM	Space division multiplexing
SDS	Smart distributed system
Serial transmission	The most common transmission mode in which information bits are sent sequentially on a single data channel.
Session layer	Layer 5 of the ISO/OSI reference model, concerned with the establishment of a logical connection between two application entities and with controlling the dialogue (message exchange) between them.
SFD	The start of frame delimiter
Short haul modem	A signal converter that conditions a digital signal for transmission over dc continuous private line metallic circuits, without interfering with adjacent pairs of wires in the same telephone cables.
Signal to noise ratio	The ratio of signal strength to the level of noise.
Simplex transmissions	Data transmission in one direction only.
Slew rate	This is defined as the rate at which the voltage changes from one value to another.
SNA	Subnetwork access, or systems network architecture
SNDC	Subnetwork dependent convergence
SNIC	Subnetwork independent convergence
SNR	Signal to noise ratio
Software	Refers to the programs that are written by a user to control the actions of a microprocessor or a computer. These may be written in one of many different programming languages and may be changed by the user from time to time.
SOH	Start of header (ASCII Control-A)
Space	Absence of signal. This is equivalent to a binary 0.
Spark test	A test designed to locate imperfections (usually pinholes) in the insulation of a wire or cable by application of a voltage for a very short period of time while the wire is being drawn through the electrode field.
SRC	Source node of a message
SREJ	Selective reject
Star	A type of network topology in which there is a central node that performs all switching (and hence routing) functions.
Statistical multiplexer	A device used to enable a number of lower bit rate devices, normally situated in the same location, to share a single,

higher bit rate transmission line. The devices usually have human operators and hence data is transmitted on the shared line on a statistical basis rather than, as is the case with a basic multiplexer, on a pre-allocated basis. It endeavors to exploit the fact that each device operates at a much lower mean rate than its maximum rate.

STP	Shielded twisted pair
Straight through pinning	RS-232 and RS-422 configuration that match DTE to DCE, pin for pin (pin 1 with pin 1, pin 2 with pin 2, etc.).
STX	Start of text (ASCII Control-B).
Switched line	A communication link for which the physical path may vary with each usage, such as the public telephone network.
SYN	Synchronous Idle
Synchronization	The coordination of the activities of several circuit elements.
Synchronous transmission	Transmission in which data bits are sent at a fixed rate, with the transmitter and receiver synchronized. Synchronized transmission eliminates the need for start and stop bits.

T

Talker	A device on the GPIB bus that simply sends information on to the bus without actually controlling the bus.
TCP	Transmission control protocol
TCU	Trunk coupling unit
TDM	Time division multiplexer A device that accepts multiple channels on a single transmission line by connecting terminals, one at a time, at regular intervals, interleaving bits (bit TDM) or characters (character TDM) from each terminal.
Telegram	In general a data block which is transmitted on the network. Usually comprises address, information and check characters.
Temperature rating	the maximum and minimum temperature at which an insulating material may be used in continuous operation without loss of its basic properties.
TIA	Telecommunications Industry Association
Time sharing	A method of computer operation that allows several interactive terminals to use one computer.
TNS	Transaction bytes
Token ring	Collision free, deterministic bus access method as per IEEE 802.2 ring topology.
TOP	Technical Office Protocol – a user association in USA which is primarily concerned with open communications in offices.

Topology	Physical configuration of network nodes, e.g. bus, ring, star, tree.
Transceiver	Transmitter/receiver – network access point for IEEE 803.2 networks.
Transient	An abrupt change in voltage of short duration.
Transport layer	Layer 4 of the ISO/OSI reference model, concerned with providing a network independent reliable message interchange service to the application oriented layers (Layers 5 through 7).
Trunk	A single circuit between two points, both of which are switching centers or individual distribution points. A trunk usually handles many channels simultaneously.
TTL	Transistor-transistor logic
Twisted pair	A data transmission medium, consisting of two insulated copper wires twisted together. This improves its immunity to interference from nearby electrical sources that may corrupt the transmitted signal.

U

UART	Universal asynchronous receiver/transmitter – an electronic circuit that translates the data format between a parallel representation, within a computer, and the serial method of transmitting data over a communications line.
UHF	Ultra high frequency
Unbalanced circuit	A transmission line in which voltages on the two conductors are unequal with respect to ground e.g. a coaxial cable.
Unloaded line	A line with no loaded coils that reduce line loss at audio frequencies.
UP	Unnumbered poll
USB	Universal serial bus
USRT	Universal synchronous receiver/transmitter. See UART.
UTP	Unshielded twisted pair

V

V.35	ITU standard governing the transmission at 48 kbps over 60 to 108 kHz group band circuits.
Velocity of propagation	The speed of an electrical signal down a length of cable compared to speed in free space expressed as a percentage.
VFD	Virtual field device – a software image of a field device describing the objects supplied by it e.g. measured data, events, status etc. which can be accessed by another network.
VHF	Very high frequency

VLAN	Virtual LAN
Volatile memory	An electronic storage medium that loses all data when power is removed.
Voltage rating	The highest voltage that may be continuously applied to a wire in conformance with standards of specifications.
VRC	Vertical redundancy check
VSD	Variable speed drive
VT	Virtual terminal

W

WAN	Wide area network
Word	The standard number of bits that a processor or memory manipulates at one time. Typically, a word has 16 bits.

X

X.21	ITU standard governing interface between DTE and DCE devices for synchronous operation on public data networks.
X.25	ITU standard governing interface between DTE and DCE device for terminals operating in the packet mode on public data networks.
X.25 Pad	A device that permits communication between non X.25 devices and the devices in an X.25 network.
X.3/X.28/X.29	A set of internationally agreed standard protocols defined to allow a character oriented device, such as a visual display terminal, to be connected to a packet switched data network.
X-ON/X-OFF	Transmitter on/transmitter off – control characters used for flow control, instructing a terminal to start transmission (X-ON or control-S) and end transmission (X-OFF or control-Q).
XOR	Exclusive-OR