WinWedge®

SoftwareWedge for Windows

Connect any instrument to your PC and instantly input real-time serial data into any application.

CONTENTS

TAL Technologies, Inc.
2027 Wallace Street
Philadelphia, PA 19130   USA
Tel: (215)-763-5096 Fax: (215)-763-9711
e-mail: sales@taltech.com

What is the Software Wedge?

The Software Wedge is an extremely powerful utility that is designed to add complete 2-way serial communications capabilities to other Windows applications. The main purpose of the "Wedge" is to provide a simple way to connect any RS232, RS422 or RS485 instrument to a PC and be able to read or write data to or from the device directly from within any Windows application. Typical kinds of instruments being used with the Software Wedge include bar code and magnetic stripe readers, electronic balances, measuring tools including calipers and gages, temperature and pressure sensors, telephone caller ID boxes, telephone call logging systems, GPS receivers, pH and color meters, coordinate measuring machines, optical comparators, digital volt meters, force gages and a wide variety of other laboratory or industrial instruments that communicate over a standard serial interface.

The "Wedge" performs its magic by either converting incoming serial data to keystrokes, so that other applications receive the data as if it were being typed in, or the Wedge can pass data to and from other applications using Dynamic Data Exchange (DDE).

The Software Wedge offers an array of features including the ability to parse and filter data received through the serial port as well as add additional data including date/time stamps or cursor navigation keystrokes to incoming data. A translation table allows you to translate incoming characters to other characters or to specific keystrokes. You may even pre-define "Serial Output Strings" that can be transmitted to your serial device either automatically at regular timed intervals or by clicking your mouse on a button in the Software Wedge window. You can even set up the Wedge to issue commands to the target application after each data record has been transferred to it. Thus you could cause the application to perform actions as the data is being received; for example you could force a spreadsheet to run a macro or perform calculations on the data or even update a graph or chart in real time. All these features make it possible to create sophisticated device interfaces for virtually any serial instrument directly from within any Windows application program.

History of the Software Wedge

The Software Wedge was originally developed by TAL Technologies in 1989 as a simple interface for bar code readers. After its initial release, it quickly became apparent that a more advanced product of this type was needed for use with the hundreds of different types of devices that communicate through a serial interface. Since its humble beginnings, the Software Wedge has evolved into an extremely powerful product capable of interfacing virtually any type of serial instrument to any PC application. To date thousands of copies of the “Wedge” have been licensed to users in almost every country around the globe. There are currently five versions of the Software Wedge available for both DOS and Windows. For each platform there is a Standard Edition and a Professional Edition. The Standard Editions are less sophisticated products designed primarily for instruments like bar code readers, electronic balances and simple measuring instruments. The Professional Editions are more advanced products designed to handle more complex data streams from devices like medical and laboratory instruments, telephone systems and PLCs. The Software Wedge is also available in a native 32-bit version specifically designed for all 32 bit versions of Windows.

Who’s Using The Software Wedge

The following is just a short list of some of the many companies that rely on the Wedge for hundreds of different serial data collection applications.

3M, Abbott Labs, Allen-Bradley, Allied Signal, American Cyanamid, American Express, Ametek, Amgen, Amtrack, Amway, Anderson Consulting, ARCO Chemical, Ashland Chemical, AST Research, AT& T, Atlantic Electric, Avery, BASF Corp., Battelle, Bauer Industries, Baxter Healthcare, Becton Dickinson, Bell Atlantic, Bell Industries., Black & Decker, Boeing, Borden Chemical, Bose, Bristol-Myers-Squibb, Bridgestone Firestone, Bush Industries, Butterball Turkey, Cable & Wireless, Cambridge Aeroflo, Camcar Textron, Canon USA, Catepillar, Centers for Disease Control, CERN, Chevron Research, Ciba Corp., Ciba Corning Diagnostics, Corning Glass, Cray Research, Cryovac, Dana Corp, Danfloss Electronics, Data Electronics, DataVision, Delco Chassis, Delco Remy, DHHS, DHL International, Domino Sugar, Donnelly, Dow Chemical, Dow Deutschland, Dow Jones, Dow Pipeline, Dupont, Duracell, Eaton Corp., EDS, Elan Engineering, Eli Lilly, Epson, Esselte Meto, Eurotherm Chessell, Eveready, FDA, Federal Express, Federal Reserve, Firestone, First Brands, Fischer Technologies, Fisher Controls, Fisher Rosemount Systems, Florida Power, Florida Steel, Ford Motor Co., Friskies, Frito-Lay, Fuji-Xerox, Fujitsu Microelectronics, GSA, Gates Rubber, GE Famuc, GE Medical, General Electric, General Dynamics, General Mills, General Motors, Georgia Pacific, Gillette, Goodyear, Great Lakes Controls, Grumman, Gulf States Steel, Harley Davidson, Harp Ireland, Helene Curtis, Henry Ford Hospital, Hewlett Packard, Hoechst Celanese, Hoffer Scientific, Honeywell, Hoover, Hughes Aircraft, Hunter Industries, ITI Electronics, IBM Corp., Imperial Oil, Intel, Intellution, Intermec, J.C. Penney, Jack & Jill Ice Cream, James River Corp., Jandel Scientific, Jenco Instruments, Jensen Chevron, Johnson & Johnson, Johnson Controls, Kaiser Aluminum, Kal Can Foods, Kimberly-Clark, Koch Industries, Kodak, Konica, Kraft Foods, Lake Pharmaceutical, Land O’ Lakes, Leeds & Northrup, Lennox Industries, Levi Strauss, Liquid Container, Lockheed Martin, Lubrizol, Lucas Aerospace, M.I.T., M & M Mars, Malaysian Carbon, Marion Merrell Dow, Mars Electronics, Marsam pharmaceuticals, Martel Electronics, Maytag, Merck & Co., Metrologic, Moore Research, Motorola, NASA, National Gypsum, National Institute of Health, National Weather Service, Navel Medical Research, New York Times, Nestle, Nike, Nikon, Nippon Denso, NordicTrack, Northeast Utilities, Northern States Power, Northern Telecom, Norton Chemical, Nova Scotia Power, Nynex, Ocean Spray, Olympic Controls, Osram Sylvania, Ortho-McNeil Pharmaceuticals, Otter Tail Power Co., Owens Corning, PCC Airfoils, Pacific Gas and Electric, Packard Electric, Penzoil, Pfizer, Philips Consumer Electronics, Picker International, Pioneer Hi-Bred International, Pioneer Micro Systems, Pitney-Bowes, Plant Genetic Systems, Polaroid, Porter Instruments, PPG Industries, Prince Sports Group, Pratt & Whitney, Proctor & Gamble Pharmaceutical, Ragu Foods, Rand Mining, Ravenswood Aluminum, Raychem, Reynolds Metals, Richmomd Memorial Hospital, Robert Bosch Ind., Rosemount Analytical, Rockwell International, Rohr, RR Donnelley, RW Johnson Pharmaceutical Research, Sandoz, Sanyo Electric Co, Sartorius Corp., Seagate Tech., Scott Paper, Sears & Robuck, Siemens, Smith Corona, SmithKlein Beecham, Spaulding, Spectra Physics, SSI Services, Stanley Tools, Sony, Southern Research Institute, Starcraft Aerospace, Strathmore Paper, Summitt Controls, Sunkist, Symbol Technologies, TCBY Ent., TEC, Teledyne, Telxon Corp., Texas Beef, Texas Instruments, Texas Medical Center, The American Tobacco Co., The Coca Cola Co., The Gap, The Glidden Co., The Mennen Co., The Pister Group, The Trane Co., Timberline Instruments, Toshiba, Toyota America, Trimos Sylvac, TVA, Unisys, Upjohn, United Parcel Service, U.K. Ministry of Defense, U.S. Telecom, U.S. Army, Navy, Coast Guard, U.S. Fish & Wildlife, U.S. Geological Survey, U.S. Gypsum, U.S. Postal Service, Underwriter’s Labs, Unilever, VA Medical Center, Verifone, Verbatim, Volvo, W.L. Gore, Washington Post, Wausau, Westinghouse, Wyeth Ayerst, Xerox, Zeneca Pharmaceutical, Zymark, and many, many more....

National laboratories using the Wedge include Oak Ridge, Sandia, Los Alamos, Argonne, Fermilab and Lawrence Berkeley laboratories and hundreds of industrial and university research laboratories worldwide from Harvard and MIT to Universitat Stuttgart.

What can you do with the Software Wedge?

The Software Wedge can be used in applications ranging from simple data input (i.e. reading data from a bar code scanner or electronic balance) to complex device control interfaces (i.e. using the Wedge with Excel or an Industrial Control Program to create a real time process control application that communicates with multiple data collection devices). The best way to understand the complete capabilities of the Software Wedge is to read this manual thoroughly. The Software Wedge has many features that are not immediately apparent from the menus and dialog boxes presented in the program. Again, the only way to learn about its full capabilities is to read this manual in its entirety.

Before You Begin

All of us at TAL Technologies would like to take this opportunity to thank you for purchasing the Software Wedge. We sincerely hope that it proves to be the perfect solution for all your serial I/O applications. We also hope that you find the "Wedge" both easy and enjoyable to use. If there is a particular feature that you would like to see in a future version or perhaps a change in either the software or the documentation that would make this product better, please contact us. Your feedback is extremely important to us and will be greatly appreciated.

It is also very important that you complete the enclosed owner registration card and either mail or fax it back to us. You can also register on-line at http://www.taltech.com.
Unlike most other software companies, we do not charge for technical support, however, we cannot provide support unless you are a registered user. We would also like to take this opportunity to remind you to read the copyright notice and license agreement on page two of this manual thoroughly. You have purchased a license to use this software in a single PC and thus you may not use it on more than one PC at a time without purchasing additional licenses. Please help us to protect your investment by adhering to the terms of this agreement and by reporting copyright violations directly to: TAL Technologies, Inc., Tel: 800-722-6004 or 215-763-5096
Fax: 215-763-9711, e-mail: sales@taltech.com.

System Requirements

The Software Wedge will run on any IBM or compatible PC running MS Windows 3.x, Windows NT or Windows 95, 98 or 2000. A mouse and approximately 1 Mb free hard disk space is recommended although not entirely necessary.

If you are running Windows 3.x, it is also recommended that you have the TERMINAL font installed in your system. If you are running a 32 bit version of Windows, the Terminal font is built into the operating system and does not need to be explicitly installed. The Terminal font is used by the Software Wedge to display control codes that might appear in data received from your serial device. The Wedge will function perfectly well without the Terminal font however you may find it easier to configure the Wedge with this font installed. The Terminal font is shipped with Windows and can be found in your “\Windows\System” directory. By default it is automatically installed with Windows however it is commonly removed. You can check for it or install it using the “Font” applet in the Windows Control Panel. For instructions on using the Font applet, refer to your Windows users manual.

Why Is It Called the *Software Wedge*?

A *Wedge* (or *keyboard wedge*) is a hardware device that connects between a computer and its keyboard. Typically a wedge provides a serial port that allows data to be inputted from a device as if it were being typed in on the keyboard.

Before the development of the *Software Wedge*, keyboard wedges were the only available solution if you needed to input serial data directly into an off the shelf program like Excel, Lotus or dBase, etc. Unfortunately, keyboard wedges are severely limited in capability, are painfully slow, and they cannot support two way serial I/O. They are also expensive, prone to failure and can be difficult or impossible to install, especially on a Laptop that does not have a keyboard port.

The Software Wedge is a *software only* alternative to a keyboard wedge that allows you to connect a serial device directly to your PC's serial port with no need for any additional hardware, thus the name: *Software Wedge.*

Because the Software Wedge is implemented as a software product, it works directly with your PC’s operating system and can therefore operate much faster than a hardware wedge. It also can provide features that are either difficult or impossible to implement in a hardware product such as data parsing, filtering and translation functions, fully bi-directional serial I/O capability, date/time stamping functions and support for operating system specific features like Dynamic Data Exchange and true background operation. If you compare features, the Software Wedge is light years ahead of even the most sophisticated hardware wedge available. If you compare prices you’ll also find the Software Wedge to be far more economical as well.

Getting Started

Installing and Running the Software Wedge

The Software Wedge comes with an installation program called "SETUP.EXE" that will copy all Software Wedge program files to your hard disk and also install the Software Wedge icons into the Windows Start menu or Program Manager.

To install the Wedge, place the Software Wedge distribution diskette into drive A: (or B:) then select "Run" from the Start Menu. When the "Run" dialog box appears, type the command "A:SETUP" and click "OK". The SETUP program takes approximately two minutes to copy all files and install the Wedge into the Windows Program Manager or Start Menu.

The Software Wedge can be run by either double clicking your mouse on the Software Wedge icon located in Windows "Start" menu.

See Also: Activating The Wedge Automatically With a Specific Configuration File (pg. 14)

How To Obtain Technical Support

Technical support for Software Wedge related questions is provided free of charge, however, **only registered users are eligible for technical support**. If you have not yet sent in your owner registration card, please do so now to insure that you will have no problems when you call for support.

If you experience difficulty configuring the Software Wedge after you have read this manual thoroughly, please call TAL Technologies at **Tel: (215)-763-5096** or **Fax: (215)-763-9711** for technical support. Questions can also be sent via **e-mail** to: **support@taltech.com**.

Please have your original Software Wedge diskette and this manual available when calling for support and, if possible, be at your PC with the Wedge running when you call.
You can also find technical support information and answers to many frequently asked questions in the support section of our Internet **home page**: **http://www.taltech.com**.

The *Software Wedge Knowledge Base* located in the Support section of our web site is an excellent place to to find all sorts of assistance for configuring and using the Wedge. This includes dozens of examples for using the Wedge with other applications like Excel, Access, FoxPro, Visual Basic, Lotus 123, Wonderware, and many other common PC applications.

See Also:
A Typical Software Wedge Configuration Example (pg. 9)
Diagnosing Serial Communications Problems (pg. 85)
Troubleshooting (pg. 86)

Quick Start Guide To Using The Software Wedge

Outlined below are the most typical minimal steps that are required to use the Software Wedge successfully. Please note: This section has been provided as a general overview of the steps involved in setting up the Software Wedge and is by no means a replacement for the rest of this manual.

See Also: A Typical Software Wedge Configuration Example (pg. 9)

Step 1. Select how the Wedge will transfer incoming serial data to your other Windows programs by choosing one of the options in the *Mode* menu (i.e. Keystrokes or DDE). For a complete description of each of the two modes available, read the section of this manual entitled: **The Mode Menu** (pg. 15)

Step 2. Select the *Settings* option in the *Port* menu and then choose the serial communications settings to match the parameters for the serial device that you will be using. After you select your serial communications parameters, you should select the *Analyze* option from the *Port* menu and transmit some sample data from your serial device in order to both test that you are using the correct serial communications parameters and also to gain an understanding of how the data from your serial device is structured.
Refer to the sections: **The Port Menu** (pg. 19) and **Diagnosing Serial Communications Problems** (pg. 85) for further information.

Step 3. Select "*Input Data Record Structure...*" from the Define menu. This will present you with a series of dialog boxes that allow you to first describe the structure of your incoming serial data to the Wedge and finally describe how you want to parse and filter the incoming data as well as add additional data, keystrokes or DDE commands to the data before passing it on to the target application program. The Wedge offers a variety of options for dealing with many different kinds of data streams therefore it is highly recommended that you read the sections: **Defining the Input Data Record Structure** (pg. 24) and **Understanding The Data From Your Serial Device** (pg. 89)

Step 4. If the device that you are using requires data or commands to be sent back to it from your PC, please read the sections: **Defining Serial Output Strings** (pg. 38)

Step 5. The final step is to Activate the Software Wedge to start it working. You may want to save your configuration at this point by selecting "*Save*" from the *File* menu. To activate the Wedge simply select "*Test Mode*" from the *Activate* menu. Refer to the section: **The Activate Menu** (pg. 40) for further information.

Note: The Software Wedge will not work until you activate it by selecting one of the options in the *Activate* menu.

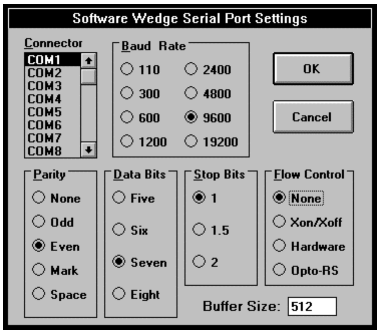
A Typical Software Wedge Configuration Example

For Bar Code Readers, Electronic Balances and Simple Measuring Instruments

This section shows the typical steps for setting up the Wedge for use with a bar code reader or a simple measuring instrument like a digital caliper or electronic balance. The steps outlined below show how to configure the Wedge to read data from your device directly into another Windows application wherever the cursor is; just as if the data were being typed in on your keyboard. This example assumes that the application you want to capture data to is a spreadsheet. It also assumes that you would like to have successive readings from your instrument collected in a column in the spreadsheet.

Begin by reading the owners manual for your device to find out the serial communications parameters that it uses. For this example we will assume that the device transmits data at 9600 baud and uses Even parity, seven data bits and one stop bit.

Step 1. Plug the device into COM1 or COM2, turn it on, and run the Software Wedge.


Software Wedge Serial Port Settings

Connector: [✓ COM1] COM2 COM3 COM4 COM5 COM6 COM7 COM8

Baud Rate: 110 300 600 1200 2400 4800 [✓ 9600] 19200

Parity: None Odd [✓ Even] Mark Space

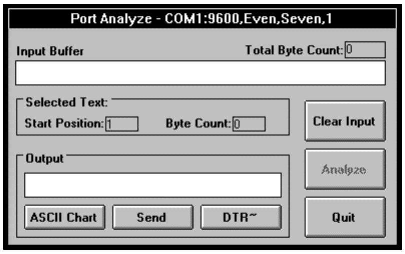
Data Bits: Five Six [✓ Seven] Eight

Stop Bits: [✓ 1] 1.5 2

Flow Control: [✓ None] Xon/Xoff Hardware Opto-RS

Buffer Size: 512

Step 2. Select the “SETTINGS” option from the “PORT” menu in the Wedge and choose the communications parameters COM1 (or COM2), 9600 baud, Even Parity, 7 Data Bits, 1 Stop Bit and no flow control from the dialog box that appears and then click the OK button to return to the main menu.


Port Analyze - COM1:9600,Even,Seven,1

Input Buffer

Total Byte Count: 0

Selected Text: Start Position: 1 Byte Count: 0

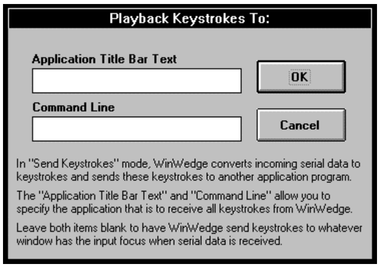
Output

Step 3. Select “ANALYZE” from the “PORT” menu and transmit a sample reading from your device into the Input Buffer in the Analyze window. This step serves two purposes, first to simply test that your communications parameters are correct and that the device is connected properly and is transmitting data, and second to gain an understanding of how the data from the device is structured. If data does not appear in the text box marked “Input Buffer”, then read the “Troubleshooting" section of this manual (pg. 86).

If data does appear in the Input Buffer, then examine all the characters that are received looking for regular features like delimiters, the existence of special characters such as carriage returns or linefeeds, etc. You may want to take several readings from the device just to make sure that each reading is structured in a similar manner.

Most simple devices such as bar code readers, calipers and electronic balances transmit a single number or short data string followed by a carriage return or carriage return linefeed. A carriage return will appear as a musical note and a linefeed will look like a small rectangle with a circle inside it. For the rest of this example, we will assume that this is how the data from your device is structured.

After you have examined the data from your device and are satisfied that it is being received correctly, click on the button marked "Quit" to return to the main menu.


Playback Keystrokes To:

Application Title Bar Text

Command Line

OK

Cancel

In "Send Keystrokes" mode, WinWedge converts incoming serial data to keystrokes and sends these keystrokes to another application program.

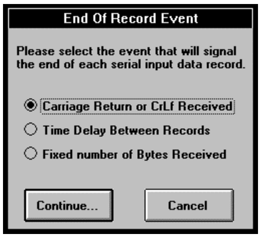
The "Application Title Bar Text" and "Command Line" allow you to specify the application that is to receive all keystrokes from WinWedge.

Leave both items blank to have WinWedge send keystrokes to whatever window has the input focus when serial data is received.

Step 4. Because we want to be able to read data directly into *any* Windows program, the easiest way to do this is to have the Wedge convert the serial data to keystrokes so that it will appear as if it were being typed in on your keyboard. Select "Send Keystrokes To..." from the MODE menu (even if it is already checked). This will present a dialog box prompting for the application title bar text and/or command line that will identify the application that you want the Wedge to send all keystrokes to.

In this example, we want the Wedge to send the data to whatever application has the input focus when the data is entered, therefore the two items "Application Title Bar Text" and "Command Line" should be completely cleared out. Leaving these items blank (as shown) causes the Wedge to send keystrokes to whatever application has the input focus when the data comes in through the serial port. After clearing out these two items, click the OK button to return to the main menu.

Step 5. The next step is to define the structure of the data that is being received from your serial device. In this example we want to send the data to a spreadsheet and have the cursor move down one cell after each reading, hence we must also instruct the Wedge to add an "Enter" keypress at the end of each reading from our device.


End Of Record Event

Please select the event that will signal the end of each serial input data record.

[✓ Carriage Return or CrLf Received]

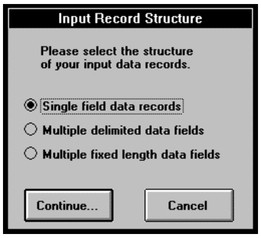
Time Delay Between Records

Fixed number of Bytes Received

Continue...

Cancel

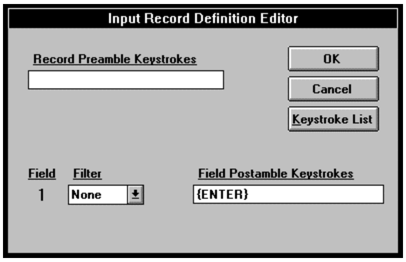
To do this, select "Input Data Record Structure..." from the DEFINE menu. The first dialog box that appears (left) will allow us to describe the event that determines the end of each individual data record transmitted by our device. In this case each data record consists of a small string of text terminated by a carriage return. Thus, we would specify "Carriage Return or CrLf Received" as the "End of Record Event". After this option is selected, click on the "Continue..." button to proceed


Input Record Structure

Please select the structure of your input data records.

(●) Single field data records
( ) Multiple delimited data fields
( ) Multiple fixed length data fields

The next dialog box to appear provides options to describe an overall "Structure" for each data record. In this example our data only consists of a single "Field" therefore we would select "Single Field Data Records" and click the "Continue..." button.


Input Record Definition Editor

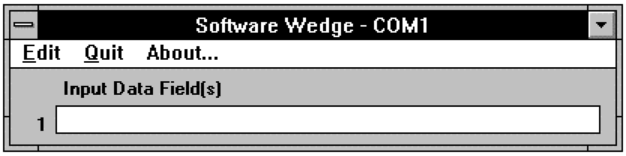
Record Preamble Keystrokes:

Field	Filter	Field Postamble Keystrokes
1	None	{ENTER}

In the next dialog box, place the cursor in the text box labeled "Field Postamble Keystrokes" and click the button marked "Keystroke List". When the keystroke list appears, scroll down the list until the word "Enter" is highlighted and click the OK button in the keystroke list box. This should cause the word "{ENTER}" to appear in the "Field Postamble Keystrokes" text box (as shown). If you are using a device that transmits numeric data and would like the Wedge to strip out any non-numeric characters (i.e. leading spaces or other text), select "Numeric" from the list of Filters. Finally, click the button marked "OK" to return to the main menu.

The Software Wedge is now completely configured to work the way we want it to. You should save your configuration at this point by selecting "Save" from the FILE menu. It is a good idea to choose names for your configuration files that are easily associated with the device that you are using. For example you might use the name BALANCE.CFG for an electronic balance.

Step 6. After configuring the Wedge you must activate it by selecting "Test Mode" from the ACTIVATE menu. When you activate the Wedge, a window containing a text box marked "Input Data Field(s)" will appear to indicate that the Wedge is ready to do its job.

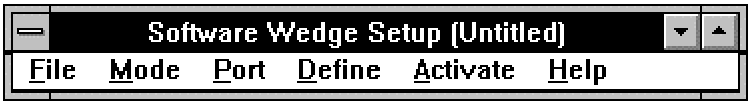

Software Wedge - COM1

Step 7. Leave the Wedge running, open your spreadsheet and place the cursor in a cell where you would like your data to appear and then transmit a reading from your serial device. If everything is working correctly the data will appear just as if it were being typed in on your keyboard with an "Enter" keypress being issued automatically after the data appears.

See Also: More Software Wedge Configuration Examples (pg. 91)

The Main Menu

When you run the Software Wedge you will be presented with the following "Main Menu":


Software Wedge Setup (Untitled)
File Mode Port Define Activate Help

The title bar of the main menu indicates that you are in *setup* mode and it also displays the file name of any currently selected configuration file. Before you can use the Software Wedge, you must configure it for your particular serial device. You configure the Wedge by selecting various options from the main menu and then after it is configured, you “Activate” it by selecting one of the options in the Activate menu.

The six main menu options and their functions are:

File - Allows opening and saving of configuration files and exiting of the Software Wedge. See Also: The File Menu (pg. 13)

Mode - Allows selection of the method used to transfer serial input data from the Software Wedge to another application & optionally specify the application that is to receive the data. See Also: The Mode Menu (pg. 15)

Port - Allows selection and testing of all serial port communications parameters. See Also: The Port Menu (pg. 19)

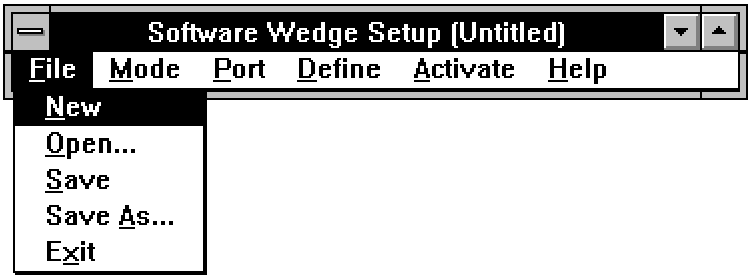
Define - The define menu contains selections that allow you to customize how the Software Wedge should work for your particular application. The different sub-menu items allow the definition of the input data record structure, parsing instructions to be applied to incoming data, additional cursor navigation keystroke or DDE commands to be issued during the input of data, and serial output data strings. See Also: The Define Menu (pg. 23)

Activate - Allows activation of the Software Wedge using the current configuration. The Software Wedge will not do anything until it is activated by selecting one of the activation options in this menu. This menu also has entries that allow you to control several additional features of the Software Wedge including how it appears when activated. See Also: The Activate Menu (pg. 40)

Help - Provides on-line help for the Software Wedge. See Also: Accessing On-Line Help (pg. 44)

The File Menu

Selecting "File" from the Software Wedge main menu presents the sub-menu shown below:



File sub-menu options and their functions are as follows:

New - Unloads any currently loaded configuration file and resets the Software Wedge to its default configuration.

Open - Opens a dialog box that allows selection of a previously saved Software Wedge configuration file.

Save - Saves changes to the currently open Software Wedge configuration file.

Save As - Opens a dialog box that allows you to save the current configuration with a filename that you specify.

Exit - Exits the Software Wedge.

Software Wedge Configuration Files

When you configure the Software Wedge for use with a particular serial device and a specific application program, you can save your configuration to a disk file that can be reloaded later. By saving your configurations to a disk file, you only have to create the configuration once. The next time you need to use the Wedge with a particular device, you can simply run the Wedge and open your previously saved configuration file and then activate the Wedge. This saves you the trouble of having to configure the Wedge each time you want to use it.

The first four options in the "FILE" menu allow you to save and load your configuration files.

Activating The Wedge Automatically With a Specific Configuration File

The Software Wedge supports one optional command line argument using the following syntax: **WinWedge.exe *filename*** where *filename* is the name of a previously saved Software Wedge configuration file.

If a filename is supplied on the command line used to launch the Wedge, the specified configuration file will be loaded and the Wedge will activate itself automatically using the configuration. The filename must be the name of a valid Software Wedge configuration file complete with the filename extension and also the drive and directory path for the file.

The filename extension ".SW1" is the default filename extension for all Software Wedge Standard Edition configuration files. The installation program for the Wedge (SETUP.EXE) automatically establishes an "Association" between the filename extension ".SW1" and the executable program "WinWedge.exe". This allows you to launch Software Wedge configuration files that have the filename extension ".SW1" directly without having to go through the process of launching the Wedge, opening the configuration file and then activating the Wedge.

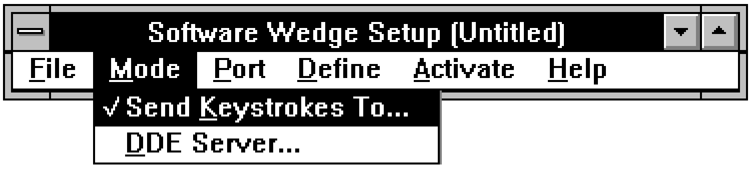
For example after you create and save a configuration file for the Wedge, you can simply double click on the name of the configuration file in the File Manager or the Windows Explorer and Windows will automatically launch the Software Wedge and activate it using the selected configuration.

Activating The Wedge Automatically When You Start Windows

To load and activate the Software Wedge automatically with a particular configuration file when you start Windows, open the Windows File Manager (or Windows Explorer in Win95), select your desired Software Wedge configuration file and drag the file to the Program Manager and drop it into your “Startup” program group. This should cause the Software Wedge program icon to appear in your Startup group. From this point on, whenever you start Windows, the Wedge will automatically load and activate itself with the selected configuration file.

In Windows 95 you can perform the same function by opening the Explorer, selecting a previously saved Software Wedge configuration file and dragging and dropping the configuration file into the folder named "StartUp". The "StartUp" folder is usually located in the "Programs" folder inside the "Start Menu" folder in either your "Win95" or "Windows" folder.

The Mode Menu



The Software Wedge can operate in one of two modes that specify how to transfer incoming data from your serial device to another application program. The Wedge can either transfer data by converting it to keystrokes and then sending the keystrokes directly to another application or it can pass data using Dynamic Data Exchange (DDE) to any application that supports DDE. To see if an application supports DDE, look for a "Paste Link" or "Paste Special" command in its main menu, usually under an "Edit" menu.

The two choices in the MODE menu, "Send Keystrokes To..." and "DDE Server", allow for the transfer of incoming serial data directly into another Windows application program. Each of these two modes has its own unique advantages and the choice of which mode to use depends partly on the capabilities of the application program that you will be sending data to. If your application does not support Dynamic Data Exchange then you should select "Send Keystrokes". See Also: Sending Keystrokes vs. Dynamic Data Exchange (pg. 18).

Send Keystrokes Mode

In "Send Keystrokes Mode", the Software Wedge will convert incoming serial data to keystrokes and then send these keystrokes directly to any application that you specify. "Send Keystrokes Mode" is the easiest way to use the Wedge with most simple devices like bar code readers, electronic balances and measuring tools like gages and meters.

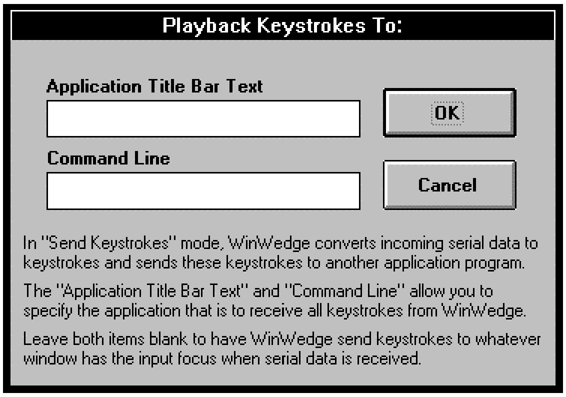
When you select "Send Keystrokes To..." from the Mode menu, a dialog box will appear that lets you specify the application program that is to receive the keystroke data. The Software Wedge is pre-configured to send keystrokes to the NotePad accessory that comes with Windows however you may change this to any application that you like. You may enter either the application's title bar text (the text that appears in the title bar of the program's main menu) or you may also specify the command line used to launch the program in Windows (the application's full path and executable filename along with any command line arguments). If you specify only the title bar text, the application must be running either as an icon or in a window before the Wedge can send it any data. If you specify only the application's command line, then the Wedge will automatically launch the application before sending it any data.

After you activate the Software Wedge in “Send Keystrokes” mode, whenever data is received from your serial device, the Wedge will try to send the data to the application that you specify. The logic used by the Wedge is to first search for an application with the specified title bar text and if the application is already running, the Wedge will activate it so that it has the input focus before sending keystrokes to it. If a window with the specified title bar text is not found or if you did not specify the title bar text, then the Wedge will try to run the program using the command line specified. If this is successful, then the Software Wedge will retrieve the title bar text from the application after it starts running so that it can find the application’s window later on.

When specifying the title bar text, you do not need to enter the complete title and may use a substring of the text. For example, to send data to Microsoft Excel, you can simply specify "Excel" for the title bar text. As long as there are no other programs running that contain the substring "Excel" in their title bar, the Wedge should be able to find Excel’s window correctly. You may specify both the title bar text and command line in upper or lower case.

Note: If you do not specify a command line or a window's title bar text, the Wedge will send keystrokes to whatever application has the input focus when data is inputted from your serial device. In other words, the Wedge will work like a second keyboard. You may prefer to use the Wedge in this manner because it allows you to input data wherever the cursor happens to be in whatever application has the focus. If the Wedge itself has the focus when data is inputted, it will simply display the data without trying to send it to any other program.

Tech Tip:
Some Windows applications have more than one window with the same Title Bar Text, therefore the Wedge may not always be able to find the right window to send keystrokes to. If you experience difficulties in getting the Wedge to send keystrokes to a particular application even if you are sure that you specified the correct Application Title Bar Text or Command Line, try clearing out both of these items. Then after you activate the Wedge simply launch and/or switch the input focus to the application that you want the Wedge to send data to and place the cursor in the location where you want data entered.



In "Send Keystrokes" mode, WinWedge converts incoming serial data to keystrokes and sends these keystrokes to another application program.

The "Application Title Bar Text" and "Command Line" allow you to specify the application that is to receive all keystrokes from WinWedge.

Leave both items blank to have WinWedge send keystrokes to whatever window has the input focus when serial data is received.

DDE Server Mode

When the Wedge is in DDE Server mode, instead of sending incoming serial data to another application as keystrokes, the Wedge allows other Windows applications that support DDE to retrieve incoming serial data through DDE links.

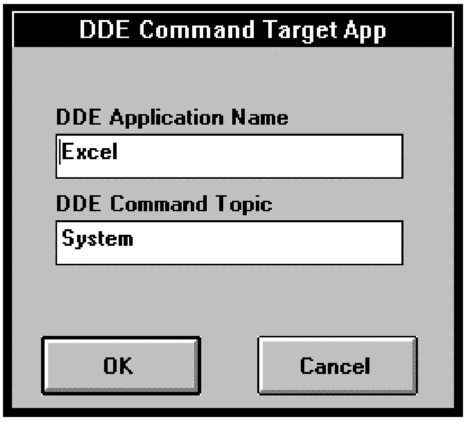
When you select "DDE Server" from the Mode menu, a dialog box will appear prompting for a DDE Application Name and a DDE Topic. In addition to passing serial data to another application, the Wedge can also be configured to issue DDE commands to another program after each input from a device is received. DDE commands can be used to force the program that is receiving data from the Wedge to do something like run a macro or subroutine or update a chart or graph. The information that you supply in this dialog box is used to identify the DDE Command Processor for the application where commands should be sent.

Note: This information is needed only if you intend to configure the Wedge to send DDE commands to the application, otherwise it can be cleared out.

See Also:
Specifying DDE Commands (DDE Server Mode) (pg. 32)
Understanding Dynamic Data Exchange (pg. 47)

The default DDE Application Name is "Excel" and the default DDE Topic is "System". These are the correct parameters for Microsoft Excel. The DDE Application Name for Microsoft Word is "WinWord" and its DDE Topic is also "System". For MS Access, the Application Name is “MSAccess” and the DDE Topic is the name of your open Access database (without the MDB file name extension). To find the Application Name and Topic for any other application's DDE command processor, refer to its users manual or on-line help.

Note: It is common practice for applications that support DDE commands to use the application's executable filename (without the EXE extension) as its DDE Application Name, and it is also typical to use the DDE Topic "System".


DDE Command Target App

DDE Application Name: Excel

DDE Command Topic: System

Sending Keystrokes vs. Dynamic Data Exchange

The primary advantage of setting up the Wedge to send keystrokes to another application rather than passing data using DDE is simplicity. By sending keystrokes, no special programming is required in the application that will receive the data. For example, when reading data into a spreadsheet, a common requirement is to have successive readings from a device appear in a column in the spreadsheet. By sending keystrokes you could simply configure the Wedge to send an "Enter" or a "Down Arrow" keystroke after each input from your device to move the cursor down one cell and thus be ready for the next input.

See Also: Specifying Pre/Postamble Keystrokes (Send Keystrokes Mode) (pg. 31)

A minor disadvantage of sending keystrokes is that the application that is receiving the data must be in the foreground and have the input focus before the Wedge can send any keystrokes to it, thus true background processing is not possible when sending keystrokes.
Sending keystrokes from one application to another is also slightly slower than using DDE.

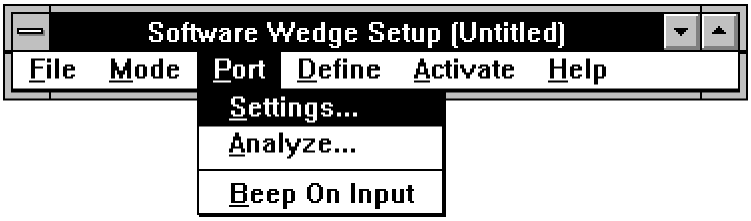
DDE on the other hand does not require the receiving application to have the input focus therefore all data transfers can occur in the background while you work with other programs in the foreground. One difficulty with DDE is that "linked" data from the Wedge (or any other DDE Server) is always transferred to the same location in the client application (i.e. a "linked" cell in a spreadsheet). To get successive data readings into a column in a spreadsheet you must write a macro in the spreadsheet that runs automatically after each input is received. The purpose of the macro would be to either explicitly request the data from the Wedge or copy each new data item from a *linked* cell and paste it to the bottom of the column in your spreadsheet. One way to accomplish this is to have the Software Wedge issue a DDE Command after each input that causes the spreadsheet to run a macro that either requests the data or copies any linked DDE data into a column.

A powerful feature of DDE is that applications that support it can send commands directly to each other and thus you could fully automate a data collection process between your application program and the Software Wedge. This requires a small amount of programming in your application however the extra effort allows you to create extremely sophisticated device interfaces. Also, because all DDE operations can occur in the background, even with a minimized application, DDE operations typically execute much faster than sending keystrokes from one application to another.

Note: **The Software Wedge does <u>not</u> have to be in "DDE Server Mode" in order to accept and process any of its own DDE commands.** The Wedge will still execute DDE commands sent to it even if it is in "Send Keystrokes Mode". This means that you can control the Wedge or transmit data out the serial port by sending DDE commands to the Wedge from another program while the Wedge passes incoming serial data back to your application as keystrokes.

See Also: Understanding Dynamic Data Exchange (pg. 47)
DDE Examples (pg. 54)
The Software Wedge DDE Command Set (pg. 50)

The Port Menu



The three PORT menu options and their functions are:

Settings: Displays a dialog box that allows you to select all serial communications parameters, (i.e. serial port, baud rate, parity, number of data bits, flow control, buffer sizes, etc.)
See Also: The Port Settings Dialog Box (pg. 20)

Analyze: This option displays a dialog box that allows you to test all serial communications parameters and also view or analyze the structure and contents of data received from your serial device.
The Analyze feature is extremely useful for testing the communications link between your PC and your serial device as well as for determining the structure and contents of all data received from the device. It is highly recommended that you use this feature to test the communications link between the Software Wedge and your instrument before activating the Wedge from the Activate Menu.
See Also: The Port Analyze Menu Option (pg. 21)

Beep On Input: This option is a switch that instructs the Software Wedge to beep your PC's speaker when data has been received and is about to be transferred to another application program. Selecting this option toggles this function on or off where "on" is indicated by a check mark to the left of this menu selection.

Note: The Software Wedge will also beep whenever a communications error occurs. This includes parity errors, framing errors and buffer overruns. If you do not have the "Beep On Input" option checked and your PC beeps when you receive data through the Wedge, the most likely cause is a mismatch in the communications parameters between the Wedge and your serial device. Because the Wedge treats all characters as valid data, even when a communications error occurs, it is possible to have a mismatch in communications parameters and still receive data that looks correct (although the continual beeping will indicate that the Wedge is configured incorrectly).

See Also: Diagnosing Serial Communications Problems: (pg. 85)

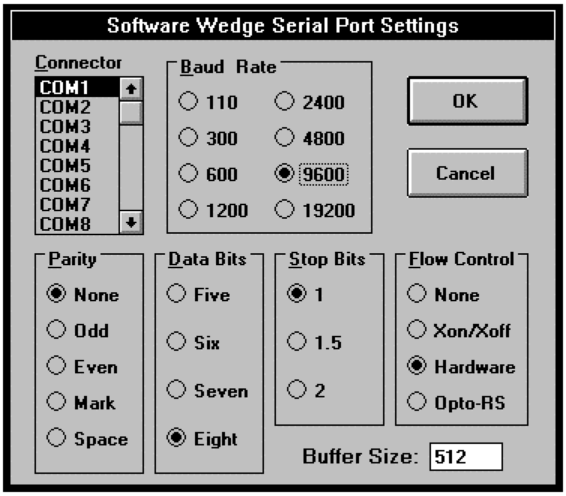
The Port Settings Dialog Box

The Port Settings dialog box allows you to select the serial communications parameters required by the device that you are using. All parameters must exactly match the settings for the serial device that you will be using. If you do not know the correct settings for your serial device you should contact its manufacturer for this information. For general instructions on how to determine the correct parameters using the Port Analyze feature of the Software Wedge, please refer to the section: Troubleshooting (pg. 86)

Note: The values for Baud Rate, Parity, Data Bits and Stop Bits chosen in the Port Settings Dialog box always override any values for these parameters set in the "Ports" section of the Windows Control Panel. Thus you do not need to set these parameters in the Control Panel.

The Buffer Size option in the Port Settings dialog box allows you to specify the size of the receive buffer. This value may range from 128 to 32000 bytes. The default buffer size of 512 bytes should be sufficient for most simple devices. If you need to input large amounts of data with no flow control, you may have to increase the buffer size to avoid losing data.

The Software Wedge supports both standard Software (XOn/XOff) and Hardware (RTS/CTS) flow control as well as a third "Opto-RS" option. The Opto-RS option has been provided to make the Wedge compatible with the Trimos-Sylvac line of measuring instruments. These devices use a special optical cable interface that requires the RTS line to be held low.
See Also: Flow Control Protocols: (pg. 106)


Software Wedge Serial Port Settings

Connector: [✓ COM1] COM2 COM3 COM4 COM5 COM6 COM7 COM8

Baud Rate: 110, 2400, 300, 4800, 600, [✓ 9600], 1200, 19200

Parity: [✓ None] Odd Even Mark Space

Data Bits: Five Six Seven [✓ Eight]

Stop Bits: [✓ 1] 1.5 2

Flow Control: None Xon/Xoff [✓ Hardware] Opto-RS

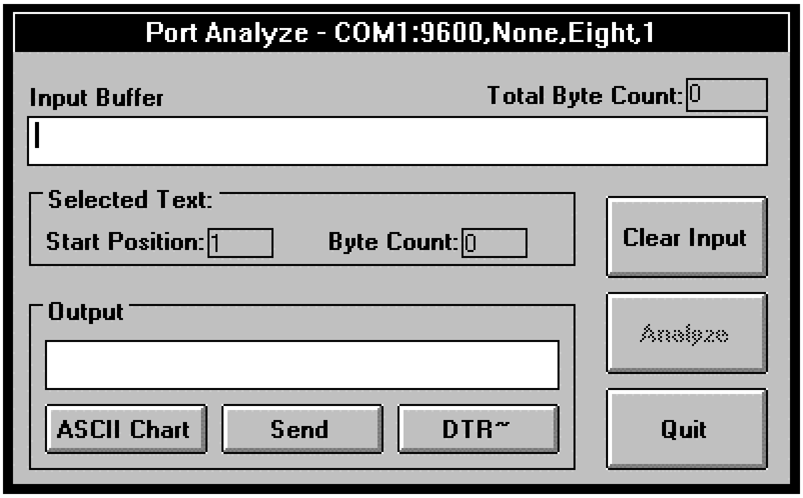
Buffer Size: 512

Note: Although the Software Wedge supports up to COM9, Windows 3.x only provides support for COM1 thru COM4. To access COM5 and above, you will need special COM drivers provided by the manufacturer of the serial adapter hardware that you intend to use.

The Port Analyze Menu Option

This option presents a dialog box that allows you to test the serial port settings that you chose in the Port Settings dialog box. The Port Analyze dialog box also lets you view or analyze the data received from your serial device. This feature can be extremely useful when you are configuring the Software Wedge for your particular application.

If you select this option and the serial port chosen in the Port Settings dialog box exists and is available for use, the following dialog box will appear:



The title bar of the Port Analyze dialog box will show the currently active communications parameters (i.e. serial port, baud rate, parity, databits and stopbits). The Input Buffer text box displays all characters received from your serial device. Characters are shown using their displayable ASCII characters thus embedded control codes also appear (i.e. a carriage return should appear as a musical note, an ASCII 1 will be a happy face, etc.). An ASCII chart is available to help decipher the meanings of characters in the input buffer and also to select characters to be placed in the "Output" text box.

Directly above and to the right of the Input Buffer text box is a count of all characters in the input buffer, labeled "Total Byte Count".

The "Clear Input" button is used to clear the input buffer and also reset the total byte count to zero. Note: The Input Buffer text box in the Standard Edition of the Software Wedge can display a maximum of 255 characters at a time.

An automatic analyze feature allows you to have the Software Wedge make an educated guess at the structure of your input data records. The Wedge can be configured to parse and filter your serial data, however, you will first need to know what the *record structure* is for your data. If you press the "Analyze" button while there is data in the input buffer, the Software Wedge will offer a guess at the record structure and ask you if you would like to automatically pre-configure the wedge using the guessed at record structure.

Note: The auto analyze feature is not very intelligent and thus you may not want to use it to pre-configure the Software Wedge. You are entirely free to define your input record structure in any way that you like.

See Also:
Defining the Input Data Record Structure (pg. 24)
Understanding The Data From Your Serial Device (pg. 89)

When viewing data in the input buffer, you can select portions of the data using your mouse or the cursor control keys. When you do this, the starting byte position of any selected characters in the input buffer is displayed next to the label "Start Position" and the total byte count for the selected data is displayed next to the label "Byte Count" (located inside the frame labeled "Selected Text"). This feature allows you to measure the lengths of specific data fields in your input data records.

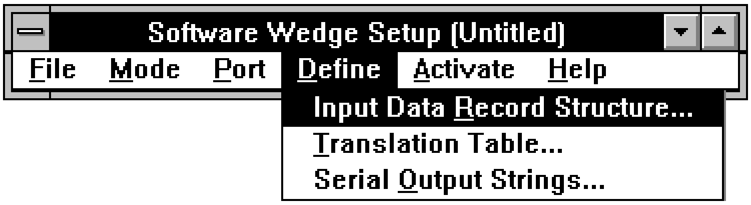
To copy data from the Input buffer to the Windows clipboard, select the data that you want to copy, then hold the control key (Ctrl) down and press the Insert key.

Inside the frame with the caption "Output" is a text box and three buttons labeled "ASCII Chart", "Send" and "DTR~". The text box is an output buffer where you can enter text to be sent to your serial device. Many devices allow you to send them command strings or control codes to do things like request a data reading. You may type and edit data directly in the Output text box. You may also select specific characters or control codes to be placed in the output text box from an ASCII chart that is available by pressing the "ASCII Chart" button. To send the data in the output buffer text box, click your mouse on the "Send" button. To clear the output buffer, select all text in the Output text box with your mouse or cursor control keys and press the Delete key.

The button marked "DTR~" is called the *Data Terminal Ready Line* Toggle button. When this button is pressed, it turns off the serial port DTR line for approx. 100 ms. Some types of serial devices interpret the toggling of the DTR line as a request for data. Sylvac brand calipers and gages use the toggling of the DTR line as a way to prompt the device for a data reading.

Note: If the Wedge beeps while you are inputting data into the Analyze window, then the serial communications parameters that you are using are incorrect. It is also possible for data to appear correctly in the Analyze window even though the communications parameters are wrong. See Also: Diagnosing Serial Communications Problems (pg. 85)

The Define Menu



The three options in the DEFINE menu and their functions are:

Input Data Record Structure

This option presents a series of dialog boxes that allow you to define the basic structure of data that you will be receiving from your serial device. The dialog boxes presented also allow you to specify how the Wedge should parse and filter incoming data before transferring it to another application. When you define the input data record structure, you can also add additional data or keystrokes to each input field or specify DDE commands to be issued following the input of each field.

See Also: Defining The Input Data Record Structure (pg. 24)

Translation Table

Selecting the Translation Table option displays a translation table for translating characters to keystrokes when the Software Wedge is in *Send Keystrokes* mode or for translating characters to other characters when the Software Wedge is in *DDE Server* mode. Translations are performed after the Wedge has parsed and filtered an incoming data record, just before the data is transferred to another application.

See Also: Translation Table (pg. 35)

Serial Output Strings

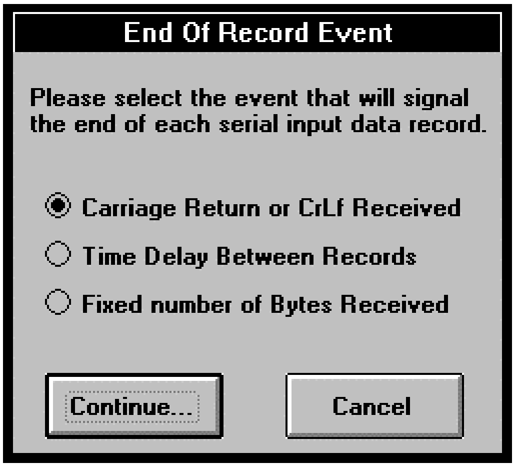
The Serial Output String option opens a window where you can pre-define character strings to be transmitted back to your serial device by the Software Wedge. Several output strings may be defined including an *acknowledgment* string as well as up to 20 "button controlled" output strings and one "timer controlled" output string.

See Also: Defining Serial Output Strings (pg. 38)

Defining the Input Data Record Structure

Most instruments transmit data in some pre-defined *record structure* with each record containing one or more parts or *fields* of data. Because the Software Wedge has no way to know how the data from your particular device is structured or how to determine what parts of the data are important to your application and which parts should be ignored, you must first define these parameters before activating the Wedge. The *Input Data Record Structure* option in the “DEFINE” menu allows you to first define the record structure of your data and then define how the Wedge should parse and filter the individual data fields within each record before sending the data on to another application. This option also provides a mechanism for adding date and time stamps, cursor navigation keystrokes or DDE commands to the data before or after each data field is sent from the Wedge to your target application program.

When defining the input data record structure, several basic descriptors must be specified starting with the event that determines the end of each data record. When you select *Input Data Record Structure* from the Define menu, the following dialog box appears:


End Of Record Event

Please select the event that will signal the end of each serial input data record.

[✓ Carriage Return or CrLf Received]
Time Delay Between Records
Fixed number of Bytes Received

The End Of Record Event

The *End Of Record Event* options allow you to specify the event that will signal the end of each input data record. The Software Wedge will not transfer any data to a target application until the selected *End Of Record Event* occurs. Therefore, you should select the *End of Record Event* that will always reliably determine when each serial data record ends.

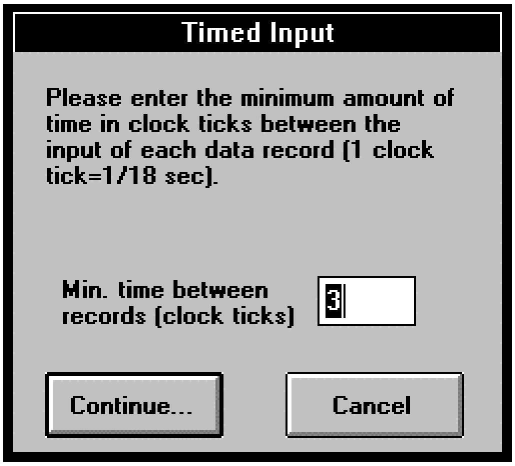
If each data record is terminated by a carriage return or carriage return linefeed, the option *Carriage Return or CrLf Received* should be chosen. Note: Carriage return and line feed characters are automatically removed from the input data if this option is chosen.

If your incoming serial data is not always transmitted as fixed length strings and there also is no carriage return at the end of each data record but the data is transmitted in bursts such that there will always be a time delay between data inputs, then *Time Delay Between Records* should be chosen. Many types of bar code scanners transmit data in this manner.

If you select this option, you will be prompted to enter the minimum amount of time (in 1/18 second increments) between the arrival of each new data record. You should specify the smallest time delay value possible that will accurately determine the end of an input data record. Thus, choosing the "Time Delay Between Records" option tells the Wedge to read in all serial data until no more data comes in the serial port. After the specified time delay occurs, the Wedge determines that is has received a complete data record.

Note: If you specify too large a value, then you risk the possibility of two or more consecutive inputs being interpreted as one long input.

The internal time clock in Windows is not highly accurate therefore you may need to experiment with the value that you specify for the time delay between inputs. For most simple devices like bar code or magnetic stripe readers the default value of three clock ticks will almost always be perfectly adequate.

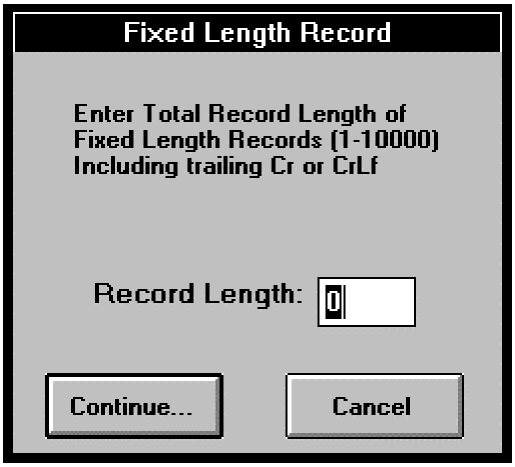

Timed Input

Please enter the minimum amount of time in clock ticks between the input of each data record (1 clock tick=1/18 sec).

Min. time between records (clock ticks) 3

If the data records transmitted by your device always consist of a fixed number of bytes with one or more fixed length data fields, then the option *Fixed Number of Bytes Received* should be chosen. If you choose this option, you will be prompted with the following dialog box to enter the complete record length for your data records.

See Also: The Port Analyze Menu Option (pg. 20)

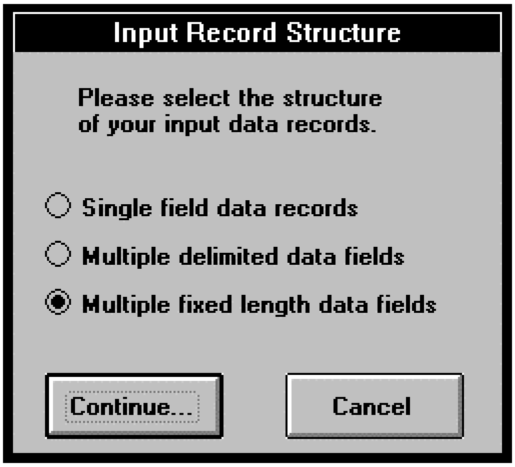


Fixed Length Record

Enter Total Record Length of
Fixed Length Records (1-10000)
Including trailing Cr or CrLf

Record Length: 0

If you choose either *Carriage Return or CrLf Received* or *Time Delay Between Records* as the End of Record Event, you will also be prompted to select a "record structure" for your data from the following dialog box:


Input Record Structure

Please select the structure of your input data records.

○ Single field data records
○ Multiple delimited data fields
● Multiple fixed length data fields

The three possible record structures and their meanings are listed below:

Single Field Data Records

Single Field Data Records means that each complete data record should be treated as a single entity thus no parsing should be performed on the data.
Note: The maximum length for a single data field is 255 bytes.

NOTE: Some types of serial devices produce an output that may be too complex or too inconsistent to be parsed into a set number of "fields" by the Software Wedge. For these types of devices, it may be easier to define the entire output from the device as a single data field and then use the capabilities of the target application (the macro language in Excel for example) to parse the data received from the Wedge.

See Also:
Understanding the Data From Your Serial Device (pg. 89)
Sending Keystrokes Vs. Dynamic Data Exchange (pg. 18)

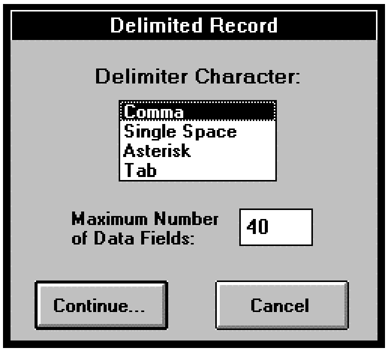
Multiple Delimited Data Fields

The *Multiple Delimited Data Fields* record structure specifies that your data records consist of two or more *fields* that should be parsed based on the position(s) of delimiter characters within each record. For example the following data record has four fields delimited by commas and a carriage return at the end of the record: **1, 90, 123.19, 98765<Cr>**

If you select this option, you will be prompted with the dialog box shown below to specify the delimiter character used to separate data fields and the maximum number of data fields that are contained in a single data record.

The Software Wedge supports data records with up to 40 data fields. If more than the specified number of fields are received for a particular record, the additional data is discarded. If less than the specified number of fields are received, the remaining fields will be left empty.

See Also: Defining The Input Data Record Structure (pg. 24)
Getting Past The 40 Field Limit In Software Wedge (pg. 65)


Delimited Record

Delimiter Character:

Comma
Single Space
Asterisk
Tab

Maximum Number of Data Fields: 40

Note: All delimiter characters will be removed from the input data and thus will not appear as part of each data field if the *Multiple Delimited Data Fields* option is chosen.

Multiple Fixed Length Data Fields

Multiple Fixed Length Data Fields should be chosen as the record structure if you need to parse your data records based on the length of each data field. If you use this option, you will later be able to specify the exact length (number of characters) contained in each data field. This option should be used only in cases where you can always rely on each data field to contain a fixed number of bytes. This option is the default if you select "Fixed Number of Bytes Received" as the End of Record Event.

Specifying Filters and Field Lengths

After you specify the End of Record Event and choose the basic Record Structure for the data transmitted by your serial device, an "Input Record Definition Editor" dialog box will appear. This dialog box is where you specify a filter to be applied to each data field and the input length of each field for fixed length fields.

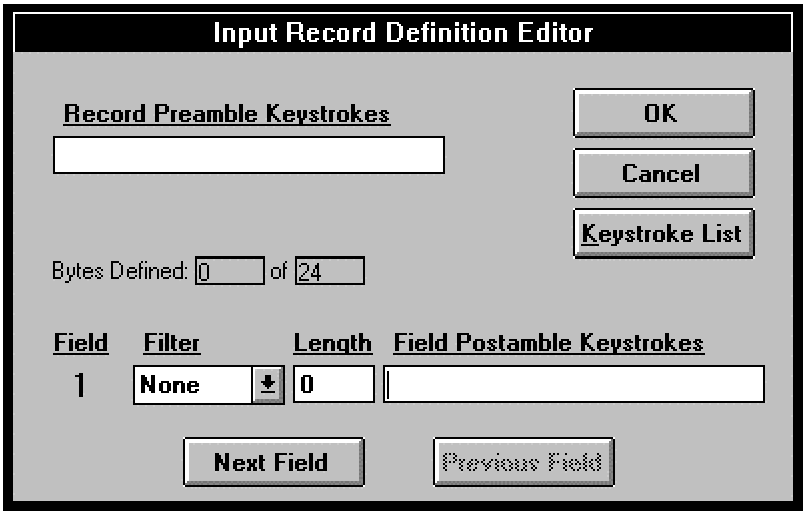
This dialog box also allows you to specify a record *Preamble* and field *Postambles* that perform different functions depending on the mode that you chose for transferring data.

In *Send Keystrokes* mode, the record preamble and field postambles are additional keystrokes that are issued before or after each data field is sent to another program.

In *DDE Server* mode, the preamble and postambles are DDE commands that are issued to the application that you specified as the *DDE Command Target Application* when you chose "DDE Server" from the Mode menu.

See Also:
Specifying Pre / Postamble Keystrokes (Send Keystrokes Mode) (pg. 31)
DDE Server Mode (pg. 17)
Specifying DDE Commands (DDE Server Mode) (pg. 32)

Note: Some of the controls in the Input Record Definition Editor may not be displayed or may have different captions than the example below depending on the chosen record structure and also depending on the selected data transfer mode.



Bytes Defined: 0 of 24

Field 1, Filter: None, Length: 0, Field Postamble Keystrokes:

The lower half of the dialog box is where you specify an "Input Filter" to be applied to each data field in your input records as well as a "Field Postamble". In cases where each data field consists of a fixed number of bytes, you must also specify a field "Length".

Under the label "Field" is a status box indicating the number of the current data field for which you are defining a Filter, a Length and a Field Postamble. If your data records consist of more than one data field, there will be two buttons marked "Next Field" and "Previous Field" in the dialog box. These buttons are used to scroll forward and backward through the parameters for each data field, (the first field is Field(1); pressing the "Next Field" button will display parameters for Field(2), etc.).

Selecting Filters

There are three choices for the "Filter" that can be applied to each data field; *None*, *Ignore* and *Numeric*. Specifying *None* means that no filter is to be applied to the data for the current field, thus all characters are to be accepted as received.

Specifying *Numeric* causes all Alpha characters and control codes to be filtered out and only allows characters: **0123456789-.** to be accepted as valid data. The *Numeric* filter is especially useful when reading numbers into a spreadsheet. Leading spaces or non-numeric characters can cause data to be interpreted as a label instead of a number.

If you select the *Ignore* filter, then no data is accepted and the entire field is discarded. The *Ignore* filter is used to ignore or remove data fields that are not required by your application.

Specifying Field Lengths

The text box labeled "Length" is where you specify the exact number of data bytes that will be received for a particular field. The "Length" text box will only appear if you selected either "Fixed Number of Bytes Received" as the "End Of Record Event" or "Multiple Fixed Length Data Fields" as the record structure. The field length thus specifies the exact number of bytes in each individual data field in the input record. Field lengths may range from 1 to 255 bytes. If you chose "Fixed Number of Bytes Received" as the End of Record Event, the sum total of all field lengths entered should equal the total "Input Data Record Length" that you specified earlier.

In the middle toward the left side of the Input Record Definition Editor (labeled "Bytes Defined") are two status boxes that display the total of all specified field lengths as well as the total record length of your data records (specified earlier for fixed length records). This information will be visible only when the end of record event is "Fixed Number of Bytes Received".

Specifying Pre / Postamble Keystrokes (Send Keystrokes Mode)

When the Software Wedge is in Send Keystrokes Mode, the Input Record Definition Editor will have a text box marked "Record Preamble Keystrokes" in the upper half of the dialog window. You can also specify "Field Postamble Keystrokes" that are associated with each input data field. Preamble and Postamble Keystrokes are additional keystrokes that the Wedge will issue before or after each data field is sent to another Windows application. They are typically used to send cursor navigation keystrokes to the target application so that the serial data ends up where it is supposed to go.

The "Record Preamble Keystrokes" are keystrokes that are issued immediately before the first data field from your serial device is transferred to the target application program. For example, the preamble keystrokes could be used to move the cursor to a specific location in the target application just before the first data field is transferred to it.

The "Field Postamble Keystrokes" are keystrokes that are issued immediately following the data for each individual data field. Field postamble keystrokes are usually used to navigate around in the application that is receiving the keystroke data. For example, if you want to read data from an instrument into a spreadsheet and you would like successive readings to be entered into a single column, you could define a Field Postamble Keystroke that consists of a {DOWN} arrow keypress. This would cause the cursor to move to the next cell down in the spreadsheet after each reading from the device. You define the Record Preamble and Field Postamble keystrokes to mimic the actual keys that you would type if you were entering the data by manually typing it in on the keyboard.

When specifying Record Preamble and Field Postamble Keystrokes, you must follow the rules outlined in the section: Keystroke Macro Rules (pg. 45)

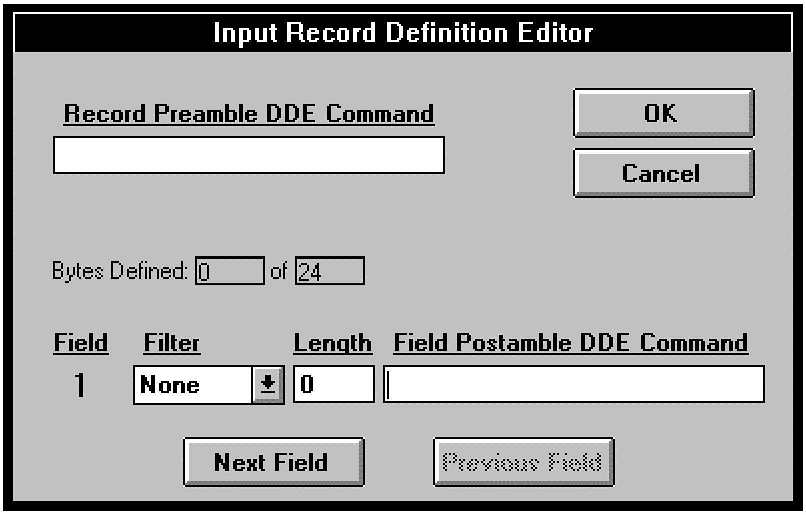
The button marked *Keystroke List* allows you to select individual keystrokes from a menu while editing preamble and postamble keystrokes. All keystrokes in the keystroke list conform to the proper Software Wedge Keystroke Macro Rules. To enter keystrokes using the keystroke list, click your mouse in either the Record Preamble or Field Postamble Keystrokes text box and then click the "Keystroke List" button. When the keystroke list appears, scroll the list down until the desired keystroke is highlighted and then click the OK button in the keystroke list dialog box. You may enter as many keystrokes as you like when defining any Preamble or Postamble keystrokes.

Preamble and Postamble Keystroke macros may also contain special date and/or time stamp functions.

See Also: Date and Time Stamps In Keystroke Macros (pg. 46)

Specifying DDE Commands (DDE Server Mode)

When using the Software Wedge in DDE server mode, the Input Record Definition Editor will appear as below.


Input Record Definition Editor
Record Preamble DDE Command
OK
Cancel
Bytes Defined: 0 of 24
Field
Filter
Length
Field Postamble DDE Command
1
None
0
Next Field
Previous Field

All of the controls in the above dialog box are the same as when the Wedge is in "Send Keystrokes Mode" except that instead of supporting *Record Preamble Keystrokes* and *Field Postamble Keystrokes*, the dialog box allows you to enter a *Record Preamble DDE Command* and *Field Postamble DDE Commands*. Most applications that support DDE also support a *DDE command language* that consists of commands that may be issued by other application programs. These commands will be unique to a particular application and should be documented in the application's users manual or on-line help.

Excel for example recognizes a wide selection of DDE commands that allow other Windows applications to take complete control of it. In fact, Excel even recognizes every one of the function names listed in the Excel Function Reference as valid DDE commands.

For example, Excel supports a function called "**RUN(**reference**)**" that can be used to run any macro or VBA subroutine defined in an Excel macro sheet where "reference" is the name of the macro or subroutine in quotes. If you create an Excel VBA subroutine named: MySub and then enter the command: **[RUN("MySub")]** as a field postamble DDE command, Excel will execute the subroutine "MySub" immediately after data for the field is transferred via DDE to Excel from the Wedge.

When used with a program like Microsoft Excel, this concept is extremely powerful because it allows you to use the entire Excel macro language including Visual Basic to perform nearly any operation on data received from the Wedge. This includes chart or graphing functions, calculations based on received data, or logical decision making - all in real time as data is being received through the serial port.

Any "Record Preamble DDE Command" specified will be issued immediately before the first data field is transferred to your application and each "Field Postamble DDE Command" is issued immediately after the data for the particular field is transferred via DDE to the target application.

The main concept here is that you are configuring the Wedge to issue a command to another application program when each new data record is received from the instrument connected to your serial port. In almost all cases you will only send a single DDE command to the other application after the very last data field that you have defined. The intention here being to cause the other application to first pull the data from the Wedge and then do something with it. The sending of the DDE command is simply a way to let the other application know that there is new data available and that it had better do something with the data before it is overwritten by the next data record.

Many applications that support DDE also provide a method for programming the application to *sense* when new data is received from a server in a DDE conversation without requiring a specific DDE command be sent to them. Excel for example supports two functions "OnData()" and “SetLinkOnData()” that can be used to cause Excel to automatically run a specific macro whenever new data is received from a DDE server.

As another example, if you "hot link" a data field from the Wedge to a text box in a Visual Basic application or to a tag in Wonderware's InTouch, the text box or tag will automatically receive a "Change" event whenever the Wedge updates the DDE data (i.e. when each new data record is received through the serial port). This Change Event can be used to trigger a subroutine or a script to run each time new data is received. By triggering a macro or subroutine off of a data "change event" or using an "OnData" type function, you do not need to have the Wedge explicitly send a DDE command to the other application to tell it to do something.

In almost all cases you will probably only want to issue a single Field Postamble DDE Command after the very last data field that you have defined in the Wedge. The purpose of the command would typically be to force another application to run a macro or subroutine. The macro or subroutine in the other program would then request data back from the Wedge (i.e. the input data fields containing the serial data) and then go on to process the data.

To learn what commands a particular application supports, you will have to refer to its users manual. You can also refer to the DDE Examples section of this manual (pg. 80) for specific examples for applications like Excel, Access, Visual Basic, FoxPro, Quattro Pro, Lotus 123 and Microsoft Word.

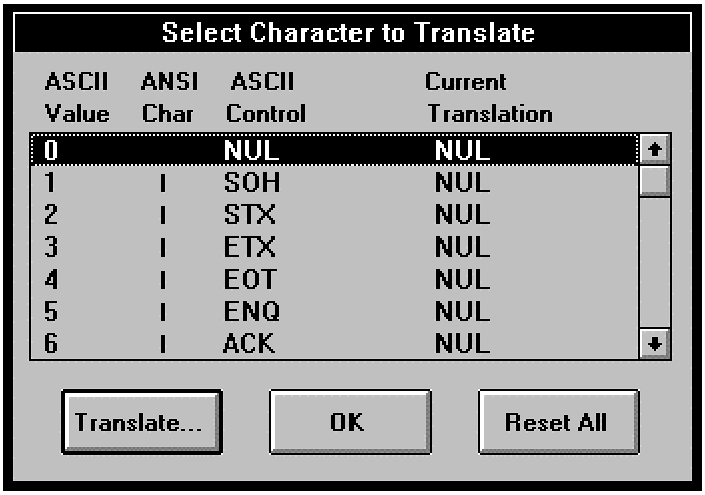
Note: Some applications that support DDE provide a way to program the application to "sense" when new data is received from a server in a DDE conversation. Excel for example supports two function “SetLinkOnData()” and "OnData()" that can be used to cause Excel to automatically run a specific macro or subroutine whenever new data is received from a DDE server. This type of capability allows you to do the same thing that a Field Postamble DDE Command would do without requiring the Wedge to actually issue a Field Postamble DDE Command. In other words instead of the Wedge sending a command to force a macro to be run, the other application would be set up to listen for changes in DDE data and when the data changes it would automatically run a macro.

See Also:
DDE Examples (pg. 54)
Understanding Dynamic Data Exchange (pg. 47).

Note: TAL Technologies maintains a website at: http://www.taltech.com where you can find numerous DDE examples (in addition to those found in this manual) as well as a Software Wedge knowledge base containing answers to many frequently asked questions. If you are having difficulty with a particular application, you may be able to find a solution to your problem by visiting our site and browsing through the "Technical Support" pages.

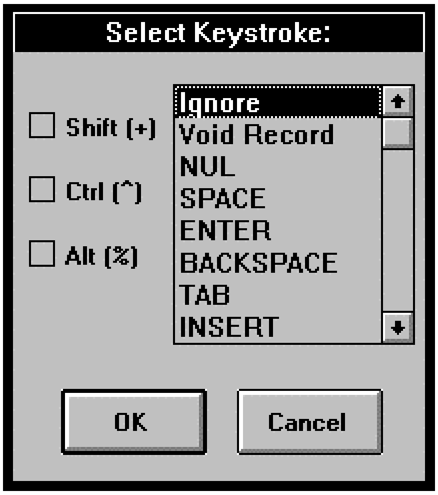
Translation Table

The translation table in the Software Wedge allows you to translate characters received from your serial device just before data is transferred to another application program - after the data has been parsed and filtered. If you select the "Translation Table" option from the Define menu, the translation table will appear as below:



ASCII Value	ANSI Char	ASCII Control	Current Translation
0		NUL	NUL
1	I	SOH	NUL
2	I	STX	NUL
3	I	ETX	NUL
4	I	EOT	NUL
5	I	ENQ	NUL
6	I	ACK	NUL

If the Software Wedge is in "Send Keystrokes Mode", then the translation table is used to convert individual characters to specific keystrokes. Data transmitted via an RS232 serial connection consists of ASCII characters. Because many ASCII characters do not correspond to a specific keystroke, the Translation Table provides a way to map characters to specific keystrokes. To translate characters, simply select the character that you need to translate with your mouse and then click the "Translate" button. This will display the following "Keystroke Selection" dialog box that allows you to select a specific keystroke or key combination.



The keystroke selection dialog box contains a list of all keystrokes as well as three check boxes that allow you to indicate the toggle key states to be used in conjunction with the selected keystroke.

The first three choices in the keystroke list (Ignore, Void Record & Null) do not represent actual keystrokes but instead have special meanings when selected.

Translating a character to *Ignore* causes the character to be ignored altogether by the Wedge. When a character translated to "Ignore" is received, it will be discarded thus it will not appear anywhere in the input data. Also, because the character is being ignored, its presence is not counted when reading data into a fixed length data record or a fixed length data field. Thus, Ignore effectively removes the character from the input data stream before it is even read in by the Wedge.

If you translate a character to *Void Record*, it causes the Wedge to invalidate and remove the entire current record. If a character translated to "Void Record" is present anywhere in an input data record, the entire record is completely ignored and will not be transferred to your application program. For example, suppose you had a device that always transmitted a header record before each set of data records and you want to get rid of the unwanted header record. If there was a specific character that always appeared in the header record but never appears in any of the data records, you could remove the header record by translating the one specific character to “Void Record”.

Translating a character to *Null* causes the character to be removed from the input, however, its presence is still counted when processing fixed length data records or fixed length data fields.

Keystrokes and ASCII characters are two completely different things. In many cases a keystroke will generate a character however some keystrokes, like function keys and cursor navigation keys, do not have a direct correlation to a specific ASCII character. If you will be using the Wedge in "DDE Server Mode" then using the translation table to translate characters to keystrokes does not make sense because DDE data consists of ASCII characters - not keystrokes. Therefore, when the Wedge is in "DDE Server Mode", you would typically use the Translation Table to translate characters to other characters.

If you translate a character to a keystroke using the translation table and then activate the Wedge in DDE Server mode, the Wedge will substitute the closest logical character to the keystroke translation. For example if you translate a character to an "Enter" keystroke and then activate the Wedge in DDE server mode, the Wedge will substitute an ASCII 13 (a carriage return character) for the Enter key.

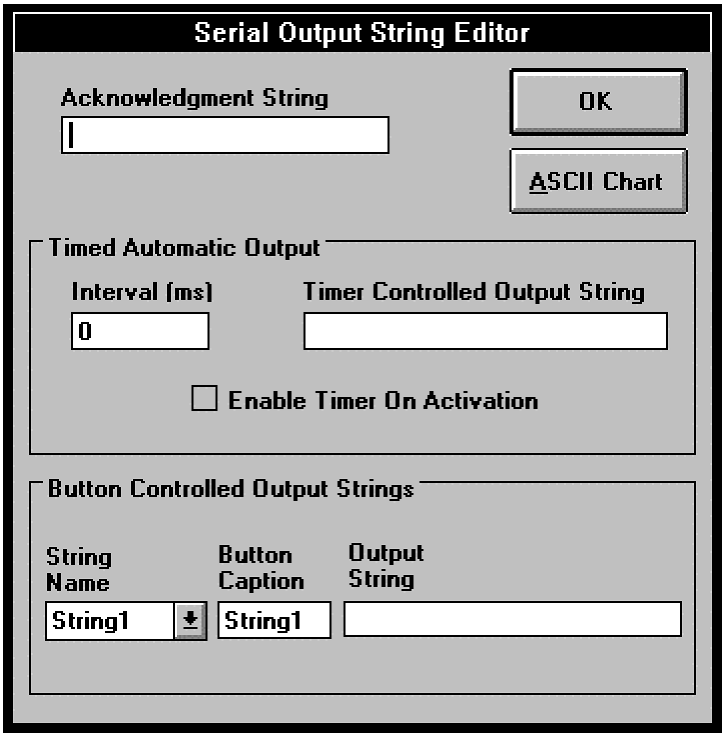
If you translate a character to a keystroke that generates a character, then that character is used as the translation when the Wedge is in DDE server mode. If you specified that a shift, control or alt key modifier be used in conjunction with a keystroke translation, the key modifier is ignored.

Just as when the Wedge is in "Send Keystrokes" mode, characters translated to "Null" in the Translation Table will be removed from the incoming serial data stream when in "DDE Server" mode.

If you are using the Wedge in DDE Server mode and you translate a character to a keystroke that does not generate a character (i.e. a cursor navigation keystroke), then the character will be treated the same as if you translated it to a Null. The only exception to this is in the case of characters translated to a backspace keystroke which will be translated to an ASCII 8 when in DDE Server mode.

Defining Serial Output Strings

Selecting "Output Strings" from the Define menu opens the dialog box shown below where you may define an Acknowledgment string, a Timer Controlled Output string and up to 20 Button Controlled Output strings that can be sent to your serial device while the Software Wedge is active.


Serial Output String Editor
Acknowledgment String
OK
ASCII Chart
Timed Automatic Output
Interval (ms)
Timer Controlled Output String
Enable Timer On Activation
Button Controlled Output Strings
String Name
Button Caption
Output String

Interval (ms): 0
String Name: String1
Button Caption: String1

The Acknowledgment String is a character string that is automatically sent out the serial port after each complete data record is received from your serial device. The capability to send an Acknowledgment string was originally intended for those devices that require one but it could also be used as a way to continually request data from a device that can be polled by sending it a character string.

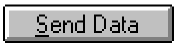
A Timer Controlled Output String may also be defined that is automatically transmitted at regular intervals. The timer interval value may range from 50 to 99,999,999 milliseconds (1/20 second to approximately 27.8 hours) and a check box allows you to specify if timed automatic outputs are initially enabled when the Software Wedge is activated. If a Timer Controlled Output String is defined, a menu item in the Software Wedge window (displayed after the Wedge is activated) will allow you to enable or disable Timed Automatic Outputs.

The "Timer Controlled Output String" is typically used with devices that transmit data in response to a poll. For example, most electronic balances will not transmit a weight reading unless you poll them by sending them a special character string. Suppose that you wanted to continually monitor weight readings from a balance over a period of time. The "Timer Controlled Output String" provides the ideal way to perform this type of function.
Note: If you specify a time interval greater than zero and leave the Timer Controlled Output String empty, the Wedge will toggle the DTR line for 100 ms at the interval rate instead of transmitting a string. This allows you to poll devices that respond to the toggling of the DTR line at regular intervals, including Sylvac digital measuring tools.

The Software Wedge also allows you to pre-define up to 20 "Button Controlled Output Strings" that can be sent to your serial device while the Software Wedge is active. These "Output Strings" (referenced as "String1" thru "String20") are each associated with a "button" that will be displayed in the Software Wedge window after the Wedge is activated. Clicking your mouse on an output string button causes the string to be sent out the serial port to your device. When defining button controlled output strings you may also specify a "Button Caption" for each button to remind you of its contents or purpose. For example, a modem can be instructed to dial a telephone number by sending it the string "ATDT" followed by the telephone number and a carriage return. For this situation, you could define an output string containing the required "ATDT" command and the telephone number & carriage return and then define its button caption as "Dial Phone". Then whenever you click your mouse on the button with the caption "Dial Phone" in the Software Wedge window, the string would be sent to the modem causing it to dial the number. (The command to hang up a modem is "ATH" followed by a carriage return.)

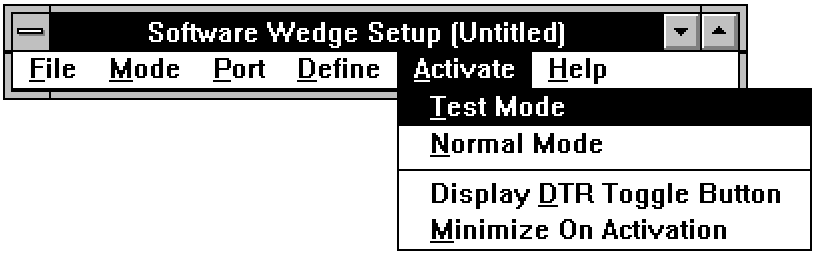
Button controlled output strings can be used for all sorts of functions including sending prompt strings, sending programming instructions, etc. Refer to the users manual for the device that you are using for information about any commands or prompts that it accepts.

When defining a caption for a button you can assign an *access key* to the button by including an ampersand (&) in the caption immediately preceding the character you want to be used as the access key. This character will appear underlined in the button caption when the Software Wedge is activated. Pressing the underlined character in combination with the Alt key while the Software Wedge is active and has the input focus has the same effect as clicking your mouse on the button. For example, specifying "&Send Data" for a button caption would cause the button to appear as below with Alt + S being the access key.



When editing output strings, an ASCII chart is available so that you can select characters that cannot be entered on your keyboard (i.e. control codes, carriage returns, etc.).

The Activate Menu


Software Wedge Setup (Untitled)
File Mode Port Define Activate Help
Test Mode
Normal Mode
Display DTR Toggle Button
Minimize On Activation

The "Activate" menu contains four entries. The first two entries, "Test Mode" and "Normal Mode" are used to activate the Software Wedge after you have finished configuring it to work the way you want. The Wedge must be activated using one of these options before it will actually perform any serial I/O functions. These two options and the differences between them are fully described in the section: Activating The Software Wedge (pg. 39)

The next two items are "ON / OFF" options that are turned on or off by selecting the them. When selected (ON), a check mark will appear to the left of each menu item.

The "Minimize On Activation", and "Display DTR Toggle Button" control the appearance of the window that is displayed when the Software Wedge is activated.

The "Minimize On Activation" option causes the Wedge to run minimized or as an icon when activated. The Wedge will continue to function even when minimized therefore, if you do not need the Wedge to be displayed in a window after it has been activated, you can get rid of it by selecting this option.

See Also: Activating The Wedge Automatically With a Specific Configuration File (Pg. 14)

The "Display DTR Toggle Button" option causes a button to appear in the Software Wedge Window with the caption "DTR~" after you activate the Wedge. This button is used to toggle the serial port Data Terminal Ready line for 100 ms. This action is interpreted by certain types of serial devices as a request to transmit a data record.

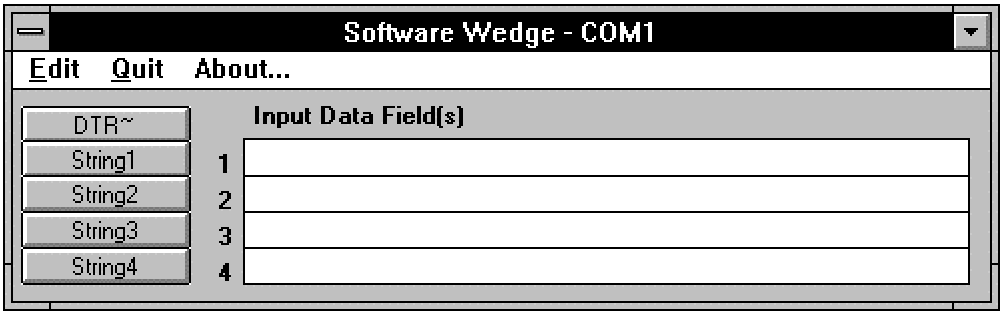
Activating the Software Wedge

The Software Wedge will not accept any serial data until it is activated by selecting either “Test Mode” or “Normal Mode” from the Activate menu. The difference between the two modes is that when you activate the Wedge in "Test Mode", you will be able to return to the Software Wedge main menu and edit your configuration parameters. Test mode should always be selected when you are first configuring the Wedge for use with a particular application and a particular serial device. Until you have everything working properly, you will find it much easier to activate the Wedge in Test Mode.

In "Normal Mode" you will not be able to go back and edit any configuration parameters without quitting the Wedge altogether and then restarting it and reloading your configuration file. When you activate the Wedge in "Normal Mode", its main menu and all sub menus, dialog boxes, and setup code are unloaded from memory in order to conserve Windows system resources. This helps free up system memory and other resources that may be needed by other application programs and it also improves overall system performance.

The Software Wedge Window

When you activate the Wedge in either Test or Normal mode, the main setup menu will disappear and a window similar to the one below will appear. This window is called the "Software Wedge Window".



The title bar of the Software Wedge Window will indicate the serial port that it is currently configured and activated for.

The menu bar in the Software Wedge Window contains three entries: Edit, Quit and About.

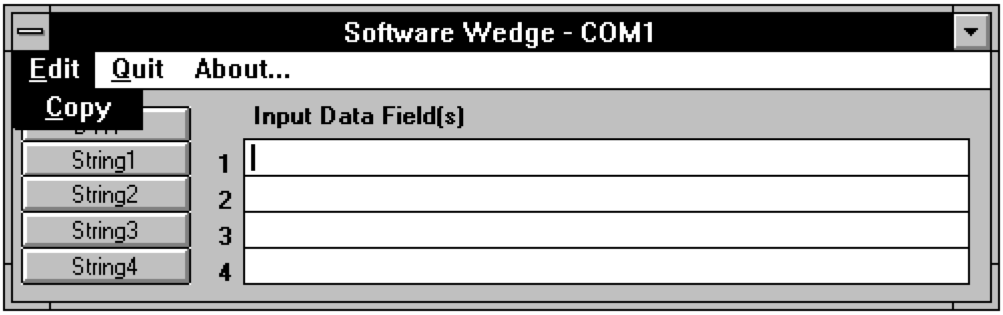
The number of "Input Data Fields" (the text boxes under the label "Input Data Field(s)") and "Serial Output Buttons" (the column of buttons on the left side of the window) that will be visible in the Software Wedge Window depends on the number of data fields previously defined when you configured the Software Wedge.

The button with the caption "DTR~" is called the "Data Terminal Ready Line Toggle Button". This button will appear only if the "Display DTR Toggle Button" option was selected in the Activate menu. Pressing this button causes the serial port Data Terminal Ready Line to be toggled from "on" to "off" for approx. 100 ms. This action is interpreted by certain types of serial devices as a request for data.

Note: The Software Wedge Window may be shrunk to an icon without affecting its operation. The only disadvantage of "minimizing" the Software Wedge Window is that you must re-open it in order to press any "output string buttons" that you have defined.

The Edit Menu

The Edit menu contains a single *Copy* entry.

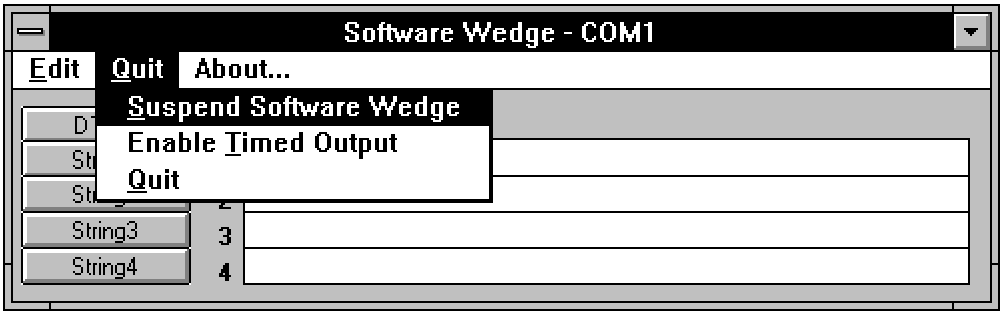


The *Copy* command allows you to copy the contents of one of your input data fields into the Windows clipboard. To copy a data field, click your mouse in the data field text box that you want to copy and select "Copy" from the Edit menu. Any data copied from the Software Wedge can be either *Pasted* or *Paste Linked* to other Windows applications.

See Also: Establishing DDE Links Using The Windows Clipboard (pg. 48)

The Quit Menu

The Quit menu provides two main choices, "Suspend" and "Quit". If a timer controlled output string was defined, a third option "Enable/Disable Timed Output" will also appear.



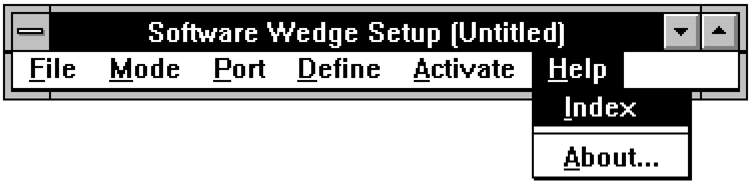
Selecting "Suspend" suspends the operation of the Software Wedge and changes the Suspend menu item to "Resume" (which allows you to resume the Software Wedge). While the Software Wedge is suspended, the title bar will contain an S in parentheses (S) to remind you that the Wedge is currently suspended. (You can also suspend and resume the Wedge by sending it special DDE commands from another application.)
See Also: The Software Wedge DDE Command Set (pg. 50)

If you quit the Software Wedge after activating it in "Test Mode", you will be returned to the main menu. If you activated the Wedge by selecting "Normal Mode" from the Activate menu or by specifying a configuration file name on the command line, the Software Wedge will quit running altogether.
See Also: Activating The Wedge Automatically With A Specific Configuration File (pg. 14)

The *Enable/(Disable) Timed Output* option is used to turn on and off any "timer controlled output string" that has been defined. If a timer controlled output has been defined and is currently Enabled, then this item will have the caption "Disable Timed Output". Likewise, if the timer controlled output string is currently Disabled, this item will have the caption "Enable Timed Output" allowing you to re-enable it. (You can also enable and disable timer controlled outputs by sending DDE commands to the Wedge from another application.)

See Also:
Defining Serial Output Strings (pg. 38)
The Software Wedge DDE Command Set (pg. 50)

Accessing On-Line Help



On-Line help is available from the main menu of the Software Wedge by selecting the Help sub menu item "Index". This option will access the Windows help system and display an index of available help topics for the Software Wedge. On-Line help is only available from the main (setup) menu and can not be accessed after the Software Wedge has been activated.

Selecting the sub menu item "About" will display a copyright notice.

Keystroke Macro Rules

When editing “Record Preamble” or “Field Postamble” keystroke macros with the Wedge in "Send Keystrokes Mode", to specify a single keyboard character, use the character itself. For example, to represent the letter A, type an "A". If you want to represent more than one character, append each character to the one before. To represent the letters a, b, and c, simply enter: abc. The plus sign (+), caret (^), percent sign (%), tilde (~) and parentheses () have special meanings to the Software Wedge. To specify one of these special characters, enclose the character inside curly braces. For example, to specify the plus sign, use {+}. Square brackets ([]) have no special meaning to the Software Wedge but you must enclose them in braces as well because square brackets have special meaning when performing DDE operations. To send curly brace characters, use "{{}" and "{}}", respectively. To specify characters that are not displayed when you press a key (such as Enter or Tab) and other keys that represent actions rather than characters, use the codes in the table below:

Key	Code	Key	Code
Backspace	{BACKSPACE}, {BS} or {BKSP}	Tab	{TAB}
Break	{BREAK}	Up	{UP}
CapsLock	{CAPSLOCK}	F1	{F1}
Clear	{CLEAR}	F2	{F2}
Delete	{DELETE} or {DEL}	F3	{F3}
Down	{DOWN}	F4	{F4}
End	{END}	F5	{F5}
Enter	{ENTER} or ~	F6	{F6}
Escape	{ESCAPE} or {ESC}	F7	{F7}
Help	{HELP}	F8	{F8}
Home	{HOME}	F9	{F9}
Insert	{INSERT}	F10	{F10}
Left	{LEFT}	F11	{F11}
NumLock	{NUMLOCK}	F12	{F12}
Page Down	{PGDN}	F13	{F13}
Page Up	{PGUP}	F14	{F14}
PrtScrn	{PRTSC}	F15	{F15)
Right	{RIGHT}	F16	{F16}
Scroll Lock	{SCROLLLOCK}		

To specify keys combined with any combination of Shift, Control, and Alt, precede the regular key code with one or more of these codes:

Key	Code
Shift	+
Control	^
Alt	%

To specify that Shift, Control, and/or Alt should be held down while several keys are pressed, enclose the keys in parentheses. For example, to hold the Shift key while pressing E then C, use "+(EC)". To hold down Shift while pressing E, followed by C without the Shift key, use "+EC". To specify repeating keys, use the form {key number} where there is always a space between key and number. For example, {LEFT 42} means press the “Left Arrow” key 42 times; {x 10} means press the letter "x" 10 times.

Date And Time Stamps In Keystroke Macros

The Software Wedge supports six date and time stamp functions that are specified by entering the following keywords in any preamble or postamble macro defined when the Software Wedge is in *Send Keystrokes Mode*.

Keyword	Function	Range
{Year}	Inserts the current year	00 - 99
{Month}	Inserts the current month	01 - 12
{Day}	Inserts the current day	01 - 31
{Hour}	Inserts the current hour	00 - 24
{Minute}	Inserts the current minute	00 - 59
{Second}	Inserts the current second	00 - 59

All date & time values are sent as a two byte string padded with a zero on the left if the value is less than ten. Only one occurrence of each stamp function may be specified in any one macro. For example, the following macro: "{Year}/{Month}/{Day}:{Hour}:{Minute}:{Second}" is valid but "{Year}{Year}" is invalid.

Date and time stamp functions cannot be used when the Wedge is in DDE Server mode, however, if you are using DDE server mode, then typically you will be able to use date and time functions provided by the application that will be receiving data from the Wedge. For example the macro language in Excel provides numerous date and time functions that you could use for this purpose.

See Also: Specifying Pre / Postamble Keystrokes (Send Keystrokes Mode) (pg.31)
Excel DDE Example #3 (pg. 61)

Understanding Dynamic Data Exchange

Dynamic Data Exchange is a feature of Windows that allows two programs to pass data or send commands directly to each other. DDE can be thought of as a direct link between two application programs. In most cases, one application is providing some form of data (either text or graphics) to another application. The application that is the source of the data is called the "server" and the application that is receiving the data is called the "client". Thus, the Software Wedge is primarily a DDE Server that "serves" incoming serial data to other (client) programs, however, it can also act as a client in some special cases.

Each data item that a server application can provide has a unique identifier consisting of three parts, a *DDE Application Name*, a *DDE Topic*, and a *DDE Item Name.*
The DDE Application Name is almost always the executable file name for the server application (without the .EXE extension), the DDE Topic typically identifies a group or category of data in the server application and each data item that a server application can provide has a unique DDE Item name. Thus the Application Name, Topic, and Item Name serve to identify the exact source of the data in a server application.

DDE links are always initiated in the client application. The client initiates a DDE link by broadcasting a message containing a *DDE Application Name*, a *DDE Topic*, and optionally a *DDE Item* to all other applications currently running. If a server application is running that can provide the data, it responds to the DDE initiate and the Windows operating system opens a "link" between the two applications. This process is actually called "Establishing a Link". Luckily, most Windows programs that support DDE insulate you from the low level details of establishing DDE links and simply allow you to specify the Application Name, Topic, and Item Name for a specific piece of data that you want and the link is then automatically established for you by your application program.

For example, if you enter the formula: =WinWedge|Com1!'Field(1)' in a cell in an Excel spreadsheet and then press the Enter key, Excel will automatically establish a DDE link between the Software Wedge and the cell in the spreadsheet. (Note: WinWedge must be running and activated for COM1 in order for the above link formula to be successful). After the link is established, any data in the text box "Field(1)" in the Software Wedge window will automatically appear in the "linked " cell in the spreadsheet. Also, whenever the data for Field(1) in the Software Wedge changes, the contents of the spreadsheet cell will automatically be updated with the new data. It's like having the operating system do an automatic cut & paste from the server to the client whenever the server's data changes. The formula: =WinWedge|Com1!'Field(1)' contains the three parts necessary to successfully link to the Wedge; the DDE Application Name (WinWedge), the DDE Topic (Com1) and the specific DDE Item Name 'Field(1)'.

Either application involved in a DDE conversation can terminate the link. Some applications have menu options that allow you to selectively terminate any open DDE links. Ending either of the linked applications also causes all links between the two applications to be terminated.

Dynamic Data Exchange also allows a client application to send commands to a server if the server program was designed to accept them. The types of commands, if any, that a server program can accept will vary depending on the application. Usually, DDE commands that can be accepted by a server application will be well documented in the application's users manual including the DDE Application Name and Topic required by a client program to establish the DDE link to the server’s command processor. The Wedge for example understands many different DDE Commands that allow you to do all sorts of things including transmitting data out the serial port to your serial device.

Establishing DDE Links Using The Windows Clipboard

Applications that support DDE usually provide an easy way to initiate DDE conversations using the Windows clipboard. If an application can function as a DDE client, it will almost always have a "Paste Link" or "Paste Special" command in its main menu (usually in an "Edit" menu). DDE server applications will likewise have a "Copy" command in their main menu (also in an "Edit" menu). Note: The presence of a "Copy" command in a program does not necessarily mean the program can act as a DDE server.

To initiate a DDE conversation, you would open up the **server** application and select the data to be linked using your mouse (the contents of an Input Data Field text box in the Software Wedge Window for example) and then "Copy" the data to the clipboard by selecting "Copy" in its Edit menu. Next, you would open up the **client** application and click your mouse in the position where you would like the "Linked" data to appear (a cell in a spreadsheet for example) and then choose "Paste Link" or "Paste Special" from the client application's Edit menu.

If the Cut & Paste Link process is successful, the data from the server will appear in the client application from that point on until the link is terminated as the result of ending either the client or the server application. That's all there is to it! No messing about with Application Names, Topics or Items because all that information is hidden away and passed implicitly through the Windows clipboard

Note: When using the Cut & Paste Link method, some client applications may display a dialog box that allows you to specify either a "Hot" or "Cold" link. A "Hot" link means that changes in the data from the server are immediately updated in the client when the change occurs. A "Cold" link means the linked data is updated only at the request of the client program, usually with some sort of "Request" command. Cold links are useful when the linked item consists of a huge amount of data, a large bitmap for example, because it is very inefficient to continually pass an entire bitmap graphic from a server to a client each time a single pixel changes in the server.
For small character based data items (as provided by the Software Wedge), you would normally specify a "Hot" link. In some programs a Hot link may also be referred to as an "Auto Update" link.

DDE and the Software Wedge

The Software Wedge can function as both a server and a client in conversations with other applications. As a server, the Wedge can supply data from the "Input Data Fields" in the Software Wedge window. It can also process DDE commands that are sent back to it from other "client" applications.

As a client, the Software Wedge can issue DDE commands to any application that supports a DDE Command language. DDE commands can only be sent to an application as a "Record Preamble DDE Command" and "Field Postamble DDE Commands" when the Software Wedge is in DDE Server Mode. Because the Wedge is acting as the client in these conversations, it needs to know the DDE Application Name and DDE Topic for the Server application that it will be sending commands to. This is why the Software Wedge prompts you for a DDE Application Name and DDE Topic when you select “DDE Server” from the Mode menu.

See Also: Specifying DDE Commands (DDE Server Mode) (pg. 32)

When using the Wedge in DDE Server Mode, other applications can initiate DDE links with the Software Wedge using the DDE Application Name "WinWedge" and the DDE Topic "COMn" where "n" is the number of the serial adapter that the Wedge is activated for. The DDE Items available will be the contents of each defined input data field in the Software Wedge window. The DDE Item names for the data fields are referenced as 'Field(1)' thru 'Field(n)' where "n" is the highest defined field number. If you use the "Copy" & "Paste Link" method to initiate DDE conversations with the Software Wedge, this information will be automatically passed through the clipboard to the client application and does not need to be explicitly specified.

To initiate DDE conversations with the Software Wedge from a macro in a spreadsheet or word processor, you will have to refer to the users manual for the specific application to learn the syntax of the commands or macro instructions that can be used to implement DDE conversations. A good place to look is in the on-line documentation for the application program that you want to use. Open up the application's help window and then search for the help topic "DDE" or "Dynamic Data Exchange".

Several excellent examples are provided in this manual for some of the more popular application programs like Excel, Lotus, MS Access, etc., however, the examples provided here do not cover all the DDE capabilities provided in each of these applications. It is highly recommended that you investigate whatever DDE functions are available in whatever application programs that you will be using with the Software Wedge.

See Also: DDE Examples (pg. 54).

The Software Wedge DDE Command Set

The Software Wedge supports all of the DDE commands listed below.
DDE commands will only execute if the Software Wedge has been activated.
See Also : Activating The Software Wedge (pg. 41)

DDE commands can be issued to the Software Wedge directly from another application using whatever macro or programming language is provided by the other application.
Note: The Software Wedge does not have to be in "DDE Server" mode in order to accept and process DDE commands. The Wedge will still accept and process all of the DDE commands listed below even if it has been activated in "Send Keystrokes" mode.

See Also:
Understanding Dynamic Data Exchange (pg. 47)
DDE Examples (pg. 54)

Command	Function
[APPEXIT]	Exits the Software Wedge.
[APPMINIMIZE]	Minimizes the Software Wedge window.
[APPNORMAL]	Opens the Software Wedge window.
[BEEP]	Causes the Software Wedge to Beep once.
[CLEAR]	Clears all input data fields in the Software Wedge window.
[COPYFIELD(n)]	Copies the data field "n" into the Windows clipboard.
[SUSPEND]	Suspends the Software Wedge.
[RESET]	Resets the Wedge and flushes the serial buffers.
[RESUME]	Resumes the Software Wedge.
[TOGGLEDTR]	Toggles the serial port DTR line for 100 ms.
[DTR=ON]	Sets the serial port DTR line ON.
[DTR=OFF]	Sets the serial port DTR line OFF.
[RTS=ON]	Sets the serial port RTS line ON.
[RTS=OFF]	Sets the serial port RTS line OFF.

[SEND(text)] Sends the text in parentheses out the serial port.

[SENDCODE(n)] Sends the character with ASCII code "n" out the serial port.

[SENDOUT('text',n...)] Sends text and control codes out the serial port.
Text must be supplied as quoted strings using single quotes and control codes must be supplied as ASCII values. Text and control codes may be combined in any combination that you like with individual elements separated by commas.
For Example, [SENDOUT('Test',13,10)] sends the word *Test* followed by a carriage return (ASCII 13) and a line feed (ASCII 10). The command: [SENDOUT(27,'P')] sends an Escape character (ASCII 27) followed by a capitol P.

[PUSH(button caption)] Pushes a button in the Software Wedge window.
The button is specified using its caption. The specified caption should be the same as that entered when it was originally defined including ampersands.
See Also: Defining Serial Output Strings (pg. 38)

[CONFIG(filename)] Reconfigures the Software Wedge.
filename specifies the configuration file to use and must include the filename extension and the full drive and directory path if it is not in the current directory. You may not specify a configuration that uses a different serial port than the one currently being used.

[TIMER-ON]
[TIMER-OFF] Enables / Disables sending of timed automatic output strings.

[TIMERINTERVAL=n] Changes the timing interval (ms) for timed automatic output strings to the value n. (between 50 and 99,999,999)

Note: All DDE commands sent to the Wedge must be enclosed in square brackets.
When sending DDE commands to the Software Wedge from another application, use the DDE Application Name "WinWedge" and the DDE Topic "COMn" where "n" is the number of the serial port that the Software Wedge is installed for. A DDE Item is not required when sending DDE commands to the Software Wedge.

Using the LINKTEST Utility to Test DDE Commands

The Software Wedge comes with a utility called LINKTEST that allows you to test all Software Wedge DDE commands or test DDE commands that can be sent to other programs. To test any Software Wedge DDE commands, the Wedge must be running and activated in either "Test Mode" or "Normal Mode".

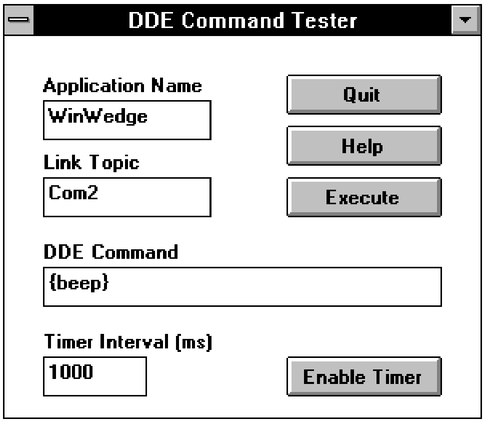
LINKTEST provides two text boxes where you specify the DDE Application Name and Topic for the application that you want to send a DDE command to. The defaults for these parameters are those required by the Wedge when it is activated for COM2. If you are using a different serial port, you should change the Link Topic to the correct value.

A third text box labeled "DDE Command" is where you enter the DDE Command that you want to send. After entering a DDE Command, you can have LINKTEST issue the command to the specified application by clicking the button marked " Execute".

If you get an error message that reads “No Foreign Application Responded to DDE Initiate” then you either did not specify the correct parameters for the DDE Application name or DDE Topic or the application is not running. If you get an error that reads “Foreign Application Won’t Perform DDE Method or Operation” then the application does not recognize either the command or the syntax for the command you are trying to have it execute.

For additional information about using LINKTEST including a list of all DDE Commands that the Wedge can recognize, click the button marked "Help" in the LINKTEST window.

If you use LINKTEST with an application other than the Software Wedge, you will have to refer to its users manual to find out the proper DDE parameters (Application Name and Topic) and also the proper syntax of any DDE commands that it can recognize.



Application Name: WinWedge
Link Topic: Com2
DDE Command: {beep}
Timer Interval (ms): 1000

Understanding How DDE Works Helps Avoid Programming Mistakes

Windows is a "message based", multitasking operating system. Multitasking means that you can have more than one application running at a time and "message based" (also referred to as "event driven") means that while an application is running, it is simply sitting in memory dormant, waiting for event messages to be sent to it by the operating system or from another running application. When you press a key on your keyboard or click your mouse, you are generating events. At the moment an event occurs, the operating system sends a message describing the event to the application that is the target of the event (i.e. the top level window for keyboard and mouse events). Messages are sent by writing them to a message "queue" for the application. When the system is idle (i.e. no programs are busy doing anything), Windows passes control to each running application in turn to give it a chance to process any event messages that it finds in its message queue. When the application is finished processing all its messages, it passes control back to the operating system so that the next running application can process its event messages. The important point here is that while one application is busy doing something, all other applications are unable to process any of their own event messages until the first application finishes or releases control temporarily.

DDE is simply a way for one application to send messages directly to another. When one program passes data or sends a DDE command to another, it is simply sending a message that will get placed in the other application's message queue. The receiving application will not be able to act on that message until the sending program goes idle thus allowing it to process its messages.

If you understand what is going on behind the scenes, you can avoid some common mistakes. For example, suppose you have a device that responds to a prompt sent via the serial port. When the device receives the prompt it sends back a string of data. You also want to write an Excel macro that will send a command to the Wedge causing it to send the prompt out the serial port and then you want Excel to read the response back from the device by performing a DDERequest to the Wedge. You might come up with the following Excel macro that, although it appears to be quite logical, will never work.

```
Sub PromptMyGizmo()
  Chan =DDEInitiate("WinWedge", "COM1")           ' open a DDE link with WinWedge
  DDEExecute Chan, "[SENDOUT('?',13,10)]"          ' send a prompt out the serial port
  MyData = DDERequest(Chan, "Field(1)")  ' read back the response from Field(1)
  Sheets("Sheet1").Cells(1, 1).Formula = MyData    ' write the data to cell A1 in Sheet1
  DDETerminate Chan                                ' terminate the DDE link
End Sub
```

The reason the above subroutine will not work is because the DDEExecute statement only sends a message to the Wedge. The subroutine does not relinquish control and let the Wedge process the message until it reaches the "End Sub" statement. The DDERequest that follows the DDEExecute statement is asking for data that will not be available until the Wedge gets a chance to run; i.e. after this subroutine has finished executing.
Don't worry, there are many ways to work inside a "message based" system as you will find in the following pages of example macros and subroutines.

DDE Examples

The following pages contain example macros and subroutines for some of the more common PC application programs (i.e. Microsoft Excel, Access, Word, Visual Basic, FoxPro, Lotus 123 and Quattro Pro). The examples demonstrate how to do things like transmit and receive serial data using Dynamic Data Exchange with the Wedge or launch and unload the Wedge from memory as well as perform various other related functions.

The examples are primarily designed to demonstrate how to perform the most common types of serial I/O operations with the Wedge using DDE. All of the examples listed in the following pages can be found in a file named EXAMPLES.WRI located in your Software Wedge program directory along with all of the other Software Wedge program files. In most cases you should be able to simply cut and paste the example code from the EXAMPLES.WRI file directly into the application program that you want to use. The comments in the examples will explain the details of how each routine works and will also point out any lines that may need changing in order to work with your particular system configuration.

Important Note Regarding DDE:

DDE is an extremely powerful and flexible way to pass data or commands directly from one application to another, however, this power and flexibility does not come without a small price in additional complexity. The following examples should help greatly to make the process easier. One thing to consider is that there are many situations where DDE may not be required and using the Wedge in "Send Keystrokes" mode may be more suitable. "Send Keystroke" mode is much easier and much quicker to configure than using the Wedge "DDE Server" mode. For example if you are reading data from a simple instrument (i.e. a bar code reader, electronic balance or an electronic caliper), and you just want to input a small amount of data into a spreadsheet or database, then using "Send Keystrokes" mode may be a better and much simpler approach than using DDE.

There may also be situations where it might be convenient to input data from the Wedge using "Send Keystrokes" mode and also send data out the serial port by sending DDE commands to the Wedge. **The Wedge does not have to be in DDE Server mode in order to accept and process DDE commands**. You can still send DDE commands to the Wedge to have it transmit data or prompts out a serial port even when it is in "Send Keystrokes" mode.

See Also: A Typical Software Wedge Configuration Example (pg. 9)
The Mode Menu (pg. 15)
Sending Keystrokes vs. Dynamic Data Exchange (pg. 18)
Understanding Dynamic Data Exchange (pg. 47)
The Software Wedge DDE Command Set (pg. 50)

Note: TAL Technologies also maintains a website where you can find additional DDE examples along with other technical information and answers to frequently asked questions.
Our web address is: **http://www.taltech.com**

DDE Examples For Microsoft Excel Version 4.0

Important – The examples in this section are for Excel 4.0 only. If you are running Excel 5.0 or newer, please use the examples in the following section for Excel 5.0 to Excel 2000.

Example #1 - Launching and Linking To The Software Wedge

The following macro named "Test", will first run the Software Wedge with a configuration file named "MyConfig.SW1", then establish a DDE link between the spreadsheet cell R1C1, and Field(1) in the Software Wedge window. This example assumes that WinWedge.Exe and the configuration file "MyConfig.SW1" exists and both are located in a directory named "WinWedge" and that the config file "MyConfig.SW1" will configure the Software Wedge for serial port COM1.

```
Test
=EXEC("c:\WinWedge\WinWedge.exe c:\WinWedge\MyConfig.SW1")
=SELECT("R1C1")
=FORMULA("=WinWedge|Com1!'Field(1)'")
=RETURN()
```

Example #2 - Sending Data Out The Serial Port

The following Excel macro named "SendPrompt" initiates a DDE conversation with the Software Wedge and then uses the Software Wedge DDE command [SendOut()] to send a question mark followed by a carriage return-linefeed out the serial port to a device. Finally, this macro terminates the DDE link with the Wedge. This macro assumes that the Software Wedge is already activated for serial port COM1.

```
SendPrompt
Chan=Initiate("WinWedge","Com1")              Initiate DDE link with WinWedge
=Execute(Chan,"[SENDOUT('?',13,10)]")         Send question mark with CrLf
=Terminate(Chan)                              Terminate the DDE link
=Return()
```

The following macro does a similar thing as the previous macro except that it sends the contents of cell A1 in spreadsheet Sheet1 out the serial port.

```
SendCellData
=SET.NAME("OutBuf",SHEET1.XLS!$A$1)           Give cell A1 the name "OutBuf"
chan=INITIATE("WinWedge", "COM4")             Initiate a link with WinWedge
=EXECUTE(chan,"[Send("&OutBuf&")]")           Send contents of cell A1
=TERMINATE(chan)                              Terminate the link
=RETURN()
```

Example #3 - Collecting Data To A Column In A Spreadsheet (Excel 4.0)

The following is an example of two Excel macros that work together to take data readings from a DDE conversation with the Software Wedge and place them in a column such that each new reading is added to the bottom of the column. This example first establishes a DDE link between the spreadsheet cell R1C1 and WinWedge|Com1!Field(1) and then defines a counter variable located at Row 2, Column 1. To use this example, you would run the macro "Begin" once when you want to start collecting data. The ON DATA statement in this macro causes Excel to automatically run the second macro "PlaceData" each time the data in R1C1 (the data linked to Field(1) in the Wedge) changes. This macro assumes that the Software Wedge is already activated for serial port COM1.

```
Begin
=SELECT("R1C1")                                      Move to R1C1
=FORMULA("=WinWedge|Com1!'Field(1)' ")               Set up DDE link to Field(1)
=SELECT("R1C3:R50C3")                                Select column 3
=CLEAR(1)                                            Clear it out
=SELECT("R2C1")                                      Move to R2C1
=SET.NAME("RowCount",1)                              Make R2C1 a counter
=ON.DATA("WinWedge|COM1!'Field(1)'","PlaceData")     Set up OnData function
=RETURN()

PlaceData
=SELECT("R1C1")                                      Select the linked cell (Field(1))
=COPY()                                              Copy linked data to clipboard
=SELECT("R"&RowCount&"C3")                           Move to the end of the column
=PASTE.SPECIAL(3,1,FALSE,FALSE)                      Paste the data to the cell
RowCount=RowCount+1                                  Point to the next cell down
=RETURN()
```

The PlaceData macro selects the cell that contains the DDE data in R1C1 from Field(1) in the Software Wedge and copies it into the clipboard. Next it moves the cell pointer to Column 3 and the Row specified by the Counter variable defined earlier. Finally, the data in the clipboard is pasted into the cell and the counter is incremented to point to the next cell down before returning from the macro.
By adding your own macro commands before the RETURN command, you could have Excel do just about anything you can imagine with your data including updating a graph or chart or perform statistical calculations on your data. Thus, you can turn Excel into an extremely powerful serial data acquisition and analysis program with a little creativity.
Note: The "PlaceData" macro in this example will only execute when the linked data in R1C1 (i.e. Field(1)) changes. If you want to record all data inputs even if two or more consecutive inputs are identical, refer to the following example.

Example #4 - Collecting Data To A Column - Another Way (Excel 4.0)

In Example #3, we used the Excel function "OnData()" to cause Excel to automatically run the macro "PlaceData" each time data from Field(1) in the Software Wedge changes. Another way to accomplish the same functionality is to remove the line containing the OnData() statement from the macro "Begin" and instead, have the Software Wedge issue the DDE command: [Run("MyMacro.xlm!PlaceData")] after each input record is received from your serial device. (This example assumes that the PlaceData macro resides in an open macro sheet named "MyMacro.xlm").

For details on how to set up the Software Wedge so that it can issue DDE commands after each input, refer to: Specifying DDE Commands (DDE Server Mode) (pg. 36).

The following two macros are more efficient versions of the Begin and PlaceData macros from example #3 and when used with the technique described above, will operate much faster than example #3. The reason that the following two macros operate much faster is because the cell pointer is not moved each time data is received and also the clipboard is not used to transfer any data. The following PlaceData macro also uses the technique of requesting data from WinWedge instead of maintaining a Hot DDE link between field(1) in the Software Wedge and a cell in the spreadsheet.

```
Begin
=SELECT("R1C1")                         Put cell pointer in R1C1
=SET.NAME("RowCount",1)                 Create name RowCount in R1C1
=SELECT("R2C1")                         Put cell pointer in R2C1
=SET.NAME("hConv",0)                    Create name hConv in R2C1
hConv=INITIATE("WinWedge","Com1")       Initiate a DDE link to WinWedge
=RETURN()                               All done here

PlaceData
SWData=REQUEST(hConv,"field(1)")        Get Field(1) from WinWedge
=FORMULA(SWData,"r"&RowCount&"c2")      Stack the data in column 2
RowCount=RowCount+1                     Increment the row number
=RETURN()                               All done here
```

See Also: Understanding Dynamic Data Exchange (pg. 47)
DDE Example for Excel 5 and 7 - Visual Basic for Applications (pg. 58)

DDE Examples for Excel 5 - Excel 2000 (Visual Basic for Applications)

The Excel VBA subroutines in the following examples must be entered in an Excel module sheet. To create a new module sheet select "Macro" and "Module" from Excel's "Insert" menu.

Example #1 - Sending DDE commands to WinWedge from a VBA subroutine
The following Excel subroutine sends a string of control codes out the serial port by issuing the DDE command “[SendOut()]” to WinWedge. This example sends an escape character (ASCII 27), a capitol “P” and a carriage return (ASCII 13).

```
Sub SendEscapeP()
Chan = DDEInitiate("WinWedge", "COM1")
DDEExecute Chan, "[SENDOUT(27,’P’,13)]"
DDETerminate Chan
End Sub
```

The following example shows how to send a text string passed as an argument to the routine.

```
Sub SendText(StringToSend$)
Chan = Application.DDEInitiate("WinWedge", "COM2")
' The following line concatenates the SEND command with the data to be sent
DDEExecute chan, "[SEND(" & StringToSend$ &")]"
' The & operator is the Excel string concatenation operator thus we are concatenating
' The three strings "[SEND(" , the data in the string variable StringToSend$ and ")]"
' If StringToSend$ ="ABC" then the command sent to WinWedge would be: "[SEND(ABC)]"
DDETerminate chan
End Sub
```

The following subroutine sends a column of data out the serial port starting in the currently active cell. After each cell's data is sent, it reads the cell directly below the current cell. If the cell contains no data then it stops otherwise it continues sending until it hits an empty cell.

```
Sub SendCells()
chan =DDEInitiate("WinWedge", "COM1")          ' open a link to WinWedge on com1
MyPointer = 0                                  ' starting offset from current cell is 0
x$ = ActiveCell.Offset(rowOffset:= MyPointer, columnOffset:=0).Formula
' put string from current cell in x$
While Len(x$)                                  ' while the length of the string is not 0
  DDEExecute chan, "[SENDOUT(' " & x$ & " ',13)]"  ' send the string out the serial port
' the above line sends the data followed by a carriage return (ASCII 13)
   MyPointer = MyPointer + 1                   ' point to the next cell down
   x$ = ActiveCell.Offset(rowOffset:= MyPointer, columnOffset:=0).Formula   ' get next cell
Wend                                           ' loop until we reach an empty cell
DDETerminate chan
End Sub
```

The following subroutine demonstrates how to send the contents of any string variable out the serial port - even strings that contain control characters or non printable ASCII characters. The string to be sent out the serial port is passed as an argument to the SendString subroutine.

```
Sub SendString(StringToSend$)
' The following loop reads the StringToSend$ variable and generates a string containing
' the ASCII values for each character in the string separated by commas.
' This resulting string is then used as the argument to the SENDOUT command.
' Ex: if the variable StringToSend$ = "ABC" then the variable Arg$ will be "65,66,67"
' See the syntax of the SENDOUT command in the Wedge users manual for details.
For x = 1 To Len(StringToSend$)
 Arg$ = Arg$ + LTrim$(Str$(Asc(Mid$(StringToSend$, x, 1))))
 If x <> Len(StringToSend$) Then Arg$ = Arg$ + ","
Next
chan = DDEInitiate("WinWedge", "COM2")
' The following line concatenates the SENDOUT command with the data to be sent
DDEExecute chan, "[SENDOUT(" & Arg$ &")]"
' The & operator concatenates the three strings "[SENDOUT(" , the data in Arg$ and ")]"
' Thus if StringToSend$="ABC" then the command that is sent to WinWedge would be:
' [SENDOUT(65,66,67)]"  - (the ASCII values for chars A, B and C are 65, 66 and 67)
DDETerminate chan
End Sub
```

The following subroutine demonstrates how you would call the above routine passing it the contents of a spreadsheet cell thus sending the cell contents out the serial port.

```
Sub TestSendString()
      X$ = Sheets("Sheet1").Cells(1, 1).Value
      SendString(X$)
End Sub
```

Tech Tip: Excel allows you to define shortcut keys or place buttons directly in a spreadsheet that will run any of the above subroutines. To assign a shortcut key to a subroutine, select "Macro" from the "Tools" menu and when the macro dialog box appears, highlight the subroutine name that you want to assign a shortcut key and click the "Options" button. In the "Macro Options" dialog box check the "Shortcut Key" checkbox and enter a character in the textbox labeled "Ctrl +". Click the "OK" button in the Macro Options dialog and then click the "Close" button in the Macro dialog to return to your spreadsheet. If you hold the Ctrl key down and press your shortcut key, your subroutine will run. To place a button in a spreadsheet that will run a subroutine, select "Toolbars" from the "View" menu and check the "Drawing" checkbox and click the OK button. In the Drawing toolbar click the "Create Button" button and then draw the button in your spreadsheet. When you release the mouse after drawing the button, Excel will open a dialog box allowing you to assign a subroutine to the button. You can even enter a caption for the button by selecting and editing the existing button caption.

Example #2 - Launching the Software Wedge from Excel 5 - Excel 2000

The following VBA subroutine executes the Software Wedge and passes it the name of a WinWedge configuration file (MyConfig.SW1) on the command line. This causes the Software Wedge to automatically load the specified configuration file and then activate itself in NORMAL mode.

```
Public Const MyPort$ = "COM1"                    ' Change port if necessary
Sub RunWedge()
On Error Goto ErrorHandler                       ' Set up an error trap
AppActivate "Software Wedge - " + MyPort$        ' Try to activate the Wedge Window
' The above line will generate an error if Wedge is not running or activated
On Error Goto 0                                  ' Remove error trap
AppActivate Application.Caption             ' Set the focus back to excel
Exit Sub

ErrorHandler:                                    ' Wedge is not running - try to launch it
RetVal = Shell("C:\WINWEDGE\WINWEDGE.EXE C:\WINWEDGE\MyConfig.SW1")
If RetVal = 0 Then                               ' Launch failed
    Beep
    MsgBox ("Cannot Find WinWedge.Exe")          ' Warn user and quit
    Exit Sub
Else                                                   ' Launch succeeded
    Application.Wait (Now + TimeValue("00:00:02"))     ' Give Wedge time to load
    AppActivate "Microsoft Excel"                      ' Switch focus back to Excel
End If
Resume Next
End Sub
```

Note: If you name an Excel subroutine "Auto_Open()", Excel will automatically run the routine whenever the spreadsheet that contains it is initially opened. The Auto_Open subroutine is an excellent place to launch the Wedge (as well as do any other initialization functions that you might require). If you re-name the above subroutine to "Auto_Open()" then you can save yourself the step of having to manually run the above subroutine.
Similar to the Auto_Open subroutine, Excel also supports an Auto_Close subroutine that runs automatically when you close your spreadsheet. This might be a good place to tell the Wedge to quit and unload itself from memory as in the following example.

```
Sub Auto_Close()                             ' This sub runs when you close the sheet
 On Error Resume Next                        ' Ignore errors and try to launch WinWedge
 chan = DDEInitiate("WinWedge", MyPort$)     ' Open a dde link with the wedge
 DDEExecute chan, "[Appexit]"                ' Tell Wedge to quit
' No need for a DDETerminate statement here because the wedge is no longer running
' When either application in a DDE conversation quits, all links are terminated
End Sub
```

Example #3 - Requesting Data From WinWedge Using A VBA Subroutine (Excel 5 - Excel 2000)

Steps for setting up the Software Wedge:

1. Select “DDE Server” from the Software Wedge “Mode” menu. When the dialog box appears asking for a DDE Command Destination Application, enter: “**EXCEL**” as the Application Name and then enter: “**SYSTEM**” as the DDE topic.

2. Select “Input Record Structure” in the “Define” menu and define the structure of the input record(s) to WinWedge. When you get to the final Window with the caption “Input Record Definition Editor”, enter the string: **[RUN("GetSWData")]** as the *Field Postamble DDE Command* after the last data field that you have defined. This is a DDE command that will be sent to EXCEL after each data record is received by the Wedge.

3. Set up the rest of the Software Wedge parameters and then activate it.

Steps for setting up EXCEL:

1. Create or edit a macro module and enter the following code in the module:
(To create a new module select “Macro” and “Module” from Excel’s “Insert” menu.)

```
Sub GetSWData()
Static RowPointer As Long, ColPointer As Long
' preserve variable values between calls
If RowPointer = 0 Then RowPointer = 1: ColPointer = 1
' If this is the first time through then initialize RowPointer to point to Row 1 and
' ColPointer to point to Column 1 - i.e. collect data into column 1 starting in Row 1
Chan = DDEInitiate("WinWedge", "Com1")          ' open DDE link to Wedge on Com1
F1 = DDERequest(Chan, "Field(1)")               ' get Field(1) data into a variant array
WedgeData$ = F1(1)                              ' convert variant array to a string
DDETerminate Chan                               ' kill the DDE link
Sheets("Sheet1").Cells(RowPointer, ColPointer).Formula = WedgeData$
' write the data to cell address: (RowPointer,ColPointer) in Sheet1
Sheets("Sheet1").Cells(RowPointer, ColPointer + 1).Formula = Time$
' The above line writes the current time to cell address: (RowPtr,ColPtr+1) in Sheet1
RowPointer = RowPointer + 1                     ' point to the next cell down
End Sub
```

The example above sets up the Wedge to issue a DDE command consisting of the Excel “**RUN**” command forcing Excel to run the subroutine “**GetSWData()**” after each data record is received from your serial device. The “GetSWData” subroutine performs a DDERequest to the Wedge that returns the contents of FIELD(1) to a variable named “WedgeData$”. The data is then assigned to a cell in “Sheet1” and a pointer variable is updated so that the next input will be written to the cell directly below the last input. The example above assumes that the open workbook contains a worksheet named **Sheet1**.

Example #4 - Requesting Data From WinWedge Using The SetLinkOnData Method.
(And other neat Excel tricks) (Excel 5 - Excel 2000)

The Excel VBA macro language has a function called "SetLinkOnData" that can be used configure Excel to automatically run a subroutine whenever data from a DDE Server is updated. In the previous example we showed how to configure the Wedge to send a command to Excel to force it to run a macro after each data record is received from a device on a serial port. This example macros show how to use the Excel SetLinkOnData method instead of having the Wedge send a command to Excel to trigger a macro to run.

The two techniques are similar in that the end result is that after each data record is received by the Wedge, Excel automatically runs a macro that grabs the data from the Wedge and does something with it. The difference between them is simply that in the previous example the Wedge triggers the macro to run, whereas in this example Excel automatically runs the macro with no coaxing from the Wedge.

A benefit of the following method is that the data from the Wedge will be transferred to Excel slightly faster than it would if the Wedge were sending a command to Excel after each input.

In addition to demonstrating the SetLinkOnData method in Excel, the following macros also show how to launch and activate the Wedge automatically when you open your spreadsheet as well as how to quit the Wedge automatically when you close your spreadsheet.

To use the following macros, set up the Software Wedge to work with your instrument by choosing the correct serial communications parameters and defining the input data record structure to fit the data that is returned from your serial devices. If you set up the Wedge in DDE Server mode, make sure that you do not have any "Field Postamble DDE Commands" defined. Next, activate the Wedge in Test Mode, switch to Excel and enter the following macros into a module in your Excel spreadsheet. (You may have to modify the values for some of the Global constants as described in the comments in the subroutines.) After you have entered the following subroutines, save your spreadsheet and close it. Finally, re-open your spreadsheet and start collecting data. The Wedge should automatically load and activate itself when you open the sheet.

```
Global RowPtr As Long, ColPtr As Long   ' Define global variables
Global Const MyPort = "COM1"  ' Change comport if necessary for your configuration
' If Wedge is not set up for COM1, then change the line above to the right port
Global Const CmdLine = "C:\WinWedge\WinWedge.Exe C:\WinWedge\Config.SW1"
' Change the above command line to point to your copy of wedge and your config file



Sub Auto_Open()                    ' This sub runs when you open the spreadsheet
  On Error Resume Next             ' Ignore errors and try to launch WinWedge
  RetVal = Shell(CmdLine)          ' Launch Wedge using command line defined above
' The above line launches and activates the wedge with a config file: Config.SW1
  If RetVal = 0 Then                                  ' Error - Wedge not found
    Beep : MsgBox ("Cannot Find WinWedge.Exe")' Display warning
    Exit Sub                                          ' and quit
  End If                                              ' Otherwise -
  Application.Wait (Now + TimeValue("00:00:03"))  ' Give Wedge time to load
  AppActivate Application.Caption                     ' Set the focus back to excel
  StartCollecting                                     ' Set up Excel to collect data
End Sub



Sub StartCollecting()
RowPtr = 1: ColPtr = 1             ' Initialize global variables
Sheets("Sheet1").Activate          ' Activate sheet 1 and set up a DDE link to WinWedge
Sheets("Sheet1").Cells(1, 50).Formula = "=WinWedge|" & MyPort & "!'Field(1)' "
ActiveWorkbook.SetLinkOnData "WinWedge|" & MyPort & "!'Field(1)' ", "GetSWData"

End Sub
```

The SetLinkOnData method causes Excel to run the GetSWData macro automatically when new data is available in the Wedge (when Field(1) is updated).

The SetLinkOnData method in Excel eliminates the need to have the Wedge send a Field Postamble DDE Command to Excel to cause it to run the GetSWData subroutine. Excel will automatically run the GetSWData sub when the Wedge updates the Field(1) DDE item.

If you have more than one field defined in the Wedge, use the last data field to trigger the SetLinkOnData method. This will insure that all data fields are available when the GetSWData subroutine runs.

```
Sub Auto_Close()            ' This macro runs automatically when you close the spreadsheet.
 StopCollecting                         ' Set up excel to stop collecting data
 On Error Resume Next                   ' Ignore errors
 chan = DDEInitiate("WinWedge", MyPort)' Open a dde link with the wedge
 DDEExecute chan, "[Appexit]"           ' Tell wedge to quit
' No need for a DDETerminate statement here because WinWedge is no longer running.
' When an application involved in a dde conversation quits, all links are terminated.
End Sub



Sub StopCollecting()
Sheets("Sheet1").Activate                          ' Activate sheet1
Sheets("Sheet1").Cells(1, 50).Formula = ""
' Remove the dde link from R1C50 so that Excel does not generate an error
' when you open the spreadsheet with WinWedge not running and activated.
ActiveWorkbook.SetLinkOnData "WinWedge|" & MyPort & "!'Field(1)' ", ""
' Shut down the SetLinkOnData function by assigning an empty string to the macro name.
End Sub



Sub GetSWData()
If RowPtr = 0 Then RowPtr = 1: ColPtr = 1
' If this is the first time through then initialize RowPointer to point to Row 1 and
' ColPointer to point to Column 1 -  i.e. collect data into column 1 starting in Row 1
chan = DDEInitiate("WinWedge", MyPort)             ' Initiate DDE link with wedge
MyVariantArray = DDERequest(chan, "Field(1)")      ' Get Field(1) from Wedge
MyString$ = MyVariantArray(1)                      ' Convert to a string
' The DDERequest function in Excel returns a variant array data type.
' This is a peculiarity of Excel - the return type should be a string data type.
' The above line fixes the Excel "inconsistency" and converts the variant array
' to a string by assigning element 1 of the variant array to a string variable.

' Add your code here to do something with the data maybe?
' Or possibly send a command back to the Wedge as in the following line
' DDEExecute chan, "[Beep]"  ' Send a beep command to the Wedge

Sheets("Sheet1").Cells(RowPtr, ColPtr).Formula = MyString$
' The above line writes the data to cell address: (RowPtr,ColPtr) in Sheet1
DDETerminate chan                                  ' Terminate the DDE link
RowPtr = RowPtr + 1                                ' Point to the next row down
End Sub
```

Example #5 - Getting past the 40 field limit in the Wedge (Excel 5 - Excel 2000)

Excel 5 has two "idiosyncrasies" with its DDERequest() function. The first is that the DDERequest() function always returns a variable with the data type "Variant Array".
For some unknown reason the DDERequest function in Excel expects to receive a Tab delimited array of data. In fact, if the data from the Wedge were tab delimited, Excel would automatically parse it and fill up a Variant Array such that each tab delimited data item would be stored in successive array elements. Typically, data from most instruments is not tab delimited, therefore when you perform a DDERequest to the Wedge from Excel, you end up with a Variant Array variable with only a single element (element 1) containing all the data from the field in the Wedge that you are requesting the data from.

Excel will let you assign a Variant Array to a cell in a spreadsheet, however, if you try to perform string functions on a Variant Array, Excel generates an "Invalid Data Type" error. The Variant Array must be converted to a string data type before it can be used in a string function. This problem is easily overcome by simply assigning the Variant Array element 1 to a string. The following code fragment uses the DDERequest function to return a Variant Array and then uses an assignment statement to convert element 1 of the Array to a string.

```
DDEChannel = DDEInitiate("WinWedge", "Com2")
DDEVariantArray = Application.DDERequest(DDEChannel, "Field(1)")
DDETerminate DDEChannel
StringVariable$ = DDEVariantArray(1)          ' assign Array element 1 to a String variable
```

The second problem is that the DDERequest() function can only return string data that is less than 255 bytes long unless the string is delimited with tabs (see the following section "Advanced Excel Tricks" pg. 67). If you try to DDERequest() a data string that is longer than 255 bytes, Excel truncates the data and you lose anything after the 255th byte.

The following subroutine shows how to get around both of the problems described above, as well as how to get around the 40 field limit in the Software Wedge. To get past the 40 field limit in the Wedge, instead of using the parsing capabilities of the Wedge, we will use our own parsing routine written in Excel VBA. To do this we first have to get all the data from the device into a single string variable in Excel. This is where the problems in Excel catch up to us, especially when the total length of the data stream could be more than 255 bytes.

The following example shows how to configure the Wedge and Excel to deal with a situation where a device transmits a data record containing over 40 data fields that are delimited with commas and terminated with a carriage return. The total length of the data could also be over 255 bytes. If Excel were capable of pulling in over 255 bytes with the DDERequest function, then all we would have to do is define the "End of Record Event" in the Wedge as "Carriage Return or CrLf Received" or "Time Delay Between Records" and then define the structure of the data records as "Single Field". Finally, we would use a VBA macro to parse the string after we pass it to Excel. Because the data string could be over 255 bytes long, we have to break it into pieces before passing it to Excel and then put it back together again in our VBA macro.

The way to do this is to again select "Carriage Return or CrLf Received" or "Time Delay Between Records" as the "End of Record Event" in the Wedge and then select "Multiple Fixed Length Data Fields" as the "Record Structure". Finally, define several data fields with all their field lengths set to 250 bytes. You need to define enough 250 byte data fields so that the largest data record from the device will fit in all the fields defined in the Wedge. For example if the largest record transmitted by the device is 800 bytes, then you would need to define four 250 byte data fields. Make sure that the Wedge is in DDE Server mode and after the very last data field that you define, enter: **[RUN("NewGetSWData")]** as the "Field Postamble DDE Command".
The above command will cause Excel to run a subroutine named "NewGetSWData" after each complete data record is received by the Wedge.
Note: Be sure not to apply any Filters to any of the data fields that you define in the Wedge.

The following subroutine performs the job of pulling the data from the Wedge and then putting it back together into one long string. Finally the subroutine parses the data into individual data fields and plugs each field into separate cells in a column of the spreadsheet.

```
Sub NewGetSWData()     ' This sub is run by the Wedge after each data record is received
Static R As Long, C As Long        ' R & C point to the Row and Column where data will go
If R = 0 Then R = 1: C = 1         ' Make sure neither R or C is zero
Chan = DDEInitiate("WinWedge", "Com2")
' Assume that there are 4 data fields defined - change to more or less if necessary
Field1 = DDERequest(Chan, "Field(1)")     ' Get field 1
Field2 = DDERequest(Chan, "Field(2)")     ' Get field 2
Field3 = DDERequest(Chan, "Field(3)")     ' Get field 3
Field4 = DDERequest(Chan, "Field(4)")     ' Get field 4
DDETerminate Chan                                       ' Kill the link

MyVar$ = Field1(1) & Field2(1) & Field3(1) & Field4(1) & ","
' Convert each variant array to a string, concatenate them all together and add a final
' comma delimiter for the following parsing routine. The following code parses the string
' MyVar$ by searching for commas and placing each delimited field in a separate row
StartPos = 1                              ' Starting position in the string - start at 1st byte
While StartPos < Len(MyVar$)              ' Scan until we reach the end of the string
  DelimPos = InStr(StartPos, MyVar$, ",")  ' Find the next comma delimiter
  DataPoint$ = Mid$(MyVar$, StartPos, DelimPos - StartPos)
' Pull out a data point between the starting position and the position of the delimiter
  StartPos = DelimPos + 1           ' Update the starting position (skip over the delimiter)
  Sheets("Sheet1").Cells(R, C).Formula = DataPoint$ ' save the current data point
  R = R + 1                         ' Point R to the next row down for the next data point
Wend                                ' Go get the next point

R = 1              ' Point R back to row 1
C = C + 1          ' Increment C - move to the right one column for the next set of data
End Sub
```

Example #6 - Advanced Excel Tricks - Reading Large Arrays Of Data Into Excel (Excel 5 - Excel 2000)

The native format for Excel data is tab delimited text with a carriage return at the end of each row of data. When you perform a DDERequest function in Excel, it returns a variable with the data type "Variant Array". If the data that is requested with the DDERequest function is tab delimited with carriage returns at the end of each row of data and we assign this type of data to variant array, Excel will automatically parse the data and fill up the array with the data. If you use the FormulaArray function to assign a variant array variable to a range of cells, Excel will write the data to the specified range.

For example, if you have a device that generates data in the following format:

```
123,456,789,123<cr>
456,789,123,456<cr>
etc...
```

you can use the "Translation Table" in the Software Wedge to convert all commas to tabs so that the data would appear in the Wedge as follows:

```
123<tab>456<tab>789<tab>123<cr>
456<tab>789<tab>123<tab>456<cr>
etc...
```

To do this, select "Translation Table" from the DEFINE menu and highlight the comma character (ASCII 44) in the table and click the button marked "Translate". When the keystroke list appears, scroll the chart down to the TAB key and click the OK button to perform the translation. Click the OK button in the translation table to return to the main menu.

If you define the record structure in the Software Wedge as "Single Field Data Records" and thus pull the entire array into a single field in the Wedge and then pull the array into a variant array variable using Excel's DDERequest function and finally use the FormulaArray function to assign the variant array to a range of cells, Excel will automatically parse the data and write it to a range of cells such that each cell will contain a single value from the array. Data elements separated by tabs will be written to cells across a row and each carriage return in the data causes Excel to place data following each carriage return in the next row down.

The following example demonstrates how to set up the Wedge and Excel to pull in an entire comma delimited array (where each line of data in the array is carriage return terminated). By pulling in the entire array with a single DDERequest and letting Excel parse the data for us, you can pass huge amounts of data extremely quickly into a large range of cells.

Suppose you have a device that transmits a comma delimited array of data similar to the following where the entire array is transmitted once every ten seconds or in response to a prompt and you would like to input the data into a range of cells in Excel extremely quickly: (<cr> represents carriage returns and <lf> represents linefeed characters)

```
123, 456, 789, 123<cr><lf>
456, 789, 123, 456<cr><lf>
789, 123, 456, 789<cr><lf>
123, 456, 789, 123<cr><lf>
456, 789, 123, 456<cr><lf>
789, 123, 456, 789<cr><lf>
```

Steps for setting up the Software Wedge:

1. Select “DDE Server” from the Software Wedge “Mode” menu. When the dialog box appears asking for a DDE Command Destination Application, enter: “**EXCEL**” as the Application Name and then enter: “**SYSTEM**” as the DDE topic.

2. Select “Input Record Structure” in the “Define” menu and define the structure of the input record(s) to WinWedge. Select the "End of Record Event" as "Time Delay Between Records" and then click the "Continue" button. In the next dialog box to appear prompting for the time between records, leave the default value of 3 clock ticks and click the "Continue" button again. Next, select "Single Field Data Records" from the "Record Structure" dialog box and click the "Continue" button again. When you get to the final Window with the caption “Input Record Definition Editor”, enter the string: **[RUN("GetDataArray")]** as the *Field Postamble DDE Command* for Field(1). This is a DDE command that will be sent to EXCEL after each data array is received by the Wedge.

3. Because Excel is expecting tab delimited data with carriage returns at the end of each line, we will need to use the "Translation Table" to translate the commas to tabs. Select "Translation Table" from the DEFINE menu and select the comma character (ASCII 44) in the table and click the button marked "Translate". When the keystroke list appears, scroll the chart down to the TAB key and click the OK button to perform the translation. Finally, click the OK button in the translation table to return to the main menu.

4. Set up the rest of the Software Wedge parameters and then activate it.

Steps for setting up EXCEL:

1. Create or edit a macro module and enter the following code in the module:
(To create a new module select “Macro” and “Module” from Excel’s “Insert” menu.)

```
Sub GetDataArray()
Static StartCol as Integer, StartRow as Integer       ' Retain values between calls
' This sub performs a DDERequest for Field(1) in the Wedge and reads in a tab delimited
' array with carriage returns at the end of each line. It then fills a range of cells with the data.
' The native format for Excel data is tab delimited text with a carriage return at the end of
' each row of data.  If we assign this type of data to a range of cells using  the FormulaArray
' function, Excel automatically parses the data and fills it into the specified range.
Chan = DDEInitiate("WinWedge", "COM2")                ' Initiate dde channel
MyArray = DDERequest(Chan, "Field(1)")                ' Request field(1) from wedge
DDETerminate chan                                     ' Terminate the dde channel

' The first time through, StartCol and StartRow will be 0 - initialize StartCol and StartRow
If StartCol = 0 Then StartCol = 1   ' Set the starting column where data will go in our sheet
If StartRow = 0 Then StartRow = 1' Set the starting row

x = 1                               ' Set default x dimension to 1
On Error Resume Next                ' Ignore errors (error occurs if array has one dimension)
x = UBound(MyArray, 2) + StartCol - 1       ' Find far right column from ubound of 2nd dim
' Get upper bound of array x dimension and add offset from StartCol
On Error GoTo 0                     ' Allow errors
y = UBound(MyArray, 1) + StartRow - 1       ' Find bottom row from ubound of 1st dimension
' Get upper bound of array y dimension and add offset from StartRow
If x = 1 And y > 1 Then x = y :  y = 1      ' If array has one dimension then swap x and y

Sheets("Sheet1").Range(Cells(StartRow, StartCol), Cells(y, x)).FormulaArray = MyArray
' Fill cells in the range starting in "StartCol:StartRow" with the data array from WinWedge
' StartCol = x + 1 ' un-comment this line to move right the width of the array after each input
' StartRow = y + 1' or un-comment this line to move below the previous data after each input
' Keep the above 2 lines commented out to have each new array overwrite the previous one
End Sub
```

The example above sets up the Wedge to issue a DDE command to Excel forcing it to run the subroutine “**GetDataArray ()**” after each complete array is received from the serial device. The “GetDataArray” subroutine performs a DDERequest to the Wedge that returns the contents of FIELD(1) to a variant array variable named “MyArray”. The entire array is then assigned to a range of cells in “Sheet1” using the FormulaArray function. The example above assumes that the open workbook contains a worksheet named **Sheet1**.

Note: This subroutine will also work with non array type data (i.e. single data values).
Single data values are actually equivalent to a single dimension array containing a single data element. Thus you could use this example in place of Example #3 on pg. 61.

Example #7 - Continually Polling Multiple Devices In Sequence or Sending Multiple Polls to a Single Device (Excel 5 - Excel 2000)

Suppose you had a group of devices connected via a multi-drop RS422 or RS485 line such that each device responded to a prompt by sending back a record of data. Suppose also that you wanted to set up the Wedge and Excel to poll each device in sequence and retrieve the data placing the readings from each device in separate columns in your spreadsheet. In a multi-drop situation you typically have to wait for each device to respond to its prompt before you can send the next prompt to the next device. The following instructions and macro examples show how to handle a situation like this.
Note: The following example can also be used to send multiple prompts to a single device over an RS232 connection and thus retrieve multiple data elements from the device.

Steps for setting up the Software Wedge:

1. Select “DDE Server” from the Software Wedge “Mode” menu. When the dialog box appears asking for a "DDE Command Target Application", clear out both text boxes prompting for an Application Name and a DDE topic.
Note: In this example we will not be configuring the Wedge to send a DDE Command to Excel, therefore these two items are not necessary and can be left blank.

2. Select “Input Data Record Structure” in the “Define” menu and define the structure of the input data record(s) to WinWedge. When you get to the final Window with the caption “Input Record Definition Editor”, make sure that you do not have any Field Postamble DDE Commands defined.

3. Set up the Wedge to transmit the first prompt for the first device that you want to poll using either a timer controlled output string or a button controlled output string - Select "Output Strings" from the DEFINE menu. If you use a timer controlled output string, set the timer interval to a value that is high enough to guarantee that all devices will be polled before the first prompt gets sent again. Also, do not check the option "Enable Timer on Activation" - we will activate the timer later, after we are finished setting up Excel.

4. Set up the rest of the Software Wedge parameters and then activate it.

Steps for setting up EXCEL:

1. Create or edit a macro module and enter the following code in the module:
(To create a new module select "Macro" and "Module" from Excel's "Insert" menu.)

```
Global Const MyPort="COM1"      ' Change port to match configuration of WinWedge
Global Const MaxPrompts% = 3                    ' Number of prompts or devices
Dim RowPointer(MaxPrompts%) As Long ' Create an array of RowPointers
Dim Prompt$(MaxPrompts%)                        ' Create an array of prompts

Sub GetSWData()
'  The variable PNum keeps track of which prompt was sent last  - start with device 1
Static PNum As Long             ' Preserve the value of PNum between calls
If PNum = 0 Then PNum = 1       ' Set the initial value of PNum to 1
' Define the specific prompts for each device - add a line for each prompt
' See the Wedge manual for details about the SENDOUT command
' WinWedge will send the first prompt so we do not need to define it here
Prompt$(2) = "[SENDOUT('PString2',13,10)]"           ' Prompt for device 2
Prompt$(3) = "[SENDOUT('PString3',13,10)]"           ' Prompt for device 3
For x%= 1 to MaxPrompts%                             ' Initialize RowPointer array
     If RowPointer(x%) = 0 Then RowPointer(x%) = 2   ' Start saving data in row 2
Next
On Error Resume Next                                 ' Ignore errors
Chan = DDEInitiate("WinWedge", MyPort)          ' Open a link to WinWedge
MyData = DDERequest(Chan, "FIELD(1)")                ' Get data from field(1)
' The above line retrieves the response from the last prompt that was sent
MyDataString$ = MyData(1)       ' Convert variant array type to a string variable
' The data from Field(1) of WinWedge is now in the variable "MyDataString$"
Sheets("Sheet1").Cells(RowPointer(PNum), PNum).Formula = MyDataString$
' The above line writes the data to column "PNum" in row "RowPointer(PNum)"
RowPointer(PNum) = RowPointer(PNum) + 1         ' increment RowPointer(PNum)
' PNum indicates which prompt requested the data that we just got
' Add code here to do further processing of the data if required.
Select Case Pnum
   Case 1   ' Do something with data from device #1 here if desired
   Case 2   ' Do something with data from device #2 here
   Case 3   ' Do something with data from device #3 here
End Select
PNum = PNum + 1            ' Increment Prompt Number - count from 1 to MaxPrompts%
If PNum > MaxPrompts% Then      ' If PNum>MaxPrompts% then loop back to 1
 PNum = 1                       ' and quit - Wedge will send prompt for first device
Else                            ' Otherwise send the prompt for the next device
  DDEExecute Chan, Prompt$(PNum)                ' Send the next prompt
End If
DDETerminate Chan                               ' Terminate the link
End Sub
```

```
Sub SetUpDDE()
Sheets("Sheet1").Activate           ' Activate sheet 1 and set up a DDE link to WinWedge
Sheets("Sheet1").Cells(1, 50).Formula = "=WinWedge|" & MyPort & "!'Field(1)' "
ActiveWorkbook.SetLinkOnData "WinWedge|" & MyPort & "!'Field(1)' ", "GetSWData"

'  The SetLinkOnData method causes Excel to run the GetSWData macro automatically
'  when new data is available in the Wedge (when Field(1) is updated).
'  The SetLinkOnData method in Excel eliminates the need to have the Wedge send a
'  Field Postamble DDE Command to Excel to cause it to run the GetSWData subroutine.
'  Excel will automatically run the GetSWData sub when the Wedge updates the
'  Field(1) DDE item.

'  If you have more than one field defined in the Wedge, use the last data field to trigger
'  the SetLinkOnData method. This will insure that all data fields are available when the
'  GetSWData subroutine runs.

End Sub
```

After entering the two subroutines above, run the subroutine "SetUpDDE" to establish a DDE link between Excel and the "Field(1)" DDE item in the Wedge.

After Excel is set up and you are ready to start prompting for data you can enable the timer in the Wedge (select "Enable Timer" from the "Quit" menu in the Wedge). What is going on here is that the Wedge will send the first prompt to the first device causing it to send back a response. When each response comes back, the Wedge updates the "Field(1)" DDE item in Excel forcing it to run the macro above (we used the SetLinkOnData method in the SetUpDDE subroutine to cause this to happen). The first thing the subroutine does is pull in the data from the Wedge and place it in a column using the "PNum" variable as the column selector. For example, if PNum is equal to one, the data is written to column A. If PNum is equal to two then the data is written to column B, etc.. Next, the macro increments a "PNum" variable and checks its value. If PNum is greater than the maximum number of prompts then it simply resets to 1 and quits; otherwise it transmits the next prompt in the sequence. When the response to this prompt comes in, the process repeats itself until we receive the data in response to the last prompt that we have defined and PNum is incremented past the number of prompts. The timer controlled output string in the Wedge starts the whole process all over again by prompting for data from the first device.

Note: You do not have to use a timer controlled output string. To kick off a the sequence, simply transmit the prompt for the first device. You can do this from Excel by writing a macro that sends a DDE command to the Wedge telling it to send the first prompt or you can define a button controlled output string in the Wedge that would transmit the first prompt.

See Also: Defining Serial Output Strings (pg. 38)

DDE Examples for Microsoft Access

Example #1 - Collecting Data Directly To A Table In An Access Database

Steps for setting up the Software Wedge:

1. Select “DDE Server” from the “Mode” menu in the Wedge. When the dialog box appears asking for a DDE Command Target Application, enter: “**MSACCESS**” as the Application Name and the name of your open database (without the MDB Extension) as the DDE topic.

2. Select “Input Record Structure” in the “Define” menu and define the structure of the input data to WinWedge. When you get to the final window entitled “Input Record Definition Editor”, enter the string: **[GrabData]** as the Field Postamble DDE Command after the last data field that you have defined. This is a DDE command that will be sent to MSACCESS after each data record is received by the Wedge.

3. Set up the rest of the Wedge as needed for your serial device and then activate it.

Steps for setting up MS ACCESS:

MSAccess does not have a “GrabData” DDE command so the first step is to create one. When you define a macro in an Access database, the name of the macro automatically becomes a valid MSAccess DDE command for as long as the database is open.

1. Create a new macro that performs the “RunCode” Action. For the function name that is required by the RunCode action, enter the name “**GetWedgeData()**”. Save the macro with the macro name “**GrabData**” and finally, create a new module with a function named “GetWedgeData()” and enter the following commands in the body of the function:

```
Function GetWedgeData ()
Dim Chan, MyData As Variant                          ‘Create a variable
Chan = DDEInitiate("WinWedge", "Com1")               ‘Open a link to WinWedge
MyData = DDERequest(Chan, "FIELD(1)")                ‘Get data from field(1)
DDETerminate Chan                                    ‘ Terminate the link
If Len(MyData) = 0 Then Exit Function                ' If no data then quit

' Data from Field(1) in WinWedge is now in the variable “MyData”. The following code adds
' a new record to TABLE1 and stores the data in a field named "SerialData"
Dim MyDB as Database, MyTable As Recordset
Set MyDB = DBEngine.Workspaces(0).Databases(0)
Set MyTable = MyDB.OpenRecordset("TABLE1", DB_OPEN_TABLE)
MyTable.AddNew           ' Add a new record - the new record becomes the current record
MyTable("SerialData") = MyData        ' Write our data to a field named "SerialData"
MyTable.Update                        ' Update the table
MyTable.Close                         ' Close the table
End Function
```

The example above sets up the Wedge to issue a DDE command consisting of the name of an Access macro (directly to the Access database that contains the macro) after each data record is received from your serial device. The macro calls an AccessBASIC function that performs a DDERequest back to the Wedge and returns the contents of FIELD(1) to a variable named “MyData”. It then creates a new record in a table and inserts the data into the record.

Example #2 - Launching And Terminating The Software Wedge From Access

```
Function LaunchWedge ()
' This function will launch the Wedge feeding it the name of a configuration file on the
' command line causing the Wedge to automatically load the config file and activate itself
CmdLine = "C:\winwedge\winwedge.exe C:\winwedge\test.SW1"
' Change "CmdLine" to specify the correct path for your copy of WinWedge.Exe
' Make sure that the complete path is specified for your configuration file as well
RetVal = Shell(CmdLine)
x = Now + TimeValue("00:00:02") ' Wait for 2 seconds
Do While Now < x                ' Give wedge time to load and activate itself
    DoEvents                    ' Allow other Windows processes to continue
Loop
If RetVal = 0 Then                                 ' Launch failed
  Beep : MsgBox ("Cannot Find WinWedge.Exe")       ' Warn user and exit
  Exit Function
Else                                               ' Launch succeeded
    AppActivate "Microsoft Access"                 ' Set the focus back to Access
End If
End Function

Function KillWedge ()                     ' This function unloads the Wedge from memory
chan = DDEInitiate("WinWedge", "Com1")' Initiate DDE channel with wedge on COM1
DDEExecute chan, "[AppExit]"              ' Send AppExit command to Wedge
' No need for a DDETerminate statement here because Wedge is no longer running
End Function
```

Example #3 - Transmitting Variables Out The Serial Port From Access

```
Function SendString (StringVar$)
' The Wedge must be running and activated for COM1 for this code to work
chan = DDEInitiate("WinWedge", "Com1")' Initiate DDE channel with wedge
DDEExecute chan, "[SEND(" + StringVar$ + ")]"     ' Send String out the serial port
DDETerminate chan                                  ' Terminate the link
End Function
```

For more examples of how to transmit data, refer to the examples for Microsoft Excel VBA on pg. 58-59. Excel VBA and AccessBasic are almost identical macro languages.

DDE Examples for Microsoft Word for Windows

Steps For Setting Up The Software Wedge:

1. Select “DDE Server” from the MODE menu and enter: **WinWord** as the Application Name and **System** as the DDE topic in the dialog box that appears and click the OK button.

2. Select “Input Data Record Structure” from the DEFINE menu and define the structure of the input records to the Wedge. When you get to the final Window entitled “Input Record Definition Editor”, enter the string: **[GRABDATA]** as the "Field Postamble DDE Command" after the last data field that you have defined. This is a DDE command that will be sent to Word after each data record is received by the Wedge.

3. Set up the rest of the Software Wedge parameters and activate it.

Steps For Setting Up Word:

When you create a WordBasic macro or subroutine in Word and save it in the Global macro sheet, the name of the subroutine automatically becomes a valid Word DDE command. Create a new macro by selecting "Macro" from the TOOLS menu. When the dialog box appears prompting for a macro name, enter the name: GRABDATA and then click the button labeled "Create". Enter the following code in the macro editor window:

```
Sub MAIN
chan = DDEInitiate("WinWedge", "COM2")           ' Open a link to the Wedge
MyData$ = DDERequest$(chan, "Field(1)")' Get the data from the Wedge
DDETerminate chan                                ' Close the link
Insert MyData$                                   ' Enter the data into the open document
InsertPara                                       ' Insert a paragraph (hard return)
                    ' Add your own code here
End Sub
```

After you are finished entering the code, select "Close" from the FILE menu to save the macro. This example sets up the Wedge to issue a DDE command (**[GRABDATA]**) to Word after each data record is received from your serial device. This causes the **GRABDATA** subroutine to run which then performs a DDERequest back to the Wedge asking for the data in Field(1) in the Wedge Window. This data is returned to Word in the variable: MyData$. Once the data from the Wedge is in a variable, you can do whatever you like with it by including additional code of your own design. This example simply inserts the text in the MyData$ string variable into the open document and then inserts a paragraph marker.

The following code fragment shows how to transmit a string out the serial port from Word:

```
channel = DDEInitiate("WinWedge", "COM2")        ' Open a link to the Wedge
DDEExecute channel, "[Sendout('Test',13)]"       ' Transmit "Test" and a carriage return
DDETerminate channel                             ' Close the link
```

DDE Examples for Visual FoxPro for Windows

Example #1 - Collecting data into MS FoxPro using the Software Wedge.

Steps For Setting Up The Software Wedge:

1. Select “DDE Server” from the Software Wedge MODE menu. When the dialog box appears asking for a DDE Command Destination Application, enter: **MyServer** as the Application Name and enter **MyTopic** as the DDE topic.

2. Select “Input Data Record Structure” from the DEFINE menu and define the structure of the input records to WinWedge. When you get to the final Window entitled “Input Record Definition Editor”, enter the string: **[GRABDATA]** as the "Field Postamble DDE Command" after the last data field that you have defined. This is a DDE command that will be sent to your FoxPro application after each data record is received by the Wedge.

3. Set up the rest of the Software Wedge parameters and activate it.

Steps For Setting Up FoxPro:

FoxPro does not have a “GrabData” command so we must create one. The FoxPro code example on the following page does exactly that. Create a new program module in the CODE section of FoxPro and enter the code on the following page. After you are finished entering the code, save your work and run the program.

What this example does is to set up the Software Wedge to issue a DDE command **[GRABDATA]** to your FoxPro application after each data record is received from your serial device. This causes the procedure **DoCommand** to run which then performs a DDERequest back to the Wedge asking for the data in Field(1) in the Wedge Window. This data is then returned to FoxPro in the variable: MyData. Once the data from the Wedge is in a variable, you can do whatever you like with it by including additional code of your own design.

```
*** Set this application up as a DDE server. The following code must be run at the start***
*** of your application to make it able to respond to DDE commands.***
*** Set the DDE application name to: "MyServer" and Enable DDE command execution***

= DDESetService('MyServer', 'DEFINE')
= DDESetService('MyServer', 'EXECUTE', .T.)
*** Set the DDE topic to "MyTopic" ***
*** Set up procedure "DoCommand" to run on all DDEExecutes to topic "MyTopic"***
= DDESetTopic('MyServer', 'MyTopic', 'DoCommand')



*** The following procedure will run whenever another application***
*** issues a DDE command to application name 'MyServer' and topic 'MyTopic'***
*** The DDE command is passed in the variable "gData"***

PROCEDURE DoCommand
PARAMETERS gnChannel, gcAction, gcItem, gData, gcFormat, gnAdvise
glResult = .F.
*** It's necessary to return .T. from an INITIATE action or no connection is made ***
IF gcAction = 'INITIATE'
    glResult = .T.
ENDIF
IF gcAction = 'EXECUTE'

*** Support a GRABDATA command that will perform a DDERequest to WinWedge***
*** and request the data from FIELD(1) and stores it in a variable: "MyData"***

    IF gData="[GRABDATA]"
      gnChanNum = DDEInitiate('WinWedge', 'COM2')
      IF gnChanNum != -1
        MyData=DDERequest(gnChanNum,'Field(1)')
        = DDETerminate(gnChanNum)     && Close the channel
*** The variable 'MyData' now contains the data from Field(1) in WinWedge***
*** Your code goes here to do something with it ***
        WAIT WINDOW MyData NOWAIT  && For Example - Display the data
      ENDIF
    ENDIF
ENDIF

IF gcAction = 'TERMINATE'
    glResult = .T.
ENDIF
RETURN glResult
```

Example #2 - Transmitting Data Out The Serial Port From FoxPro

```
*** the following procedure demonstrates how to transmit a string out the serial port
*** by sending a DDE command from FoxPro to WinWedge

PROCEDURE TransmitTesting

CmdString="[SENDOUT('testing 123',13)]"
*** transmit the string "testing 123" followed by a carriage return (ascii 13)
gnChanNum = DDEInitiate('WinWedge', 'COM2')
IF gnChanNum != -1
 glExecute = DDEExecute(gnChanNum, CmdString)
 = DDETerminate(gnChanNum)     && Close the channel
ENDIF
RETURN



*** the following procedure is a variation of the above procedure that transmits a
*** string passed as a parameter to the procedure (gData)


PROCEDURE TransmitData
PARAMETERS gData
cmdstring="[SENDOUT(' " & gData & " ' )]"
*** transmit the string passed in the variable gData
*** note the use of both single and double quotes around the gData variable
*** refer to the syntax for the SENDOUT command in the Wedge manual for details
gnChanNum = DDEInitiate('WinWedge', 'COM2')
IF gnChanNum != -1
 glExecute = DDEExecute(gnChanNum, CmdString)

 = DDETerminate(gnChanNum)     && Close the channel
ENDIF
RETURN
```

DDE Examples for Microsoft Visual Basic for Windows

Linking a VB Text Box to a Software Wedge Data Field

To link a data field in the Wedge to a text box in a VB application, set the properties for the text box to specify a DDE link. For example the following code would establish a "Hot" link between field(1) in the Wedge (active on COM2) so that when data changes in Field(1), it would also automatically change in the text box Text1. When a field in the Wedge is linked to a text box on a VB form using a "Hot" link, the text box will receive a "Change" event whenever the data in the Wedge changes. If you set the LinkMode property to "Notify", the text box will receive a "LinkNotify" event when the Wedge updates the data for Field(1) - even if the data does not change. Note: If you set the LinkMode to "Notify" your Text1_LinkNotify event handler will have to perform a "LinkRequest" method to update the textbox. (i.e. Text1.LinkRequest)

```
Hot = 1 : Warm = 2 : Notify=3 : None = 0
Text1.LinkTopic = "winwedge|com2"          ' Set the link topic
Text1.LinkItem = "field(1)"                ' Set the link item
Text1.LinkMode = Hot                       ' Set the link mode to "Hot"
```

Sending DDE Commands To The Software Wedge From A VB Application.

After a link is established between a text box and a data field in the wedge, you could issue commands to the Wedge using the text box "LinkExecute Method". For Example the following code sends the string "Testing" followed by a carriage return out the serial port.
Note: The Wedge must be active on the specified serial port for the following code to work.

```
Hot = 1 : Warm = 2 : Notify=3 : None = 0
Text2.Linkmode = None                      ' Disable links before setting topic
Text2.LinkTopic = "winwedge|com2"          ' Set the link topic
' There is no need to set the LinkItem property when just sending DDE commands
Text2.LinkMode = Warm                      ' Set the link mode to "Warm"
Text2.LinkExecute "[Sendout('Testing',13)]"  ' Send the string out the serial port
```

Launching The Software Wedge From A Visual Basic Application

```
Sub RunWedge()
' The shell statement passes a wedge config file as a command line arg.
' This causes the wedge to automatically load the config file and activate itself.
x% = Shell("c:\WinWedge\WinWedge.exe c:\WinWedge\DDETest.SW1")
TwoSecsFromNow = TimeValue(Now) + TimeValue("00:00:02")
Do While TimeValue(Now) < TwoSecsFromNow
 DoEvents                        ' Give WinWedge time to load
Loop
AppActivate Me.Caption           ' Set focus back to the VB application
End Sub                          ' Wedge should be loaded now
```

DDE Example for Lotus 123 for Windows

The following is an example of two Lotus 123 macros that work together to take data readings from a DDE conversation with the Software Wedge and place them in a column such that each new reading is added to the bottom of the column.

```
Start_Wedge  {LET A1,1}
             {LAUNCH “C:\WinWedge\WinWedge.Exe C:\WinWedge\MyConfig.SW1”}
             {DDE-OPEN "WinWedge","COM1"}
             {DDE-ADVISE "Get_Data","Field(1)"}
             {DDE-EXECUTE "[BEEP]"}

Get_Data     {DDE-REQUEST ("B"&@STRING(A1,0)),"Field(1)"}
             {LET A1,A1+1}
```

The macro named “Start_Wedge” first sets up a counter value (equal to one) in cell A1 and then launches the Software Wedge so that it will automatically load and activate itself using the configuration file named “MyConfig.SW1”. Next, it opens a DDE conversation between Lotus and the Software Wedge using the DDE-OPEN macro command. The following DDE-ADVISE statement sets up Lotus to run the second macro “Get_Data” each time the data in the RecordNumber DDE item in the Software Wedge changes. The last line in the “Start_Wedge” macro sends the DDE command “[BEEP]” to the Software Wedge and is included here as an example of how to issue DDE commands to WinWedge from Lotus.

If the Start_Wedge macro is successful, then the second macro named “Get_Data” will run automatically whenever the RecordNumber DDE item in the Software Wedge is incremented (i.e. after each new data record is received). The DDE-REQUEST command in the Get_Data macro simply retrieves the data from Field(1) in the Software Wedge and places the data in column B in the row pointed to by the value in cell A1. The line {LET A1, A1+1} increments the value in cell A1 so that the next time through, the data from Field(1) will be placed in the cell directly below the last data value.

To use this example, you would simply run the macro "Start_Wedge" one time when you want to start collecting data. This macro assumes that the Software Wedge will be activated for serial port COM1.

DDE Examples for Quattro Pro for Windows

Steps for setting up the Software Wedge:

1. Select “DDE Server” from the Software Wedge “Mode” menu. When the dialog box appears asking for a DDE Command Target Application, enter: “**QPW**” as the Application Name and then enter “**SYSTEM**” as the DDE topic.

2. Select “Input Record Structure” in the “Define” menu and define the structure of the input record(s) to WinWedge. When you get to the final Window entitled “Input Record Definition Editor”, enter the string: **{Get_Wedge_Data}** as the Field Postamble DDE Command after the last data field that you have defined. This is a DDE command that will be sent to Quattro Pro after each data record is received by the Wedge.

3. Set up the rest of the Software Wedge parameters and then activate it.

Steps for setting up Quattro Pro:

Quattro Pro does not have a “Get_Wedge_Data” DDE command so the first step is to create one. When you create and name a macro in a Quattro Pro notebook, the name of the macro automatically becomes a valid Quattro Pro DDE command for as long as the notebook is open.

1. Enter the following lines into cells a Quattro Pro notebook and name each cell or group of cells using the block names indicated below.

Block Name	Block Contents
row	**1**
column	**A**
cell_number	**A1**
dde_channel	**0**
dde_command	**[beep]**
result	**0**

get_wedge_data	**{ONERROR _error_handler}**
	{REQUEST dde_channel,"FIELD(1)",@@("cell_number")}
	{LET row,row+1}
	{LET cell_number,+column&@string(row,0)}
	{EXECUTE dde_channel,+dde_command,result}

_error_handler	**{INITIATE "WinWedge","COM1",dde_channel}**

This example sets up the Software Wedge to issue a DDE command to Quattro Pro consisting of the name of a macro (get_wedge_data) after each data record is received from your serial device. The macro performs a DDE Request to the Software Wedge that returns the contents of FIELD(1) to a spreadsheet cell pointed to by the variables named “row” and “column”. (i.e. row 1, column A).

The blocks with the names “row”, “column”, “cell_number”, “dde_channel”, “dde_command” and “result” are simply variables that are used by the “get_wedge_data” macro. The “get_wedge_data” macro first sets up an error handler pointing to the macro named “_error_handler”. The following line then attempts to request the data for “Field(1)” from the Software Wedge and store the data in a cell whose address is pointed to by the variable “cell_number”. The first time the macro is invoked the “request” command will fail because a DDE channel has not yet been opened. The purpose of the “_error_handler” macro is to simply initiate the DDE channel when the first call to “get_wedge_data” fails. After the “request” macro command, the variable “row” is incremented and the variable “cell_number” is updated using the new row number so that it points to the cell directly below the one where the last data reading from the Software Wedge was stored. Thus, each time a new data record is received by the Software Wedge, the contents of “Field(1)” will be retrieved and stacked up in a column. The final macro line in the “get_wedge_data” macro uses Quattro Pro’s “Execute” command to issue a “Beep” command to the Software Wedge. This line has been included as an example of how to send DDE commands to WinWedge from Quattro Pro.

Passing Data Across a Network Using Network DDE

One of the more interesting features of Windows for Workgroups, and all 32 bit versions of Windows is that they all support a feature called "Network DDE". Network DDE allows DDE data to be passed across a network from one PC to another. This means that it is possible to have the Software Wedge running on one or more PCs providing real time data to an application running on an entirely different PC in the network. Note: Network DDE must be enabled in all 32 bit versions of Windows by running the program NETDDE.EXE. You may want to place the NETDDE program in your StartUp folder so that it runs automatically when you start Windows.

The process of setting up a Network DDE connection is fairly simple and involves using the ClipBook application found in the MAIN program group in Windows. The ClipBook extends the concept of the Windows Clipboard across a network. The ClipBook allows you to create "ClipBook Pages" that can be "Shared" with other computers on a network. Once you create and share a page on one computer, other computers on the network can "connect" to that page. If the page contains data from a DDE server (like the Wedge), then you can set up DDE links to the data and the data will be passed to any client application on the network just as if the client were running in the same local PC as the server application.

Suppose you have the Wedge running in one PC and you want to DDE link data from "Field(1)" in the Wedge to a cell in an Excel spreadsheet running in another PC on your network. The following steps outline the procedure:
Note: All versions of Excel prior to 5.0c have a bug that causes Network DDE connections to fail. If you have an earlier copy of Excel you can get an update from Microsoft for free.

1. Load and activate the Wedge on the first PC. When the Wedge window appears, place the cursor in input data field "Field(1)" and select "Copy" from the EDIT menu.

2. Open the ClipBook program in the PC where the Wedge is running and select "Local ClipBook" from the WINDOW menu and then select "Paste" from the EDIT menu. In the dialog box that appears, type **MyPage** in the text box labeled "Page Name" and make sure that the check box labeled "Share Item Now" is checked.

3. Open up the ClipBook program on the second PC where Excel will be run and select "Connect..." from the FILE menu. When the dialog box appears, enter the network name of the PC where the Wedge is running and click the OK button. (If you do not know the name of the computer, click the "Browse" button and the ClipBook will display a list of all computers in the Network). This will display a new window that will contain all the "shared" items in the ClipBook running on the other PC. You should see an item in the window with the name "MyPage" and then select "Copy" from the EDIT menu. This operation takes the data from the ClipBook on the other PC and places it in the local clipboard.

4. Open Excel and place the cursor in the cell that you want to link the data to and select "Paste Special" from Excel's EDIT menu. In the dialog box that appears, check the option labeled "Paste Link" and then click the OK button. If the link is successful, you will see the data from Field(1) in the cell and you will also see the formula for the link in the formula bar. Whenever new data is received by the Wedge it will be updated in the spreadsheet cell.

If you look at the formula that Excel returns from the above Copy / Paste Link procedure, it will look something like the following:

=\\ComputerName\NDDE$|$MyPage.dde!Field(1)

The formula for a DDE link in Excel is expressed using the DDE Application Name followed by a pipe character (|), a DDE Topic followed by an exclamation mark and then the DDE Item Name.

If you were to use the Copy / Paste Link method of establishing a DDE link between the Wedge and Excel where both programs were running on the same PC, the linked cell in Excel would contain a formula similar to the following:

=WinWedge|COM1!Field(1)

The difference between a local DDE connection and a Network DDE connection is that instead of using the DDE Server's executable file name, the Network DDE Application Name becomes two slashes followed by the name of the computer running the DDE Server program followed by a single slash and the word: **NDDE$**. The DDE Topic also changes to a dollar sign followed by the "Page Name" that was assigned in the ClipBook application followed by a period and the word **dde**. The DDE Item Name stays the same.

After you create a "Page Name" using the ClipBook application and paste data from a DDE server into the page, the page name and the DDE connection data contained in it will continue to exist even after you close the ClipBook application on both computers. You can even shut down Windows on both computers and when you restart you will be able to re-establish the DDE links using the same Network DDE Application, Topic and Item Names.
In other words, after you set up a Network DDE connection once, you do not need to go through the same procedure in the ClipBook every time you want to re-establish the link.

If you wanted to send a DDE command from Excel to a copy of the Wedge running on a remote PC, all you need to do is to change the DDE Application Name and DDE Topic to the new Network DDE Application and Topic Names. For example the following Excel macro could be used to send the "BEEP" command to a Wedge on a remote PC. (We will assume that the name of the remote PC is "Bob" and that a ClipBook page was created on that PC using the name "MyPage" and that the page contains a DDE data field from the Wedge.)

```
Sub BeepRemoteWedge()
chan = DDEInitiate("\\BOB\NDDE$", "$MyPage.DDE")
DDEExecute chan, "[BEEP]"
DDETerminate chan
End Sub
```

Diagnosing Serial Communications Problems

Step1: Select **Settings** from the **Port** menu in the Wedge and choose the communications parameters required for your serial device. **Important:** All serial communications parameters should exactly match the settings for your device. If you do not know the correct parameters, then you will have to contact the manufacturer of the device to obtain this information.

Step 2: Open the Analyze window in the Wedge by selecting **Analyze** from the **Port** menu. If you get an error message that reads "Device COMn Not Available" then either the specified serial port is currently being used by another program (possibly another instance of the Wedge), the port does not exist, or the serial adapter itself or the Windows COM Driver for the serial adapter have not been installed or configured correctly in the Windows Control Panel. If you suspect that the port may be configured incorrectly in the Control Panel, then refer to the documentation that came with your PC (or the documentation that came with your serial adapter) for information on how to correct or work around the problem. You may need to contact the manufacturer of the PC to troubleshoot a hardware problem.

Step 3: If the Analyze window appears, transmit some data from your serial device to the Wedge. The data from the device should appear in the text box marked "Input Buffer". If no data appears in the input buffer then either your instrument is not connected to your PC using the correct serial cable; you are not connected to the serial adapter that you specified in the port settings dialog box or your device is not transmitting any data.

Try plugging the cable from the device into a different serial port and transmitting data again to rule out the possibility that you are plugged into the wrong serial port.

If you are certain that the device is transmitting yet you still receive no data in the Analyze Window, then you may need a 'Null Modem" adapter or a different serial cable. If you normally connect your device to a serial printer and are able to print successfully, then you definitely need a Null Modem adapter. A Null Modem adapter connects between a serial cable and your computer and crosses the transmit and receive lines in the cable so that the transmit line from your serial device is connected to the receive line on your PC and vice versa. Null Modem adapters are available at most computer supply stores for under five dollars.

If your PC beeps one or more times whenever you try to read data into the Input Buffer in the Analyze window or if data appears in the Input Buffer but part or all of it is either garbled or consists of unreadable characters, then you are connected to the correct port however the communications parameters that you chose in the Port Settings dialog box are wrong and must be corrected.

Note: The Wedge has an option "Beep On Input" (located in the "Port" menu) that configures the Wedge to beep your PC's speaker whenever it receives a data record through the serial port. The "Beep On Input" option does not apply to data received in the Port - Analyze window. If your PC beeps while inputting data into the Port Analyze window then it is an indication of a serial communications error; typically either a mismatch in serial communications parameters (baud rate, parity, databits, etc.) between the Wedge and your device or a serial buffer overrun.

Troubleshooting

<u>If you are not receiving any data all in the Port Analyze Window:</u>

1. Make sure that the PC you are using is equipped with the serial port selected in the Port Settings dialog box. If you do not have the chosen serial port or if your serial port is configured improperly then you must correct the problem and try again.

Important: Make sure that the Port Address and IRQ number in the Advanced section of the PORTS dialog box in the Windows Control Panel are correctly specified for your serial adapter hardware. The standard values for COM1 through COM4 are shown below:

Connector	Port Address	IRQ
COM1	03F8	4
COM2	02F8	3
COM3	03E8	4
COM4	02E8	3

See Also: Running the Wedge On Multiple Serial Ports Simultaneously (pg. 88)

2. Check that you are connected to your PC using the proper RS232 cable for connection to a PC and that you are plugged into the serial adapter that you specified in the Port Settings dialog box. Also make sure that your serial device is getting power and is turned on. You may also want to contact the manufacturer of the device to confirm that you are using the correct cable and that you are doing everything that is necessary to connect to a PC and also to get the device to transmit data.

3. Use the Analyze window in the Software Wedge and try, at least, to get some data from the device to appear in the Input Buffer text box. If you cannot get any data at all to appear (even garbage) then either the device is not transmitting anything, or you need a "Null Modem" adapter. A Null Modem adapter is a small plug that connects between the serial cable from your device and serial port on your PC. Its purpose is essentially to cross the transmit and receive lines in the cable so that the transmit line from your serial device is connected to the receive line on your PC and vice versa. If you normally connect your serial device to a printer and you can successfully print to the printer, then you definitely need a Null Modem adapter. You can pick one up at any computer supply store for about five dollars. Take the device or the cable to the store with you so that you can match the adapter to the cable.

4. The Analyze Window in the Wedge is similar to the Terminal program that comes with Windows 3.1 or the HyperTerminal program that comes with Windows 95 (located in your Accessories program group). The manufacturer of your serial device may not be familiar with the Software Wedge however they will more than likely be familiar with either Terminal or HyperTerminal. A good course of action at this point would be to contact the manufacturer of your serial device and have one of their technical support representatives walk you through the process of connecting the device to the Terminal or HyperTerminal program. Once you are able to get data into Terminal or HyperTerminal you should have no trouble doing the same thing using the Analyze window in the Wedge.

<u>If you are receiving garbled or unreadable data in the Analyze window:</u>

5. Check that the serial device is set up using exactly the same communications parameters as the Wedge, i.e. baud rate, parity, number of data bits, and number of stop bits. If you do not know the parameters used by your device, then you will need to either consult the users manual for the device or contact its manufacturer for this information. You can also use the Port Analyze feature in the Software Wedge to try different parameters until you get data that appears correct. Try different baud rates first using No Parity, Eight Data Bits, and One Stop Bit until you get data that looks either partially or completely correct. Finally, try different combinations of Parity, Number of Data Bits and Number of Stop Bits until all data is correct. Most devices use either seven data bits with Even or Odd parity or eight data bits with No Parity. One stop bit is also used more frequently than two.

<u>If data appears using the Port Analyze feature but not after you activate the Wedge:</u>

6. Make sure that the Wedge is activated and enabled and try reading data into the NotePad application that is shipped with Windows. If no data appears in NotePad, then you most likely have the Wedge configured incorrectly (either the "End Of Record Event" is wrong or your parsing and filtering parameters are specified incorrectly). If you are inputting very large data records, you may also need to increase the size of the serial input buffer. Re-configure the Wedge and try again.

<u>If your computer locks up when you open the Analyze window or activate the Wedge:</u>

Microsoft has confirmed that there is a bug in an early release of the Windows 3.x COM driver file: SERIAL.386. The bug causes a system crash when you try to activate a serial port and typically occurs on an older 486 or Pentium PC with 16550 UARTs (high speed serial adapters). The bug has been repaired in all copies of Windows that shipped after March of 1994. If you have an older version of the SERIAL.386 driver dated before March of 1994, you should replace it with the latest release. The newest version of the driver can be found on the Software Wedge distribution diskette in a self extracting ZIP file named WG1001.EXE.

To extract the new driver, copy the WG1001.EXE file into a temporary directory on your hard drive and then run it to un-compress the files that it contains. When you un-compress the WG1001.EXE file, a README file will also be un-compressed along with the new SERIAL.386 driver. This README file contains complete instructions for replacing the defective driver as well as some technical notes from Microsoft regarding the symptoms of the bug.

If you cannot get the Wedge to operate properly after you have gone through the above procedures, then call or fax TAL Technologies for assistance at: Tel: (215)-763-5096 or Fax:(215)-763-9711. Questions can also be sent via e-mail to: **support@taltech.com**.
Please have this manual and your original Software Wedge diskette(s) at hand and, if possible, be at your PC with the Wedge running and your serial device connected when you call for support. You can also find technical support information in the tech support section of our Internet home page at: **http://www.taltech.com**.

Running The Wedge On Multiple Serial Ports Simultaneously

The Software Wedge may be activated for more than one serial port simultaneously. To do so, simply launch the Wedge twice and activate each *instance* of the Wedge using configuration files that were set up for different serial ports.

Using More Than Two Serial Adapters At A Time

If you will be using the Wedge on a PC that is not a Micro Channel or EISA Bus PC and more than two serial ports will be active at the same time, then you may have to change the configuration of your serial adapters for ports COM3 and above.

The standard bus architecture (ISA bus) for all IBM or compatible 80x86 PC's provide dedicated interrupt request lines for only two serial adapters (COM1 uses IRQ4 & COM2 uses IRQ3). Many serial adapters that provide COM3 & COM4 also use the same IRQ lines as COM1 & COM2. When interrupt driven serial I/O is used, as by the Wedge, then two active serial ports will end up fighting with each other for control of the shared interrupt request line and both will lose. One adapter will try to drive the IRQ line "high" while the other adapter tries to drive it "low". EISA and Micro Channel bus PC's avoid this problem by using an edge triggered interrupt control mechanism instead of the original level triggered scheme used in the ISA bus.

One way around this problem is to take advantage of an unused IRQ line on your PC (usually IRQ 5, 7,10,11,12 or 15) by configuring your COM3 or above adapter's jumper or switch settings to use one of these IRQ lines instead of IRQ4 or IRQ3. Most add-on serial adapters provide jumpers or switches just for this purpose. This must be done with care because many other peripheral devices including mice, network cards, internal modems and sound cards may also use these IRQ lines. You can check to see which IRQ lines are free by using the Microsoft Diagnostics program that comes with DOS and Windows. Exit Windows and type: **MSD** at the DOS prompt and when the menu appears, select "IRQ Status". IRQ lines that are available will be specified as "Reserved". If you are running Win95, consult the manufacturer of the add on serial adapter that you are using for instructions on how to set up the board.

Note: In Windows 3.x, if you change the IRQ for a serial adapter, you also have to inform Windows of the change by modifying the Interrupt Request Line value for the port in the Windows Control Panel. (Open the Control Panel from the MAIN program group, double click on the PORTS icon then select the COM port that you are changing the IRQ for. Next, click on the SETTINGS button and then click the ADVANCED button in the Settings dialog box. Finally, choose the new IRQ for the COM port in the Advanced Settings dialog box).

Another solution is to use a multi-port serial adapter that provides an "interrupt multiplexing" scheme. Interrupt multiplexing is simply a mechanism that allows two or more ports to use the same interrupt. Several companies manufacture multi port serial adapters that support multiplexed interrupts. If you would like us to assist you in finding an add on serial board that fits your requirements please contact us at Tel: (215)-763-5096, Fax: (215)-763-9711 or
e-mail: support@taltech.com.

Understanding The Data From Your Serial Device

Before you can correctly configure the Software Wedge to parse and filter data received from your serial device, you must first have an understanding of the data that your device transmits. You should also be able to recognize what parts of the data are important to you and also completely understand what features of the Software Wedge can be used to manipulate your data. To use the Wedge effectively, you must think of each input from your device as *record* containing one or more specific *fields* of data. The next step is to decide what features of the Wedge can be used to consistently separate each data record from the next and also what features can be used to separate or *parse* individual data fields within a record. Finally you need to identify which data fields are important to your application and which fields should be removed or ignored.

The best place to start is to refer to the users manual for the device that you are using. The users manual should have a section that describes the structure and contents of all data that can be outputted from the device. If you do not have a manual for your device or if the output data structure is not described, you can use the Port-Analyze dialog box in the Software Wedge or the Terminal program that is provided with Windows to manually analyze the output from your device by viewing the data that is transmitted from it.

When you configure the Software Wedge for a particular device you must define the input record structure for your serial data to the Wedge by selecting "Input Data Record Structure" from the *Define* menu. When defining the input record structure, you specify an "End Of Record Event" and a general record structure for each data record (i.e. single field records, multiple delimited fields, or multiple fixed length fields).

With some devices, the record structure may be immediately obvious. For example, the following data record contains three numeric data fields delimited by commas and terminated by a carriage return-linefeed pair:

110,250,801<CrLf>

For this type of data record, you would select "Carriage Return or CrLf Received" as the *End Of Record Event* and for the *Record Structure* you would choose "Multiple Delimited Data Fields" and specify that there are three data fields per record with a comma delimiter separating each field.

Other devices may output data with a record structure that is less obvious than the example above. The important thing to look for is regular patterns in the data and also to understand all the features in the Wedge that can be used to correctly parse the data. If you have a situation where the data is too complex or inconsistent for the Wedge to parse correctly, you can also treat each transmission from the device as a single field data record and have the target application parse the data using whatever script or macro language it provides. For example, you could use the macro language in a spreadsheet to parse the data.

The important points to remember are that the Wedge always waits until the specified "End Of Record Event" occurs before it will do anything else. When the "End Of Record Event" occurs, the Wedge takes the data record that it just received and parses it according to the definition of the

Record Structure that you selected (i.e. Single Field, Multiple Delimited Fields or Multiple Fixed Length Fields). For each field that you have defined, the Wedge applies the chosen field Filter and then passes each field to the target application in sequence sending each field followed by the field's "Postamble" that you defined.

In many cases you could define the structure of your input data in many different ways. For example if your device transmitted the following data in bursts with some time between bursts, you could define the "End Of Record Event" as either "Carriage Return or CrLf Received" or as "Time Delay Between Records".

```
1,   23,  12,  24        <CrLf>
2,   27,  13,  8         <CrLf>
3,   27,  13,  9         <CrLf>
```

If you specified "Carriage Return or CrLf Received" as the "End Of Record Event" then the Wedge would see the data as three records with four fields per record.

If you specified "Time Delay Between Records" then the entire transmission would be read in as a single data record.

If the data above were also transmitted with the same number of characters each time and each of the data fields were always in the same position within data stream, you could specify "Fixed Number of Bytes Received" as the "End of Record Event" and then parse the entire transmission into 12 fixed length data fields.

In other words you can think of the original data above as either three records containing four fields each or as a single record with one field or a single record with 12 fields.

Again, as long as you understand the data from your serial device and have a clear understanding of the full capabilities of the Software Wedge, you can transform almost any serial data collection problem into an extremely simple task.

See Also: Defining The Input Data Record Structure (pg. 24)
Translation Table (pg. 35)

More Software Wedge Configuration Examples

Interfacing A Measuring Instrument To Excel Using A GagePort And The Wedge

A GagePort is a device that converts the output signal from many common measuring instruments to an RS232 data record. A common task for the Wedge is to interface a GagePort to a PC and to take readings from a measuring tool like a gage or caliper directly into a spreadsheet, (typically Excel), and have each reading from the gage entered into the spreadsheet so that they are stacked up in a column. This section describes a typical configuration for both the GagePort and the Wedge.

Note: A GagePort transmits data records containing four comma delimited data fields with a carriage return at the end of the record. Many other devices transmit data in a similar manner therefore this example may be applicable to other devices. For example X-Rite Densitometers (used for measuring Cyan, Yellow, Magenta and Black color densities) also transmit records with four fields of data and a carriage return at the end of each record. The only difference between data from the densitometer and the data from a GagePort is that the densitometer uses a single space as the delimiter between each of the four data fields. It is very possible that whatever device you happen to be using also has a similar output to a GagePort therefore even if you are not using a GagePort, you may find this example to be very educational.

Steps for setting up the GagePort:

1. Set up your GagePort in "Printer Mode" as per the instructions provided with the GagePort. This is done using the DIP switches on the GagePort. Also, use the DIP switches to set the communications parameters for the GagePort to 9600 baud, No Parity, Eight DataBits and One StopBit.

When set to Printer Mode, a reading is sent from the GagePort in either of the three following cases:

a. When you press the "transmit" button on your gage.
b. When you press the foot switch connected to the GagePort.
c. When a prompt is transmitted through the serial port to the GagePort requesting data.

2. Connect the GagePort to a COM port on your PC using the cable provided with it.

Steps for Setting up the Software Wedge:

For this situation, the easiest way to use the Wedge is to set it up in "Send Keystrokes" mode so that it will convert incoming serial data to keystrokes and thus cause your the data from your GagePort to appear as if it is being typed in on your keyboard.

1. Run the Software Wedge by double clicking on its icon in either the Windows Program Manager or from the Start Menu in Windows 95.

2. From the Software Wedge MODE menu, select "Send Keystrokes To..." (even if it is already checked). This will cause a dialog box to appear asking for the "Title Bar Text" and "Command Line" for the application where you want the keystrokes sent. Clear out both of these items so that they are completely empty. By clearing out these items and thus not specifying an application for the Wedge to send the keystrokes to, the Wedge will simply act as a second keyboard and send the incoming serial data to whatever application has the current input focus, i.e. the current foreground program.

3. Select "Settings" from the PORT menu in the Wedge and when the Port Settings dialog box appears, choose the communications parameters: 9600 baud, No Parity, Eight DataBits and One StopBit and then click the OK button to return to the main menu.

4. To test that the GagePort is connected properly and that your serial cable is working, select "Analyze" from the Port menu in the Wedge. When the Port Analyze dialog box appears, transmit a reading to the Wedge by either pressing the Transmit button on your gage or by pressing the foot switch connected to the GagePort. If your gage does not have a transmit button or a foot switch, then place the cursor in the "Output Buffer" text box in the Analyze dialog box and type in an upper case "A" followed by a lower case "r" and then click the button marked "Send". If a reading is transmitted successfully from the GagePort, you should see some data in the text box marked "Input Buffer". You should receive 25 characters and the data should be in the following format:

```
NNNN,##########,UUUUU,PP<cr>
```

The meanings of the four fields are as follows:

NNNN	Four digit sequential reading number
##########	Ten digit gage reading with decimal place and minus sign if negative.
UUUUU	Five blank spaces.
PP	Two digit port number.
<cr>	Carriage return. (Should appear as a musical note in the Analyze window.)

If you do not receive any data in the Input Buffer in the Analyze window, then either the GagePort is not connected properly or it is not configured correctly. If you do get data in the Input Buffer but the data consists of unreadable characters, then you most likely have the wrong communications parameters selected in the Wedge. (The com parameters set in both the Wedge and the GagePort must match.) Refer to the troubleshooting section of the Software Wedge users manual for further assistance. After you get good data in the Analyze window in the Software Wedge, click the Quit button to return to the main menu.

5. The next step is to configure the Wedge to parse and filter the data from the GagePort so that you get only the data that you are interested in to appear in your spreadsheet. In this case we will assume that you are only interested in the actual reading on the gage and that none of the other information transmitted by the GagePort is wanted.

Select "Input Data Record Structure" from the DEFINE menu in the Wedge. The first dialog to appear will prompt you for an "End of Record Event". For the End of Record Event, choose "Carriage Return or CrLf Received" and click the CONTINUE button to proceed. In the next dialog box to appear prompting for a "Record Structure", select "Multiple Delimited Data Fields" and click the CONTINUE button. This will display another dialog box prompting for a "Delimiter Character" and the "Maximum Number of Data Fields". Choose the Comma (,) for the delimiter character and enter 4 for the maximum number of data fields and click the CONTINUE button to proceed.

The final dialog box to appear is the "Input Record Definition Editor" dialog box. This dialog box allows you to select filters that can be applied to each data field in the incoming serial data as well as define any additional keystrokes that you want issued before or after each individual data field is sent from the Wedge to the application where you want the data to go. Select "Ignore " for the filter for Field 1 and click the button marked "Next Field". Select "Numeric" for the filter for Field 2 and place the cursor in the text box marked "Field Postamble Keystrokes" and click the button marked "Keystroke List". When the keystroke list appears, scroll the list until the word "ENTER" is highlighted and then click the OK button. This should cause the word "{ENTER}" to appear in the Field Postamble Keystrokes text box. Click the button marked "Next Field" and select "Ignore" for the filter for Field 3 and click the "Next Field" button again and select "Ignore" for the filter for Field 4. Finally, click the OK button to return to the main menu of the Wedge.

6. Select "Minimize on Activation" from the ACTIVATE menu and then save your work by selecting "Save" or Save As..." from the FILE menu. Save your configuration file using the file name GAGEPORT.SW1

7. The final step is to activate the Wedge by selecting "Test Mode" from the ACTIVATE menu. After the Wedge has been activated, you can open up Excel and place the cursor in a cell and whenever you take a reading on your gage, the reading will appear in the cell and the cursor will automatically move down one cell.

Interfacing A pH Meter To Excel With The Wedge

This example explores some of the less obvious features of the Software Wedge and how to apply them to a more complex situation. In this example we will assume that you have already gone through the steps for setting up the Software Wedge in "Send Keystrokes" mode and that you have also analyzed the data transmitted by a particular pH meter and discovered that the data is structured as follows with underscores representing blank spaces, <Cr> representing Carriage Returns and <Lf> representing Linefeed characters:

03/25/97 _ _ _12:53<Cr><Lf>7.01_ _ pH _ 67.5 _ mV<Cr><Lf>21.5 _ _ _C<Cr><Lf><Cr><Lf>

The data transmitted by the meter normally contains 48 characters. If we think of the entire output from the meter as a single data "Record" containing a group of data fields, the individual fields can be categorized as follows:

Bytes	Field Contents	Description	Length
1 - 11	"03/25/97 _ _ _"	the date followed by 3 blank spaces	11 digits
12 - 16	"12:53"	the time in military format	5 digits
17 - 18	<Cr><Lf>	carriage return & linefeed	2 bytes
19 - 22	"7.01"	pH value	4 digits
23 - 24	"_ _"	2 blank spaces	2 bytes
25 - 27	" pH _ "	the letters "pH" followed by 1 blank space	3 bytes
28 - 31	"67.5"	millivolt value	4 digits
32 - 34	"_ mV"	1 blank space followed by the letters "mV"	3 bytes
35 - 36	<Cr><Lf>	carriage return & linefeed	2 bytes
37 - 40	"21.5"	temperature in centigrade	4 digits
41 - 43	"_ _ _"	3 blank spaces	3 bytes
44	"C"	the letter "C"	1 byte
45 - 48	<Cr><Lf><Cr><Lf>	two carriage return linefeed pairs	4 bytes

When we examine the data more closely we find that it is always transmitted with the same number of bytes except that after every 10th reading the pH meter transmits an extra carriage return linefeed pair making every 10th reading two bytes longer than a normal reading.

For our particular application, we are interested in capturing only the time, the pH value and the temperature to a an Excel spreadsheet such that the three values appear next to each other on the same row but in separate columns. i.e. with one column of "time" values, one column of "pH" values and one column of "temperature" values.

When we select "Input Data Record Structure" from the Define menu in the Wedge, the first thing we must do is to select an "End Of Record Event" that will always reliably signal the end of each data record. Because there is more than one carriage return in each record, we cannot use "Carriage Return or CfLf Received" as the "End Of Record Event". We also cannot use "Fixed Number of Bytes Received" as the "End Of Record Event" because every 10th reading contains two additional bytes.

The way we normally use the meter is to press a button on the meter when we want to transmit a reading and we do not anticipate taking readings any faster than once a second. For this situation the "Time Delay Between Records" option is ideal. When you select this option the Wedge prompts you for a "Time Delay Between Records". The default value of three clock ticks (3/18 seconds) should be more than adequate for this situation. The next dialog to appear prompts us for a "Record Structure" for our data records. In this case the individual data fields within each data record always occupy the same byte positions within each record therefore we would select "Multiple Fixed Length Data Fields".

The final job that we need to do is to apply "Filters" to each of the data fields to remove the fields that we do not want and also add "Field Postamble Keystrokes" so that the data that we want to keep is entered the way we want in Excel. In the "Input Record Definition Editor" dialog in the Wedge we would select the following parameters for each of the data fields that we identified our data record:

Field	Filter	Length	Postamble Keystroke(s)	Notes
1	Ignore	11		Ignore Date
2	None	5	{Right}	Time field
3	Ignore	2		Ignore CrLf
4	Numeric	4	{Right}	pH value field
5	Ignore	5		Ignore "_ _ pH _ "
6	Ignore	4		Ignore mV value
7	Ignore	5		Ignore "mV _ "<Cr><Lf>
8	Numeric	4	{Down}{Left}{Left}	Temp field
9	Ignore	100		Ignore everything after temp

If you compare the above field definitions to the original structure described on the previous page, you can see that in several places we took the liberty of combining two or more adjacent fields in a couple of places. This was done deliberately to point out the fact that you are free to define your data fields in whatever way you like (even for the purpose of simplifying the setup of the Wedge). We could have gone even further and combined fields 5, 6 and 7 to further reduce the number of fields.

In this particular setup, we are telling the Wedge to read in all data until no more data comes in the serial port for 3/18th of a second or more. This "time delay between records" is used to signal the Wedge that we have a complete data record. We configured the Wedge to parse our records into nine fixed length data fields where we ignore all fields except the Time field, the pH field and theTemp field. The postamble keystrokes that we defined cause the Wedge to type in the Time field followed by a Right arrow, the pH field followed by another Right arrow and finally the Temp field followed by a Down arrow and two Left arrows. The postamble keystrokes were chosen to mimic the keystrokes that we would use if we were to type the data into Excel by hand.

See Also: Defining the Input Data Record Structure (pg. 24)

Cool Wedge Tricks

Sending keystrokes to a DOS Window with WinWedge:

Although it is technically not possible for a Windows program to send keystrokes directly to a DOS application running in a DOS window, there is a way to cheat and get Windows to do the job for us. An interesting feature of Windows is that it supports the ability to "Cut and Paste" data between DOS and Windows programs using the Windows ClipBoard. In order to get the Wedge to send data to a DOS application in a DOS window you first have to send the data as keystrokes to a Windows program and then send additional keystrokes that first "cut" the data from the Windows application, switch the focus to the DOS window and finally "paste" the data into the DOS window.

To try this technique, configure the Wedge in "Send Keystrokes" mode specifying the NotePad program as the target for all keystrokes from the Wedge. Next, configure the Wedge to work with your serial device and then add the following characters to the end of the "Field Postamble Keystrokes" for the very last data field that you have defined in the Wedge:
%ea^x%{TAB}%{ }ep

Finally, open a DOS session in a window (not full screen) and transmit some data from your device. The data should first pop into the NotePad program and then be cut from NotePad and pasted into the DOS window.

The keystrokes: "%ea" mean "Alt+e a" which causes the NotePad program to select all the text in the NotePad Window. The keystrokes: "^x" mean "Ctrl+x" which causes NotePad to cut the currently selected text to the ClipBoard. The keystrokes: "%{TAB}" mean "Alt+Tab" which causes Windows to switch the focus to the last window that was active before the NotePad program received the focus (i.e. the DOS window). Finally the keystrokes: "%{ }ep" mean "Alt+space e p" which opens the control box in the DOS window and selects "Edit" and "Paste".

The next time you transmit a data record from your device, the Wedge will switch the focus back to the NotePad application automatically and do the same thing all over again. If you have a reasonably fast PC, you will barely notice the NotePad program popping open in front of your DOS window.

Note: A true DOS version of the Software Wedge is available that may be better suited for passing data to a DOS application program. The Software Wedge for DOS is a RAM resident (TSR) program that passes serial data directly to other DOS programs by stuffing the DOS keyboard buffer. Although it is a DOS application, it runs extremely well in Windows inside a DOS Window and is an excellent tool for passing serial data to other DOS based programs.

Sending data as keystrokes to more than one application at a time:

In the previous example we showed how to send data from the Wedge to a DOS application by first sending the data as keystrokes to the NotePad program and then cutting the data out of NotePad, switching the focus to a DOS window and pasting the data in. You can use the same technique to have the Wedge send data to two (or more) Windows or DOS programs at a time. For example, if in the previous example we were to copy the data from NotePad instead of cutting the data, we would end up with both the NotePad and the DOS programs receiving the data.

All you need to do to pass data to two Windows programs is to send the data to the first application and then use whatever keystrokes are necessary to copy the data to the ClipBoard, switch the focus to the second application and finally paste the data in. Even if neither of the applications provide an explicit copy or paste function, you can always copy text to the clipboard in any Windows program by first selecting the text and issuing the keystroke Ctrl+Insert. In a "Field Postamble Keystroke" macro in the Wedge, Ctrl+Insert is represented by: ^{INSERT}. Likewise, you can paste text data from the clipboard into any Windows program using the keystrokes Shift+Insert. In the Wedge Shift+Insert is represented by: +{INSERT}. To select text in any program you can hold down the shift key while using the arrow keys, Home, End, Page Up and Page Down keys to highlight and select characters.
Windows will also switch the focus between all running application when you hold the Alt key down and press the Tab key. Windows maintains a list of all currently running applications with the application that currently has the input focus placed at the top of the list. If you switch to another application, the new application moves to the top of the list and the previously active application moves down one spot in the list. When hold the ALT key down, each time you press the tab key, Windows moves down the list one item and selects the application at that position in the list. When you release the Alt key, the selected application is given the focus and is moved to the top of the list pushing all other application that were above it in the list down one spot.

For example, if you had the Wedge set up to send keystrokes to NotePad and you also wanted to send the data to the Windows Write (WordPad in Windows 95) you could use the following procedure:

Set up the Wedge in "Send Keystrokes" mode specifying the NotePad program as the target for all keystrokes from the Wedge. Next, configure the Wedge to work with your serial device and then add the following characters to the end of the "Field Postamble Keystrokes" for the very last data field that you have defined in the Wedge:
+({LEFT}{HOME})^{INSERT}{DOWN}%{TAB}+{INSERT}
Activate the Wedge and then launch the Write program followed by the NotePad program. At this point NotePad has the focus and is therefore at the top of the Window list with the Write program directly underneath it. When you input data from your device, it will be sent to both NotePad and Write. The keystrokes: +({LEFT}{HOME}) means hold the shift key while pressing the left arrow and the home key. This selects all text in the current line. The keystrokes: ^{INSERT} means Ctrl+Insert and causes all selected text to be copied to the ClipBoard. The {DOWN} keystroke that follows is there to simply de-select the currently selected text. The keystrokes: %{TAB} means Alt-Tab and causes Windows to switch the focus to the Write program. Finally, the keystrokes: +{INSERT} means shift+insert and performs the job of pasting the data from the clipboard into the Write program.

To send the data to three Windows programs at a time you could use the following "Field Postamble Keystrokes":
+({LEFT}{HOME})^{INSERT}{DOWN}%{TAB}+{INSERT}%{TAB 2}+{INSERT}

Using The Wedge with the Recorder application that comes with Windows 3.1:

Windows 3.x is shipped with an extremely useful utility called "Recorder" that allows you to record sequences of keystrokes, mouse moves and mouse clicks. You can then assign your recorded sequences to hot keys so that all you need to do to play back a long sequence is press a single hot key. The Recorder program can be found in your Accessories program group in Window 3.1. Recorder is probably the single most time saving utility on your PC but unfortunately it is also the least understood and least used accessory. In fact, Microsoft does not even provide Recorder as part of Windows 95 because they found that hardly anyone was using it in Windows 3.1. (Unfortunately, the 16 bit version of Recorder that is shipped with Windows 3.1 does not work in Windows 95 or NT so if you are running Windows 95 or NT this discussion will be of little value.) Because the Recorder program allows you to assign macros to a single hot key keystrokes, you could invoke a Recorder macro from the Wedge by placing the hot key keystroke for a Recorder macro in either the "Record Preamble Keystrokes" or the "Field Postamble Keystrokes" when you define your "Input Data Record Structure" in the Wedge. You could also use the "Translation Table" to translate specific characters that might appear in your serial data to keystrokes that invoke different Recorder macros. Using the Wedge in combination with the Recorder thus allows you to do practically anything you can imagine with the data coming in the serial port. For example you could set up the Wedge and Recorder so that whenever a specific character was received through the serial port, an entire sequence of operations took place including opening and closing applications, deleting files, etc.. If you can do it manually then Recorder can automate the job for you with a macro and the Wedge can send the necessary hot keys to invoke the macro.

See Also:
Specifying Pre / Postamble Keystrokes (Send Keystrokes Mode) (pg. 31)
Translation Table (pg. 35)

Invoking hot keys in other applications:

Almost all spreadsheets, databases and word processors allow you to create subroutines or macros that can be invoked with a hot key while working within the application. For example when you write a subroutine in Excel, you can assign the subroutine to a "Shortcut Key" allowing you to run the subroutine by pressing the shortcut key. Similar to the previous discussion on invoking Recorder macros with the Wedge, you could invoke Excel subroutines with the Wedge by placing the shortcut key keystroke for a subroutine in either the "Record Preamble" or the "Field Postamble Keystrokes" when you define your "Input Data Record Structure" in the Wedge. You could also use the "Translation Table" in the Wedge to translate specific characters in your serial data to keystrokes that would invoke Excel subroutines.

Cool Excel Tricks

Entering data into a range of cells:

If you highlight a range of cells in Excel and type numbers into the range pressing either the Tab key or the Enter key after each number, Excel will cycle through the selected range and jump back to the first cell after the last number in the range is entered. You can take advantage of this feature by setting the Wedge in "Send Keystrokes" mode and defining a "Field Postamble Keystroke" consisting of either the Enter or Tab key. After you activate the Wedge, switch to Excel and select the range where you would like data entered and start taking readings from your instrument. The data from the device will automatically cycle through the selected range. If you chose the Enter key as the postamble, the data will move down and to the right through the range of cells and if you chose the Tab key, the data will move to the right and down through the range of cells.

Quickly chart a range of cells:

Excel has a tool called the "Chart Wizard" that allows you to create charts and graphs by simply clicking a few buttons. To use the Chart Wizard highlight the range of cells that you want to chart and select "Chart" from the "Insert" menu and follow the instructions presented on screen. Once you have a chart associated with a range of cells, Excel will automatically update the chart the moment any data in the range changes.

Place a button in on a toolbar to run a macro:

Excel allows you to customize its toolbars to include buttons that can perform all sorts of actions including running your own macros or subroutines. If you have written a subroutine (perhaps to prompt a device for a reading) and would like to place a button in an Excel toolbar to run this subroutine, simply select "Toolbars" from the "View" menu and then click the "Customize" button in the dialog that appears. In the "Customize" dialog box, select "Custom" from the list of "Categories" to display a set of toolbar icons. Select an icon that you like and drag it up to the toolbar area below Excel's main menu and drop it into either an existing toolbar or on its own. When you release your mouse to drop the button, Excel will open a dialog box prompting for the name of the subroutine that you want associated with the button.
See Also: Tech Tip (pg. 59)

INTRODUCTION TO SERIAL COMMUNICATIONS

All IBM PC and compatible computers are typically equipped with two serial ports and one parallel port. Although these two types of ports are used for communicating with external devices, they work in different ways.

A parallel port sends and receives data eight bits at a time over 8 separate wires. This allows data to be transferred very quickly; however, the cable required is more bulky because of the number of individual wires it must contain. Parallel ports are typically used to connect a PC to a printer and are rarely used for much else. A serial port sends and receives data one bit at a time over one wire. While it takes eight times as long to transfer each byte of data this way, only a few wires are required. In fact, two-way (full duplex) communications is possible with only three separate wires - one to send, one to receive, and a common signal ground wire.

Bi-directional Communications

The serial port on your PC is a full-duplex device meaning that it can send and receive data at the same time. In order to be able to do this, it uses separate lines for transmitting and receiving data. Some types of serial devices support only one-way communications and therefore use only two wires in the cable - the transmit line and the signal ground.

Synchronous And Asynchronous Communications

There are two basic types of serial communications, synchronous and asynchronous. With synchronous communications, the two devices initially synchronize themselves to each other, and then continually send characters to stay in sync. Even when data is not really being sent, a constant flow of bits allows each device to know where the other is at any given time. That is, each character that is sent is either actual data or an idle character. Synchronous communications allows faster data transfer rates than asynchronous methods, because additional bits to mark the beginning and end of each data byte are not required. The serial ports on IBM-style PCs are asynchronous devices and therefore only support asynchronous serial communications.

Asynchronous means "no synchronization", and thus does not require sending and receiving idle characters. However, the beginning and end of each byte of data must be identified by start and stop bits. The start bit indicates when the data byte is about to begin and the stop bit signals when it ends. The requirement to send these additional two bits cause asynchronous communications to be slightly slower than synchronous however it has the advantage that the processor does not have to deal with the additional idle characters when no data is flowing.

An asynchronous line that is idle is identified with a value of 1, (also called a mark state). By using this value to indicate that no data is currently being sent, the devices are able to distinguish between an idle state and a disconnected line. When a character is about to be transmitted, a start bit is sent. A start bit has a value of 0, (also called a space state). Thus, when the line switches from a value of 1 to a value of 0, the receiver is alerted that a data character is about to come down the line.

Communicating By Bits

Once the start bit has been sent, the transmitter sends the actual data bits. There may either be 5, 6, 7, or 8 data bits, depending on the number used by the transmitting device. Both receiver and the transmitter must agree on the number of data bits, as well as the baud rate. Almost all devices transmit data using either 7 or 8 databits.

Notice that when only 7 data bits are employed, you cannot send ASCII values greater than 127. Likewise, using 5 bits limits the highest possible value to 31. After the data bits are transmitted, a stop bit is sent. A stop bit has a value of 1 and can be detected correctly even if the previous data bit also had a value of 1. This is accomplished by the fixed duration of each of the data bits and the start bit. Stop bits can be 1, 1.5, or 2 bit periods in length.

The Parity Bit

Besides the synchronization provided by the use of start and stop bits, an additional bit called a parity bit may optionally be transmitted along with the data. A parity bit affords a small amount of error checking, to help detect data corruption that might occur during transmission. You can choose either even parity, odd parity, mark parity, space parity or none at all. When even or odd parity is being used, the number of marks (logical 1 bits) in each data byte are counted, and a single bit is transmitted following the data bits to indicate whether the number of 1 bits just sent is even or odd.

For example, when even parity is chosen, the parity bit is transmitted with a value of 0 if the number of preceding marks is an even number. For the binary value of 0110 0011 the parity bit would be 0. If even parity were in effect and the binary number 1101 0110 were sent, then the parity bit would be 1. Odd parity is just the opposite, and the parity bit is 0 when the number of mark bits in the preceding word is an odd number. Parity error checking is very rudimentary. While it will tell you if there is a single bit error in the character, it doesn't show which bit was received in error. Also, if an even number of bits are in error then the parity bit would not reflect any error at all.

Mark parity means that the parity bit is always set to the mark signal condition and likewise space parity always sends the parity bit in the space signal condition. Since these two parity options serve no useful purpose whatsoever, they are almost never used.

Baud Versus Bits Per Second

The baud unit is named after Jean Maurice Emile Baudot, who was an officer in the French Telegraph Service. He is credited with devising the first uniform-length 5-bit code for characters of the alphabet in the late 19th century. What baud really refers to is modulation rate or the number of times per second that a line changes state. This is not always the same as bits per second (BPS). If you connect two serial devices together using direct cables then baud and BPS are in fact the same. Thus, if you are running at 19200 BPS, then the line is also changing states 19200 times per second. But when considering modems, this isn't the case.

Because modems transfer signals over a telephone line, the baud rate is actually limited to a maximum of 2400 baud. This is a physical restriction of the lines provided by the phone company and is the reason why the Internet is often referred to as the "World Wide Wait". The increased data throughput achieved with 9600 or higher baud modems is accomplished by using sophisticated phase modulation, and data compression techniques.

RS-232C

RS-232 stands for Recommend Standard number 232 and C is the latest revision of the standard. The serial ports on most computers use a subset of the RS-232C standard. The full RS-232C standard specifies a 25-pin "D" connector of which 22 pins are used. Most of these pins are not needed for normal PC communications, and indeed, most new PCs are equipped with male D type connectors having only 9 pins.

DCE And DTE Devices

Two terms you should be familiar with are DTE and DCE. DTE stands for Data Terminal Equipment, and DCE stands for Data Communications Equipment. These terms are used to indicate the pin-out for the connectors on a device and the direction of the signals on the pins. Your computer is a DTE device, while most other devices are usually DCE devices.
If you have trouble keeping the two straight then replace the term "DTE device" with "your PC" and the term "DCE device" with "remote device" in the following discussion.

The RS-232 standard states that DTE devices use a 25-pin male connector, and DCE devices use a 25-pin female connector. You can therefore connect a DTE device to a DCE using a straight pin-for-pin connection. However, to connect two like devices, you must instead use a null modem cable. Null modem cables cross the transmit and receive lines in the cable, and are discussed later in this chapter. The listing below shows the connections and signal directions for both 25 and 9-pin connectors.

25 Pin Connector on a DTE device (PC connection)

Pin	Name	Direction of signal
1	Protective Ground	
2	Transmitted Data (TD)	Outgoing Data (from a DTE to a DCE)
3	Received Data (RD)	Incoming Data (from a DCE to a DTE)
4	Request To Send (RTS)	Outgoing flow control signal controlled by DTE
5	Clear To Send (CTS)	Incoming flow control signal controlled by DCE
6	Data Set Ready (DSR)	Incoming handshaking signal controlled by DCE
7	Signal Ground	Common reference voltage
8	Carrier Detect (CD)	Incoming signal from a modem
20	Data Terminal Ready (DTR)	Outgoing handshaking signal controlled by DTE
22	Ring Indicator (RI)	Incoming signal from a modem

9 Pin Connector on a DTE device (PC connection)

Pin	Name	Direction of signal
1	Carrier Detect (CD) (from DCE)	Incoming signal from a modem
2	Received Data (RD)	Incoming Data from a DCE
3	Transmitted Data (TD)	Outgoing Data to a DCE
4	Data Terminal Ready (DTR)	Outgoing handshaking signal
5	Signal Ground	Common reference voltage
6	Data Set Ready (DSR)	Incoming handshaking signal
7	Request To Send (RTS)	Outgoing flow control signal
8	Clear To Send (CTS)	Incoming flow control signal
9	Ring Indicator (RI) (from DCE)	Incoming signal from a modem

The TD (transmit data) wire is the one through which data from a DTE device is transmitted to a DCE device. This name can be deceiving, because this wire is used by a DCE device to receive its data. The TD line is kept in a mark condition by the DTE device when it is idle. The RD (receive data) wire is the one on which data is received by a DTE device, and the DCE device keeps this line in a mark condition when idle.

RTS stands for Request To Send. This line and the CTS line are used when "hardware flow control" is enabled in both the DTE and DCE devices. The DTE device puts this line in a mark condition to tell the remote device that it is ready and able to receive data. If the DTE device is not able to receive data (typically because its receive buffer is almost full), it will put this line in the space condition as a signal to the DCE to stop sending data. When the DTE device is ready to receive more data (i.e. after data has been removed from its receive buffer), it will place this line back in the mark condition. The complement of the RTS wire is CTS, which stands for Clear To Send. The DCE device puts this line in a mark condition to tell the DTE device that it is ready to receive the data. Likewise, if the DCE device is unable to receive data, it will place this line in the space condition. Together, these two lines make up what is called RTS/CTS or "hardware" flow control. The Software Wedge supports this type of flow control, as well as Xon/XOff or "software" flow control. Software flow control uses special control characters transmitted from one device to another to tell the other device to stop or start sending data. With software flow control the RTS and CTS lines are not used.

DTR stands for Data Terminal Ready. Its intended function is very similar to the RTS line. DSR (Data Set Ready) is the companion to DTR in the same way that CTS is related to RTS. Some serial devices use DTR and DSR as signals to simply confirm that a device is connected and is turned on. The Software Wedge sets DTR to the mark state when the serial port is opened and leaves it in that state until the port is closed. The DTR and DSR lines were originally designed to provide an alternate method of hardware handshaking. It would be pointless to use both RTS/CTS and DTR/DSR for flow control signals at the same time. Because of this, DTR and DSR are rarely used for flow control.

CD stands for Carrier Detect. Carrier Detect is used by a modem to signal that it has a made a connection with another modem, or has detected a carrier tone.

The last remaining line is RI or Ring Indicator. A modem toggles the state of this line when an incoming call rings your phone.

The Carrier Detect (CD) and the Ring Indicator (RI) lines are only available in connections to a modem. Because most modems transmit status information to a PC when either a carrier signal is detected (i.e. when a connection is made to another modem) or when the line is ringing, these two lines are rarely used.

Flow Control Protocols

Flow control is a mechanism for controlling the flow of serial data from one device to another to insure that no data is ever lost. When one device is transmitting data to another, the receiving device uses a temporary holding area called a "buffer" to store all the incoming data in until it has a chance to process it. If a transmitting device sends more data than the receiver can hold in its buffer and the receiving device is unable to remove data from the buffer fast enough, the receiver must either discard any new data that does not fit in the buffer or it must use "Flow Control" to send a signal back to the transmitting device to tell it to stop sending. After the receiving device has had a chance to remove the data in it's buffer, it can send another signal back to the transmitter to let it know that it is OK to start sending data again.

The two most common types of flow control are "Hardware" and "Software". Hardware or "RTS/CTS" flow control uses the RTS and CTS lines in the serial cable to tell the device on the other end of the wire when it is OK to transmit data. When the receiver's buffer is almost full, it raises the signal on the RTS line. After it has had a chance to remove data from its buffer, the receiver lowers the signal on the RTS line.

Software flow control is most often used for transmitting data through a modem. When using a modem to transmit serial data over a telephone line, there are no RTS or CTS wires in the telephone line therefore hardware flow control cannot be used.

Software flow control uses two special control characters called the Xon and Xoff characters (ASCII 17 and ASCII 19) to control the flow of data. When a receiving device is unable to store any more data in its buffer, it transmits an Xoff character to the transmitting device to tell it to stop sending. After the receiver has had a chance to remove the data in its buffer, it transmits an Xon character to let the transmitter know that it is OK to start sending again.

Software flow control has an advantage that only three wires are necessary in the serial cable (transmit, receive and ground) but it has the disadvantage that special additional protocols must be used if the actual data that is transmitted could possibly contain either the Xon or Xoff characters.

Note that if the receive buffer is large enough to accomodate all the data that might ever be sentby a device, flow control is entirely unnecessary. Even if it is enabled in both the transmitter and the receiver, it will not be used because the buffer will never get close to full. Most simple serial devices like bar code readers and electronic balances transmit very small amounts of data in short bursts. With these types of devices, the chance of ever filling up even a small 512 byte buffer is extremely low therefore it will make no difference which type of flow control is used.

Cable Lengths

The RS-232C standard imposes a cable length limit of 50 feet. You can usually ignore this "standard", since a cable can be as long as 10000 feet at baud rates up to 19200 if you use a high quality, well shielded cable. The external environment has a large effect on lengths for unshielded cables. In electrically noisy environments, even very short cables can pick up stray signals. The following chart offers some reasonable guidelines for 24 gauge wire under typical conditions. You can greatly extend the cable length by using additional devices like optical isolators and signal boosters. Optical isolators use LEDs and Photo Diodes to isolate each line in a serial cable including the signal ground. Any electrical noise affects all lines in the optically isolated cable equally - including the signal ground line. This causes the voltages on the signal lines relative to the signal ground line to reflect the true voltage of the signal and thus canceling out the effect of any noise signals.

Baud Rate	Shielded Cable Length	Unshielded Cable Length
110	5000 (feet)	1000 (feet)
300	4000	1000
1200	3000	500
2400	2000	500
4800	500	250
9600	250	100

Cables, Null Modems, And Gender Changers

In a perfect world, all serial ports on every computer would be DTE devices with 25-pin male "D" connectors. All other devices to would be DCE devices with 25-pin female connectors. This would allow you to use a cable in which each pin on one end of the cable is connected to the same pin on the other end. Unfortunately, we don't live in a perfect world. Serial ports use both 9 and 25 pins, many devices can be configured as either DTE or DCE, and - as in the case of many data collection devices - may use completely non standard or proprietary pin-outs. Because of this lack of standardization, special cables called null modem cables, gender changers and custom made cables are often required.

Null Modem Cables and Null Modem Adapters

If you connect two DTE devices (or two DCE devices) using a straight RS232 cable, then the transmit line on each device will be connected to the transmit line on the other device and the receive lines will likewise be connected to each other. A Null Modem cable or Null Modem adapter simply crosses the receive and transmit lines so that transmit on one end is connected to receive on the other end and vice versa. In addition to transmit and receive, DTR & DSR, as well as RTS & CTS are also crossed in a Null Modem connection.
Null Modem adapter are available at most computer and office supply stores for under $5.

9 Pin To 25 Pin Adapters

The following table shows the connections inside a standard 9 pin to 25 pin adapter.

9-PIn Connector			25 Pin Connector	
Pin 1	DCD	---------------------	Pin 8	DCD
Pin 2	RD	-----------------------	Pin 3	RD
Pin 3	TD	-----------------------	Pin 2	TD
Pin 4	DTR	---------------------	Pin 20	DTR
Pin 5	GND	--------------------	Pin 7	GND
Pin 6	DSR	---------------------	Pin 6	DSR
Pin 7	RTS	---------------------	Pin 4	RTS
Pin 8	CTS	---------------------	Pin 5	CTS
Pin 9	RI	------------------------	Pin 22	RI

Gender Changers

The final problem you may encounter is having two connectors of the same gender that must be connected. Again, you can purchase gender changers at any computer or office supply store for under $5.

Note: The parallel port on a PC uses a 25 pin female connector which sometimes causes confusion because it looks just like a serial port except that it has the wrong gender. Both 9 and 25 pin serial ports on a PC will always have a male connector.