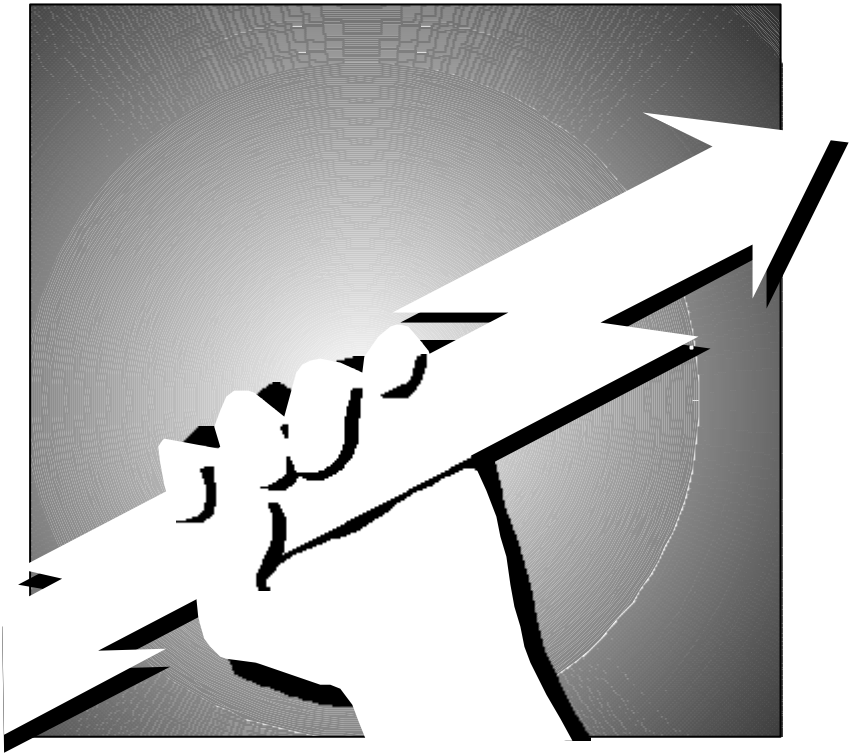
brain boxes



Serial Solutions for Windows 95

1.5 EDITION March 1997

Guarantee.

FULL 36 MONTHS GUARANTEE.

BRAIN BOXES guarantee your Serial Port Card for a full 36 months from purchase, parts and labour, provided it has been used in the specified manner. In the unlikely event of failure return your interface to BRAIN BOXES or to your Dealer, with proof of purchase, who will determine whether to repair or replace this product with an equivalent unit.

COPYRIGHT.

Serial Solutions are designed, manufactured, and supported by

BRAIN BOXES.

Unit 3F, Wavertree Boulevard South,
Wavertree Technology Park,
Liverpool, L7 9PF,
England.

Telephone: 0151-220 2500
Fax: 0151-252 0446
E-mail: help@brain_boxes.cybase.co.uk
Web: www.connect.org.uk/merseymall/brain_boxes/

ACKNOWLEDGEMENTS.

Warning !.

Unauthorised copying of Serial Solutions is a crime !

Although public domain programs allow unlimited copying, and much commercial software allows copying provided the program purchased is only in use on only one machine at a time, many programs specifically forbid copying of proprietary material. The copyright laws of most countries forbid unauthorised copying of computer programs without the authors permission. Thus you will be committing a crime when unlawfully copying programs from one machine to another, and so will be liable to arrest.

Software authors, Like the author of Serial Solutions, earn their living from revenue due to the sales of their programs, it is legally and morally wrong to steal their income by misuse of a computers copy routines. In these circumstances you should buy extra copies of the program.

Don t make yourself into a thief! Moral debilitation will result from habitually committing these crimes. You have been warned.!

Licences allowing multiple copies of Serial Solutions are available from your supplier!

Thank You For Buying
Serial Solutions for Windows 95

Windows Serial Solutions For Windows 95.

Serial Solutions for Windows 95 requires the following hardware:-

a) An Industry compatible computer running Windows 95.

b) Any PC standard serial ports, either on main system board, or on expansion cards based on the 8250, 16450, 16550, 16650 or 16750 UARTs. Up to 255 ports are directly supported. Of course we recommend our own AT One port, Dual Port, Triple Port, Quad Port and Lynx 8 Port cards which are made in Britain and have a 36 month guarantee.

c) Correct cabling to connect the serial port(s) to the remote device

Serial Solutions For Windows 95 Overview

Serial Solutions for Windows 95 supports any mix of RS232, RS422 and RS485 Serial cards in the PC.

Serial Solutions support interrupt sharing boards including our own Quad and Lynx cards as well as most third party manufacturers including those based on the AST cluster card and the Digiboard PC/4 and PC/8 cards.

Serial Solutions supports the 16750 Velocity UARTs as well as standard 8250,16450, and 16550 UARTs.

The Layout Of This Manual

This manual describes how to install and use the Serial Solutions for Windows 95 software suite.

Chapter 1 - Introduction to Windows 95 Serial Communications, provides information about Windows 95 Serial Communication, a description of the serial solutions device driver and its components

Chapter 2 - Software Installation Guide, provides an overview on the determination of Windows 95 I/O and IRQ usage, configuration, installation of the serial card, a detailed guide to Serial Solutions installation and serial port and Serial Solutions uninstallation procedures.

Chapter 3 - Users Guide, details RS422/485 communication and the Serial Solutions RS485 mode support, and a section on related problem solving.

Chapter 4 - Sample Software Guide, Using sample code, written in Microsoft Visual C 4.0, shows how to perform simple communications with a serial port using the Windows 95 API function calls, all source code fully documented.

Chapter 5 - Communications Jargon, a discussion of serial communications, the RS232 standard, port pin outs, wiring diagrams for cross over and loop back cables, and shared interrupts.

CHAPTER 1
INTRODUCTION TO WINDOWS 95 SERIAL COMMUNICATIONS

Windows 95 Standard Communications Port

Windows 95 includes a 'Communications Port' as one of several standard port types which supports any serial port using 8250, 16450 or 16550 compatible serial interfaces, (or UARTs), and works even when several serial ports share an interrupt (IRQ number).

Configuration of this standard driver is similar to the Serial Solutions device driver, and these instructions can be used as a reference for configuring standard ports - but select **standard port types** and **Communications Port** rather than **Serial Solutions Port** at the relevant point.

Serial Solutions Device Driver

The Serial Solutions driver for Windows 95 is designed for use with Serial Solutions serial cards and offers these benefits over the standard driver:

- Through supporting the Serial Solutions Shared Interrupt Mechanism the driver can go directly to the serial ports generating an interrupt rather than search through each in turn. This means faster interrupt servicing and better throughput, and no chance for simultaneous interrupts to be lost.
- Support for RS485 communications, by automatically disabling the transmitter after a transmission completes. This helps in correctly receiving responses from interconnected equipment.

- Support for new "Velocity" 16750 UARTs, allowing serial communications at up to 1 Mega Baud, with a massive 64 Byte FIFO buffer.

Note: The first port name assigned to the Serial Solutions device driver is typically COM5.

The serial solutions for Windows 95 software will support all standard dumb PC serial Ports based on the industry standard 8250, 16450, 16550, and 16750 family of UARTs. Several multiport interrupt sharing cards including our own Lynx 8 port and Quad serial cards and the Digiboard PC/4 and PC/8 cards are also supported.

About Installing a Device Driver

Normally a device driver will be installed to control and interface with a serial card. This will assign a name, such as COM5 or COM6, to each serial port and allow simple access and control through Windows 95.

Windows 95 supports up to 255 serial ports, named COM1, COM2, all the way up to COM255, all of which are potentially available to Win32 programs. However, specific programs may be restricted in which serial ports they can use, e.g. only COM1 to COM4. Also, Microsoft state variously that either the first 9 serial ports or COM1 to COM9 are available from 16 bit Windows applications, which is similar to Windows 3.1.

Not Installing a Device Driver

It is not necessary to install a device driver for any serial port that you intend to control directly (with IN and OUT instructions) in your own programs.

It is recommended however that you **do** use a communications driver as there is little, if any performance gain to be made, by battling with the complexities of direct hardware programming in Windows.

The Serial Solutions software consists of several components, you may not wish to install all of these depending on your application. For this reason you should read this section.

The Serial Solutions for Windows 95 Driver Disk consists of the following Programmes:-

Bbcommsp.inf	The information file to aid the installation process "Have Disk...."
Bbcommui.dll	The DLL and..
BBcomm.vxd	..the virtual device driver providing the shared interrupt handler and dispatch routines.

The above files **must** be used if you wish to install the Serial Solutions Driver. They are copied to your system by the "Have Disk" process.

The Serial Solutions for Windows 95 Sample Disk is constantly being updated. You should refer to the README.txt file supplied on the disk for current contents.

The Sample Programs are required only if you do **not** have existing software that accesses Serial Ports above COM4, or intend to write you own communications software. The source code is provided alongside the sample .exe files as an aid to programming the serial ports. The supplied sample software is independent of the Serial Solutions driver, It will work with any installed Windows 95 or third party communications driver.

About Installing your Serial Card

Refer to the card's hardware manual for instructions on installing a serial card into your PC. This also gives information on configuring switches and jumpers, if there are any, and how to choose suitable addresses and interrupts (IRQ). Record which address and IRQ are used for each port, and if using a multiport card with a compatible Shared Interrupt Mechanism, the address of the **Shared Interrupt Status Register** (SISR) and the **card id** that has been selected. These will be needed when installing the Serial Solutions driver or writing your own software to control the serial ports.

Serial Solutions for Windows 95 is designed, written and manufactured in England, and our policy is one of complete support to our dealers and direct to our users. Please note that Serial Solutions is designed 'in house' and is completely understood by our staff. It's great strength is the support we give it. Our intention is to supply the software and any technical information you may need to allow you to exercise complete control over your serial ports and devices. After searching the manual, do not hesitate to contact us on our HOTLINE number given on the inside front cover, if you need help.

CHAPTER 2
SOFTWARE INSTALLATION GUIDE

Introduction

This chapter explains how to install the Serial Solutions for Windows 95 driver, and configure your serial ports for use in the shortest possible time.

Installation Procedure

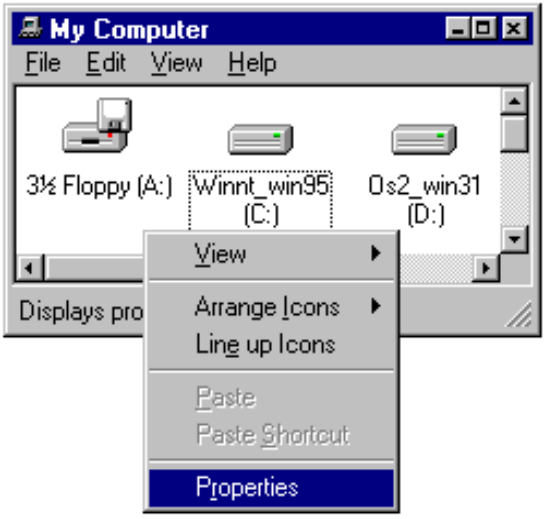
The suggested installation sequence is:

- Check Windows 95's I/O Use, to determine which IRQ's and I/O addresses are already in use on your PC, and thus which are available.
- Configure and install the serial card, noting the settings of jumpers and switches.
- Install the Serial Solutions device driver.

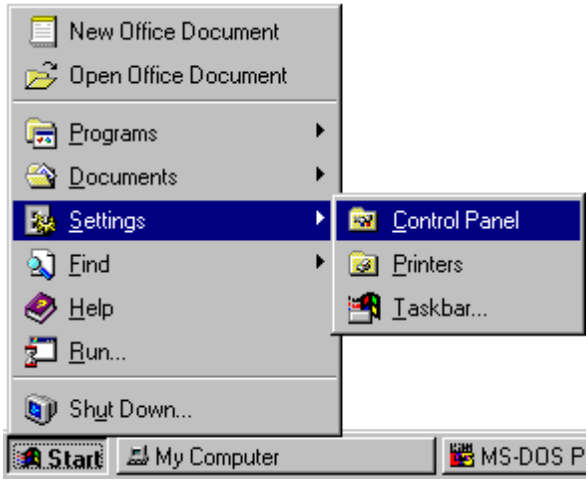
Checking Windows 95 s I/O Use

The simplest way to find out which I/O addresses and IRQ's are potentially available for the serial cards is to ask Windows 95 which ones it believes to be free.

This can be done through the **Device Manager**. There are several possible routes here, the shortest is to right click on the desktop *My Computer* icon, (or any other name you may have changed it to), select **Properties** from the menu:



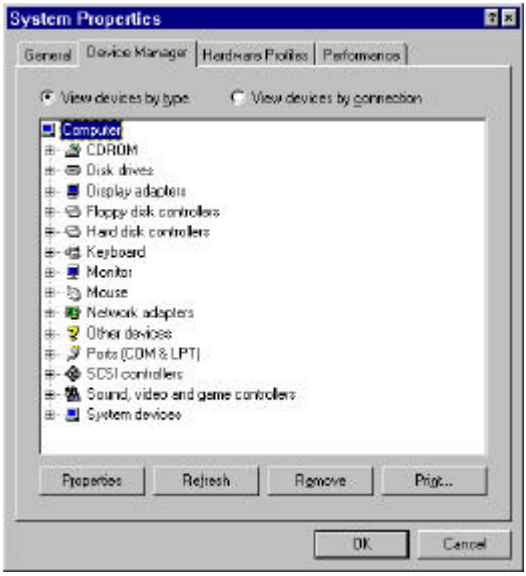
Alternatively open the **Start** menu, choose **Settings** and select the **Control Panel**,



from here double click on the **System** icon

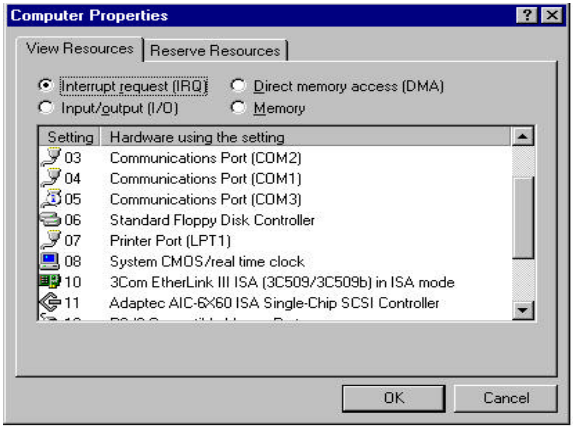
In either case, now click on the **Device Manager** tab.

Select the **View devices by type** radiobutton, if it is not already selected.



The Computer should be highlighted, as above. From here the **Print** button will print details of the interrupts and I/O addresses used or the

Properties button allows on screen inspection. The radio button at the top of the dialogue selects which information is displayed.



PCs have a limited number of interrupt inputs, many of which are used by standard peripherals such as the floppy and hard disk drives, timers and network cards. The Shared Interrupt Mechanism requires only one PC IRQ input for a Compatible Multiport card with multiple serial ports, and that interrupt can also be shared between multiple serial cards.

In theory Windows 95 allows the same IRQ to be used by multiple serial ports, (though not shared by a serial port and another peripheral). In practice this does not always work, especially if one of the ports is connected to a serial mouse.

Configuring and Installing the Serial Card

Install the serial interface card in an available slot, as documented in the manufacturers hardware manual for the card.

Depending on the type of cards being installed, and how they are configured, you should note the configuration of each serial port:

1. If you are **not** using the **Shared Interrupt Mechanism** for any ports, or it is not available on the cards you are installing, note;

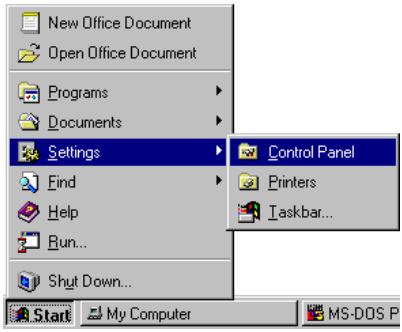
- The input/output address. (the relationship between the COM1, etc. jumper positions and the input/output address should be documented in the relevant manufactures hardware manual.)
- The IRQ number you are using.

2. If any ports **are** using the **Shared Interrupt Mechanism** note;
   - The Shared Interrupt Status Register (SISR) address (the default for Serial Solutions cards is 3A0hex).
   - The Card ID (the default for Serial Solutions cards is 0). Each logical card has 8 ports on, so the Card ID is set in units of 1 (0 to 7) for 8 port serial cards and units of 0.5 (0 to 3.5) for 4 port serial cards.
   - Determine the input/output addresses used by each serial port, (determined by the COM1, etc. jumpers or the BANK ADDR switches).
   - The IRQ number for the shared interrupt, and any ports you have selected to occupy separate interrupts.

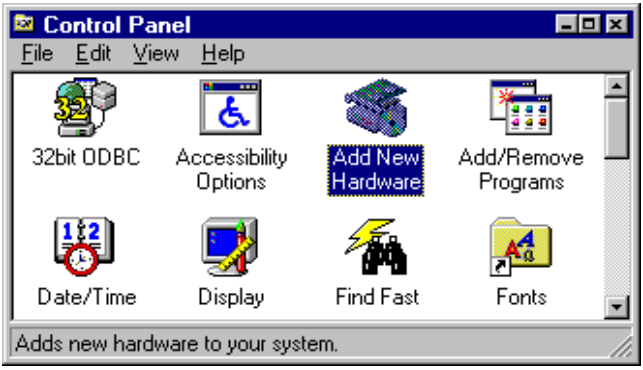
Serial Solutions Installation for Windows 95

The following steps describe the installation of the Serial Solutions driver for Windows 95, which is supplied on one 3.5" floppy disc. The listed installation procedure assumes that only 1 COM port (COM1) is present.

Open the **Control Panel** - there are several routes to the **Control Panel**, the simplest is to open the **Start** menu and select **Settings**.



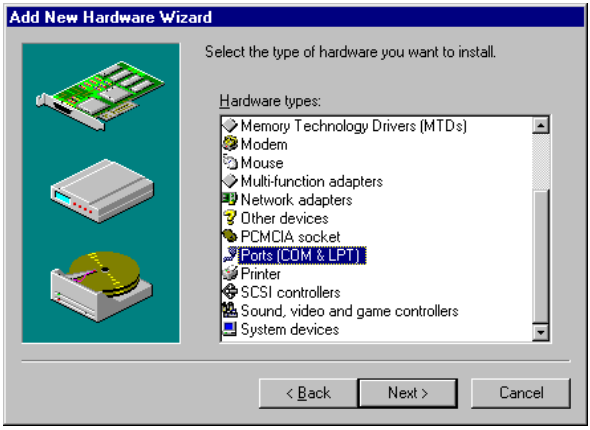
Double click the **Add New Hardware** icon in the control panel.



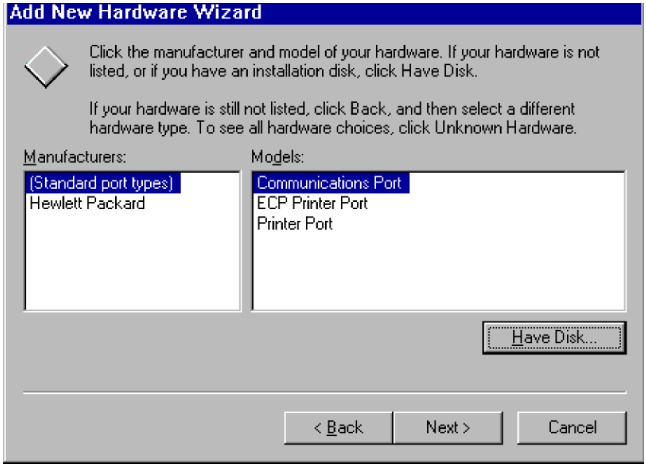
Click **next** on the applet dialogue.

The **Add New Hardware** wizard will ask you if you wish Windows to search for your hardware. Click the **No** radiobutton since Windows cannot find Multiport Serial Solutions serial ports and it will save some time. Click **next**

From the hardware types list on the next page select **Ports (COM & LPT)**.Click **next**.

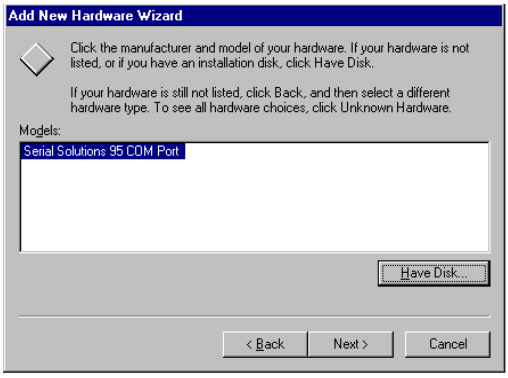


Select **Standard Port Type** and then **Communications Port**, and then click **Have Disk**.



Windows will then ask you for the location of the Serial Solutions files, assuming your floppy drive is designated A:, you will see the following:

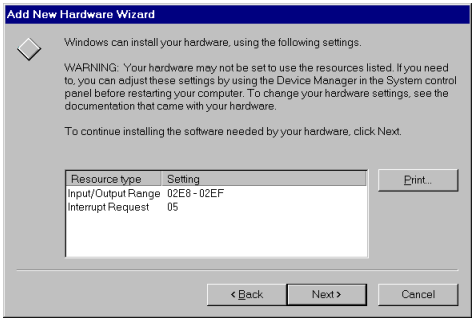
After the installation procedure, the Window will display the entry Serial Solutions for Windows 95:



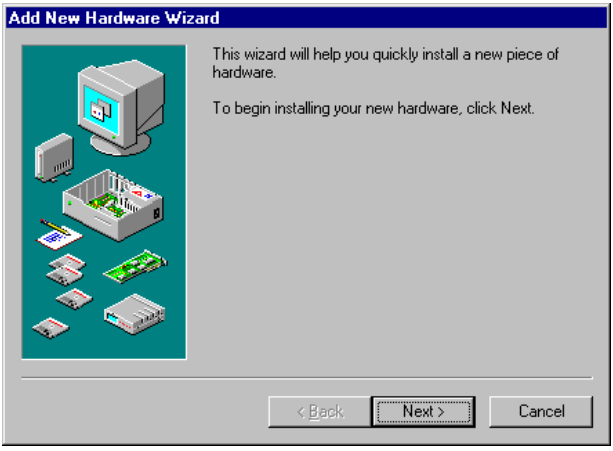
Select the **Serial Solutions for Windows95 Port**. Click **next.**

Windows 95 will then inform you of the settings it has assumed for the new ports. The Windows 95 default addresses for new serial ports do not match the default addresses on Serial Solutions serial cards. These resources will need to be modified after installation to reflect those which you have set on the card. To do this complete installing all the required ports and before re-booting modify the resource settings to those on the card by following the instructions in the section "**Configuring Ports.**"

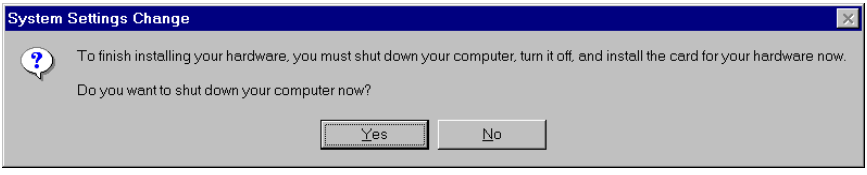
Click **next.**



Click **finish**. Windows will install the selected port.



You will then be asked if you wish to re-boot the system. Since the Serial Solution port entries will require some configuration changes after installation select **No**.

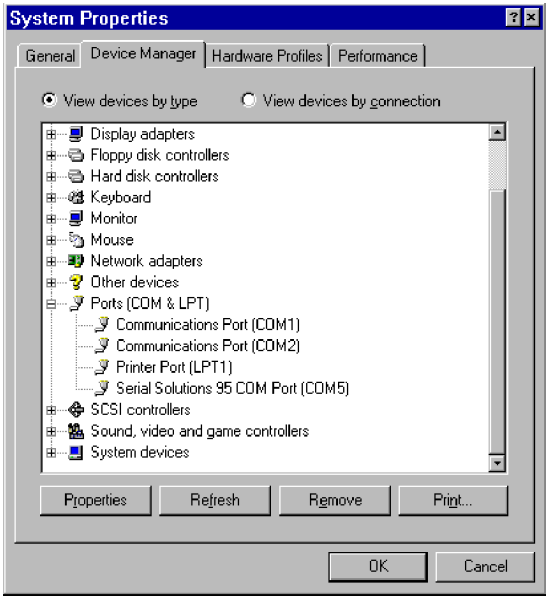


N.B. Each port needs to be installed and configured separately

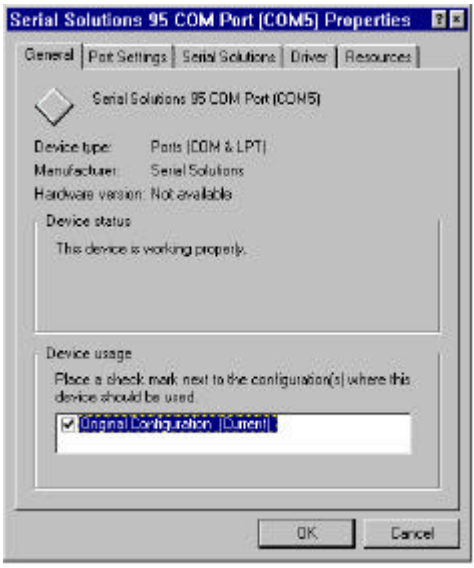
Configuring Ports

For each port you have installed you must verify through the **Device Manager** that all the resource settings are correct for the port you have installed. There are several routes here, the shortest is to right click on the desktop *My Computer* icon, (or any other name you may have changed it to), and select **Properties** from the menu. Alternatively open the **Start** menu, choose **Settings** and select the **Control Panel**, from here double click on the **System** icon.

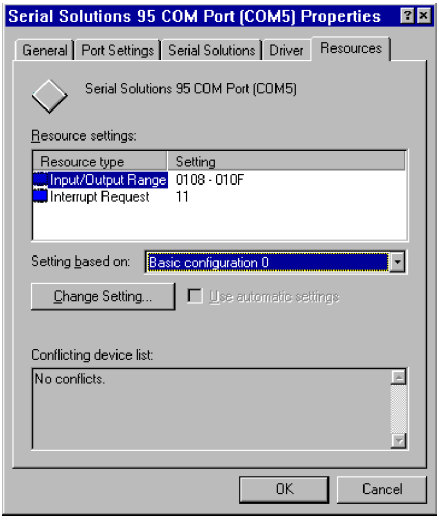
Select the **Device Manager** tab at the top of the dialogue.



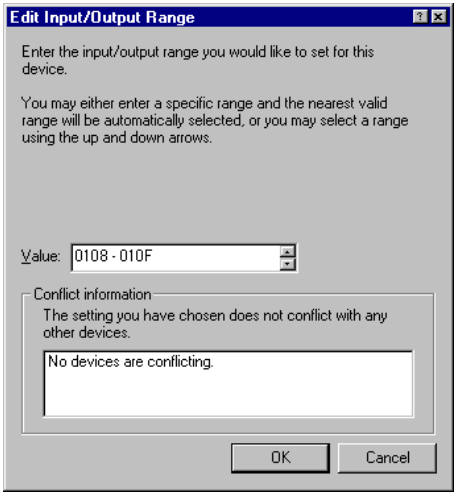
Select the **View devices by type** radiobutton, if it is not already selected. Click the + sign to the left of the **Ports (COM & LPT)** entry in the list. Double click the first of the Serial Solutions ports you have just installed.



Click the **Resources** tab.

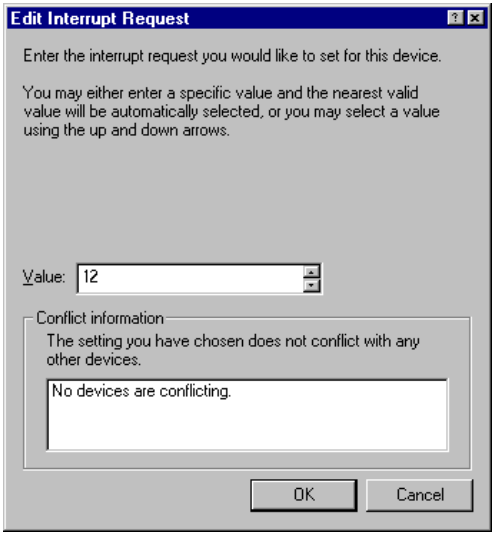


If the Input/Output Range setting does not reflect those set on the card (which will most probably be the case) double click the Input/Output Range text on the dialogue.



Use the scroll bars to change the address to that which the port has been set to on the hardware. Click OK.

If the Interrupt Request number does not reflect that set on the port, either the separate interrupt allocated to a specific port, or the shared interrupt (if the port is configured to use that) needs to be added instead. double click the Interrupt Request text on the dialogue.

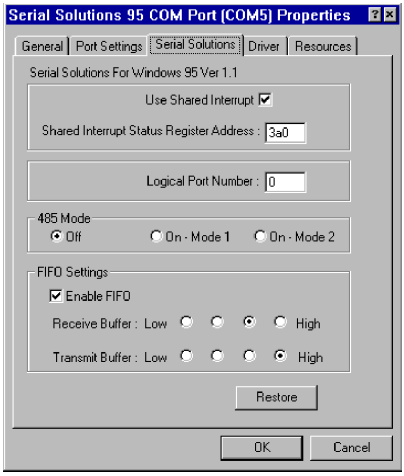


Use the scroll bars to set the port to the correct interrupt number.
Click OK. The resources dialogue should now show the correct settings for the port. When you are sure the card has been correctly configured go to the section entitled "**Restarting Windows**"

Configuring the Card for Shared Interrupts

For Serial Solution cards that have been configured to support shared interrupts (see the appropriate hardware manual for guidance) the **Use Shared Interrupt** checkbox must be ticked, the shared interrupt status register address must be entered and appropriate I/O addresses entered.

Click the **Serial Solutions** tab



By default the installation will show '3A0' which is the factory shipped default setting. This procedure is carried out for all ports that are to have shared interrupts.

The logical port number must be set as detailed in the appropriate hardware manual. Essentially, for a single card set-up, the logical port number will start from '0' for the first port on the card, '1' for the second port, '2' for the third and so on. For multiple card, daisy-chained configurations, the logical port numbers continue

incrementing through all cards, though this is dependant on the order that the cards have been daisy-chained in hardware. For card with a Card ID of 1 the first logical port is 8, for a (4 port) card with a Card ID of 0.5 the first logical port is 4.

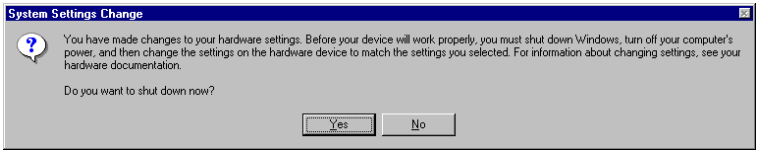
In Order to check the settings of each port are correct, refer to the previous section "**Port Configuration**"

When you are sure the card has been correctly configured go to the section entitled "**Restarting Windows**"

Restarting Windows

Whenever certain values have been entered or changed in the hardware settings window, a message prompting to restart Windows will appear. Only after having made ALL the necessary changes restart Windows so that the new settings come into effect.

Windows 95 issues a warning to ask you to confirm your changes and asks you if you wish to shut down the computer. If you have more ports to configure select No, otherwise select Yes.



Repeat steps from "**Configuring Card for Shared Interrupts**" or "**Configuring Ports**" section, as appropriate for each port that you have installed. When the last port has been configured correctly re-boot the system to use these correct settings.

After the computer has been re-booted the new ports should be available for use.

N.B. Windows 95 Will only allow ports to be called COM1, 2, 3 or 4 only if their addresses are set to the industry standards of 3F8, 2F8, 3E8 and 2E8hex, respectively.

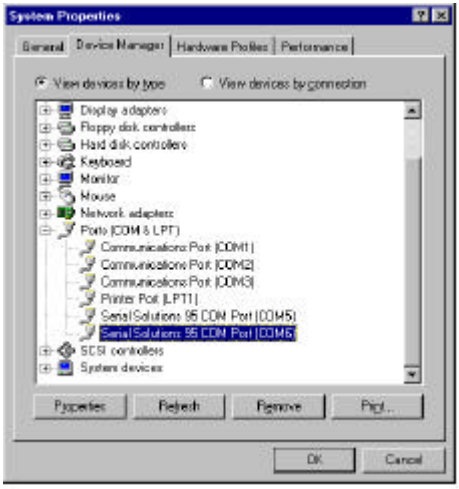
If you want a multiport card to be COM3, for example, the jumpers on the card must be set to COM3 (not bank). When a new port is added it will be named as the next available port above COM5, if the base address is changed in **Device Manager** to reflect the card settings, then when rebooted, the port will be called COM3. If you want the port to called COM1, COM2, or COM4 then the same procedure should be adopted.

Removing a Serial Solutions Port

Removal of the device driver is performed through the **Device Manager**. There are several possible routes here, the shortest is to right click on the desktop **My Computer** icon, (or any other name you may have changed it to), select **Properties** from the menu and then click on the **Device Manager** tab at the top of the dialogue. Alternatively open the **Start** menu, choose **Settings** and select the **Control Panel**, from here double click on the **System** icon and then click on the **Device Manager** tab.
From the Device Manager proceed in the following manner:

- Select the **View devices by type** radiobutton if it is not already selected.

- Click the + sign to the left of the **Ports (COM & LPT)** entry in the list.



- Select the port you wish to remove from the system.
- Click the **Remove** button at the bottom of the dialogue.
- You will see a warning message giving you the option of cancelling the operation.
- If you want to continue click **OK** otherwise click **Cancel**.
- The device will be removed from the system.
- Repeat from step 3 for any further serial ports you wish to remove.

Removing Serial Solutions Drivers

Due to the nature of the installation procedure Serial Solutions for Windows 95 is not uninstallable via the Windows Uninstall Wizard, therefore you will need to delete the files that comprise Serial Solutions. These files are:

Bbcommsp.inf	The information file to aid the installation process "Have Disk...."
Bbcommui.dll	The DLL and..
BBcomm.vxd	..the virtual device driver providing the shared interrupt handler and dispatch routines.

However, before the file deletion is attempted **ALL** Serial Solutions ports must be removed in the manner described in the section above, entitled "**Removing a Serial Solutions Port**". When you are certain you have removed all of the Serial Solutions for Windows 95 port, i.e. none are visible in the **Ports (COM & LPT)** entry in the **Device Manager System** tab, the above files can then be deleted.

The location of these files must be determined. The standard installation process places the files Bbcommui.dll and Bbcomm.vxd within the Windows 95 system directory, and Bbcommsp.inf in the profiles directory. However due to variations in a users installation of Windows 95 etc. these files may not be located in the above named directories, so it is wise to use the **Find** function located in the **Start** menu, and use it to search for the file locations. Once they have been located, the files can then be manually deleted from their locations.

Once you are sure that the appropriate files have been deleted, shut down all applications and exit and restart Windows; Windows should then restart in the normal way.

CHAPTER 3
USER GUIDE

Introduction

Once installed, the use of the Serial Solutions serial ports should be straightforward. In practice most problems are likely to come from incorrect assumptions about the default or previous state of the serial port mode. It is therefore important to ensure that details such as parity, handshake control, etc. are set to the values you want.

If you are configuring RS485 serial ports there are a couple of further issues to be careful of: correct selection of mode jumpers on the serial cards; whether to use one of the RS485 control modes supported by the serial device driver.

RS422 and RS485 Communication

RS422/485 serial ports are often used for applications such as controlling or monitoring industrial equipment in factories.

RS422 (with pairs of balanced wires), like RS232 (with single wires), connects the receive and transmit from the PC to the transmit and receive connections on an item of equipment. This allows simultaneous communication in both directions, known as **full duplex**. This can potentially be extended so that the transmit data from the PC is connected to the receive data connection on multiple items of equipment, so that the PC can send data to all of them. However this implies that data transfer is basically one way, or that a communication protocol is used to identify which item of equipment can reply, (with the receive data on the PC connected to the transmit data of all the items of equipment).

RS485 interconnection of equipment, through connecting all the transmit and receive connections together, similarly requires a protocol that allows only one item of equipment to transmit data at a time. When the PC intends to transmit data it must raise the RTS signal, which

enables the RS485 transmitter, and wait until after the transmission is complete before lowering RTS. Conversely RTS must be lowered before external data can be received. (Depending on the hardware configuration of the port whilst RTS is high the receiver may either be disabled or receiving the very same data that it is transmitting.)

With RS485 the time taken between completion of a transmit and being ready to receive a reply is typically limited by the delay in disabling RTS, this is known as **receive turnaround**. The time taken between receiving data and transmitting a reply is known as **transmit turnaround** and should be longer than the receiving equipment's worst receive turnaround. It is therefore important that RTS is not disabled either too quickly, (corrupting the end of the transmitted data), or too slowly, (possibly missing the reply).

Serial Solutions RS485 Modes

The RS485 modes supported by the Serial Solutions driver can help.

Mode 1 turns off RTS immediately the last character is transmitted, but sits watching the data leave the serial port. Although this assures rapid electrical turnaround it can prevent some interrupts being noticed, possibly corrupting other serial data channels or network communication.

Mode 2 notices when the last data byte is written to the serial port. Then it watches at 'idle' moments of the system for the transmission to finish, when it disables RTS. The disadvantage is that a maximum delay cannot be guaranteed. On a lightly loaded 486 system the typical delay has been observed of between 8 and 12 milliseconds.

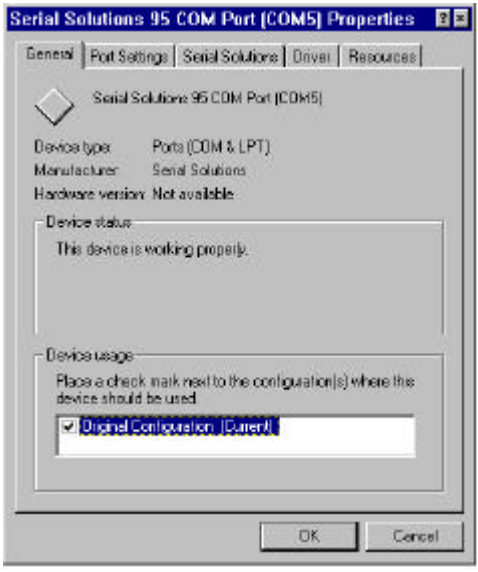
Making Configuration Changes

Changing the settings of an installed Serial Solutions port is performed through the **Device Manager**. There are several possible routes here, the shortest is to right click on the desktop *My Computer* icon, (or any other name you may have changed it to),

select **Properties** from the menu and then click on the **Device Manager** tab at the top of the dialogue. Alternatively open the **Start** menu, choose **Settings** and select the **Control Panel**, from here double click on the **System** icon and then click on the **Device Manager** tab.

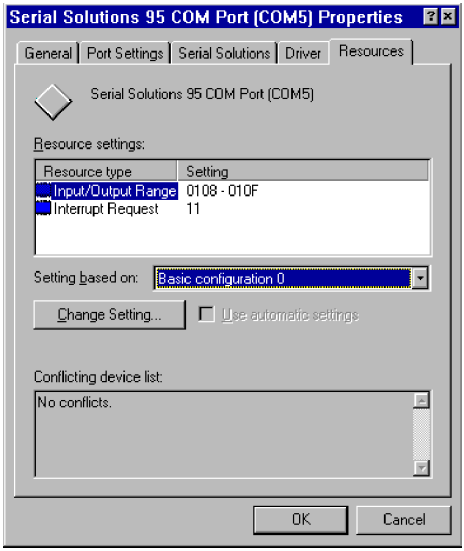
- Select the **View devices by type** radiobutton, if it is not already selected.
- Click the + sign to the left of the **Ports (COM & LPT)** entry in the list.
- Double click the port you wish to modify.

A five page dialogue appears:



Changing Resources

To change resource settings such as Input/Output address or interrupt (IRQ) used select the **Resources** tab.



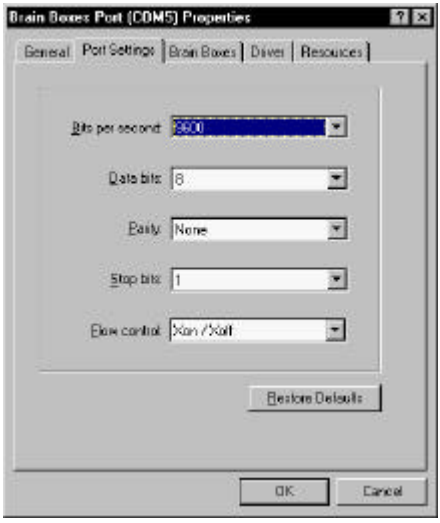
To change the specified I/O address:

- Double click the **Input/Output range** item. Type or select the new address in the dialogue. This address should match that set on the card itself for the port.

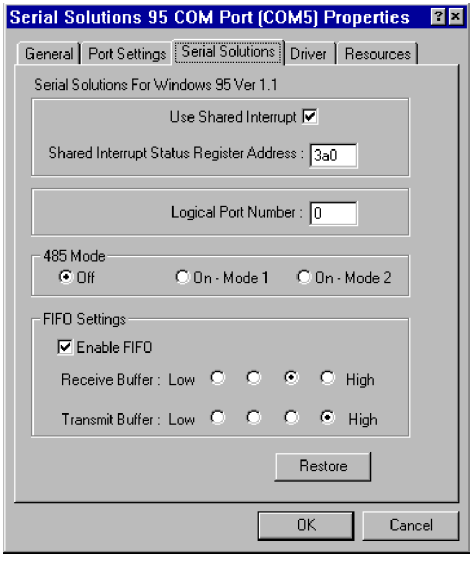
To change the specified interrupt number:

- Double click the **Interrupt Request** item. Either type or use the scroll buttons to set the new interrupt. Windows will inform you if the system has other devices which may conflict with your setting. Assuming all is well click **OK**.

To change the default baudrate, data bits, parity, stop bit and flow control settings you should use the **Port Settings** tab of the dialogue:



To change either logical port numbers or the shared interrupt status register address use the **Serial Solutions** tab of the properties dialogue.



All Serial solutions serial cards are based on the 16550 UART which includes a FIFO buffer that makes communications much more reliable. The **FIFO Enable** checkbox enables this feature and is on by default. It is recommended that this default is left as it is, though if you suspect that a fault may be related to this then uncheck the box.

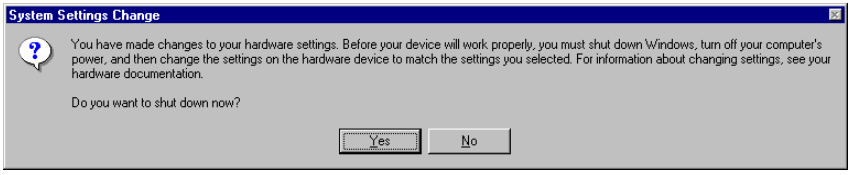
The Serial solutions driver includes the facility to automatically set the RTS line of the port to control 485 transmission. By default this is disabled. There are two methods of implementing RTS line control in the Serial Solutions driver, these are described in detail under **Serial Solutions RS485 Modes** earlier in the User Guide

When you are sure that you have correctly adjusted the port settings and you wish to make use of your new ports refer to the section directly below.

Restarting Windows

Whenever certain values have been entered or changed in the hardware settings window, a message prompting to restart Windows will appear. Only after having made ALL the necessary changes restart Windows so that the new settings come into effect.

Windows 95 issues a warning to ask you to confirm your changes and asks you if you wish to shut down the computer. If you have more ports to configure select No, otherwise select Yes.



After the computer has been re-booted the new ports should be available for use.

Problem Solving

Serial Ports don t work

Check that the Serial Solutions card and driver are correctly configured. Check that the interrupt (IRQ) number used is not also being used by another type of device, (not even a serial mouse). Check for any software you are running that may interfere with interrupts being serviced.

Serial or LAN communication is corrupted, or screen mouse moves jerkily

The Serial Solution Driver's RS485 modes attempt to accurately disable the RS485 transmitter at the correct time. This can cause delays to processing other events or interrupts, try a different RS485 mode or disable the transmitter in your own code.

CHAPTER 4
SAMPLE SOFTWARE GUIDE

Introduction

The sample code, written in Microsoft Visual C 4.0, shows how to perform simple communications with a serial port using the Windows 95 API (Application Programming Interface) function calls.

The terminal programs source code has been documented to show what each function does, and any ambiguity may be cleared by reading this chapter.

The terminal program simply writes a byte (Character) to the serial port, and reads any incoming bytes from the serial port, displaying the characters that have been sent to, or received from the serial port. It has been written to show how to write a program to accomplish this task without doing any complex processing on the characters that are received or transmitted.

Some sources of help are given, which you may find useful, at the end of this chapter.

Sample Software Files.

The following files will be extracted from the EZTERM.EXE file on the supplied disk.

Easycwin.c	C Program source code
Easycwin.rc	Resource script
Ezterm.mak	Program make file
Ezterm.mdp	Project file
Ezterm.ncb	Program file
Resource.h	Program header
Easycwin.exe	Executable program

Windows Programming Environment.

Windows works on a time slicing principle in that all programs are assigned various numbers of time slices. Each program then has to perform as many of it's tasks as possible, and when the time slices are used up by a program, Windows then allows another program to use a certain number of time slices. Windows keeps information in the form of messages, which it can send to a program to tell it the state of operations and to give the program various levels of information access, i.e. Windows uses a messaging system to communicate with the program. On this principle, Windows programming requires the implementation of so-called "callback procedures" which allow the communication with Windows.

The Windows programming environment prevents direct access to the hardware when device drivers are installed for that particular device. For example serial ports, video chips, etc. Hardware is thus made accessible to programs through the use of an Applications Programming Interface (API). This provides a range of functions, which interface between the user application(s) and the device driver.

Program Start-up Routine.

There are two functions required to produce a functional skeleton Windows executable program. A function to provide message translation and dispatch, and a call-back procedure which will handle these messages. Other languages may implement this differently.

The WinMain() function pumps the messages to the callback procedure, and the call-back procedure, which we have called TermProc(), performs tasks dependent on the message received. WinMain() is used to initialise the window that our program will use to display text and window controls. A class is defined for the window to tell Windows what call-back to use, what type of window to use, what background to use, what icon to use, etc. After registering that class, the program is able to use that class on the creation of a window.

The case statements in the call-back procedure determine the

type of message that we want to process. When the program is run, before the window is shown the message WM_INITDIALOG will be received by the call-back function, here we have 2 functions that get the terminal program ready for communications, and one API call.

The function OpenPort() contains an API call Createfile() which will open the logical communications port specified in the application dialogue. The return from Createfile() is used to identify if the device has initialised and opened OK. A device file is created and a handle assigned by the function which is then used for further comms.

The function SetupComms() contains an API call to SetCommState() which will apply serial port communications settings to use in communicating to the remote serial port, which must match in communications settings.

The API WaitCommEvent() which allows the notification message WM_COMMNOTIFY to be sent when there are bytes in the queue. Setting the number of characters to a higher number will result in fewer messages being sent if the bytes are transmitted fast and therefore allows the program to cope with faster data rates by not having to service notification messages too often.

Sending Characters.

This is a very straightforward process, the keypress is acknowledged by the receipt of the WM_CHAR message posting to the callback procedure, on which we can send the byte by using the WritePortFunction() using the character stored in the character buffer.

Reading Characters.

This is a not so straightforward process. A receipt of bytes in the queue is acknowledged by the receipt of the WM_COMMNOTIFY message. On receiving this message we read as many bytes from the receive queue as our temporary storage buffer will allow, using the function ReadPort(). Use the return from ReadComm() to determine whether an error has occurred or we have a

number of bytes that have been read.

Showing Characters.

To show the characters in this program, we have used multi-line edit controls which can hold a maximum of 64,000 bytes. To keep it to a limit we remove the first few characters every time our buffer is unable to hold any more characters, which will prevent the multi-line edit control from overflowing. This is a slow process and does not perform well for high baud rates.

A better solution would make the program very complex, but involves defining a rectangular region, and writing the string to display using the TextOut() function. This is very fast but involves handling scrolling functions and using algorithms to manipulate the text strings, which is used in the non-simple Serial Solutions Terminal program.

Exit Procedure.

This is accomplished when the user clicks on the OK or CANCEL buttons, or when the user clicks on the close system menu icon which is located at the top left corner of the Terminal program.

It simply ends communications with the serial port using the API function CloseComm() and then removes the window by calling the API function DestroyWindow().

Sources For Help.

Use of APIs are documented on most Windows compilers in the on-line help, but some of the example programs can be useless or non-existent, leaving the programmer to flounder by experimenting, or resorting to find other sources of help.

To optimally use programming techniques, a very good source of help and sample code is the Microsoft Developer Network Library which is a subscription based set of CDs geared towards developers.

There are also magazines that are highly recommended, but which may tend to C++ programming techniques, such as:

EXE - The Software Developers Magazine
Microsoft Systems Journal
C/C++ Users Journal
Dr Dobbs Journal
Windows/Dos Developers Journal
MSDN Subscription Library

and also of help are various books, of which Charles Petzold's book Programming Windows, is the most infamous amongst developers.

CHAPTER 5
COMMUNICATIONS JARGON.

Introduction.

This chapter gives details of the communications concepts and terminology which should be helpful in using your serial ports to their fullest capabilities.

About This Chapter.

There are three main sections to this chapter, an outline of which is given below.

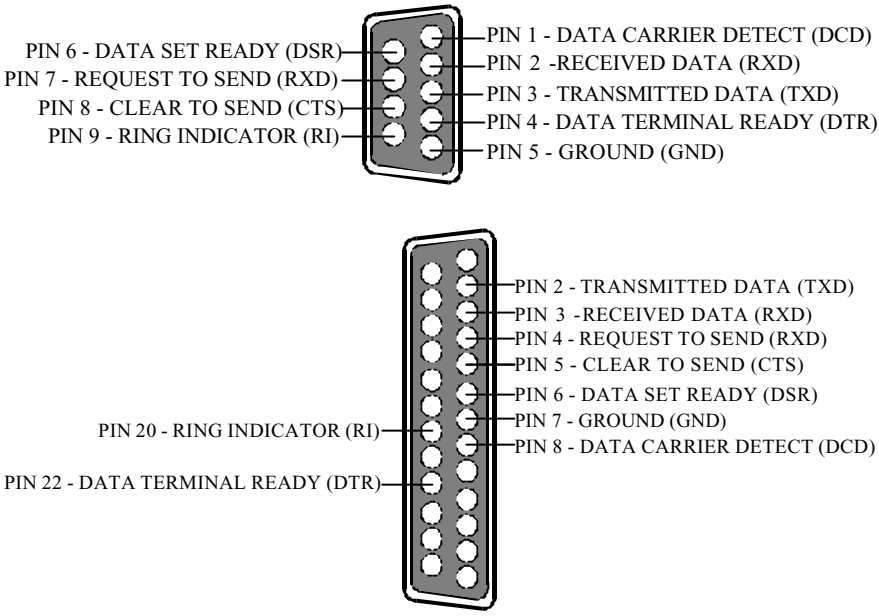
i) **Serial Communications**. This section gives details of what serial communications is and how it can be achieved, what chips can be used for good performance, giving pin-outs for the cabling requirements where necessary.
ii) **Communications Settings**. This section gives details of the serial communications settings which are used to specify speeds, methods of data checking, and data lengths used to communicate with a remote device or terminal.
iii) **Hardware Settings**. This section details what the hardware settings are, what they are used for, default settings and any related material which you may find useful.

Serial Communications.

Serial communications is the exchange of data using a device called a serial port, at the heart of which is the UART (Universal Asynchronous Receiver Transmitter). This is a programmable element which has an outlet to connect to a communications cable. The outlet is either a 9-pin Male D-type or a 25-pin Male D-type connector as

specified by the RS232 standard. The pin-outs for these connectors are shown in Figure 5-1.

Figure 5-1. Serial Port Pin Outs.



The signals at some of these pins, usually known as handshake lines, travel in the directions shown below.

SIGNAL	DIRECTION
RTS REQUEST TO SEND	OUTPUT FROM PC
CTS CLEAR TO SEND	INPUT TO PC
DSR DATA SET READY	INPUT TO PC
DCD DATA CARRIER DETECT	INPUT TO PC
DTR DATA TERMINAL READY	OUTPUT FROM PC
RI RING INDICATOR	INPUT TO PC

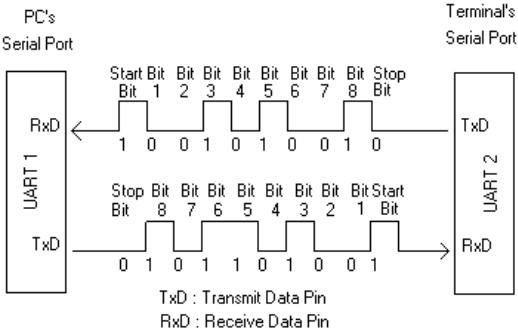
Two of the pins are used only for the exchange of data whose signal directions are given below.

TxD TRANSMITTED DATA	OUTPUT FROM PC
RxD RECEIVED DATA	INPUT TO PC

The other pins are either not connected or grounded (the pins can be used in whatever fashion you require provided you have software that will recognise the scheme you are going to use, however this may not conform to the RS232 standard).

A serial port transmits data by converting the parallel data from the CPU to a serial form, and then packs this data in between start and stop bits before being clocked out of the UART at a programmed baud rate. Conversely, when receiving data, the UART receives data in serial form, one bit at a time, and then removes the start and stop bits before converting it into a parallel form. Figure 5-2 illustrates communications using 8 data bits, no parity.

Figure 5-2. Serial Communication.



RS232 Standard.

The RS232 standard, introduced in 1962, is a widely established set of rules for the exchange of data between a computer and Data Communication Equipment (DCE) such as modem, or terminal.

The standard sets ranges or limits to such things as the speed of data transmissions, connector types, hardware line electrical ranges etc. This provides a common base to allow for one manufacturer of serial equipment to supply consistent hardware to a customer, which the customer can then mix with serial equipment bought from another manufacturer.

UART Performance.

The original IBM PC had a serial port based on the Intel 8250 but was replaced in the IBM AT with the 16450 chip. This has a faster access time which allowed the 4.77 MHz bus to be upgraded to 8 MHz but was replaced in IBM's PS/2 MCA range with the 16550 chip. This has a faster access time than the 16450, and is identical to the 16450, but with one important difference. It has two 16-byte FIFOs (First In First Out) buffers. The FIFO buffers gives the chip vastly increased performance (i.e. less chances for data overrun errors and so higher baud rates can be used). It does this by reducing the overheads of software, which does not have to deal with each and every character as it is received, because the characters can be stored in the FIFO buffer until it is full, or nearly full depending on a trigger level the software is written to cope with.

You may want to try our Velocity range of cards with 64 byte FIFO buffers which supports data rates of 1 Mega Baud using the 16750 chip!

Note: The 16550 chips or better chips, whose FIFO buffers are automatically detected and enabled by the Serial Solutions Windows communications driver, are very strongly recommended both by Microsoft documentation for Windows, and by ourselves, for highest performance.

Communications Cabling.

The data transmission path is set up using a serial cable, one end connected to the serial port of the computer by a 9-pin Female D-type connector or a 25-pin Female D-type connector and the other end

connecting to the serial port of the DCE using one of the aforementioned D-type connectors, which may be male or female dependent on the equipment used. It is sound practice to use cables with screws fitted that will allow you to fasten the cables securely to the PC card.

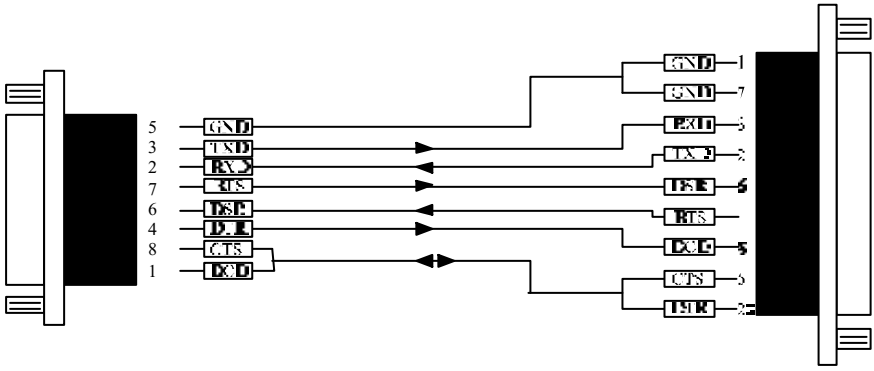
The 9-pin connector can replace the 25-pin connector in most cases depending on how many wires are to be used for communications, which is usually 9 wires or less, so then the 9- pin D-type connector is sufficient for the cable termination's. The 25-pin D-type connector may be favoured due to it being the first connector type to be available in the original IBM PC.

There are many serial communications equipment that use 25-pin connectors, and many serial boards offer 9-pin connection ports, so it may be necessary to use an adapter converter, which simply allows a 25-pin D-type connector to used to connect to a 9-pin D-type connector.

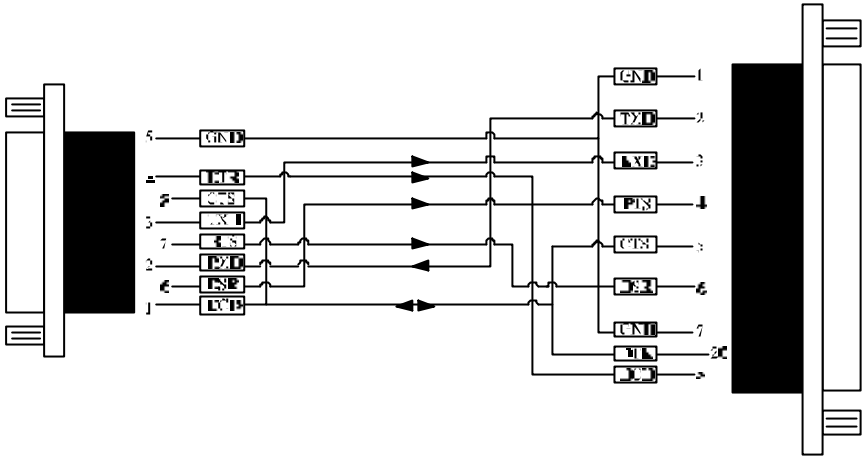
Typically RS232 maximum cable length is 50 feet and is generally a cross-over cable (pin-out and wiring diagram given in Figure 5-3) to connect a computer's serial port to the serial port of another device or terminal.

Figure 5-3. Cross-over Cable

SCHEMATIC REPRESENTATION



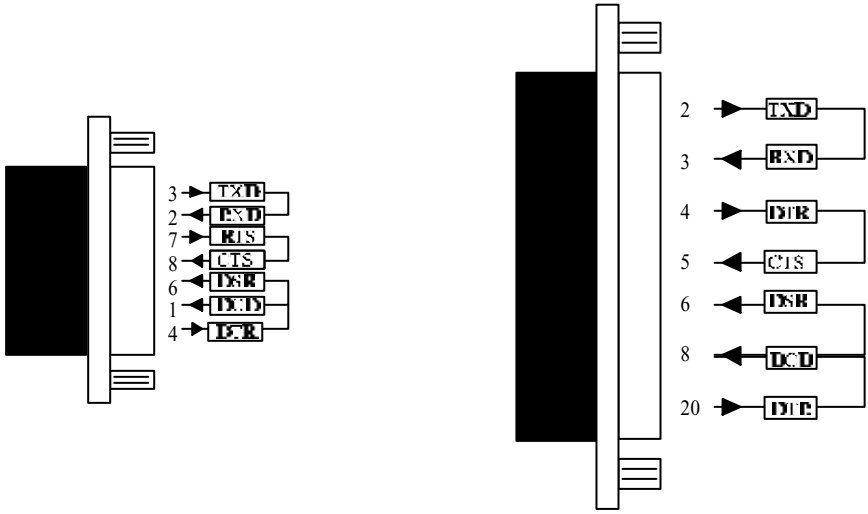
ACTUAL REPRESENTATION



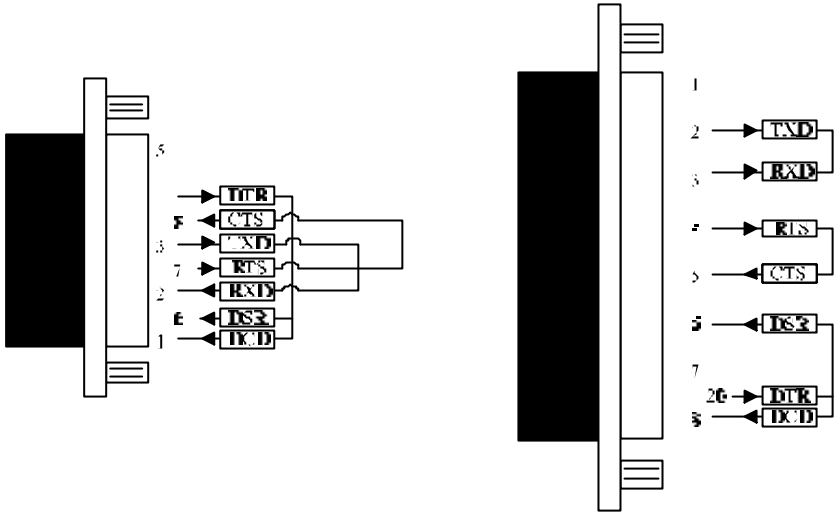
For testing the serial ports, it is not unusual to use a loop back connector (Figure 5-4), which is just a 9 or 25-pin female D-type connector with 3 groups of signals shorted together.

Figure 5-4. Loop Back Connector.

SCHEMATIC REPRESENTATION:



ACTUAL REPRESENTATION:



A loop back connector can be used to echo RS232 data transmitted by a serial port back into its own RS232 receiver. In this way the function of the serial port can be tested.

Modems do not require the cross-over cable, thus a straight-through cable, in which the transmit and receive lines are not crossed over, should be used.

Communications Settings.

Communications settings must match on both communicating machines. This is because inconsistencies can cause problems due to, for example, different speeds of transmission between the communicating serial ports, where the receiving UART is clocking in data at a slow rate, but the data is being received too fast so data gets overwritten causing the received data to be garbled.

Character Transmission.

When a character is transmitted, usually as 8 data bits (can be in packets of 5, 6, 7 or 8 data bits), it is packaged with extra bits to permit asynchronous (no system clock is used for synchronisation) communications and to check that consistent transmissions take place by using:

1) A start bit to show the start of a character transmission, which is always logic low.
2) Stop bit(s) to show the end of a character transmission, which is/are always logic high.
3) An optional parity bit to check for consistent data.

The Parity Bit.

The parity bit is a checksum bit that enables the receiving port to determine whether the received data has been corrupted. This bit can employ odd, even, mark or space parity checking and is tagged to the end of the data bits (the bits sandwiched between start and stop bits).

Odd parity is used to make the data bits in the character package odd. If the data bits on the receiving end of a transmission are even, then this will signal an error condition.

Even parity is used to make the data bits in the character package even. The data bits on the receiving end of the transmission should be even, otherwise an error condition is detected and reported by the UART.

Mark parity is used to keep the parity bit logic high so that the receiving end of the transmission should always receive this parity bit as a logic high, otherwise indicating an error condition.

Space parity is used to keep the parity bit logic low so that the receiving end of the transmission should always receive this parity bit as a logic low, otherwise indicating an error condition.

There are many causes of parity error conditions, the bits are usually flipped because of power spikes, noisy lines, or poor quality of the serial cable.

Transmission Speed.

The transmission speed of data is usually measured in bits per second (baud rate), usual values are found in the 9600 to 19,200 baud range, with 115,200 set as the maximum commonly available baud rate which is useful for the newer serial communications cards with built in FIFOs (16, 64 or more, byte buffers of the first in first out queue type).

Higher baud rates are possible (1 Mega Baud) with our Velocity range of serial cards.

Flow Control.

Flow control is a method used to prevent the loss of data. Flow control, or handshaking, allows the receiving port to tell the transmitting port that it is ready to take more data. There are two types of flow control detailed below:

Hardware Handshaking.

The serial port hardware pins are used to achieve this type of flow control. The output lines are set high or low dependent on the state of the received data buffer.

The output lines are the Data Terminal Ready (DTR), and Request To Send (RTS) lines, which can be used to provide flow control signals to the remote machine.

The input lines are Data Set Ready (DSR), Data Carrier Detect (DCD), Clear To Send (CTS), and Ring Indicator (RI), which are used to receive flow control signals from the remote machine.

Software Handshaking.

Two different control characters are used in this type of flow control, which is usually referred to as XON/XOFF handshaking. One character is used to initiate the flow of data and the other character is used to terminate the flow of data.

Hardware Settings.

Hardware settings comprise the base address for a serial port and interrupt line used for notification of serial communications events.

If interrupt sharing is to be used, then further hardware settings also need to be known, namely the SISR (Shared Interrupt Status Register) base address and the physical port location on the multiport card(s).

Hardware settings are registered with Windows using the supplied Control Panel Applet. The hardware settings are detailed below. The hardware settings must match your card settings otherwise Windows could crash.

Serial Port Base Address.

This is an area in memory which is used by the communications driver to talk with the serial port. It is an eight bits wide area of memory and it's location for bit one, usually specified in hexadecimal notation (base 16), is used to identify the serial port.

SISR Base Address.

The SISR (Shared Interrupt Status Register) base address is an 8 bits wide area in memory (usually specified in hexadecimal notation (base 16), which is read by the communications driver to find out which serial port gave rise to an interrupt condition, from which the interrupt can be investigated and cleared and then the appropriate notification, if any, can be sent to the program which is using the serial port which caused the interrupt condition.

Interrupt Line.

This is a hardware line that is used for notification of hardware events only when the events occur. This removes the aspect of having to continually poll the hardware for an event to occur, and processor time is therefore released for other activities.

There are sixteen IRQ lines, but many of these are unusable for expansion cards:

IRQ 0 - System timer
IRQ 1 - Keyboard interrupt
IRQ 8 - Real Time Clock interrupt
IRQ 13 - Maths coprocessor interrupt

and some lines are needed by most machines for system on-board devices e.g.

IRQ 2 - VGA graphics card (very few use this)
IRQ 3 - On-board Serial 2 interrupt
IRQ 4 - On-board Serial 1 interrupt
IRQ 6 - Floppy disk controller
IRQ 7 - Parallel port interrupt used in OS/2, Novell, WinNT
IRQ 12 - On-board Mouse interrupt
IRQ 14 - On-board IDE 1 controller, free if SCSI drive
IRQ 15 - On-board IDE 2 controller, used for CD-ROMs

which leaves about five interrupt lines free for use by network cards, serial cards, scanner cards, sound cards etc. This may give rise to the necessity for interrupt sharing mechanisms to be used.

Index