Version 3.X
Standard Edition

CONTENTS:

License Agreement:

1. GRANT OF LICENSE

TAL Technologies, Inc. grants you the right to use one copy of the enclosed software program (the SOFTWARE) on a single terminal connected to a single computer (i.e., with a single CPU). You may not network the SOFTWARE or otherwise use it on more than one computer or terminal at the same time.

2. COPYRIGHT

The SOFTWARE is owned by TAL Technologies, Inc. and is protected by United States copyright laws and treaties. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g., a book or musical recording) except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the users manual accompanying the SOFTWARE.

3. OTHER RESTRICTIONS

You may not rent or lease the SOFTWARE, but you may transfer the SOFTWARE and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. You may not reverse engineer, de-compile, or disassemble the SOFTWARE. If SOFTWARE is an update, any transfer must include the update and all prior versions.

4. DISCLAIMER

No Warranty of any kind on the SOFTWARE is expressed or implied.

In no event shall TAL Technologies or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profit, business interruption, loss of business information or other pecuniary loss) arising out of the use of or inability to use this product. If you have questions concerning this agreement, or wish to contact TAL Technologies, please call or write:

TAL Technologies, Inc.
2101 Brandywine Street, Ste. 102
Philadelphia, PA 19130 USA
Tel: (215)-496-0222
Fax: (215)-496-0322
email: sales@taltech.com
Skype: taltech1
Web Site: http://www.taltech.com

What is WinWedge?

WinWedge is an extremely powerful utility that is designed to add serial communications capabilities to any Windows application. The primary function of WinWedge is to provide a way to connect any RS232, RS422 or RS485 device (and some USB devices) to a PC and be able to read or write data to or from the device directly from within any Windows application without the need for any custom serial communications software development. Typical kinds of serial instruments being used with WinWedge include bar code and magnetic stripe readers, electronic balances, measuring tools including calipers and gages, temperature sensors, telephone call logging systems, pH and color meters, PLCs, coordinate measuring machines, optical comparators, digital volt meters, GPS receivers and a wide variety of other laboratory or industrial instruments that communicate over a standard serial interface.

WinWedge can transfer serial data to *any* Windows application either by sending the data as keystrokes or by providing data as a DDE (dynamic data exchange) server. You can also set up the "Wedge" to issue commands to another application after each record of incoming serial data has been received. Thus you could cause the receiving application to perform actions based on the data being received; for example you could force an application to run a macro or update a graph or chart in real time using the data received from the device.

WinWedge 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. Two translation tables 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 WinWedge window or by pressing a "Hot Key" on your PC's keypad. You can use a special set of DDE commands to transmit data or command strings out the serial port directly from another program.

All these features make it possible to create extremely sophisticated device control interfaces for virtually any serial instrument directly from within any Windows application program.

What can you do with WinWedge?

WinWedge can be used in applications ranging from simple data input (i.e. reading data from a bar code scanner or scale into a spreadsheet) to complex device control interfaces (i.e. using WinWedge with Excel, Access, InTouch or other HMIs or LIMS to create an On-Line, real time, process control program that communicates with multiple laboratory instruments). The best way to understand the complete capabilities of WinWedge is to read this manual thoroughly. WinWedge has many features including powerful DDE capabilities that are not immediately apparent from the menus and windows presented when you run WinWedge on your PC. Again, the only way to learn about its full capabilities is to read this manual in its entirety.

See Also: Cool Wedge Tricks (pg. 99)

Before You Begin

We at TAL Technologies would like to take this opportunity to thank you for purchasing WinWedge. We sincerely hope that it proves to be the perfect solution for all your serial I/O applications. We also hope that you find WinWedge 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 easier to use, please contact us. Your feedback will be greatly appreciated.

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 215-496-0222 Fax: 215-496-0322 email: support@taltech.com
Skype: taltech1

System Requirements

The 32 bit version of WinWedge will run on any Windows 95, 98, ME, NT, 2000, XP, Vista, or Windows 7 system (i.e. any 32 or 64 bit version of Windows and any newer 32 or 64 bit versions of Windows that Microsoft may release in the future). WinWedge uses roughly 2 Mb of disk space and 16 or more Mb of RAM memory is recommended. Most PCs have several Gigabytes of RAM and several hundred Gigaytes of hard disk space therefore you should not have any problems installing and running WinWedge on even the most barebones Windows workstation.

Devices Compatible with WinWedge

WinWedge was designed primarily as an interface to most commonly used RS232 data collection devices including Electronic Scales, Bar Code, RFID Tag and Magnetic Stripe Readers, Electronic Measuring Tools, Medical, Industrial and Laboratory Instruments, Telephone PBX and Caller ID systems, Data Loggers, Flow Meters, Sensors, Plate Readers, Spectrum Analyzers, VOMs, pH meters, Densitometers, CMMs, Strip Chart Recorders, Titrators, Quality and Process Control Instruments, Coin Counters, PLCs, GPS receivers etc.. Any type of device with a serial interface whose output data is transmitted in a somewhat structured ASCII text format is a good candidate for use with WinWedge.

Note: Many RS422 and RS485 serial can also be used with WinWedge provided that you have the correct type of serial I/O adapter or converter installed in your PC. WinWedge can also be used with many types of USB devices provided that the USB device comes with a driver that causes the device to appear as a standard COM port in Windows. This includes all commercially available USB to Serial converters. For example, if your PC does not have any available RS232 serial ports, you can always add serial ports to it by purchasing a USB to Serial add-on adapter available at any computer or office supply store for roughly $10 to $30. All USB to Serial add-on adapters come with a driver that you install in Windows so that the operating system recognizes the adapter and causes it to appear in the Windows Device Manager as a standard RS232 COM port.

What's New In Version 3.X

Version 3.X of WinWedge adds several new capabilities to previous versions including: user definable “Hot Keys” that allow you to control many serial I/O functions with a simple key press, support for up to 32 serial ports, a Pre-Input Character Translation Table, additional Data Parsing and Filtering capabilities as well as a Timed Automatic Serial Output Function and an expanded set of DDE Commands. Complete context sensitive help (available by pressing function key F1) has also been added to all windows and dialog boxes thus making WinWedge easier to set up and use. A new “Virtual Instrument” mode even allows you to test the functionality of WinWedge without requiring an actual serial instrument to be attached to your PC. The users manual has also been re-written to include many examples for setting up and using WinWedge with different serial devices and other software applications. All new features were designed to support a wider range of RS232 instruments and I/O data formats as well as make WinWedge much easier to set up and use.

Why Is It Called WinWedge?

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 serial device as if it was being typed in on the keyboard.

Before the development of WinWedge, keyboard wedges were the only available solution if you needed to input serial data directly into an off the shelf program like Excel, Access, Word, 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 numerous problems and can be difficult or impossible to install, especially on a laptop that does not have a keyboard port or a desktop PC that uses a UBS keyboard.

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

Because WinWedge 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 can also provide features that are either difficult or impossible to implement in a hardware product such as data parsing, filtering, formatting 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, WinWedge is light years ahead of even the most sophisticated hardware wedge available. If you compare prices you’ll also find WinWedge to be far more economical as well.

Who's Using WinWedge

WinWedge was originally developed by TAL Technologies in 1989 as a simple interface for bar code readers. It quickly became apparent that a product of this type was needed for use with the hundreds of different types of devices that communicate through a serial interface. Since then, WinWedge has evolved into an extremely powerful product capable of interfacing virtually any type of serial instrument to a PC. Several hundred thousand copies of the "Wedge" have been licensed to users in every country around the globe. The following is just a short list of some of the many companies that rely on WinWedge 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, 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, SmithKline 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 WinWedge 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.

Getting Started

Installing and Running WinWedge

WinWedge comes with an installation program called "SETUP.EXE" that will copy all WinWedge program files to your hard disk and also install the WinWedge icons into the Windows "Start Menu".

To install WinWedge, place the WinWedge CD ROM into your CD ROM drive and select "Run" from the Start Menu. When the "Run" window appears, type the command "d:SETUP" (where d is the drive letter for your CD ROM drive) and click "OK". The SETUP program takes approximately one minute to copy all files and install WinWedge into the Windows Start Menu.

WinWedge can be run by clicking your mouse cursor on the WinWedge icon in the Start menu.

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

How To Obtain Technical Support

If you experience difficulty configuring WinWedge after you have read this manual thoroughly, please call TAL Technologies at Tel: (215)-496-0222 or Fax: (215)-496-0322 for technical support. Questions can also be sent via e-mail to: **support@taltech.com**
You may also reach us on Skype using the Skype username: taltech1

Only registered users of WinWedge are eligible for technical support. Please have your original WinWedge CD ROM and this users manual available when calling for support and, if possible, be at your PC with WinWedge running when you call. TAL Technologies also has a web page at: **http://www.taltech.com** where you can find technical information and examples as well as a WinWedge "Knowledge Base" containing answers to many frequently asked questions.

Important: If you have not already done so, please take the time fill out the software registration form on our web site at http://www.taltech.net/register.asp. If you call for technical support and you are not a registered user then we will not be able to provide you with telephone or e-mail support.

See Also:

Diagnosing Serial Communications Problems (pg. 86)
Troubleshooting (pg. 87)

Quick Start Guide To Using WinWedge

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

Step 1. Select how WinWedge will transfer incoming serial data to your other Windows programs by choosing one of the options in the *Mode* menu. WinWedge can transfer incoming serial data to other programs by either converting the data to "keystrokes" and typing the data into another application's window or it can pass the data using Dynamic Data Exchange (DDE). Sending data as keystrokes is generally much simpler than using DDE however DDE provides greater flexibility in how you can deal with the data in the other program. For a complete description of each of mode, read the section of this manual entitled: **The Mode Menu** (pg. 15)

Step 2. Select the *Settings* option in the *Port* menu and choose the communications parameters that match the settings for the serial device that you will be using. After you select your 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. 88) for further information.

Step 3. Select "*Input Data Record Structure..."* from the *Define* menu. This will present you with a series of windows that allow you to first describe the structure of your incoming serial data and finally describe how you want to parse, filter and format the data as well as add additional data, keystrokes, or date/time stamps to the data before passing it on to the target application program. This step can be fairly complex therefore it is highly recommended that you read the sections: **Defining the Input Data Record Structure** (pg. 25) and **Understanding The Data From Your Serial Device** (pg. 90)

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) and **Defining Hot Keys and Hot Key Actions** (pg. 40)

Step 5. The final step is to Activate WinWedge to start it working. You may want to save your WinWedge configuration at this point by selecting "*Save*" from the *File* menu. To activate WinWedge, select "*Test Mode*" from the *Activate* menu.
Refer to the section: **The Activate Menu** (pg. 44) for further information.

Note: WinWedge will not work until you activate it by selecting one of the options in the Activate menu. We recommend selecting "Test Mode" until you have everything working the way you desire.

See Also:

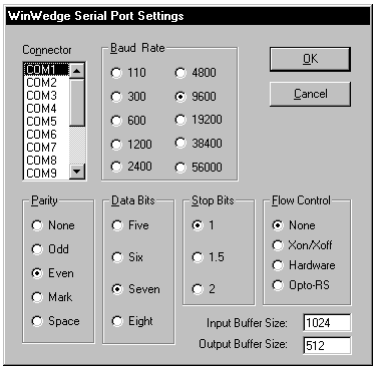
A Typical WinWedge Configuration Example (pg. 9)
More WinWedge Configuration Examples (pg. 94)
Diagnosing Serial Communications Problems (pg. 86)
Troubleshooting (pg. 87)

A Typical WinWedge Configuration Example

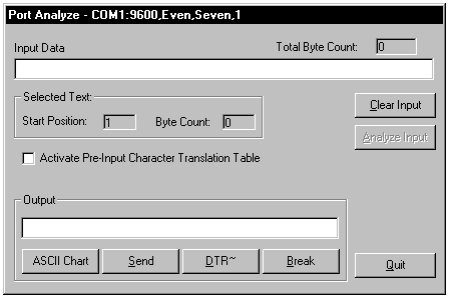
This section shows the typical steps for setting up WinWedge for use with a bar code reader or a simple measuring instrument like a digital caliper, force gage, weigh scale, or pH meter. The steps outlined below show how to configure WinWedge to read data from your device directly into another Windows application wherever the cursor is; as if the data were being typed in on your keypad. This example assumes that the application that you primarily want to capture data to is a spreadsheet. It also assumes that you would like to have successive readings from your instrument stacked up in a column in the spreadsheet.

You should begin by reading the owners manual for the serial device that you are using to find out the communications parameters that it uses. For this example we will assume that the device uses the parameters: 9600 baud, even parity, 7 data bits and one stop bit.

Step 1. Plug the instrument into serial port COM1 or COM2, turn it on, and run WinWedge by clicking on its icon in the Windows "Start" menu.



Step 2. Select the “SETTINGS” option from the “PORT” menu in WinWedge and choose the communications parameters COM1 (or COM2), 9600 baud, Even Parity, 7 Data Bits, one stop bit and no flow control from the window that appears and then click the OK button to return to the main menu.

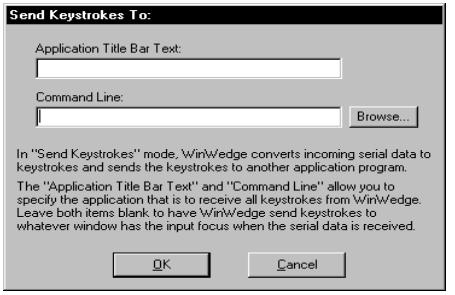


Step 3. Select “ANALYZE” from the “PORT” menu and transmit a sample reading from your device into the "Input Data" textbox in the Analyze window. This step serves two purposes, first, to test that your serial communications parameters are correct and that the device is connected properly and second, to gain an understanding of how the data from the device is structured. If data does not appear in the text box marked “Input Data”, then read the section of this manual entitled “Troubleshooting” (pg. 87).

If data appears in the Input Data textbox, then examine all the characters that are received looking for regular features like delimiters, or special characters like carriage returns or linefeeds, etc.. (A carriage return (ASCII 13) will appear as a musical note and a linefeed (ASCII 10) will look like a small rectangle with a circle inside it.) 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 electronic balances, bar code readers, calipers, and force gages transmit a single number or short data string followed by a carriage return or carriage return linefeed pair. 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 thoroughly and are satisfied that the device is transmitting your data correctly, click on the button marked “Quit” to return to the main menu.



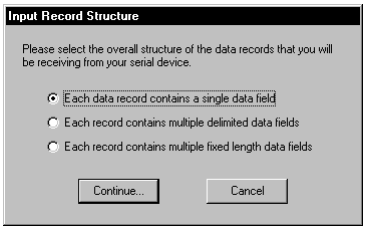
Step 4. Because we want to be able to read data directly into *any* Windows program, the easiest way to do this is to have WinWedge convert the serial data to keystrokes so that it will appear as if it is being typed in on your keyboard. Select “Send Keystrokes To...” from the MODE menu (even if it is already checked). This will present a window prompting for the Application Title Bar Text and/or Command Line that will identify the application that you want WinWedge to send all keystrokes to.

In this example, we want WinWedge to send the data to whatever application has the input focus when the data is received therefore the two items “Application Title Bar Text” and “Command Line” should be completely cleared out. Leaving these items blank (as shown) causes WinWedge to send the data from your device as 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.

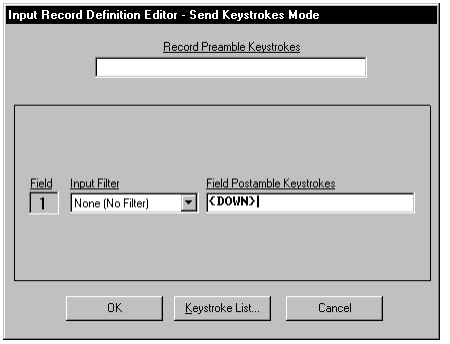
Step 5. The next step is to define the structure of the data that is being received from your serial device. We primarily want to send the data to a spreadsheet and have the cursor move down one cell after each reading, hence we must also instruct WinWedge to add a Down Arrow keypress at the end of each reading from our device.

To do this, select “Input Data Record Structure...” from the DEFINE menu. The first window that appears (left) will allow us to describe the events that determine the start and end of each data record transmitted by our device. In this case each data record is a small text string ending with a carriage return (ASCII 13). Thus, we would specify "Any Character Received" as the "Start of Record Event" and “Carriage Return or CrLf Received” as the “End of Record Event”.

After these options are selected, click on the “Continue...” button to proceed



The next window 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.



In the next window to appear, place the cursor in the text box labeled “Field Postamble Keystrokes” and then click the button marked “Keystroke List”. When the keystroke list appears, scroll down through the list until the word “Down” is highlighted and then click the OK button in the keystroke list window. This should cause the word “{DOWN} to appear in the “Field Postamble Keystrokes” text box (as shown). Then click the button marked “OK” to return to the main menu.

WinWedge 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 BALANCECOM1.SW3 for a balance connected to COM1.

Step 6. After configuring WinWedge you must activate it by selecting “Test Mode” from the ACTIVATE menu in order to start it working. After you activate WinWedge, a window containing a text box marked “Input Data Field(s)” will appear on your screen to indicate that it has been activated successfully.

Step 7. At this point, WinWedge is now ready and waiting to do its job. If you open your spreadsheet (or any other Windows program) and place the cursor in a cell and then transmit a reading from your serial device, the data will appear just as if it were being typed in on your keyboard with a “Down Arrow” keypress being issued automatically after the data appears.

See Also:

More WinWedge Configuration Examples (pg. 94)
Cool Wedge Tricks (pg. 99)

The Main Menu

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

The title bar of the main menu indicates that you are in *setup* mode and it also displays the filename of any currently selected configuration file in parentheses. Before you can use WinWedge, you must configure it for your particular serial device. You configure WinWedge 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 main menu options and their functions are :

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

Mode - Allows selection of the method used to transfer serial input data from WinWedge to another application and 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)

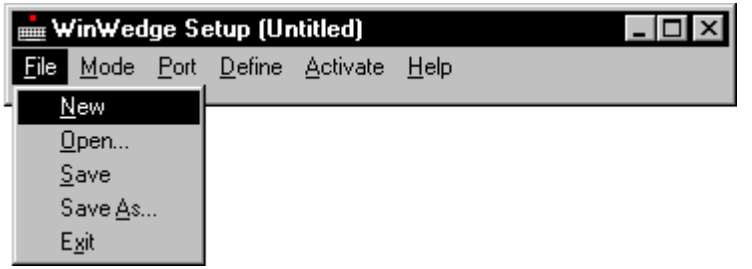
Define - The define menu contains options that allow you to customize how WinWedge should work for your particular application. The different sub menu items allow the definition of the input data record structure, parsing instructions & filters to be applied to incoming data, additional cursor navigation keystroke to be issued during the input of data, translations of incoming characters, definition of hot keys and serial output data strings.
See Also: The Define Menu (pg. 23)

Activate - Allows activation of WinWedge using the current configuration.
WinWedge will not do anything until it is activated by selecting one of the activation options in this menu.
See Also: The Activate Menu (pg. 44)

Help - Provides On-Line help for WinWedge.
In addition to the help menu, WinWedge has complete context sensitive help that is available by pressing function key F1.
See Also: Accessing On-Line Help (pg. 49)

The File Menu

Selecting "File" from the WinWedge 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 WinWedge to its default configuration.

Open - Opens a window that allows selection of a previously saved WinWedge configuration file.

Save - Saves changes to the currently open WinWedge configuration file.

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

Exit - Exits WinWedge.

WinWedge Configuration Files

When you configure WinWedge 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 WinWedge with a particular device, you can simply run WinWedge and load in your previously saved configuration file and then activate WinWedge. This saves you the trouble of having to configure WinWedge each time you want to use it. If you save your configuration files to your Windows Desktop folder, you can even launch and activate WinWedge with the saved configuration file by double clicking on the file's icon on your desktop without ever having to open WinWedge first.

Activating WinWedge Automatically With a Specific Configuration File

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

If the filename for a WinWedge configuration file is supplied on the command line used to launch WinWedge, the specified configuration file will be loaded automatically and WinWedge will activate itself using that configuration. The filename must be the name of a valid WinWedge configuration file complete with the filename extension and also the drive and directory path for the file if it is not in the same file folder with the WinWedge program files.

The filename extension ".SW3" is the default filename extension for all WinWedge 3.x configuration files. The installation program for WinWedge (SETUP.EXE) automatically establishes an "Association" between the filename extension ".SW3" and the executable program "WinWedge.exe". This allows you to launch WinWedge configuration files directly without having to go through the process of launching WinWedge, opening the configuration file and then activating WinWedge.

For example, after you create and save a configuration file for WinWedge, you can simply double click on the name of the configuration file in the Windows Explorer and Windows will automatically launch WinWedge and activate it using the selected configuration. You could also create shortcuts to your WinWedge configuration files and then drag and drop the shortcuts onto your desktop. This would place an icon on your desktop that you could use to launch WinWedge with the particular configuration file.

Activating WinWedge Automatically When You Start Windows

To load and activate WinWedge automatically with a particular configuration file when you start Windows, open the Windows Explorer and select a previously saved WinWedge configuration file and drag and drop the configuration file into the folder named "StartUp". The "StartUp" folder is usually located in the "Programs" folder inside the "Start Menu" folder in your "Windows" folder. From this point on, whenever you start Windows, WinWedge will automatically load and activate itself with the selected configuration file.

Note: WinWedge can be configured to automatically minimize itself to the system tray on your taskbar therefore if you place a WinWedge configuration file in your StartUp folder that has the "auto-minimize option enabled, all that the user will see is a small icon in the system tray after WinWedge has been launched and activated.

See Also: The Preferences Window (pg. 42)

The Mode Menu

WinWedge can operate in one of two "modes" that control how it will transfer incoming data from your serial device to another application program. WinWedge can transfer data either by converting the data to keystrokes and then sending the keystrokes directly to another application or it can use Dynamic Data Exchange (DDE) to pass data to any application that supports DDE.

The two choices in the MODE menu, "Send Keystrokes" and "DDE Server", allow for the transfer of incoming serial data directly into another running Windows program. Each of these two modes has its own unique advantages and the choice of which mode to use depends greatly on what you are trying to accomplish as well as on the capabilities of the application program that you will be sending data to. It is extremely important that you fully understand the differences between these two modes.

See Also: Sending Keystrokes vs. Dynamic Data Exchange (pg. 18).

Send Keystrokes Mode

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

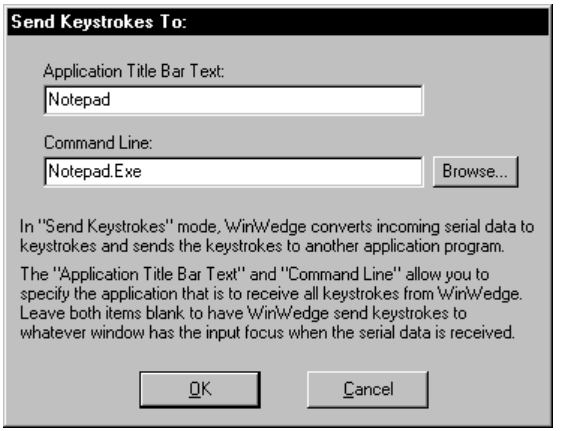
When you select "Send Keystrokes To..." from the Mode menu, a window will appear that lets you specify the application program that you want WinWedge to send the data to. WinWedge is pre-configured to send keystrokes to the NOTEPAD accessory that comes with Windows however you will probably want to change this to a different application program. 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 (the application's full pathname and EXE filename along with any command line arguments). If you specify only the title bar text, then the application must be running, either minimized or in a window, before WinWedge can send it any data. If you specify only the application's command line, then WinWedge will automatically launch the application before sending it any data.

After you activate WinWedge in “Send Keystrokes” mode, whenever data is received from your serial device, WinWedge will try to send the data to the application that you specify. The logic used by WinWedge is to first search for an application with the specified title bar text and if the application is already running, WinWedge 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 the title bar text is not specified, then WinWedge will try to run the program using the command line specified. If this is successful, then WinWedge 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 specify **Excel** for the title bar text and as long as there are no other programs running that contain the word "Excel" in their title bar, WinWedge should be able to find Excel’s window correctly. You may specify both the title bar text and command line in either upper or lower case.

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

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



DDE Server Mode

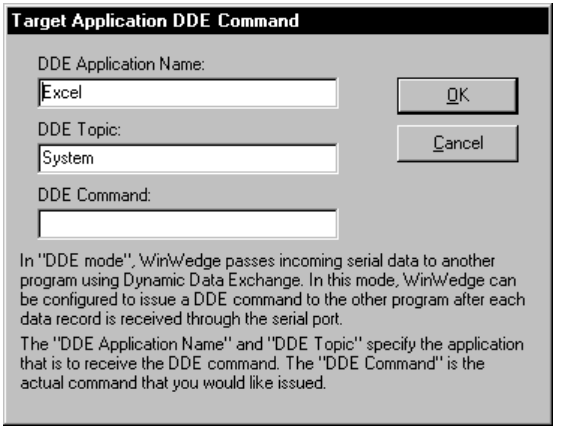
When WinWedge is in DDE Server mode, instead of sending incoming serial data to another application as keystrokes, WinWedge allows other Windows applications that support DDE to retrieve the incoming serial data through DDE (Dynamic Data Exchange) links.

When you select "DDE (Server)" from the Mode menu, a window will appear prompting for a DDE Application Name, a DDE Topic and a DDE Command. In addition to sending the serial data, WinWedge can be configured to issue a DDE command to another program after each input from a serial device is received. The DDE command is typically used to force the program that is receiving the data from WinWedge to perform an action like running a macro or subroutine. For example, if you were to specify **Excel** for the DDE Application Name, **System** for the DDE Topic and **[RUN("MyMacro")]** for the DDE Command, WinWedge would force Excel to run a VBA subroutine named "MyMacro" after each data record is received through the serial port by WinWedge.

The information that you supply in this window is used to specify the DDE Command Processor for the program that will receive the DDE command from WinWedge.
Note: This information is needed only if you intend to configure WinWedge to send a DDE command to the application, otherwise it can be cleared out or simply left as is.

See Also: Understanding Dynamic Data Exchange (pg. 56)
DDE And WinWedge (pg. 58)
DDE Examples (pg. 63)

The default DDE Application Name is "Excel" and the default DDE Topic is "System". These are the correct parameters for Microsoft Excel's DDE command processor. The DDE Application Name for MS Word is "WinWord" and its DDE Topic is also "System". For MS Access, the DDE Application Name is “MSAccess” and the DDE Topic is the name of your open Access database (without the file path or the .MDB filename extension). To find the Application Name and Topic for any other program's DDE command processor, refer to its user documentation. It is common for applications that support DDE commands to use the application's executable filename (without the EXE extension) as its DDE Application Name. It is also typical to use the DDE Topic "System".



Sending Keystrokes vs. Dynamic Data Exchange

The primary advantage of setting up WinWedge 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 configure WinWedge to send 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 directly below the previous input.

See Also: Specifying Pre/Postamble Keystrokes (Send Keystrokes Mode) (pg. 33)

A disadvantage of sending keystrokes is that the application that is receiving the data must be in the foreground and have the input focus before WinWedge 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 WinWedge (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 WinWedge 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 WinWedge issue a DDE Command after each input that causes Excel 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 WinWedge. This requires a small amount of programming in your application but 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: **WinWedge does <u>not</u> have to be in "DDE Server Mode" in order to accept and process DDE commands.** WinWedge will still execute any DDE commands sent to it even if it is in "Send Keystrokes Mode".

See Also: Understanding Dynamic Data Exchange (pg. 56)
DDE Examples (pg. 63)
WinWedge DDE Commands (pg. 59)

The Port Menu

The PORT menu options and their functions are:

Settings: Displays a window where you can select all serial communications parameters, (i.e. COM port, baud rate, parity, number of data bits, flow control, buffer sizes, etc.)
See Also: The Port Settings Window (pg. 20)

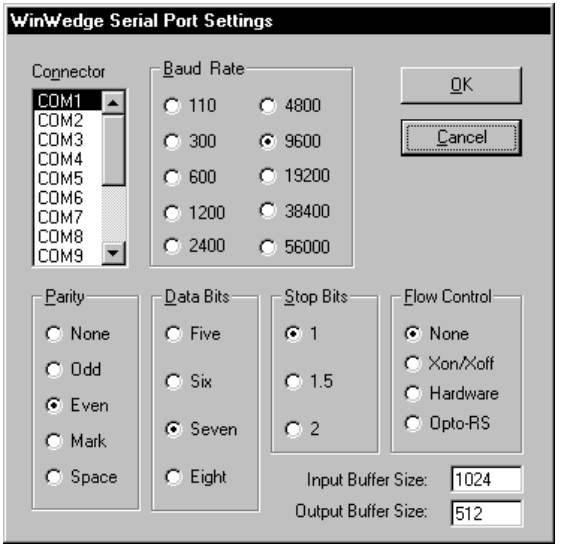
Analyze: This option displays a window that allows you to test all serial communications parameters and also view or analyze the structure and contents of data received from your serial input device. The Analyze feature is extremely useful for testing the communications link between your PC and your 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 WinWedge and your serial device before activating WinWedge from the Activate Menu.
See Also: The Port Analyze Menu Option (pg. 21)

See Also: Diagnosing Serial Communications Problems: (pg. 86)

The Port Settings Window

The Port Settings window allows you to select the serial communications parameters required by the device that you are using. All communications parameters selected in the Port Settings window must exactly match the settings for the serial device that you will be using. If you do not know the correct settings for your device, then you should contact its manufacturer for this information. For general instructions on how to determine the correct parameters using the Port Analyze feature of WinWedge, please refer to the section: Troubleshooting (pg. 87)

The Buffer Size options in the Port Settings window allow you to specify the size of the receive and transmit buffers. Buffer size values may range between 128 and 32000 bytes. The default input buffer size of 1024 bytes should be sufficient for most simple devices. If you need to input larger amounts of data with no flow control, you may need to increase the input buffer size to avoid losing data. The Output Buffer Size is relatively unimportant and should be left alone for most applications. You may see a slight performance increase in applications that transmit large amounts of data if you enlarge the output buffer size.

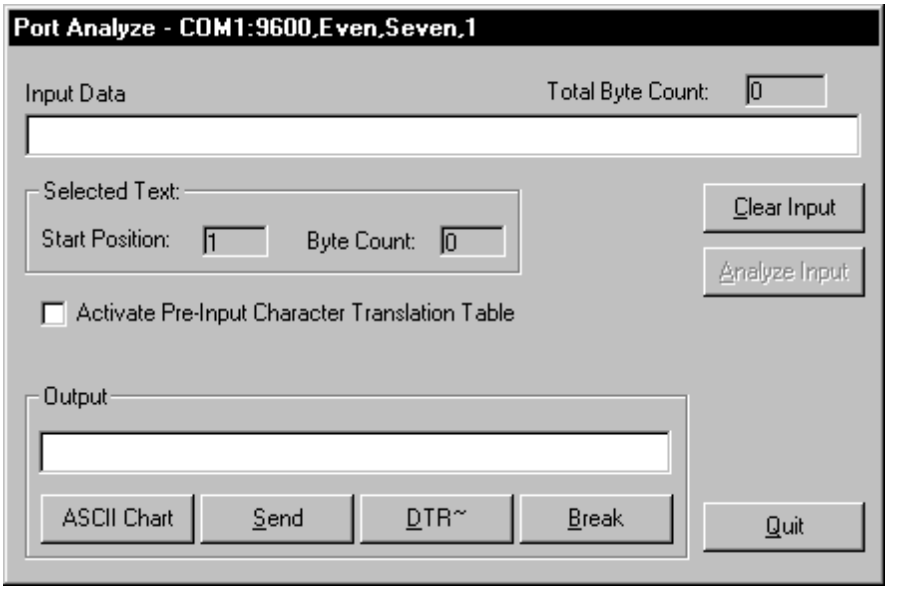


Note: If you need additional serial ports for your PC, TAL Technologies carries a line of inexpensive 2, 4 and 8 port add-on adapters. Please visit http://www.taltech.com or call 800-722-6004 for information and pricing. If you just need to add a single serial port to your PC, then one very simple way to do this is to purchase a "USB to Serial" adapter at any office or computer supply store. They typically cost under $30.00 and they come with a driver that you must install so that the new port will be recognized by Windows. USB to Serial adapters are typically assigned COM port numbers starting at COM3 or above. After you install the USB to Serial adapter's driver and connect it to your PC, you may need to look in the Windows Device Manager to find out what COM port number has been assigned to the adapter.

The Port Analyze Menu Option

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

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



The title bar of the Port Analyze window will show the currently active communications parameters (i.e. serial port, baud rate, parity, databits and stopbits). The "Input Data" text box displays all characters received from your serial device. All characters are shown using their displayable ASCII characters including control codes. (i.e. a carriage return (ASCII 13) 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. To display the ASCII chart, click the button marked "ASCII Chart".

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

The "Clear Input" button is used to clear the input data textbox and reset the total byte count to zero. Note: The Input Data text box can display a maximum of 32767 characters at a time.

An automatic analyze feature allows you to have WinWedge make an educated guess at the structure of your input data records. WinWedge can be configured to parse and filter your serial data, however in order to have it do this, 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 data textbox, WinWedge will offer a guess at the record structure and ask you if you would like to automatically pre-configure it using the guessed at record structure. Note: The capability of the auto analyze feature is limited to recognizing only the most common record structures and thus you may not always be able to use it to pre-configure WinWedge. 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. 25)
Understanding The Data From Your Serial Device (pg. 90)

When viewing data in the input data textbox, you can select portions of the data using your mouse or the cursor control keys. When you do this, the starting byte position of the selected data in the input data textbox is displayed next to the label "Start Position" and the 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.

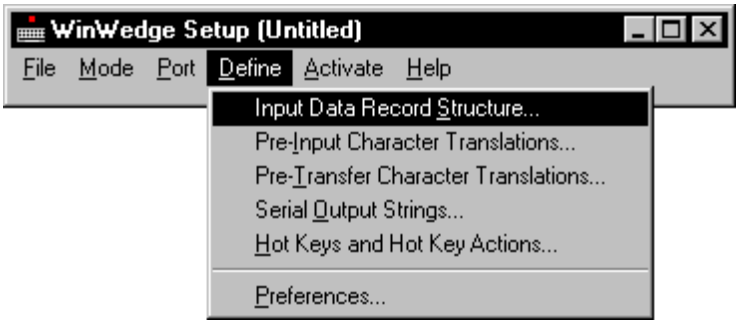
In the center of the Analyze window is a check box with the caption "Activate Pre-Input Character Translation Table". When this option is checked, all data received will be translated using the "Pre-Input Character Translation Table" before being displayed. If it is not checked, the data that appears will be the exact input from your serial device with no character translations.

See Also: Translation Tables (pg. 34)

Inside the frame with the caption "Output" is a text box and four buttons labeled "ASCII Chart", "Send", "DTR~" and “Break”. The text box is an output buffer where you can enter text or control codes to be sent out to your serial device. Many devices allow (or require) you to send them command strings or control codes to request data. 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 transmit the data in the output text box out the serial port, 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. The button marked "Break" can be used to issue a serial Break signal. These buttons are provided because some types of serial devices interpret the toggling of the DTR line or the issuing of a Break signal as a request for data.

The Define Menu



The options in the DEFINE menu and their functions are:

Input Data Record Structure

This option presents a series of windows that allow you to define the structure of the input data from your serial device. The windows presented also allow you to specify how you would like WinWedge to parse and filter the incoming serial data before transferring it to another application. When you define the input data record structure, you can also add additional data or keystrokes to be issued following the input of each data field.

See Also: Defining The Input Data Record Structure (pg. 25)

Pre-Input Character Translations

Displays a translation table that allows you to translate incoming ASCII characters to other characters before they are acted upon by WinWedge. This is an extremely powerful concept that allows a great deal of flexibility in the way incoming data is parsed or filtered by WinWedge.

See Also: Translation Tables (pg. 34)

Pre-Transfer Character Translation Table

Displays a translation table for translating characters to keystrokes when WinWedge is in *Send Keystrokes* mode or for translating characters to other characters when WinWedge is in *DDE Server* mode. Pre-Transfer translations are performed after WinWedge has parsed and filtered an incoming data record, just before the data is transferred to another application program.

See Also: Translation Tables (pg. 34)

Serial Output Strings

Opens a window where you can pre-define character strings to be transmitted back to your serial device by WinWedge. 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.
The output strings are typically used to prompt a device or control an instrument that supports two way communications.

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

Hot Keys and Hot Key Actions

This option presents a window where you can define up to 50 different hot keys that can be used to perform any of thirteen different actions including transmitting strings out the serial port, toggling the state of the RTS or DTR serial I/O lines and enabling, disabling and resetting WinWedge.

See Also: Defining Hot Keys and Hot Key Actions (pg. 40)

Preferences

Allows you to choose various preferences including options that control how the WinWedge window will appear after it is activated.

See Also: The Preferences Window (pg. 42)

Defining the Input Data Record Structure

Most serial devices transmit data records in a well-defined *structure* with each record containing one or more parts or *fields* of data. Because WinWedge 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 WinWedge. The *Input Data Record Structure* option in the “DEFINE” menu allows you to first define the overall record structure of your data and then define how WinWedge 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 additional data or keystrokes to the serial data before or after each data field is sent from WinWedge to your target application program.

When defining the input data record structure, several basic descriptors must be specified starting with the events that determine the start and end of each data record. When you select *Input Data Record Structure* from the Define menu, the following window appears:

The Start Of Record Event

The *Start Of Record Event* options specify the event that triggers WinWedge to start accepting characters as part of each new data record. If you select *Any Character Received* as the start of record event, the first character received will be considered to be the start of each record. All subsequent characters received will be treated as part of the record until an *End Of Record Event* occurs. The *Alpha/Numeric Char Received* option instructs WinWedge to ignore all input data until a valid Alpha/Numeric character is received (i.e. non control characters or chars with ASCII codes greater than 31 and less than 127). The *Numeric Char Received* option causes WinWedge to wait until a numeric character is received (i.e. 0123456789.-).

The *Special Char Received* option allows you to specify which characters are to signal the start of each new data record. If you use this option, you may enter one or more characters in the text box next to this option. To enter control codes in the text box, press the *ASCII Chart* button and choose the specific character(s) from the displayed ASCII Chart. If you enter more than one character, the reception of any of the characters will trigger the start of a record. For example, if you specify "AB", either an "A" or a "B" will signal the start of a record (not the complete string "AB").

See Also: Pre-Input Character Translation Table (pg. 34)

The End Of Record Event

Similar to the *Start 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. WinWedge 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 data record ends.

If each data record is terminated by a carriage return or carriage return linefeed, the *Carriage Return or CrLf Received* option should be chosen. Carriage returns and line feeds are **not** removed automatically if this option is chosen. If you do not want these characters as part of your data, use the "Pre-Transfer Character Translation Table" to remove them by translating them to "Nul".

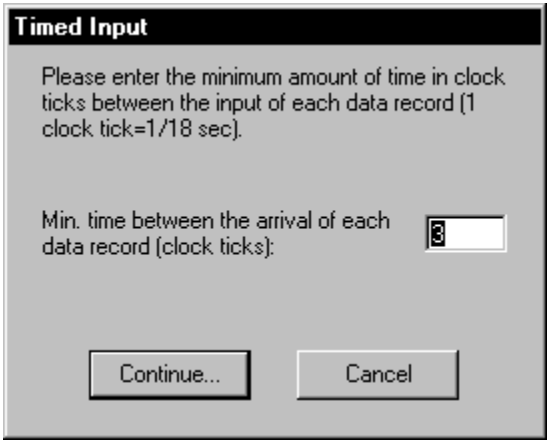
See Also: Translation Tables (pg. 34)

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 window to enter the complete record length for your data records.

See Also: Pre-Input Character Translation Table (pg. 34)
The Port Analyze Menu Option (pg. 21)

If your incoming serial data is not always transmitted as fixed length strings and there also is no specific character within the data that can be used to signal the end of each data record but the data is transmitted in bursts such that there will always be a small time delay between data inputs, then *Time Delay Between Records* should be chosen.
Note: 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 to accurately determine the end of an input data record. If you specify too large a value, then you risk the possibility of two or more inputs being interpreted as one long input. The default value of 3/18 of a second is almost always the best value to use for the types of devices that WinWedge is typically used with.

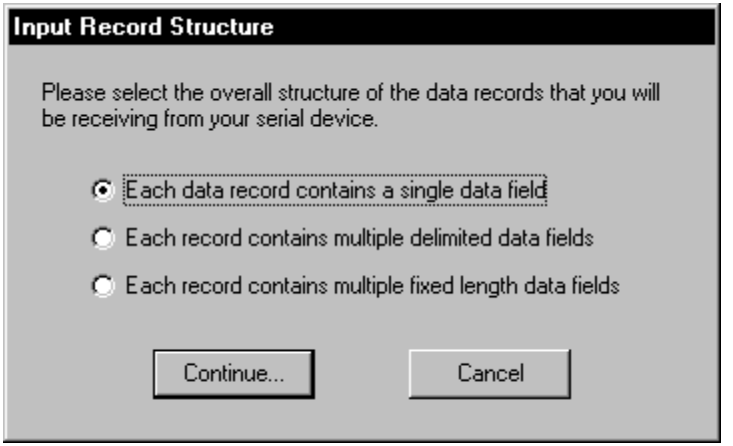


The *Special Char Received* End of Record Event option allows you to specify characters that will signal the end of an input record. If you choose this option, you may enter one or more characters in the text box next to this option. To enter control codes in this text box, press the "ASCII Chart" button and choose the characters from a displayed ASCII Chart. The reception of any of the characters entered will signal the end of an input data record.

NOTE: Special characters are **not** automatically removed if this option is chosen. If you do not want these characters to appear as part of your data, then you can use the "Pre-Transfer Character Translation Table" to remove them by translating them to "Nul".

See Also: Pre-Transfer Character Translation Table (pg. 36)

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



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

Single field data records

The "Each data record contains a single data field" option means that each entire data record should be considered as a single entity thus no parsing should be performed on the data.
Note: The maximum length for a single data field is 32766 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 WinWedge. 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 WinWedge.

See Also: Understanding the Data From Your Serial Device (pg. 90)
Pre-Input Character Translation Table (pg. 34)
Sending Keystrokes Vs. Dynamic Data Exchange (pg. 18)

Multiple delimited data fields

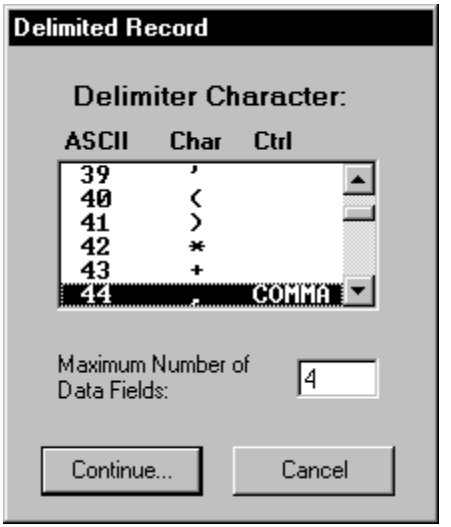
The "Multiple Delimited Data Fields" record structure option specifies that your data records consist of two or more unique *fields* that should be parsed based on the position(s) of a delimiter character within each record. For example the following data record consists of four data fields delimited by commas and with a carriage return linefeed at the end of the record:
12345, 9090, Value=123, 98765<CrLf>

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

WinWedge 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.

Note: If you are trying to interface a device that outputs data records with more than 40 data fields per record, there are several methods that you can use to overcome the 40 field limit in WinWedge. When WinWedge is in "Send Keystrokes" mode, one method is to define the record structure as "Single Field" data records and then using the "Pre-Transfer Translation Table" to translate the delimiter characters to cursor navigation keystrokes such as the Right Arrow, Down Arrow, Tab or Enter keys. Another method involves passing each data record as a single data field (in either "Send Keystrokes" or "DDE Server modes") and then using the macro language in the target application to parse the data and put the individual data fields wherever you would like them to go.

See Also: Defining The Input Data Record Structure (pg. 25)
Overcoming the 40 field limit in WinWedge. (pg. 71)



Special Considerations Regarding Delimiter Characters

All delimiter characters will be removed from the input data and thus will not appear as part of each data field. If you choose a space character as your delimiter, WinWedge will treat strings of consecutive spaces as a single delimiter. Also, if a chosen delimiter character matches a character that signals the end of an input record, (for example, if you choose a carriage return as your delimiter character and you also chose "Carriage Return or CrLf Received" as your End of Record Event), WinWedge will treat the character as the delimiter until the specified number of data fields have been received after which the next delimiter/end of record character will signal the end of the record.

See Also: Defining The Input Data Record Structure (pg. 25)

For example, suppose you have a device that transmits two lines of data with a carriage return at the end of each line. If you specify the “End of Record Event” as “Carriage Return or CrLf Received” and also specify a carriage return as your delimiter and then specify “2” for the “Maximum Number of Data Fields”, WinWedge will treat the data from the first line as the first data field and the data from the second line as the second data field.

See Also: Understanding the Data From Your Serial Device (pg. 90)

In cases where your input data records contain two or more different delimiter characters, you can use the "Pre-Input Character Translation Table" to convert all delimiters to a single character. For example, consider the following record:

Instrument# 123, Reading#3=23.45<CrLf>

Normally this data record would be interpreted to contain two fields: "*Instrument# 123*" and "*Reading#3=23.45*", with a single comma delimiter between the two. If you were to use the Pre-Input Character Translation Table to convert both pound signs (#) and equal signs (=) to commas, then after the translation, the record would appear to WinWedge as:

Instrument, 123, Reading,3,23.45<CrLf>

The translated record thus now contains five individual, comma delimited, data fields.

See Also: Pre-Input Character Translation Table (pg. 34)
The Port Analyze Menu Option (pg. 21)

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 chose "Fixed Number of Bytes Received" as the End of Record Event.

See Also: Pre-Input Character Translation Table (pg. 34)

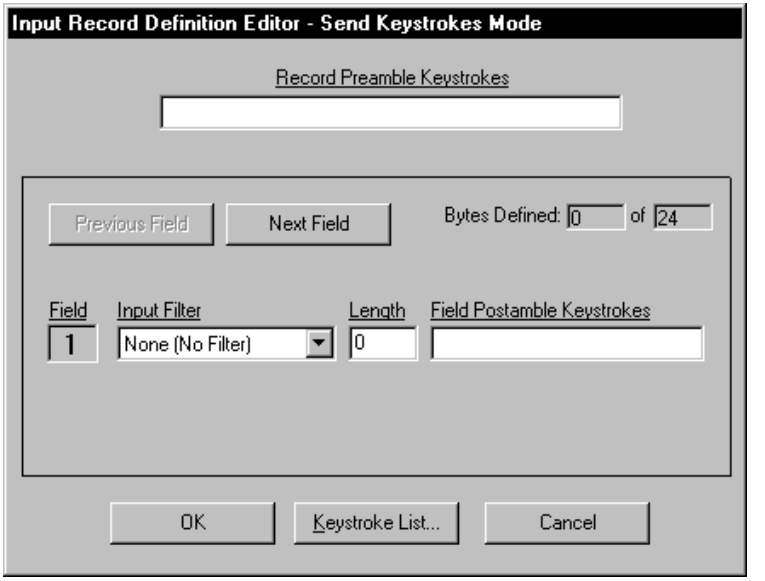
Specifying Filters and Field Lengths

After you specify the Start and End of Record Events and choose the basic structure of your data records, an "Input Record Definition Editor" window will appear. This window is where you specify a filter to be applied to each data field and the input length of each field.

If WinWedge is in "Send Keystrokes Mode", this window also allows you to specify a record *Preamble* and field *Postambles* that consist of additional keystrokes that are issued before or after each data field is sent to another program.

See Also: Specifying Pre / Postamble Keystrokes (Send Keystrokes Mode) (pg. 33)

Note: Some of the controls in the Input Record Definition Editor window 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.



The lower half of the window is where you specify an "Input Filter" to be applied to each data field in your input data records as well 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 that you are defining a Filter, a Length and a Field Postamble for. If your data records consist of more than one data field, there will be two buttons marked "Next Field" and "Previous Field" in the window. 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

The choices for the "Filter" that can be applied to each data field are; "None", "Ignore This Field", and "Numeric Data Only". 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. If you select "Ignore This Field", then no data is accepted and the entire field is ignored. The Ignore filter is thus used to ignore or remove data fields that are not required by your application. Specifying "Numeric Data Only" causes all Alpha characters and control codes to be filtered out and only allows characters: **0123456789-.** to be accepted as valid data.

The “Numeric Data Only” 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. The Numeric filter can also be used to reduce the number of data fields that need to be parsed as well as remove trailing carriage returns and linefeeds. For example, suppose you had a device that transmits the following comma delimited, carriage return terminated data records:

X=0123.45,Y=0321.22<cr>

If you are only interested in the two numeric values, you could define the record structure as "multiple delimited data fields" choosing the comma as the delimiter and specifying two as the maximum number of fields and finally applying a "Numeric Data Only" filter to each of the two data fields. In this case you would end up with the two numeric values only and the additional characters "X=", "Y=" and the carriage return (<cr>) would be removed by the numeric filter.

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 data field in the input record. Field lengths may range from 1 to 32766 bytes. For fixed length data records, the sum total of all field lengths entered should equal the total length of each input data record.

See Also: Pre-Input Character Translation Table (pg. 34)

In the middle toward the right 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 is only visible when the end of record event is set to "Fixed Number of Bytes Received".

Specifying Pre / Postamble Keystrokes (Send Keystrokes Mode)

When WinWedge 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 window. Also, for each data field that you have defined, you may enter a sequence of "Field Postamble Keystrokes". Preamble and Postamble keystrokes are additional keystrokes that you would like issued before or after each data field is sent to another Windows application. They are typically used to control the receiving application or send cursor navigation keystrokes before or after each data field so that the serial data ends up where it is supposed to go.

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

The "Field Postamble Keystrokes" are issued immediately following the data for each particular data field. Field Postamble keystrokes are usually used to navigate around in the application that is receiving the data. For example, if you are reading data from a device into a spreadsheet and you want successive readings entered in a single column, you could issue a field postamble keystroke consisting of a {DOWN} arrow. This would cause the cursor to move to the next cell down in the spreadsheet after each input from the device. You define Record Preamble and Field Postamble keystrokes to mimic the keys you would press if you were typing the data manually on the keyboard

When specifying Record Preamble and Field Postamble keystrokes, you must follow the rules outlined in the section: Keystroke Macro Rules (pg. 50)

The button marked *Keystroke List* allows you to select individual keystrokes from a list while editing preamble or postamble keystrokes. All keystrokes in the list conform to the proper Keystroke Macro Rules. Preamble and Postamble Keystroke macros may also contain special date and/or time stamps.

See Also: Date and Time Stamps In Macros (pg. 51)

Translation Tables

WinWedge provides two translation tables; a *Pre-Input Character Translation Table* and a *Pre Transfer Character Translation Table*. The two translation tables allow you to translate characters received from your serial device at two different points, before the data is acted upon by WinWedge, and just before data is transferred to another application program - after the data has been parsed and filtered.

Pre-Input Character Translation Table

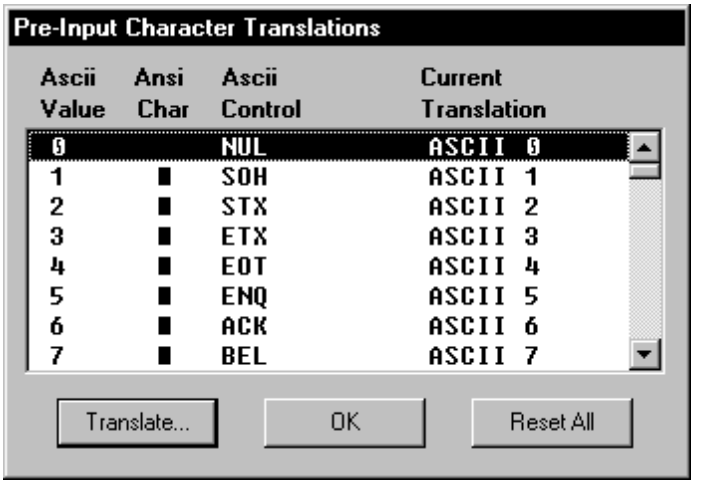
The Pre-Input Character Translation Table allows you to translate incoming ASCII characters to other ASCII characters before they are acted upon by WinWedge. This is an extremely powerful concept because it can dramatically alter the operation of WinWedge as well as the way in which you think about your serial data when you configure WinWedge.

For example, suppose that your serial device transmitted data records containing multiple delimited data fields but some of the fields were delimited with commas and others were delimited with semi-colons. In this case you could use the Pre-Input Character Translation Table to translate semi-colons to commas and then define the "Record Structure" in WinWedge as "Multiple Delimited Data Fields" choosing the comma as the delimiter. In this case WinWedge would act on all semi-colons as if they were comma delimiters.

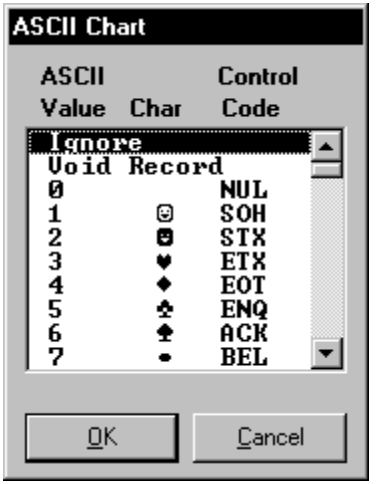
See Also: Special Considerations Regarding Delimiter Characters (pg. 30)

You can also use the Pre-Input Translation Table to strip out specific characters from the serial data or even remove an entire data record if it contains a specific character.

If you select *Pre-Input Character Translation Table* from the *Define* menu, the following window containing the translation table will appear:



The translation table shows the ASCII values for all possible characters as well as their corresponding ANSI characters, ASCII control code mnemonics and the current translation for each character. To translate characters, highlight the character that you need to translate and click the "Translate" button. This will cause the following ASCII Chart to be displayed where you can select a specific translation. The "Reset All" button in the translation table causes all translations to revert to their default settings.



The first two choices "Ignore" & "Void Record" in the ASCII Chart do not represent actual characters but instead have special meanings when selected.

"Ignore" causes the translated character to be ignored by WinWedge. When a character translated to "Ignore" is received, it will be discarded and thus it will not appear in the input data. Also, because the character is being ignored, its presence is not counted when reading data into a fixed length record or a fixed length data field. Thus, “Ignore” removes the character from the input data before it reaches WinWedge.

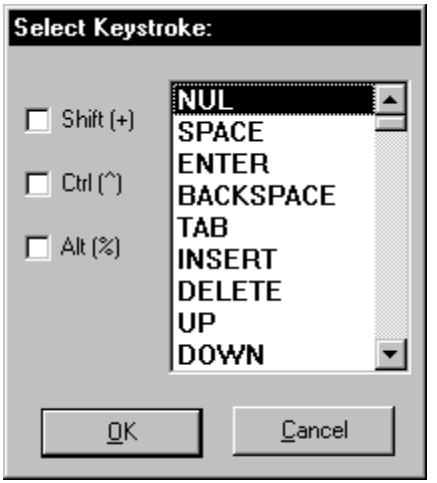
"Void Record" causes WinWedge to invalidate the entire current data record. If a character that has been translated to "Void Record" is present anywhere in an input data record, the entire record is completely ignored and will not be transferred to another application program or disk file. This feature can be useful in situations where you would like to reject certain data records from a device. For example, suppose you had a device that always transmitted a special "initialization record" each time the device is turned on. If the initialization record contained any character that would never appear in any following record, you could use the "Void Record" translation to remove the unwanted initialization record.

Pre-Transfer Character Translation Table

The Pre-Transfer Character Translation Table is used to translate characters after WinWedge has parsed, filtered and formatted your data; i.e. just before the data is sent to another application program.

The “Pre-Transfer Translation Table” is almost identical to the "Pre-Input Character Translation Table" except that this translation table will work differently depending on the current output mode that WinWedge is set for (i.e. "Send Keystrokes" or "DDE Server" Mode).

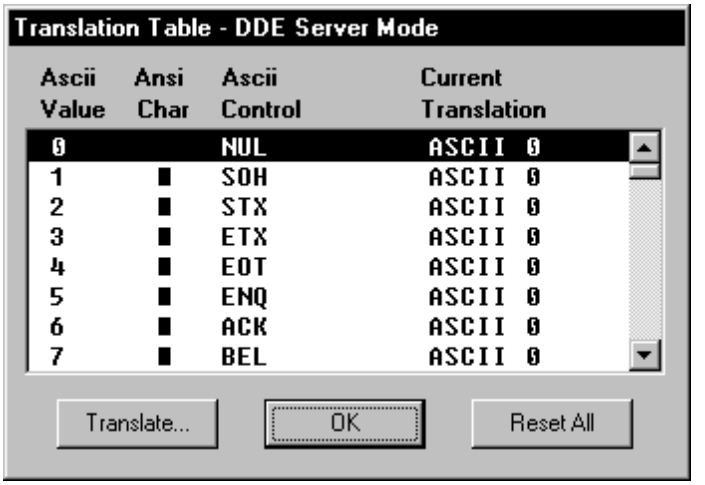
If WinWedge is in "Send Keystrokes Mode", then this translation table is used to convert individual characters to specific keystrokes. Serial data consists of ASCII characters. Because many ASCII characters do not correspond to a specific keystroke, the Pre-Transfer Character Translation Table provides a way to map characters to specific keystrokes. To translate characters, highlight the character that you need to translate and then click the "Translate" button. This will display the following "Keystroke Selection" window that allows you to choose a specific keystroke or key combination.



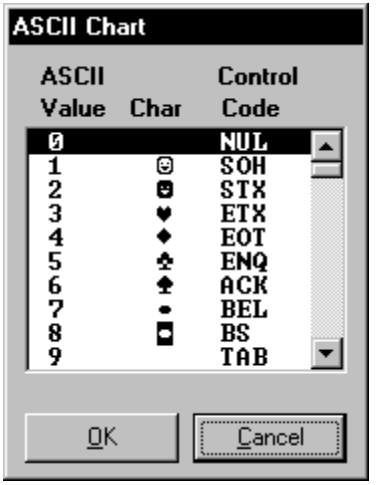
The keystroke selection window contains a list of all keystrokes as well as three check boxes that allow you to indicate the Shift, Ctrl, and Alt toggle key states to be used in conjunction with the selected keystroke.

The first item in the keystroke list, "NUL", does not represent an actual keystroke but instead is used to remove unwanted characters such as carriage returns and control codes or other characters before data is sent to another application program.

If WinWedge is in "DDE Server Mode" then the Pre-Transfer Character Translation Table will appear as shown below. This translation table allows you to translate characters to other ASCII characters.



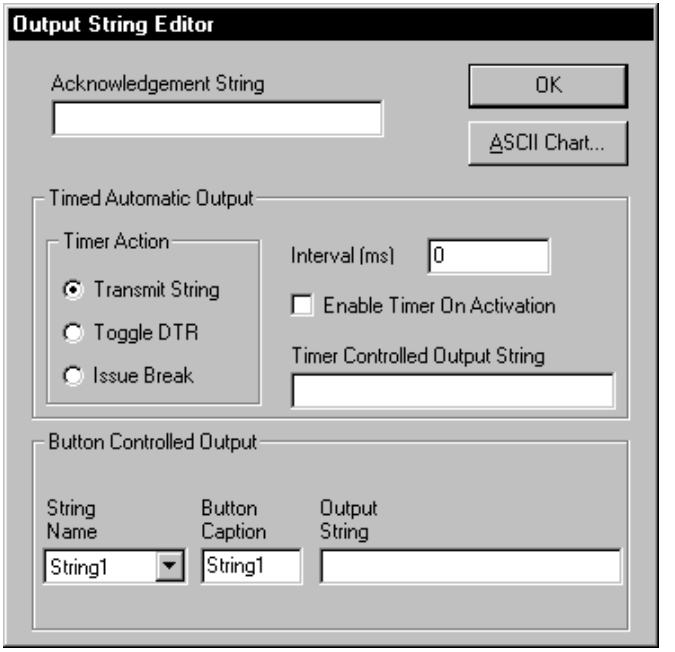
When you click on the "Translate" button to translate a character, the following ASCII Chart will appear allowing you to translate the selected character to another ASCII character. This ASCII Chart is similar to the one displayed for the Pre-Input Translation Table except that it does not have an "Ignore" or "Void Record" entry.



NOTE: When using WinWedge in DDE Server Mode, characters translated to "Nul" in the Pre-Transfer Character Translation Table will be removed from the outputted data. Translating characters to NUL can thus be used to remove unwanted characters.

Defining Serial Output Strings

Selecting "Output Strings" from the Define menu opens the window 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 WinWedge is active.



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 an acknowledgment (an ACK character for example) 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 (i.e. 1/20th of a second to once every 27 hours). A check box also allows you to specify if timed automatic outputs are initially enabled as soon as you activate WinWedge. If a timer controlled output string is defined, a menu item in the WinWedge window (displayed after WinWedge is activated) will allow you to enable or disable Timed Automatic Outputs. You can also define hot keys that can be used to enable and disable the sending of the timer controlled output string.

See Also: Defining Hot Keys and Hot Key Actions (pg. 40)

In the bottom of the Output String Editor window, you may also define up to 20 "Button Controlled Output Strings" (referenced as "String1" through "String20") that are each associated with a "button" in the WinWedge window after you activate WinWedge. Clicking your mouse on an output string button in the WinWedge window 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, many electronic balances can be prompted to transmit a weight reading by sending them a certain prompt string. For a Sartorius balance the prompt is an Escape character (ASCII 27) followed by a capital P. For a Mettler balance the prompt is a capital S followed by a carriage return (ASCII 13) and a linefeed (ASCII 10).

For this situation, you could define a button controlled output string containing the required prompt characters and also define its button caption as "Send Data". Then whenever you click your mouse on the button with the caption "Send Data" in the WinWedge window, the prompt string would be sent to the scale causing it to send back a weight reading.

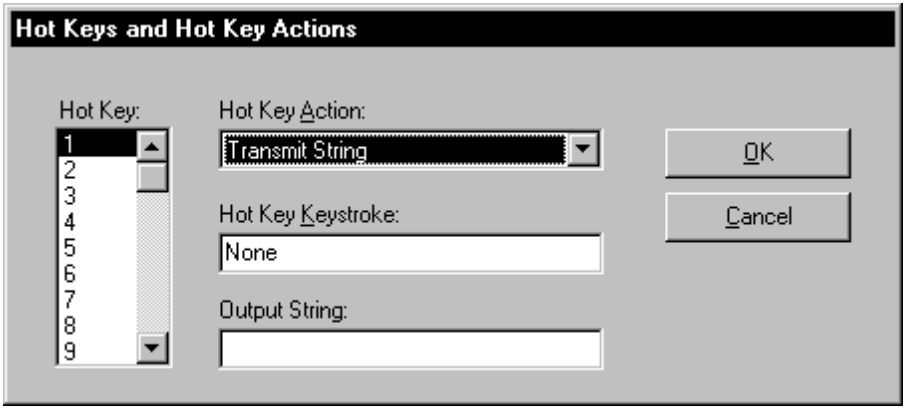
When defining a caption for a button you can assign an *access key* to the button by including an ampersand (&) in the caption immediately preceding the character you want to be used as the access key. This character will appear underlined in the button caption when WinWedge is activated. Pressing the underlined character in combination with the Alt key while WinWedge 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.

See Also: Defining Hot Keys and Hot Key Actions (pg. 40)

When editing any of the output strings in the Output String Editor window, an ASCII chart is available so that you can select ASCII characters or control codes that cannot be entered on your keyboard. For example, to enter a specific ASCII character in an output string, move the cursor to the point in the output string where you would like the character to go and then click your mouse on the button marked "ASCII Chart". When the ASCII chart appears, scroll down the list until the character that you want is highlighted and click the OK button in the ASCII chart. The character will appear where the cursor is in the output string.

Defining Hot Keys and Hot Key Actions

WinWedge allows you to define up to 50 hot keys that can be used to control several serial I/O functions and also control the operation of WinWedge. When you select “Hot Keys and Hot Key Actions” from the Define menu the following window will be displayed.



Each hot key that you can define has a number from one to 50 and you can select a particular hot key by choosing its hot key number from the list box with the caption “Hot Key”.

Selecting Hot Key Actions

You define a hot key by first selecting a *Hot Key Action* from the list of available actions and then you specify the keystroke that will be used to invoke the action. WinWedge supports the following 13 hot key actions:

	Hot Key Action	**Meaning**
1.	**Transmit String**	Transmits a character string out the serial port

When you select “Transmit String”, a text input box will appear in the bottom of the window with the caption “Output String”. This is where you enter the character string that you want transmitted out the serial port when the hot key is pressed. When editing the output string, a button with the caption “ASCII Chart...” will also appear in the window so that you can choose control codes (or ASCII characters that cannot be typed on your keyboard) that you want to place in the output string.

2.	**Issue BREAK Signal**	Issues a serial BREAK signal
3.	**Toggle DTR for 100ms**	Toggles the serial port DTR line for 100ms
4.	**Enable Timer**	Enables timer controlled outputs
5.	**Disable Timer**	Disables timer controlled outputs
		See Also: Defining Serial Output Strings (pg. 38)
6.	**Raise DTR**	Raises the serial port DTR line
7.	**Lower DTR**	Lowers the serial port DTR line
8.	**Raise RTS**	Raises the serial port RTS line
9.	**Lower RTS**	Lowers the serial port RTS line

10. Reset WinWedge Resets WinWedge
When you perform a reset, the serial input buffer is flushed and WinWedge is reset to the state it was in when it was first activated. See Also: Activating WinWedge (pg. 44)

11. Suspend WinWedge Suspends WinWedge
While suspended, WinWedge continues to collect serial data and store it to a serial input buffer however the data will not be transferred to another program until you resume WinWedge.

12. Resume WinWedge Resumes WinWedge if it is currently suspended
13. Resume for 1 Data Record Resumes WinWedge for a single data record only

If WinWedge is suspended and you resume it for one data record, the next data record available in the serial receive buffer will be processed after which WinWedge will automatically suspend itself again. The Resume for 1 Data Record action thus allows a greater degree of control over where and when data is entered into an application because it lets you accept data specifically when you are ready to receive it.

Selecting Hot Key Keystrokes

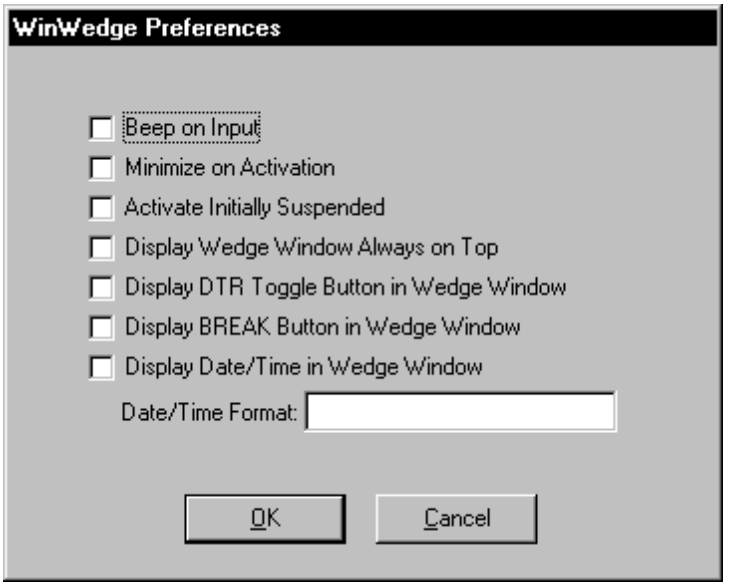
To select a hot key keystroke that will invoke a particular hot key action, place the cursor in the text box labeled “Hot Key Keystroke” and then press the key or key combination that you want to use. A description of the keystroke will appear in the text box after you press it. When the cursor is in the “Hot Key Keystroke” text box, a button will also appear with the caption “Keystroke List...”. Clicking this button pops open a list box so that you can choose your hot key keystroke from the list.

Notes: Hot keys defined in WinWedge are system wide hot keys and thus are always detected by WinWedge no matter what application has the input focus. All hot key keystrokes are also discarded after they are detected by WinWedge and thus are not passed through to the application you are working in when you press the hot key. It is also possible to define multiple hot key actions for the same hot key keystroke however if you do this, only the hot key action with lowest hot key number will be invoked.

Tech Tip: When using WinWedge in “Send Keystrokes” mode, it is possible for WinWedge to send itself hot key keystrokes. This interesting side effect could be taken advantage of to cause WinWedge to suspend or reset itself if a certain character is received. (Use the Pre-Transfer Character Translation Table to translate the character to a hot key keystroke whose action is to suspend or reset WinWedge). You could also continuously poll an instrument by having WinWedge issue a “Field Postamble Keystroke Macro” that contains a hot key that invokes the “Transmit String” action.

The Preferences Window

When you select the Preferences option from the Define menu, the following window will appear.



The **Beep On Input** option instructs WinWedge to beep your PC's speaker when data has been received and is about to be transferred to your application.

The **Minimize On Activation** option causes WinWedge to run minimized or as an icon when activated. When WinWedge is minimized, its icon will appear in the system tray on your Windows taskbar. To open the WinWedge window after it has been minimized, right click on the WinWedge icon in the system tray and select the option "Open WinWedge" from the menu that appears.

The **Activate Initially Suspended** option causes WinWedge to be suspended when it is initially activated. When WinWedge is suspended, it will still collect serial data and store it in a buffer until you resume WinWedge either by selecting a “resume” option from the activated WinWedge’s menu or by pressing a hot key or issuing a DDE command to WinWedge that causes it to resume its operation.

The **Display WinWedge Window Always On Top** option causes the WinWedge window to always be displayed on top of all other windows after you activate it. Having the WinWedge window always on top of other Windows can make it easier to click on any buttons in the WinWedge window and also to view the data that WinWedge is receiving.

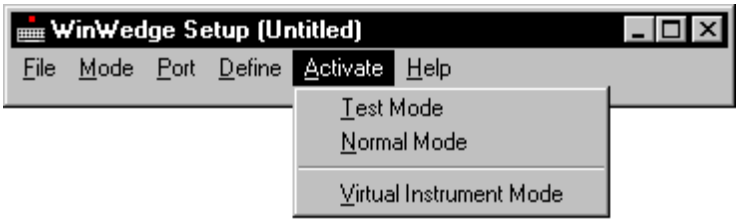
The **Display DTR Toggle Button** option causes a button to appear in the WinWedge Window with the caption "DTR~". This button is used to toggle the serial port Data Terminal Ready line for 100 ms. This action is interpreted by certain types of serial devices as a request to transmit a data record.

The **Display Break Button** option causes a button to appear in the WinWedge Window with the caption "BREAK". This button is used to issue a serial BREAK signal. Certain types of serial instruments use the Break signal as a request to either transmit a data record or perform a reset function. For example, most modems will hang up and reset themselves if a BREAK signal is issued to them.

The **Display Date/Time** option turns on and off a Date/Time display in the WinWedge Window. If you enable this option, you will also be prompted to enter a format expression for the displayed Date and or Time using formatting characters described in the section: Formatting Date / Time Expressions (pg. 52).

When WinWedge is activated with the Date/Time display enabled, a text box will appear in the WinWedge Window that contains the Date or Time formatted using the expression that you specify. If no formatting expression is specified, a default of "mm/dd/yy hh:nn:ss AM/PM" is used. The purpose of the Date/Time display is to provide a real time clock that can be linked via DDE to other Windows programs. The DDE Application name for the Date/Time display is **WinWedge** the DDE Topic is **COMn** where n is the serial port (1-16) that WinWedge is currently activated for, and the DDE Item name is **DATETIME**.

The Activate Menu



The "Activate" menu contains the entries, "Test Mode", "Normal Mode" and “Virtual Instrument Mode” that are used to activate WinWedge and start it working. WinWedge must be activated using one of these options before it will actually perform any serial I/O functions or transmit data to an application program. These three options and the differences between them are described in detail below.

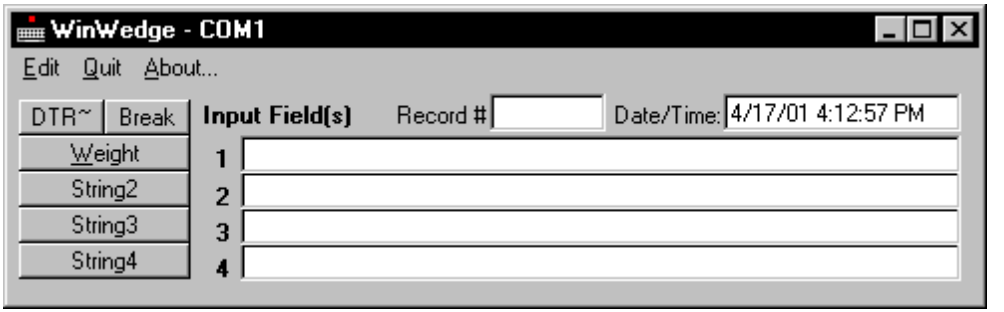
Activating WinWedge in Test Mode or Normal Mode

WinWedge 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 WinWedge in "Test Mode", you will be able to return to the WinWedge main menu and edit your configuration parameters. Test mode should always be selected when you are first configuring WinWedge 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 WinWedge in Test mode.

If you activate WinWedge in "Normal Mode", you will not be able to go back and edit any configuration parameters without quitting WinWedge and then restarting it and reloading your configuration file. When you activate WinWedge in "Normal Mode", its main menu and all sub menus, windows, and setup code are unloaded from memory in order to conserve 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 WinWedge Window

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



The title bar of the WinWedge Window will indicate the serial port that it is currently configured for.

The menu bar in the WinWedge Window contains the entries, Edit, Quit and About.

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

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

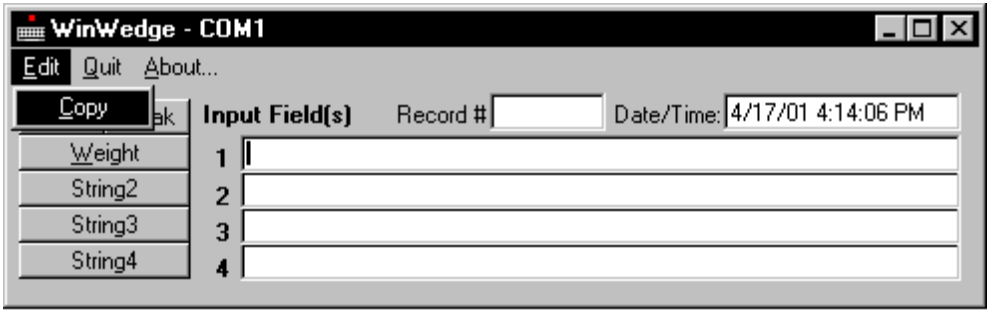
The button with the caption "DTR~" is called the "Data Terminal Ready Line Toggle Button". This button will appear only if the "Display DTR Toggle Button" option is selected in the Preferences window. Pressing this button causes the serial port Data Terminal Ready Line to be toggled from "on" to "off" for approx. 100 ms. This action is interpreted by certain types of serial devices as a request for data. Similar to the DTR toggle button, the “Break” button can be used to issue a BREAK signal to a device.

The “Record #” text box will display a count of the total number of input data records that have been processed since WinWedge was activated. The Record # value is stored as a double precision number which means that its value can go as high as ten raised to the 300^{th} power. (If you were to input data at a rate of 10 records per second, it would take several hundred years before the Record # would reach its maximum value.)

Note: The WinWedge Window can be minimized without affecting its operation. The only disadvantage of minimizing the WinWedge Window is that you must re-open it in order to press any output string buttons that you may have defined. If you want to run WinWedge minimized and still be able to send data out the serial port, you would have to use other features in WinWedge such as Hot Keys (pg. 40) or DDE Commands (pg. 59) to transmit data out the serial port.

The Edit Menu

The Edit menu contains a single *Copy* entry.



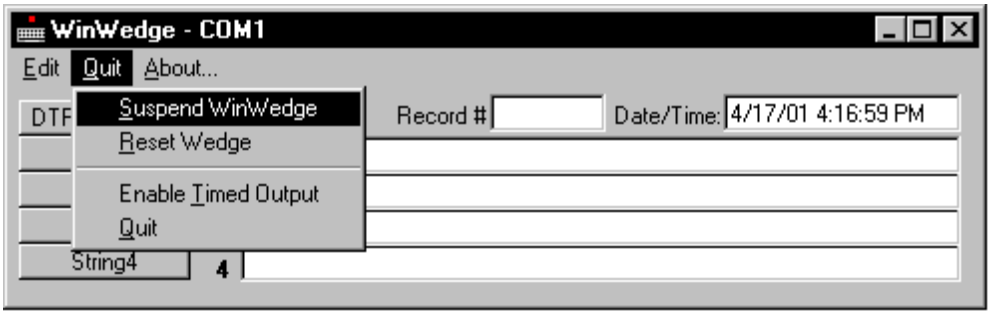
The *Copy* command allows you to copy the contents of one of your input data fields, the record number or the date/time display into the Windows clipboard. To use this feature, click your mouse in the text box that you want to copy and select "Copy" from the menu. Any data copied from WinWedge can be either *Pasted* or *Paste Linked* to other Windows applications.

See Also: Establishing DDE Links Using the Windows Clipboard (pg. 57)

The Quit Menu

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

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



Selecting "Suspend" suspends the operation of WinWedge and changes the Suspend menu item to "Resume" (which allows you to resume WinWedge). While WinWedge is suspended, the title bar will contain an S in parentheses (S) to remind you that WinWedge is currently suspended. When WinWedge is suspended, there will also be an additional Quit menu option with the caption “Resume For One Record” that allows you to resume WinWedge for a single data record after which WinWedge will be automatically re-suspended. This feature allows you to pull in data one record at a time.

See Also: Defining Hot Keys and Hot Key Actions (pg. 40)
WinWedge DDE Commands (pg. 59)

If you select Reset from the Quit menu, WinWedge will flush its serial receive buffer and also reset all internal variables to their default values. This has the same effect as quitting and then reactivating WinWedge.

If you quit WinWedge after activating it in "Test Mode", you will be returned to the main menu. If WinWedge was originally activated in "Normal Mode", WinWedge will quit running altogether.

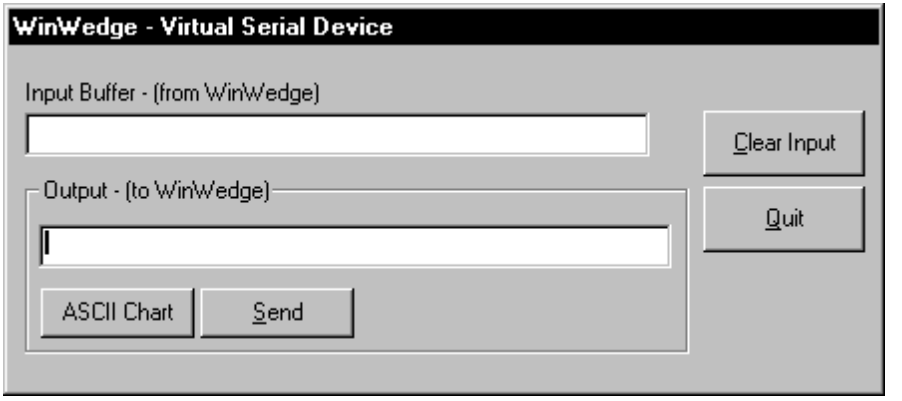
See Also: Activating WinWedge Automatically With A Specific Configuration File (pg. 14)

The *Enable/(Disable) Timed Output* option is used to turn on and off any automatic 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" thus allowing you to re-enable it.

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

Activating WinWedge in Virtual Instrument Mode

When you activate WinWedge in “Virtual Instrument mode”, it will behave exactly as it does when you activate it in “Test Mode” except that instead of communicating with a physical instrument connected to one of the serial ports on your PC, an additional window called a "virtual serial device" window will appear on your screen.



The virtual serial device window allows you to test the functionality of WinWedge without requiring any real hardware device to be connected to a serial port on your PC. The window has an “Input Buffer” that will display any data that would normally be transmitted from WinWedge out a serial port and it also provides an “Output Buffer” where you can enter and send data to WinWedge to simulate data that normally would be received from a device connected to an actual serial port.

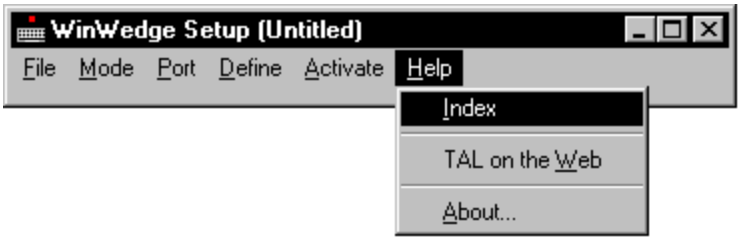
The virtual serial device window thus allows you test the functionality of WinWedge without requiring an actual instrument or device to be connected to your PC. To simulate a device, enter some typical data in the “Output Buffer” and click on the button marked “Send”. This will transmit the data from the “Output Buffer” to WinWedge just as if it were being transmitted through a serial port. To enter control codes or other characters that cannot be typed on your keyboard into the Output textbox, an ASCII chart is available by clicking the button marked “ASCII Chart”. Any data that is transmitted from WinWedge will appear in the Input Buffer.

The button marked “Clear Input” is used to clear out the contents of the Input Buffer text box and the button marked “Quit” will return you to the main menu of WinWedge.

Note: When you send data to WinWedge from the Virtual Serial Device window (and from a real instrument), WinWedge will not do anything with the data until the specified “End of Record Event” occurs. For example, if you specified the “End of Record Event” as a carriage return or a special character received, you should include these characters in any data that you transmit to WinWedge from the Virtual Serial Device window. You can enter a carriage return - linefeed pair (ASCII 13 & ASCII 10) in the Output textbox by pressing the Enter key or you can select these characters from the ASCII chart.

See Also: Defining the Input Data Record Structure (pg. 25)

Accessing On-Line Help



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

Context sensitive help is also available by pressing function key F1.

If you select the "TAL on the Web" option and your PC is connected to the Internet and you have a web browser installed, your web browser will open to the TAL Technologies web site (http://www.taltech.com). This web site contains a great deal of additional information about the WinWedge product line as well as an excellent technical support section where you can receive additional help and answers to common technical support questions relating to WinWedge.

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

Keystroke Macro Rules

When editing “Record Preamble” or “Field Postamble” keystroke macros with WinWedge 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 (~) parentheses () and square brackets [ ] have special meanings to WinWedge. To specify one of these special characters, enclose the character inside curly braces. For example, to specify the plus sign, use {+}. 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 Macros

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

Keyword	Function	Range
{Century}	Inserts the current century	19 - 99
{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.

Formatting Date/Time Expressions

Date/Time format expressions can be used to format the Date/Time display in WinWedge window and can also be used with the SENDDATE(format) DDE command.

See Also: The Preferences Window (pg. 42)
WinWedge DDE Commands (pg. 59)

To format dates and times, you can specify either the name of a commonly used format that has been pre-defined in WinWedge or you can create user-defined date/time formats using special formatting characters that have special meaning when used in a format expression. The following table shows the pre-defined data format names you can use and the meaning of each:

Format Name	**Description**
General Date	Display a date and/or time. For real numbers, display a date and time. (e.g. 4/3/93 05:34 PM); If there is no fractional part, display only a date (e.g. 4/3/93); if there is no integer part, display time only (e.g. 05:34 PM).
Long Date	Display a Long Date, as defined in the International section of the Control Panel.
Medium Date	Display a date in the same form as the Short Date, as defined in the International section of the Control Panel, except spell out the month abbreviation.
Short Date	Display a Short Date, as defined in the International section of the Control Panel.
Long Time	Display a Long Time, as defined in the International section of the Control Panel. Long Time includes hours, minutes, seconds.
Medium Time	Display time in 12-hour format using hours and minutes and the AM/PM designator.
Short Time	Display a time using the 24-hour format (e.g. 17:45).

The following table shows the characters you can use to create user-defined date/time formats and the meaning of each:

Character	Meaning
:	Time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator depends on the Time Format specified in the International section of the Control Panel.
/	Date separator. The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in the formatted output depends on Date Format specified in the International section of the Control Panel.
c	Display the date as ddddd and display the time as t t t t t, in that order. Only date information is displayed if there is no fractional part to the date serial number; only time information is displayed if there is no integer portion.
d	Display the day as a number without a leading zero (1-31).
dd	Display the day as a number with a leading zero (01-31).
ddd	Display the day as an abbreviation (Sun-Sat).
dddd	Display the day as a full name (Sunday-Saturday).
ddddd	Display a date serial number as a complete date (including day, month and year) formatted according to the Short Date setting in the International section of the Windows Control Panel. The default Short Date format is m/d/yy.
dddddd	Display a date serial number as a complete date (including day, month and year) formatted according to the Long Date setting in the International section of the Control Panel. The default Long Date format is mmmm dd, yyyy.
w	Display the day of the week as a number (1 for Sunday through 7 for Saturday).

ww	Display the week of the year as a number (1-53).
m	Display the month as a number without a leading zero (1-12). If m immediately follows h or hh, the minute rather than the month is displayed.
mm	Display the month as a number with a leading zero (01-12). If m immediately follows h or hh, the minute rather than the month is displayed.
mmm	Display the month as an abbreviation (Jan-Dec).
mmmm	Display the month as a full month name (January-December).
q	Display the quarter of the year as a number (1-4).
y	Display the day of the year as a number (1-366).
yy	Display the year as a two-digit number (00-99).
yyyy	Display the year as a four-digit number (100-9999).
h	Display the hour as a number without leading zeros (0-23).
hh	Display the hour as a number with leading zeros (00-23).
n	Display the minute as a number without leading zeros (0-59).
nn	Display the minute as a number with leading zeros (00-59).
s	Display the second as a number without leading zeros (0-59).
ss	Display the second as a number with leading zeros (00-59).
ttttt	Display a time serial number as a complete time (including hour, minute and second) formatted using the time separator defined by the Time Format in the International section of the Control Panel. A leading zero is displayed if the Leading Zero option is selected and the time is before 10:00 A.M. or P.M.. The default time format is h:mm:ss.

AM/PM Use the 12-hour clock and display an uppercase AM with any hour before noon; display an uppercase PM with any hour between noon and 11:59 PM.

am/pm Use the 12-hour clock and display a lowercase AM with any hour before noon; display a lowercase PM with any hour between noon and 11:59 PM.

A/P Use the 12-hour clock and display an uppercase A with any hour before noon; display an uppercase P with any hour between noon and 11:59 PM.

a/p Use the 12-hour clock and display a lowercase A with any hour before noon; display a lowercase P with any hour between noon and 11:59 PM.

AMPM Use the 12-hour clock and display the contents of the 1159 string (s1159) in the WIN.INI file with any hour before noon; display the contents of the 2359 string (s2359) with any hour between noon and 11:59 PM. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as it exists in the WIN.INI file. The default format is AM/PM.

The following are examples of user-defined date and time formats:

Format	**Display**
m/d/yy	12/7/58
d-mmmm-yy	7-December-58
d-mmmm	7 December
mmmm-yy	December 58
hh:mm AM/PM	08:50 PM
h:mm:ss a/p	8:50:35 p
h:mm	20:50
h:mm:ss	20:50:35
m/d/yy h:mm	12/7/58 20:50

Understanding Dynamic Data Exchange

Dynamic Data Exchange is a feature of Windows that allows two programs to pass data directly to each other or send commands directly to each other. DDE can be thought of as a direct conversation between two running application programs. In most cases, one application is providing some form of data 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, WinWedge is primarily a DDE Server that "serves" incoming serial data to other (client) programs. WinWedge can also act as a DDE 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 Name*, and a *DDE Item Name.*
The DDE Application Name is almost always the executable filename 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 identify the exact source of the data in a server application. (Sort of like a telephone number with the three parts: Area Code, Exchange and Number.)

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. Fortunately, most Windows programs that support DDE insulate the user 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 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 WinWedge and the spreadsheet cell. (Note: WinWedge must be running and activated on 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 WinWedge window will automatically appear in the "linked" cell in the spreadsheet. Also, whenever the data for Field(1) in WinWedge changes, the contents of the spreadsheet cell will automatically be updated with the new data. It is 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 WinWedge; 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. Closing either of the linked applications also causes all links between the two programs to be terminated.

DDE also allows a client application to send commands to a server. The types of DDE commands, if any, that a server program can accept will vary depending on the application and should be well documented in the application's user's manual along with the DDE Application Name and Topic Name required by client programs to establish the DDE link to the server's command processor. WinWedge supports over three dozen DDE commands that allow other applications to control WinWedge or transmit data out a serial port. All of these commands are described in detail in the section: WinWedge DDE Commands (pg. 59).

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 WinWedge 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 - Paste Link" 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.

DDE and WinWedge

WinWedge can function as both a DDE Server and a DDE Client in conversations with other applications. As a server, WinWedge can supply data from the "Input Data Fields", the Record Number and the Date/Time display field in the WinWedge window. It can also process DDE commands that are sent to it from another application.

As a client, WinWedge can also issue a DDE command to another application after each data record is received through the serial port. When you select "DDE Server" from the Mode menu, WinWedge will display a window where you can specify a DDE Application Name, a DDE Topic, and a DDE command to issue after each data record is received by WinWedge. The DDE command is typically used to force another application program to run a macro or subroutine after WinWedge receives each data record from a serial device. The purpose of the macro or subroutine would typically be to retrieve the serial data from WinWedge. The DDE command is therefore used as a means for triggering an "event" in another application program signaling that WinWedge has just received a new record of data through the serial port and that the data is available for the other program to retrieve.

See Also: DDE Server Mode (pg. 17)

Other applications can initiate DDE links with WinWedge using the DDE Application Name "**WinWedge**" and the DDE Topic "**COMn**" where "n" is the number of the serial adapter that WinWedge is activated for. The DDE Items available will be the contents of each defined input data field and the contents of the Date/Time and Record# display in the WinWedge window. The DDE Item names for the data fields are referenced as "**Field(1)**" through "**Field(n)**" where "n" is the highest defined field number. The DDE Item name for the Date/Time display is "**DATETIME**" and the DDE Item name for the Record# display field is "**RECORDNUMBER**". If you use the "Copy" & "Paste Link" method to initiate DDE conversations with WinWedge, this information will be automatically supplied to the client application and does not need to be explicitly specified.

To initiate DDE conversations with WinWedge from a macro in a spreadsheet or other application, you will have to refer to the user's manual or on-line help for the specific application to learn the syntax of the commands or macro instructions that can be used to initiate DDE links, send DDE Commands or perform DDE Requests. Many example are provided in the "DDE Examples" section of this manual and should therefore provide a good overview of how to communicate with WinWedge using DDE.

See Also: DDE Examples (pg. 63)

WinWedge DDE Commands

WinWedge supports all of the DDE commands listed below.
Note: DDE commands will only execute if WinWedge has been activated.

See Also : Activating WinWedge (pg. 44)
Understanding Dynamic Data Exchange (pg. 56)
DDE Examples (pg. 63)

Command	Function
[APPEXIT]	Exits WinWedge.
[APPMINIMIZE]	Minimizes WinWedge Window.
[APPNORMAL]	Opens WinWedge Window.
[BEEP]	Causes WinWedge to Beep once.
[BREAK]	Issues a Serial Break signal.
[CLEAR]	Clears all input data fields in WinWedge Window.
[COPYFIELD(n)]	Copies the data field "n" into the Windows clipboard.
[COPYDATE]	Copies the date/time display field into the clipboard.
[COPYRECNUM]	Copies the record number display field into the clipboard.
[SUSPEND]	Suspends WinWedge.
[RESUME1]	Resumes WinWedge for one data record.
[RESUME]	Resumes WinWedge.
[RESET]	Flushes all buffers and resets WinWedge.
[RTS=ON]	Sets the serial port RTS line on.
[RTS=OFF]	Sets the serial port RTS line off.
[DTR=ON]	Sets the serial port DTR line on.
[DTR=OFF]	Sets the serial port DTR line off.
[TOGGLEDTR]	Toggles the serial port DTR line for 100 ms.
[SEND(text)]	Sends the text in parentheses out the serial port.
[SENDCODE(n)]	Sends the character with ASCII code "n" out the serial port.
[SENDDATE(format)]	Sends the date or time out the serial port using a specified Date/Time format string.
[SENDFILE(filename)]	Sends the contents of the specified file 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 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).

[PUSH(button caption)] Pushes a button in the WinWedge 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 WinWedge.
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 in use.

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

[TIMERINTERVAL=n] Changes the timing interval (ms) for timed automatic output strings to the value n (between 1 and 99,999,999).

Note: All DDE commands must be enclosed in square brackets.

When sending DDE commands to WinWedge from another program, use the DDE Application Name "WinWedge" and the DDE Topic "COMn" where "n" is the number of the serial port that WinWedge has been activated for.

Using the LINKTEST utility to test DDE commands

WinWedge comes with a utility called LINKTEST that allows you to test all WinWedge DDE commands or test DDE commands that can be sent to other programs. To test any WinWedge DDE commands, WinWedge must be running and activated in either "Test Mode", "Normal Mode" or “Virtual Instrument Mode”.

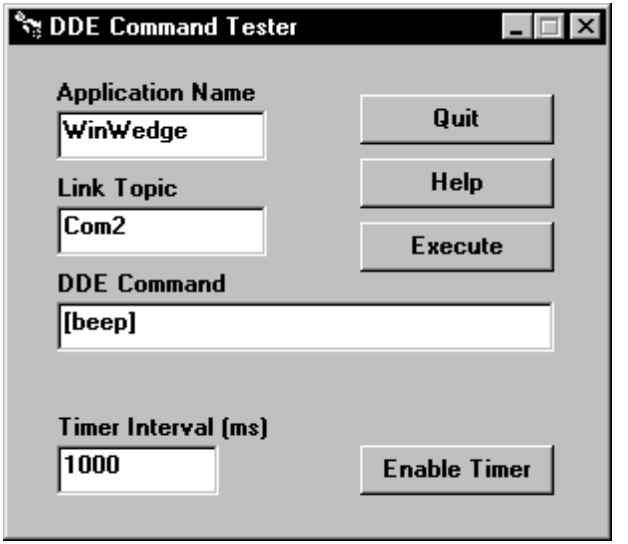
LINKTEST provides two text boxes where you specify the DDE Application Name and DDE Topic Name for the application that you want to send a DDE command to. The defaults for these parameters are those required by WinWedge when it is activated for COM2. If you are using a different serial port, you should change the DDE Topic Name 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 currently 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 WinWedge can recognize, click the button marked "Help" in the Link Test window.

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



Understanding How DDE Works Helps Avoid Common 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 sitting in memory dormant, waiting for event messages to be sent to it by the operating system. 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, other applications may not be able to process their own event messages until the first application finishes or until Windows specifically passes control to the application to let it process its events.

DDE provides 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 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. Suppose that you also want to write an Excel macro that will send a command to WinWedge 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 WinWedge. You might come up with the following Excel VBA subroutine that, although it appears to be quite logical, will not 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 WinWedge. The subroutine does not relinquish control and let WinWedge 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 WinWedge 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 and FoxPro). The examples demonstrate how to do things such as transmit and receive serial data using Dynamic Data Exchange with WinWedge or launch and unload WinWedge from memory as well as perform various other related functions.

The examples are designed to demonstrate how to perform the most common types of serial I/O operations with WinWedge using DDE. All of the examples listed in the following pages can be found in a file called **EXAMPLES.RTF** located in your WinWedge program directory along with all of the other WinWedge program files. In most cases you should be able to cut and paste the example code from the EXAMPLES.RTF file directly into the application 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.

Important:

Dynamic Data Exchange 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 examples provided below 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 WinWedge in "Send Keystrokes" mode may be more suitable. Using WinWedge in "Send Keystroke" mode is much easier and much quicker to configure than using WinWedge "DDE Server" mode.

For example if you are reading data from a simple instrument like a bar code reader, electronic balance or an electronic caliper and you 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 WinWedge using "Send Keystrokes" mode and also send data out the serial port by sending DDE commands to WinWedge. **WinWedge does not have to be in DDE Server mode in order to accept and process DDE commands**. You can still send DDE commands to WinWedge to have it transmit data or prompts out a serial port even when it is in "Send Keystrokes" mode.

See Also: A Typical WinWedge Configuration Example (pg. 9)
The Mode Menu (pg. 15)
Sending Keystrokes vs. Dynamic Data Exchange (pg. 18)
Understanding Dynamic Data Exchange (pg. 56)
WinWedge DDE Commands (pg. 59)

Note: TAL Technologies also maintains an extensive "support" section on our 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 (Visual Basic for Applications)

Excel Example #1 - Sending DDE commands to WinWedge from a VBA subroutine

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

```
Sub SendEscapeP()
Chan = DDEInitiate("WinWedge", "COM1")
DDEExecute Chan, "[SENDOUT(27,'P',13,10)]"
DDETerminate Chan
End Sub
```

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

```
Sub SendText(StringToSend$)
Chan = DDEInitiate("WinWedge", "COM1")
' The following line concatenates the SEND command with the data to be sent
DDEExecute Chan, "[SEND(" & StringToSend$ &")]"
' DDE commands are sent as text strings therefore we need to build our command.
' The & operator is the Excel string concatenation operator thus we are concatenating
' The three strings "[SEND(" , the data in the string variable StringToSend$ and ")]"
' Thus if StringToSend$ ="ABC" then the command that is 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 variable 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 new string containing
' the ASCII character values for each byte in the string separated by commas.
' This resulting string is then used as the argument to the SENDOUT command below.
' Ex: if the variable StringToSend$ = "ABC" then the variable Arg$ will be "65,66,67"
' See the syntax of the SENDOUT command in the WinWedge 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", "COM1")
' The following line concatenates the SENDOUT command with the data to be sent
DDEExecute Chan, "[SENDOUT(" & Arg$ &")]"
' The & operator is the Excel string concatenation operator thus we are concatenating
' The three strings "[SENDOUT(" , the data in the string variable 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
```

Excel Example #2 - Launching WinWedge from Excel

The following VBA subroutine executes WinWedge and passes it the name of a WinWedge configuration file (MyConfig.SW3) on the command line. This causes WinWedge to automatically load the specified configuration file and then activate itself in NORMAL mode.

```
Public Const MyPort As String = "COM1"          ' change port if necessary
Sub RunWedge()
On Error Goto ErrorHandler                      ' Set up an error trap
AppActivate "WinWedge - " & MyPort              ' try to activate the WinWedge Window
' the above line will generate an error if WinWedge is not running or activated
On Error Goto 0                                 ' remove error trap
AppActivate Application.Caption                 ' set the focus back to excel
Exit Sub

ErrorHandler:                                   ' WinWedge is not running - try to launch it
RetVal = Shell("C:\WINWEDGE\WINWEDGE.EXE C:\WINWEDGE\MyConfig.SW3")
If RetVal = 0 Then                                      ' launch failed
    MsgBox ("Cannot Find WinWedge.Exe")                 ' warn user and quit
    Exit Sub
Else                                                    ' launch succeeded
    Application.Wait Now + TimeValue("00:00:03")        ' give WinWedge time to load
End If
Resume Next
End Sub
```

Note: If you name an Excel subroutine "Auto_Open()", Excel will automatically run the subroutine whenever the spreadsheet that contains it is initially opened. The Auto_Open subroutine is an excellent place to launch WinWedge (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 after you open your workbook.
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 WinWedge 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
 Chan = DDEInitiate("WinWedge", MyPort) ' open a dde link with WinWedge
 DDEExecute Chan, "[Appexit]"           ' tell WinWedge to quit
 DDETerminate Chan
End Sub
```

Excel Example #3 - Requesting data from WinWedge using a VBA subroutine.

Steps for setting up WinWedge:

1. Select “DDE Server” from the WinWedge “Mode” menu. When the window appears asking for a Target Application DDE Command, enter: **EXCEL** as the DDE Application Name and then enter: **SYSTEM** as the DDE topic and finally, enter the string: **[RUN("GetSWData")]** as the DDE Command. This is a DDE command that will be sent to EXCEL after each data record is received by WinWedge. This command will force Excel to run a VBA subroutine called "GetSWData" which will read in the data from WinWedge and place it in a column in a worksheet.

2. Set up the rest of the WinWedge parameters as necessary for the device that you are using and then activate it.

Steps for setting up EXCEL:

1. Create a VBA subroutine and enter the following code: To create a macro select "Tools", "Macro" and "Macros..." then enter a subroutine name "GetSWData" and click the "Create" button.

```
Sub GetSWData()
Dim RowPointer As Long, Chan As Long, F1 As Variant, WedgeData As String
Dim MaxRows As Long
' Find the next empty cell in Column A looking from the bottom up
MaxRows = ThisWorkbook.Sheets("Sheet1").Rows.Count
RowPointer = Thisworkbook.Sheets("Sheet1").Cells(MaxRows,1).end(xlup).row + 1
Chan = DDEInitiate("WinWedge", "Com1")          ' establish DDE link to WinWedge on Com1
F1 = DDERequest(Chan, "Field(1)")               ' get Field(1) data into a variant array
WedgeData = F1(1)                               ' convert variant array to a string
Thisworkbook.Sheets("Sheet1").Cells(RowPointer, 1).Formula = WedgeData
' Write the data to cell address: (RowPointer,1) in Sheet1, i.e.  fill up Column 1 (Column A)
' If you are inputting multiple data fields from WinWedge you would duplicate the code above for
' each additional data field. For example, the following three lines pull in data from Field(2)
' F1 = DDERequest(ChannelNum, "Field(2)")       ' get Field(2) data into a variant array
' WedgeData = F1(1)                             ' convert variant array to a string
' ThisWorkbook.Sheets("Sheet1").Cells(RowPointer, 2).Formula = WedgeData ' put data in Col 2
DDETerminate Chan                               ' kill the DDE link
' Insert a date/time stamp in the sheet in column 3 of the same row as the data
ThisWorkBook.Sheets("Sheet1").Cells(RowPointer, 3).Value = Now
' YOUR CODE GOES HERE TO FURTHER PROCESS THE DATA IF NECESSARY
End Sub
```

The example above sets up WinWedge 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 WinWedge 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. This example assumes that the name of the worksheet is **Sheet1**.

Excel Example #4 - Requesting Data from WinWedge using VBA SetLinkOnData method. (And other Excel tricks)

The Excel VBA macro language has a function called "SetLinkOnData" that can be used to cause Excel to automatically run a subroutine whenever data from a DDE Server is updated. In the previous example we showed how to configure WinWedge 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 shows how to use the Excel SetLinkOnData method instead of having WnWedge send a command to Excel to trigger the macro to run.

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

One of the benefits of the following method is that WinWedge does not have to be in "DDE Server Mode" in order to have Excel run the macro after each data record is received by WinWedge. Although WinWedge has an explicit "DDE Server Mode", it is actually a DDE Server no matter what mode it is in (i.e. "Send Keystrokes" or "DDE Server" modes). The reason that WinWedge has a DDE Server Mode at all is so that you can configure it to send a DDE command to another application after each input. Suppose that you wanted to send the serial data as keystrokes to one application *and* pass the data to Excel via DDE in real time. You could set up WinWedge in Send Keystrokes mode and use the SetLinkOnData method in Excel to trigger a macro that would pull the data from WinWedge into a spreadsheet. Another benefit of this method is that the data from WinWedge will be transferred to Excel slightly faster than the it would if WinWedge 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 WinWedge automatically when you open up your spreadsheet as well as how to quit WinWedge automatically when you close your spreadsheet.

To use the following macros, set up WinWedge 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 device. If you set up WinWedge in DDE Server mode, make sure that you do not have a "DDE Command" defined. Next, activate WinWedge 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. WinWedge should automatically load and activate itself when you open the sheet.

```
Global RowPtr As Long, ColPtr As Long   ' define global variables
Global MyPort As String
Global Const CmdLine = "C:\WinWedge\WinWedge.Exe C:\WinWedge\Config.SW3"
' change the above command line to point to your copy of WinWedge and your config file

Sub Auto_Open()          ' this sub runs automatically when you open the spreadsheet
  FindWedge              ' check if WinWedge is already running
  If MyPort = "" then                ' WinWedge is not running - try to launch it
     On Error Resume Next            ' ignore errors and try to launch WinWedge
     RetVal = Shell(CmdLine)         ' launch WinWedge using command line defined above
     ' the above line launches and activates WinWedge with a config file: Config.SW3
      If RetVal = 0 Then             ' error - WinWedge not found
        MsgBox ("Cannot Find WinWedge.Exe")              ' display warning
        Exit Sub                                         ' and quit
     End If                                              ' otherwise -
     Application.Wait Now + TimeValue("00:00:04")        ' give WinWedge time to load
     AppActivate Application.Caption                     ' set the focus back to Excel
     FindWedge                       ' find out which port WinWedge was activated for
  End If
StartCollecting                      ' set up Excel to collect data from WinWedge
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 & "!RecordNumber"
ActiveWorkbook.SetLinkOnData "WinWedge|" & MyPort & "!RecordNumber", _
     "GetSWData "
' the SetLinkOnData method causes Excel to run the GetSWData macro automatically when new
' data is available in WinWedge (when the RecordNumber is updated). Using the SetLinkOnData
' method in Excel eliminates the need to have WinWedge send a DDE Command to Excel to cause
' it to run the GetSWData sub. The RecordNumber DDE item is updated after all other DDE data
' items (the input data fields) have been updated.
End Sub

Sub Auto_Close()         ' this macro runs automatically when you close the spreadsheet
 Sheets("Sheet1").Activate                ' activate sheet1
 ActiveWorkbook.SetLinkOnData "WinWedge|" & MyPort & "!RecordNumber", ""
' shut down the SetLinkOnData function by assigning it an empty string to the macro name
 Sheets("Sheet1").Cells(1, 50).Formula = ""          ' remove the dde link from R1C50
 On Error Resume Next                    ' ignore errors
 Chan = DDEInitiate("WinWedge", MyPort)' open a dde link with WinWedge
 DDEExecute Chan, "[Appexit]"            ' tell WinWedge to quit
 DDETerminate Chan                       ' terminate the DDE link
End Sub
```

```
Sub GetSWData()
Dim VArrray As Variant, MyStiring As String, Chan As Long
ColPtr = 1
RowPtr = RowPtr + 1                          ' point to the next row down
' 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 WinWedge
VArray = DDERequest(Chan, "Field(1)")        ' get Field(1) from WinWedge
MyString = VArray(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
' Add your code here to do something with the data maybe?
' or possibly send a command back to WinWedge as in the following line
' DDEExecute chan, "[Beep]"  ' send a beep command to WinWedge

ThisWorkBook.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
End Sub

Sub FindWedge()  ' This sub tries to find WinWedge and identify  the port that it is activated on
  MyPort = "COM1": CurrentPort = 1           ' start with COM1
  On Error GoTo errhandler                   ' AppActivate causes an error if WinWedge is not active
  AppActivate "WinWedge - " & MyPort         ' try to set focus to WinWedge
  On Error GoTo 0                                  ' disable error trap
  AppActivate Application.Caption                  ' set the focus back to Excel
  Exit Sub                                         ' quit & return to caller

errhandler:
 CurrentPort = CurrentPort + 1               ' AppActivate generated an error - try next com port
 MyPort = "COM" + CStr(CurrentPort)          ' set MyPort to the new port (COM2, COM3, etc...)
 If CurrentPort > 8 Then                ' if ports 1-8 have been tested and WinWedge not found
      MyPort = ""                            ' set MyPort="" to indicate WinWedge was not found
      Resume Next                            ' resume at next line after AppActivate statement
 End If
 Resume                                      ' try next port - resume at AppActivate Statement
End Sub
```

Excel Example #5 - Overcoming the 40 field limit in WinWedge

Excel 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 serial data from WinWedge 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 WinWedge 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 WinWedge 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 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 = 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 unless the string is delimited with tabs (see the following section "Advanced Excel Tricks" pg. 73). 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 WinWedge. To get past the 40 field limit in WinWedge, instead of using the parsing capabilities in WinWedge, 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 WinWedge 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 WinWedge as "Carriage Return or CrLf Received" 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 first step is to select “DDE Server” from the WinWedge “Mode” menu. When the window appears asking for a Target Application DDE Command, enter: **EXCEL** as the DDE Application Name and then enter: **SYSTEM** as the DDE topic. and finally, enter the string: **[RUN("NewGetSWData")]** as the DDE Command. This command will cause Excel to run a subroutine named "NewGetSWData" after each complete data record is received by WinWedge.

To configure WinWedge to input the data in a way that the can be used by the Excel subroutine, choose "Input Data Record Structure" from the Define menu and then select "Carriage Return or CrLf Received" as the "End of Record Event". Click the Continue button and 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 WinWedge. For example if the largest record transmitted by the device is 1125 bytes, then you would need to define five 250 byte data fields. Note: Be sure not to apply any Filters to any of the data fields that you define.

The following subroutine performs the job of pulling the data in from WinWedge 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 row of the spreadsheet.

```
Sub NewGetSWData()      ' this sub is run by WinWedge after each data record is received
Static R As Long, C As Long ' R & C point to the Row and Column where the data will go
If (R = 0) Or (C = 0) Then R = 1: C = 1                  ' make sure neither R or C is zero
Chan = DDEInitiate("WinWedge", "Com1")           ' change the serial port if necessary
' assume that there are 5 data fields defined - change to more or less if necessary
Field1 = Application.DDERequest(Chan, "Field(1)")          ' get field 1
Field2 = Application.DDERequest(Chan, "Field(2)")          ' get field 2
Field3 = Application.DDERequest(Chan, "Field(3)")          ' get field 3
Field4 = Application.DDERequest(Chan, "Field(4)")          ' get field 4
Field5 = Application.DDERequest(Chan, "Field(5)")          ' get field 5
DDETerminate Chan                                        ' kill the link
MyVar$ = Field1(1) & Field2(1) & Field3(1) & Field4(1) & Field5(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
```

Excel Example #6 - Advanced Excel Tricks - Reading large arrays of data into Excel

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, the function returns a variable with the data type "Variant Array". If the data that is requested by 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 "pre-transfer translation table" in WinWedge to convert all commas to tabs so that the data would appear in WinWedge as follows:

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

Note: When WinWedge is in DDE Server mode, its default behavior is to strip out carriage returns by translating them to "Nulls" therefore we will also need to translate carriage returns back to carriage returns and not to nulls. Select "Pre Transfer Translation Table" from the DEFINE menu and select the comma character in the table and click the button marked "Translate". When the ASCII chart appears, scroll the chart down to the TAB character (ASCII 9) and click the OK button to perform the translation. Next, select the carriage return character (ASCII 13) in the translation table and click the "Translate" button again. When the ASCII chart appears this time, select ASCII 13 and click the OK button. Click the OK button in the translation table to return to the main menu.

If you define the record structure in WinWedge as "Single Field Data Records" and thus pull the entire array into a single field in WinWedge 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 the 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 the carriage return in the next row down.

The following example demonstrates how to set up WinWedge 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 WinWedge:

1. Select “DDE Server” from the WinWedge “Mode” menu. When the window appears asking for a Target Application DDE Command, enter: “**EXCEL**” as the Application Name and then enter: “**SYSTEM**” as the DDE topic. Finally, enter the string: **[RUN("GetDataArray")]** as the DDE Command and click the OK button to return to the main menu.

2. Select “Input Record Structure” in the “Define” menu and define the structure of the input record(s) to WinWedge. Select the "Start of Record Event" as "Any Character Received" and the "End of Record Event" as "Time Delay Between Records" and then click the "Continue" button. In the next window 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" window and click the "Continue" button again. When you get to the final Window with the caption “Input Record Definition Editor”, click the OK button to return to the main menu.

3. Because Excel is expecting tab delimited data with carriage returns at the end of each line, we will need to use the "Pre-Transfer Translation Table" to translate the commas to tabs. Also, when WinWedge is in DDE Server mode, its default behavior is to strip out carriage returns by translating them to "Nulls" therefore we will also need to translate carriage returns back to carriage returns and not to nulls. Select "Pre Transfer Translation Table" from the DEFINE menu and select the comma character in the table and click the button marked "Translate". When the ASCII chart appears, scroll the chart down to the TAB character (ASCII 9) and click the OK button to perform the translation. Next select the carriage return character (ASCII 13) in the translation table and click the "Translate" button again. When the ASCII chart appears this time, select ASCII 13 and click the OK button. Finally, click the OK button in the translation table to return to the main menu.

4. Set up the rest of the WinWedge parameters as required by your serial device and then activate it by selecting either "Test Mode" from the Activate menu.

Steps for setting up EXCEL:

1. Create or edit a VBA module and enter the following code in the module:
(To create a new subroutine select "Tools", "Macro" and "Macros..." then enter a subroutine name "GetDataArray" and click the "Create" button.)

```
Sub GetDataArray()
Static StartCol as Integer, StartRow as Integer      ' retain values between calls
' This sub performs a DDERequest for Field(1) in WinWedge 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", "COM1")          ' initiate dde channel
MyArray = DDERequest(Chan, "Field(1)")          ' request field(1) from WinWedge
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 = 0                               ' set default x dimension to 0
On Error Resume Next                ' ignore errors (error occurs if array has one dimension)
x = UBound(MyArray, 2)              ' get upper bound of array x dimension
On Error GoTo 0                     ' allow errors
y = UBound(MyArray, 1)              ' get upper bound of array y dimension
If x = 0 And y >= 1 Then x = y : y = 1  ' if array has one dimension then swap x and y
EndRow = StartRow + y - 1
EndCol =  StartCol + x - 1
Sheets("Sheet1").Range(Cells(StartRow, StartCol), Cells(EndRow, EndCol)).FormulaArray = _
MyArray
' fill cells in the range starting in "StartCol:StartRow" with the data array from WinWedge
' StartCol = EndCol + 1 ' un-comment this line to move right the width of the array after each input
 StartRow = EndRow + 1 ' or un-comment this line to move below 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 WinWedge 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 WinWedge that returns the contents of FIELD(1) to a variant array variable named "MyArray". The dimensions of the array are obtaines using the UBOUND function and the entire array is then assigned to a range of cells the same size as the array in the worksheet "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 page 87.

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

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 WinWedge and Excel to poll each device in sequence and retrieve the data stacking up 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 example VBA subroutines 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 WinWedge:

1. Select “DDE Server” from WinWedge “Mode” menu. When the window appears asking for a "Target Application DDE Command", clear out all text boxes prompting for an Application Name, a DDE topic and a DDE command.
Note: In this example we will not be configuring WinWedge to send a DDE Command to Excel therefore these 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 as required for your application.

3. Set up WinWedge 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 WinWedge parameters as required by your device and then activate it.

Steps for setting up EXCEL:

1. Create or edit a VBA module and enter the following code in the module:

```
Public Const MyPort As String="COM1"    ' Change port to match configuration of WinWedge
Public Const MaxPrompts As Long = 3          ' number of prompts or devices
Dim RowPointer As Long                       ' row number for where data is stored
Dim Prompt(MaxPrompts) As String             ' create an array of prompts

Sub GetSWData()
Dim MyData As Variant, MyDataString As String, Chan As Long
'  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
' The first time through, RowPointer will be 0 - find the first empty row in the sheet
If RowPointer = 0 Then RowPointer = Sheets(1).Cells(65000,1).end(xlup).row + 1

' define the prompts for each device - add a line below for each prompt
' see the WinWedge 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

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 gets 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(1).Cells(RowPointer, PNum).Formula = MyDataString
' the above line writes the data to column "PNum" in row "RowPointer"
' PNum indicates which prompt requested the data that we just got

' add code here to further process the data if required.

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 - WinWedge will send prompt for first device
  RowPointer = RowPointer + 1   ' increment RowPointer - next set of data goes to next row
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 & "!RecordNumber"
ActiveWorkbook.SetLinkOnData "WinWedge|" & MyPort & "!RecordNumber", _
     "GetSWData "

'  the SetLinkOnData method causes Excel to run the GetSWData macro automatically
'  when new data is available in WinWedge (when the RecordNumber is updated).
'  Using the SetLinkOnData method in Excel eliminates the need to have WinWedge
' send a DDE Command to Excel to cause it to run the GetSWData sub.
'  Excel will automatically run the GetSWData sub when WinWedge updates the
'  RecordNumber DDE item. The RecordNumber DDE item is updated after each
'  data record is received.

End Sub
```

After entering the two subroutines above, run the subroutine "SetUpDDE" to establish a DDE link between Excel and the RecordNumber DDE item in WinWedge.

After Excel is set up and you are ready to start prompting for data you can enable the timer in WinWedge. What is going on here is that WinWedge will send the first prompt to the first device causing it to send back a response. When each response comes back, WinWedge updates the RecordNumber DDE item in Excel forcing it to run the macro above (because we used the SetLinkOnData method in the SetUpDDE subroutine to cause this to happen). The first thing the macro does is pull in the data from WinWedge 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 (i.e. devices) then it 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 from the last device and PNum is incremented past the number of devices. The timer controlled output string in WinWedge starts the whole process 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 WinWedge telling it to send the first prompt or you can define a hot key or a button controlled output string in WinWedge that would transmit the first prompt.

See Also: Defining Hot Keys and Hot Key Actions (pg. 40)
Defining Serial Output Strings (pg. 38)

DDE Examples for Microsoft Access

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

Steps for setting up WinWedge:

1. Select “DDE Server” from the “Mode” menu in WinWedge. When the window appears asking for a Target Application DDE Command, enter: **MSACCESS** as the DDE Application Name. For the DDE Topic, enter the file name of your open database file without the file path or the MDB Extension. For the DDE Command, enter the string: **[GrabData]**. This is a DDE command that will be sent to MSACCESS after each data record is received by WinWedge.

2. Set up the rest of WinWedge 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 VBA Module with a function named “GetWedgeData()” and enter the following code in the body of the function:

```
Function GetWedgeData ()
Dim MyData As String, Chan As Long                    ‘create variables
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, MyTable                ' dim variables for database and recordset objects
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 WinWedge to issue a DDE command consisting of the name of an Access macro to your database after each data record is received from your serial device. The macro calls a VBA function that performs a DDERequest back to WinWedge and returns the contents of FIELD(1) to a string variable named “MyData”. It then creates a new record in a table named "Table1" and inserts the data into a text field named "SerialData" the new record. This example assumes that a table named "Table1" exists in the database and that the table contains a text field named "SerialData"

Access Example #2 - Launching And Terminating WinWedge From Access

```
Function LaunchWedge ()
' This function will launch WinWedge feeding it the name of a configuration file on the
' command line causing WinWedge to automatically load the config file and activate itself
CmdLine = "C:\winwedge\winwedge.exe C:\winwedge\test.SW3"
' 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("00:00:03") ' wait for roughly 3 seconds
Do While Now < x                ' give WinWedge 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 WinWedge from memory
Chan = DDEInitiate("WinWedge", "Com1")' initiate DDE link with WinWedge on COM1
DDEExecute Chan, "[AppExit]"            ' send AppExit command to WinWedge
DDETerminate Chan                       ' terminate the DDE link
End Function
```

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

```
Function SendString (StringVar$)
' WinWedge must be running and activated for COM1 for this code to work
Chan = DDEInitiate("WinWedge", "Com1")          ' initiate DDE link with WinWedge
DDEExecute Chan, "[SEND(" + StringVar$ + ")]"   ' 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 (pg 64). Excel VBA and Access VBA are almost identical programming languages.

DDE Examples for Microsoft Word for Windows

Steps For Setting Up WinWedge:

1. Select “DDE Server” from the “Mode” menu in WinWedge. When the window appears asking for a Target Application DDE Command, enter: **WinWord** as the DDE Application Name and for the DDE topic enter the string: **System**. For the DDE Command, enter the string: **[GrabData()]**. This is a DDE command that will be sent to Word after each data record is received by WinWedge.

2. Set up the rest of the WinWedge parameters as required for your serial device 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 window 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 GrabData()
chan = DDEInitiate("WinWedge", "COM1")          ' open a link to WinWedge
MyData$ = DDERequest$(chan, "Field(1)")         ' get the data from WinWedge
DDETerminate chan                               ' close the link
Selection.TypeText Text:= MyData$               ' enter the data into the open document
Selection.TypeParagraph                         ' insert a paragraph marker (hard return)
End Sub
```

After you are done entering the code select "Close" from the FILE menu to save the macro. This example sets up WinWedge 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 WinWedge asking for the data in Field(1) in WinWedge Window. This data is returned to Word in the variable: MyData$. Once the data from WinWedge is in a variable, you can do whatever you like with it by including additional code of your own design. This example simply inserts the text in the MyData$ string variable into the open document and then inserts a paragraph marker to move the cursor to the next line down.

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

```
Chan = DDEInitiate("WinWedge", "COM2")          ' open a link to WinWedge
DDEExecute Chan, "[Sendout('Test',13)]"         ' transmit "Test" and a carriage return
DDETerminate Chan                               ' close the link
```

DDE Examples for Visual FoxPro for Windows

FoxPro Example #1 - Collecting data into MS FoxPro using WinWedge.

Steps For Setting Up WinWedge:

1. Select “DDE Server” from WinWedge MODE menu. When the window appears asking for a DDE Command Destination Application, enter: **MyServer** as the Application Name and enter **MyTopic** as the DDE topic. For the DDE Command, enter the string: **[GRABDATA]**. This is a DDE command that will be sent to your FoxPro application after each data record is received by WinWedge.

2. Set up the rest of the WinWedge parameters as required by your serial device 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 done entering the code save your work and run the program.

What this example does is to set up WinWedge to issue the 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 WinWedge that returns the data in Field(1) in the WinWedge Window. This data is returned to FoxPro in the string variable named **MyData**. Once the data from WinWedge 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 - performs a DDERequest to WinWedge***
*** on COM1, requests data from FIELD(1) and stores it in a variable: "MyData"***

     IF gData="[GRABDATA]"
        gnChanNum = DDEInitiate('WinWedge', 'COM1')
        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
```

FoxPro 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', 'COM1')
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 WinWedge manual for details
gnChanNum = DDEInitiate('WinWedge', 'COM1')
IF gnChanNum != -1
  glExecute = DDEExecute(gnChanNum, CmdString)
  = DDETerminate(gnChanNum)    && Close the channel
ENDIF
RETURN
```

DDE Examples for Microsoft Visual Basic 6 for Windows

Linking a VB Text Box to a WinWedge Data Field

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

```
Text1.LinkTopic = "winwedge|com2"          ' set the link topic
Text1.LinkItem = "field(1)"                ' set the link item
Text1.LinkMode = vbLinkAutomatic           ' set the link mode to "Automatic"
```

Sending DDE Commands To WinWedge From A VB Application.

After a link is established between a text box and a data field in WinWedge, you could issue commands to WinWedge 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: WinWedge must be active on the specified serial port for the following code to work.

```
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 sending DDE commands
Text2.LinkMode = vbLinkManual              ' set the link mode to "Manual"
Text2.LinkExecute "[Sendout('Testing',13)]"  ' send the string out the serial port
```

Launching WinWedge From A Visual Basic Application

```
Sub RunWedge()
' the shell statement passes a WinWedge configuration file as a command line argument.
' This causes WinWedge to automatically load the configuration file and activate itself.
x = Shell("c:\winwedge\winwedge.exe c:\winwedge\ddetest.SW3")
ThreeSecsFromNow = TimeValue(Now) + TimeValue("00:00:03")
Do While TimeValue(Now) < ThreeSecsFromNow
 DoEvents                          ' give WinWedge time to load
Loop
End Sub                            ' WinWedge should be loaded now
```

Diagnosing Serial Communications Problems

To diagnose communications problems go throught the following steps:

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

Step 2: Open the Port Analyze window by selecting **Analyze** from the WinWedge **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 WinWedge), the port does not exist on your PC, or, in the case of add-on serial adapters, the Driver for the add-on serial adapter has not been installed or set up correctly. If you suspect that the port may be set up incorrectly, then you may need to contact the manufacturer of the add-on serial adapter that you are using to correct the problem.

Step 3: If the Port Analyze window appears, transmit some data from your serial device to WinWedge. The data from the device should appear in the text box marked "Input Data".
If no data appears in the input data textbox then either your instrument is not connected to your PC using a proper serial cable for the device; you are not connected to the serial adapter that you specified in the port settings window in WinWedge or your device is not transmitting any data. Even if you have the wrong baud rate, parity, databits or stopbits selected, you should see something (even garbage characters) appear in the Analyze window.

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 data and yet you still have no success receiving data in the Analyze Window, then you may need a 'NUL 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 almost certainly need a nul modem adapter. A nul modem adapter is a small plug that connects between a serial cable and your computer that 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. Nul 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 data textbox in the Port Analyze Window or if data appears in the Input Data textbox but part or all of it is either garbled or consists of unreadable characters, then you are connected to the correct port and your device is sending data however the communications parameters that you chose in the Port Settings window are wrong and must be corrected to match parameters that the device is using.

Note: WinWedge has a "Beep On Input" option in the "Preferences" window that causes WinWedge 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 the beeping is an indication of a communications error; typically either a mismatch in serial communications parameters (baud rate, parity, databits, etc.) between WinWedge and your device or you are experiencing a serial buffer overrun.

Troubleshooting

If you are not receiving any data all in the Port Analyze Window:

1. Make sure that the PC you are using is equipped with the serial port selected in the Port Settings Window. 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.

2. Check that your serial device is 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 Window. Also, make sure that your serial device is getting power and is turned on. You may want to contact the manufacturer of the device to confirm that you are using the correct cable, have the right communications parameters and that you are doing everything that is necessary to connect to a PC and also to get the device to transmit data. Many serial devices will transmit data either when you press a button on the device or after you perform some type of operation with the device. Other type of devices may require that you send a command out the serial port to the device to trigger it to send back data. If the device that you are using requires that a command be sent out the serial port to trigger it to send data back, then you will need to know exactly what the required command is. You would then need to enter the required command in the "Output" textbox in the Analyze window and then click the "Send" button to send the command out the serial port. With devices that require commands to be sent to them, it is extremely important to have the serial communications parameters set in WinWedge to exactly match what the device uses. It is possible to receive data from a device with the wrong serial communications parameters set in WinWedge however if you send a command to a device using the wrong communications parameters, the device will not recognize the command and therefore not respond to the command.

3. Use the Port Analyze window in WinWedge and try, at least, to get some data from the device to appear in the Input Data 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 "NUL Modem Adapter". A nul modem adapter is a small plug that connects between the serial cable from your device and serial port on your PC. Its purpose is 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 almost certainly need a NUL Modem Adapter. You can pick one up at any Radio Shack or 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 WinWedge is similar to HyperTerminal program that comes with Windows located in your Accessories program group. The manufacturer of your serial device may not be familiar with WinWedge however they will more than likely be familiar with 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 reading in data from the device to the HyperTerminal program. Once you are able to get data into HyperTerminal you should have no trouble doing the same thing with WinWedge.

If data appears garbled or unreadable in the Port Analyze window:

5. Check that the serial device is set up using the exact same communications parameters as WinWedge, 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 WinWedge to try different parameters until you get data that appears correct. Try different baud rates first using Even Parity, Seven Data Bits, and One Stop Bit until you get data that looks correct. Finally, try different combinations of Parity, Number of Data Bits and Number of Stop Bits until all data is correct and you do not hear any beeping when data is transmitted into the Analyze window in WinWedge. 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.

If data appears using the Port Analyze feature but not after you activate WinWedge:

6. Make sure that WinWedge 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 WinWedge configured incorrectly (either the Start or End Of Record Events are 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 WinWedge and try again.

If you cannot get WinWedge to operate properly after you have gone through the above procedures, then call TAL Technologies for assistance at: Tel: (215)-496-0222 during regular business hours. Please have this manual at hand and be at your PC with WinWedge running and your serial device connected when you call for support. A valid serial number is also required to obtain telephone support. Your serial number is located either on the back cover of your users manual or on a sticker on the outside of the box that WinWedge was shipped in.

TAL Technologies maintains a web site at **http://www.taltech.com** where you can find the latest technical notes and articles relating to typical support issues for WinWedge. The web site also contains many configuration examples for WinWedge as well as sample Excel spreadsheets that show how to perform the most common functions with WinWedge. Web based support is available 24 hours a day and you may also email support questions to: **support@taltech.com**. Support questions sent via email will be responded to during regular business hours (9:00 AM to 5:00 PM Eastern Time Monday to Friday).

Running WinWedge On Multiple Serial Ports Simultaneously

WinWedge may be activated for multiple serial ports simultaneously provided you have the proper serial adapter hardware installed in your PC. To input data from more than one serial port, you would create a separate WinWedge configuration file for each serial port and then launch and activate new instances of WinWedge using each configuration file.

WinWedge should work fine with all built in COM ports on your PC as well as with all add-on serial adapters that install into a card slot, a PCMCIA slot or a USB port (USB Serial Adapters) on your PC. Some add-on serial adapters including all USB add-on serial adapters, require special COM Drivers to be installed before Windows will recognize the new serial ports. The adapter will normally come with any require drivers that need to be installed. Once the drivers are installed and the adapters are connected properly, you should see them listed by their COM port numbers (COM3, COM4, etc.) in the Windows Device Manager in the "Ports (COM & LPT)" section.

WinWedge will work with all single or multi port serial adapter that interfaces to your PC through a USB port. TALTech sells a complete line of add-on serial adapters that both plug into a USB port or plug into an unused ISA slot in your PC. Add-on serial adapters that connect to a PC through a USB port do not require their own dedicated Interrupt Request line therefore there are never any conflicts with other devices. They are also "Plug and Play" devices that are trivially easy to install. For more information please visit: http://www.taltech.com

Understanding The Data From Your Serial Device

Before you can correctly configure WinWedge to parse and filter data received from your serial device, you must first have a complete 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 WinWedge can be used to manipulate your data. To use WinWedge effectively, you must think of each input from your device as a *record* of information containing one or more specific *fields* of data. The next step is to decide what features of WinWedge 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 from the next field. Finally you need to identify which fields within each record 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 users manual for your device or if the output data structure is not described, you can use the Port-Analyze window in WinWedge or the HyperTerminal program that is provided with Windows to manually analyze the output from your device by actually viewing the data that is transmitted from it.

When you configure WinWedge for a particular device you must define the input record structure for your serial data to WinWedge by selecting "Input Data Record Structure" from the *Define* menu. When defining the input data record structure, you specify a "Start Of Record Event", 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 might select "Any Character Received" as the *Start Of Record Event* and "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.

Note: The <CrLf> above refers to a carriage return and a linefeed character. A carriage return (Cr) will appear in the "Analyze" window in WinWedge as a musical note and a linefeed (Lf) will appear as a small rectangle with a circle inside it.

Other devices may output data with a record structure that is less obvious than the example above. Consider the following two possible different output records transmitted from one particular laboratory instrument:

Sample#1,*213**32****23**<CrLf>
Sample#2,*215**437**141**797**89**56<CrLf>

The two records appear similar in that they both are terminated by a carriage return-linefeed and they both seem to contain multiple delimited data fields. A minor problem with the above two records is that they both contain a different number of data fields. Another complication is that they seem to contain both a comma and one or more asterisks as delimiter characters. In the first record it appears that we have four data fields: "Sample#1", "213", "32" and "23"
In the second record it appears that we have seven data fields:
"Sample#2", "215", "437", "141", "797", "89" and "56"

There are actually several ways to configure WinWedge to correctly parse both records in the above example. The most elegant method is to use the "Pre-Input Character Translation Table" to modify the delimiters so that they are all the same. For example, we could translate all asterisks to commas to end up with the two records shown below:

Sample#1,,213,,32,,,,23,,<CrLf>
Sample#2,,215,,437,,141,,797,,89,,56<CrLf>

Now both records obviously have multiple, delimited, data fields with a carriage return-linefeed signaling the end of each record. We now determine the maximum number of fields in a record by adding up the number of delimiters in each record and adding one. (i.e. the first record has 11 fields and the second record has 13 fields therefore the maximum number of data fields is 13).

Now, when you define the record structure in WinWedge you would select "Any Character Received" as the *Start Of Record Event* and "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 the maximum number of data fields per record is 13 with a comma delimiter separating each field.

When parsing records with multiple delimited data fields, if you choose a space character as your delimiter, then consecutive spaces would be treated as a single delimiter. Therefore, in the above example, if instead of translating asterisks to commas, you were to translate both commas and asterisks to *spaces*, and then you chose the space character as your delimiter, you would effectively end up with the two records shown below with each field now separated by a single (space) delimiter:

Sample#1 213 32 23 <CrLf>
Sample#2 215 437 141 797 89 56<CrLf>

Suppose you had a device that normally sent its output to a serial printer in the following manner with more than one line of data:

```
Device#        Height     Width    Length     <CrLf>
-------------------------------------------------------- <CrLf>
     1           23         12       24       <CrLf>
     2           27         13       8        <CrLf>
     3           27         13       9        <CrLf>
     4           27         13       8        <CrLf>
-------------------------------------------------------- <CrLf>
<FF>
```

(<CrLf> represents a carriage return-linefeed and <FF> represents the form feed character)

You can still think of the entire output shown above as a single data record even though it contains several lines of data or you can think of each individual line as a complete record. (In fact, you could even think of each word or number outputted from the device as a complete, single field, data record).

Again there are many different ways to parse the entire output from the device using the different features available in WinWedge.

If you wanted to capture only the lines with numeric data and parse each of these lines into records with four fields, one way to accomplish this would be to specify the "Start Of Record Event" as "Numeric Character Received" and then specify the "End Of Record Event" as "Carriage Return or CrLf Received". Finally you would define the record structure as "Multiple Delimited Data Fields" with four fields per record and a space character as the delimiter. Because you chose "Numeric Character Received" as the "Start Of Record Event" and because there are no numeric characters in the first, second, seventh and eighth lines of data, these lines would be ignored.

If you wanted to capture the numeric data as above but with all 16 numbers in one record, you could again specify the "Start Of Record Event" as "Numeric Character Received" and then specify the "End Of Record Event" as "Special Character Received" and select the Dash ("-") character as the *special character* . Next, using the Pre-Input Character Translation Table, you would translate all carriage returns and linefeeds to spaces and finally you would define the record structure as "Multiple Delimited Data Fields" with 16 fields per record and the space character as the delimiter.

Once you understand all of the features provided in WinWedge, you can get very creative with how you deal with the data from your instrument(s).

The important points to remember are that WinWedge always waits until the selected "Start Of Record Event" occurs before it starts reading in any data; after which it keeps reading in characters until the specified "End Of Record Event" occurs. When the "End Of Record Event" occurs, WinWedge 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, WinWedge 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 Keystrokes" 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 WinWedge would see the data as three records with four fields per record.

If you specified "Time Delay Between Records" and also used the "Pre-Input Translation Table" to translate carriage returns to commas and linefeeds to "Nul" or "Ignore", then the data would appear to WinWedge as follows:

1, 23,12,24,2,27,13,83,27,13,9

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 12 fields.

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

See Also: Defining The Input Data Record Structure (pg. 25)
The Pre-Input Character Translation Table (pg. 34)

More WinWedge Configuration Examples

A common task for WinWedge is to interface a GagePort or GagePort NT 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 WinWedge.

Note: The default configuration of 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 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 or by sending commands to the GagePort through the serial port. 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 WinWedge:

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

1. Run WinWedge by double clicking on its icon in the Start Menu in Windows.

2. From WinWedge MODE menu, select "Send Keystrokes To..." (even if it is already checked). This will cause a window 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 (no blank spaces or visible characters). By clearing out these items and thus not specifying an application for WinWedge to send the keystrokes to, WinWedge 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 WinWedge and when the Port Settings window 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 WinWedge. When the Port Analyze window appears, transmit a reading to WinWedge 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 window 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 reading number.
##########	Ten digit gage reading with decimal place and minus sign if the reading is negative.
UUUUU	Five blank spaces.
PP	Two digit port number.
<cr>	Trailing 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 is either garbled or consists of unreadable characters, then you most likely have the wrong communications parameters selected in either WinWedge or in the GagePort. (The communications parameters set in both WinWedge and the GagePort must match.) Refer to the troubleshooting section of WinWedge users manual for further assistance. After you get good data in the Analyze window in WinWedge, click the Quit button to return to the main menu.

5. The next step is to configure WinWedge 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 WinWedge. The first window to appear will prompt you for a "Start Of Record Event" and an "End of Record Event". For the Start of Record Event, choose "Any Character Received" and for the End of Record Event, choose "Carriage Return or CrLf Received" and click the CONTINUE button to proceed. In the next window to appear prompting for a "Record Structure", select "Multiple Delimited Data Fields" and click the CONTINUE button. This will display another window 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 window to appear is the "Input Record Definition Editor" window. This window 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 WinWedge to the application where you want the data to go. Select "Ignore This Field" for the filter for Field 1 (leave all other text boxes blank) and click the button marked "Next Field". Select "Numeric Data Only" 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 This Field" for the filter for Field 3 and click the "Next Field" button again and select "Ignore This Field" for the filter for Field 4. Finally, click the OK button to return to the main menu of WinWedge.

6. Save your work by selecting "Save" or "Save As..." from the FILE menu. Save your configuration file using the file name GAGEPORT.SW3

7. The final step is to activate WinWedge by selecting "Test Mode" from the ACTIVATE menu. After WinWedge 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 should appear in the cell and the cursor should automatically move down one cell.

A More Complex Example

This example explores some of the less obvious features of WinWedge and how to apply them to a more complex situation. In this example we will assume that you have already gone through the first four steps for setting up WinWedge outlined in the previous example and have discovered that the data transmitted by your instrument is structured as follows with four lines of data and each line terminated by a carriage return:

Data Widget Coordinates<Cr>
x = 100.7, x deviation = .50<Cr>
y = -170.25, y deviation = .38<Cr>
z = 110, z deviation = -.5<Cr>

For our particular application, we are interested in capturing only the x, y and z values to a 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 “x” values, one column of “y” values and one column of “z” values. We also want to ignore the first line of text and also each of the three “deviation” values.

If you think of the entire output from the device as a single data “Record” containing a group of data fields, the first thing we must do is to figure out a way to isolate the three x, y and z fields from the rest of the data using the parsing and filtering capabilities of WinWedge. There are actually several ways to achieve the results that we want using different features in WinWedge. One approach is to first use the “Pre Input Translation Table” to translate all commas to carriage returns so that our data appears to WinWedge as follows with the same delimiter character separating each individual data field:

Field 1	**Data Widget Coordinates<Cr>**
Field 2	**x = 100.7<Cr>**
Field 3	**x deviation = .50<Cr>**
Field 4	**y = -170.25<Cr>**
Field 5	**y deviation = .38<Cr>**
Field 6	**z = 110<Cr>**
Field 7	**z deviation = -.5<Cr>**

To perform this translation, select “Pre Input Translation Table” from the DEFINE menu and when the translation table appears, scroll down in the table until the “comma” character is hi-lighted and then click the button marked “Translate...”. In the ASCII chart that appears, scroll down to the carriage return character (ASCII 13) and click on the “OK” button in the ASCII chart window. After you perform the translation, you can exit from the translation table by clicking the “OK” button in the translation table window.

Looking at the data now, we can identify seven distinct data “fields”, each of which is delimited from the next by a carriage return.

The next step is to describe the structure of the data to WinWedge and also specify how WinWedge should parse and filter the data so that we get the results that we want. If we examine the data (with commas translated to carriage returns), we see that we would like to ignore the first, third, fifth and seventh fields and we also need to filter the second, fourth and sixth fields so that only the numeric data is sent to our spreadsheet. Finally, to get the data into our spreadsheet in three columns, we need to add a right arrow keypress after the second and fourth field and a down arrow and two left arrow keypresses after the sixth field.

To do this, select “Input Data Record Structure” from the DEFINE menu and for the “Start of Record Event” specify “Any Character Received”. For the “End of Record Event”, specify “Carriage Return or <CrLf> Received” and click on the “Continue” button to proceed. In the next window, specify “Multiple Delimited Data Fields” as the record structure and click on the “Continue” button. The next window to appear allows us to specify the delimiter character and also specify the maximum number of data fields in each data record. In this case our delimiter character is the carriage return (ASCII 13) and the maximum number of data fields in each record is seven. After entering these parameters, click on the “Continue“ button. The final window to appear allows us to specify a filter for each field as well as any “Field Postamble Keystrokes” that we want to add to the data. Go through the following steps to specify the required filters and postamble keystrokes:

Place the cursor in the “Filter” list box and select “Ignore This Field” as the filter for Field 1. Use the “Next Field” and “Previous Field” buttons to scroll through the parameters for each of the seven data fields and select “Ignore This Field” as the filter for Fields 1, 3, 5 and 7 and select “Numeric Data Only” as the filter for Fields 2, 4 and 6.

Click the “Previous Field” button until you are back to the parameters for Field 2.

Place the cursor in the “Field Postamble Keystrokes” edit box and then click on the button marked “Keystroke List”. When the keystroke list window appears, scroll down until the word “RIGHT” is hi-lighted and click the “OK” button in the keystroke list window. This should cause the word “{RIGHT}” to appear in the Field Postamble Keystrokes edit box.

Click on the “Next Field” button twice to advance to the parameters for Field 4 and repeat the previous step to add a right arrow keypress after Field 4.

Click on the “Next Field” button twice to advance to the parameters for Field 6 and enter the following text in the Field Postamble Keystrokes text box: {DOWN}{LEFT 2}
Click on the OK button to return to the main menu.

Finally, select “Test Mode” from the ACTIVATE menu, open up your spreadsheet and move the cursor to the cell where you want the first x value to appear and start collecting data.

See Also: Defining the Input Data Record Structure (pg. 25)
Cool Wedge Tricks (pg. 99)

Cool Wedge Tricks

Defining Less Data Fields Than Are Actually Received

If you define your Input Data Record Structure with less data fields than actually appear in your data, WinWedge will ignore any additional data that it receives after the last field that you have defined in WinWedge. You could also take advantage of this behavior by in situations where a device transmits more data than you are actually interested in. For example suppose you had a device that transmitted 50 comma delimited data fields followed by a carriage return and you were only interested in the data that appears in the first three fields. You could define the input data record structure by specifying the End Of Record Event as "Carriage Return or CrLf Received" and then specifying the "Record Structure" as "Multiple Delimited Data Fields" with a comma as the delimiter and specifying *three* for the maximum number of data fields. The actual data has 50 data fields before the carriage return so by specifying three for the maximum number of fields, you will end up with only the first three data fields in WinWedge and the remaining 47 fields will be ignored.

Dealing With Continuous Data

There are a few types of instruments that transmit data either continuously or in bursts with many data values arriving in succession over a short period of time. For example many older electronic balances were designed to transmit continuous data to a digital display so that you could watch the weight reading change on the display. They typically transmit up to 10 weight values per second and do not provide a way to stop the data from flowing. (Most newer balances are configurable to transmit data only in response to a prompt sent through the serial port or by pressing a "Print" button on the balance. This is the best way to use a balance with WinWedge.) There are also some devices like tensile strength testers or friction peel testers that perform a test over a short period of time and for the duration of the test they transmit a large number of data values.

Although you may not be able to configure the device to send less data, you may be able to configure WinWedge to ignore enough of the readings so that you are not overwhelmed with data. One way to do this is to configure WinWedge to treat multiple data records as one long record containing multiple fields and then simply ignore most of the fields. For example, most electronic balances transmit a weight reading followed by a carriage return and a line feed character. Normally you would define the "Input Data Record Structure" with "Carriage Return or CrLf Received" as the "End of Record Event" and then specify "Single Field Data Records" as the "Record Structure" for this type of data. If instead, you chose "Multiple Delimited Data Fields" as the "Record Structure" and chose the Carriage Return (ASCII 13) as the delimiter and also specified 30 for the "Maximum Number Of Data Fields" and finally chose "Ignore This Field" for the filters for Fields 1 through 29, you would effectively remove 29 out of each 30 data readings transmitted by the balance. If the balance sent ten readings per second, WinWedge would only pass a single reading through to another application once every three seconds.

Another way to control continuous data is to configure WinWedge so that it is initially suspended when you first activate it (select the "Activate Initially Suspended" option from the Activate menu). If WinWedge is Activated but suspended, it will continue to input serial data into its input buffer however it will not pass any data to another application until you resume it. Next, you could define one hot key that resets WinWedge and then define another hot key that resumes WinWedge for one data record. After you activate WinWedge it will not do anything until you press your two hot keys. When you press the "reset" hot key, WinWedge will reset itself (causing it to flush its input buffer) and then when you press the "Resume for 1 data record" hotkey, it will enable itself for one data record. After the next record is received, WinWedge will suspend itself again.

If you use the above technique, make sure that you select a "Start of Record Event" that will reliably determine the beginning of each data record (i.e. do not select "Any Character Received" as the Start of Record Event) . The reason for this is because if the device is in the middle of transmitting a data record when you press your hot key, the reset hot key action causes WinWedge to clear out its input buffer thus chopping off the first half of the data record being received. If you had "Any Character Received" selected as the Start of Record Event, the second half of the data record would be received as if it were a complete record. Choosing "Special Character Received" as the Start of Record Event and using the Line Feed (ASCII 10) as the "Special Character" would be a good choice in this situation. Since the device is transmitting a continual stream of data records with a carriage return and a line feed at the end of each data record, you could just as easily think of the line feed as the start of each record and the carriage return as the end. To remove the line feed and the carriage return from each record, you can use either the "Pre-Transfer Character Translation Table to translate them to "Nul" or you can apply a "Numeric Data Only" filter to the field(s) in WinWedge that contains them.

See Also: Defining the Input Data Record Structure (pg. 25)
Special Considerations Regarding Delimiter Characters (pg. 30)
Defining Hot Keys and Hot Key Actions (pg. 40)
The Pre-Transfer Character Translation Table (pg. 34)

Sending Data as Keystrokes to an application While Passing Data to Anther Program Via DDE

Although WinWedge has an explicit "DDE Server Mode", it is actually a DDE Server no matter what mode it is in (i.e. "Send Keystrokes"). The reason that WinWedge has a DDE Server Mode at all is so that you can configure it to send a DDE command to another application after each input and thus force the other application to do something with the data that was just sent. Suppose that you wanted to send data as keystrokes to one application *and* pass the data to another program via DDE in real time. You could set up WinWedge in Send Keystrokes mode and still pass the data through DDE links to another program. The only problem that you might run into is that you will not be able to configure WinWedge to send a DDE command to the other program and thus force it to do something like run a macro after each input data record. Many applications that support DDE also provide a way to monitor active DDE links so that the application automatically runs a macro when it detects new data from a DDE server. Excel for example has a function called "SetLinkOnData" that forces it to run a macro whenever data from a DDE server changes.

See Also: DDE Examples for Excel (pg. 64)

Making Good Use Of The Pre-Input Character Translation Table

The "Pre-Input Character Translation Table" in WinWedge allows you to translate ASCII characters to other characters before being inputted to WinWedge. Typically, this feature is used to convert delimiter characters so that they will all be the same character when WinWedge receives the data. For example, suppose you had a device that transmitted the following data record:

123.53, 46.20, 03:45:21<CrLf>

In the above data record there are two different delimiter characters being used, the comma and the colon. If you translated the colons to commas in the "Pre-Input Character Translation Table", then the data record would appear to WinWedge as:

123.53, 46.20, 03,45,21<CrLf>

Now that all the delimiters are the same, you could define the record structure in WinWedge as "Multiple Delimited Data Fields" using the comma as the delimiter with five fields in each record.

The "Pre-Input Character Translation Table" also allows you to translate characters to either "Ignore" or "Void Record". The "Ignore" translation removes the character from the input data as if the character did not exist. The "Void Record" translation causes WinWedge to disregard the entire current data record if a character that has been translated to "Void Record" appears anywhere within the record. This feature can be used reject certain data records from a device. For example, suppose you had a device that transmitted the following data:

Device#	Height	Width	Length	<CrLf>
----------	----------	----------	----------	<CrLf>
1	23	12	24	<CrLf>
2	27	13	8	<CrLf>
3	27	13	9	<CrLf>
4	27	13	8	<CrLf>
----------	----------	----------	----------	<CrLf>

Since you are probably not interested in the first line or the lines containing the dashes, you could use the "Void Record" translation to remove these lines by translating the dash character and the upper case "D" to "Void Record". After the translations, WinWedge would effectively receive the data as shown below:

1	23	12	24	<CrLf>
2	27	13	8	<CrLf>
3	27	13	9	<CrLf>
4	27	13	8	<CrLf>

See Also: The Pre-Input Character Translation Table (pg. 34)

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

One way to pass data to two Windows programs is to send the data to the first application and then use whatever keystrokes are necessary in that application to copy the data to the clipboard, switch the focus to the second application and finally paste the data into the second application. Even if neither of the applications provides 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 WinWedge, 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 WinWedge Shift+Insert is represented by: +{INSERT}. To select text in any program you can usually 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. While holding 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 WinWedge set up to send keystrokes to NotePad and you also wanted to send the data to WordPad you could use the following procedure:

Set up WinWedge in "Send Keystrokes" mode specifying the NotePad program as the target for all keystrokes from WinWedge. Next, configure WinWedge to work with your serial device and then add the following characters to the end of the "Field Postamble Keystrokes" for the last data field that you have defined in WinWedge: +({LEFT}{HOME})^{INSERT}{DOWN}%{TAB}+{INSERT}
Activate WinWedge and then launch the WordPad program and then launch the NotePad program. At this point NotePad has the focus and is therefore at the top of the Window list with the WordPad program directly underneath it. When you input data from your device, it will be sent to both NotePad and WordPad. 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 de-select the currently selected text. The keystrokes: %{TAB} means Alt-Tab and causes Windows to switch the focus to WordPad. Finally, the keystrokes: +{INSERT} means shift+insert and performs the job of pasting the data from the clipboard into WordPad.

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}

Invoking hot keys in other applications

Almost all spreadsheets, databases, word processors and most other large scale application programs allow you to either record macros or create subroutines that can be invoked with a hot key while you are working within the application. For example when you write a VBA subroutine in Excel, you can assign the subroutine to a "Shortcut Key" so that all you have to do to run the subroutine is press the shortcut key keystroke. You could invoke Excel subroutines from WinWedge by placing the shortcut key keystroke for a subroutine in either the "Record Preamble Keystrokes" or the "Field Postamble Keystrokes" when you define your "Input Data Record Structure" in WinWedge. You could also use the "Pre-Transfer Character Translation Table" to translate specific characters that might appear in your serial data to keystrokes that would invoke different Excel subroutines.

INTRODUCTION TO SERIAL COMMUNICATIONS

Most PCs are equipped with one or two serial ports and one parallel port. (Some newer PCs do not come with serial ports at all however it is extremely easy to add serial ports by purchasing USB to RS232 add-on adapters. They are available at most computer or office supply stores for under $25).

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 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 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 standard serial ports on all 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 indicate 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.

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 you have selected. 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 has been transmitted, a stop bit is sent. A stop bit has a value of 1 - or a mark state - and it can be detected correctly even if the previous data bit also had a value of 1. This is accomplished by the stop bit's duration. 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. 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. WinWedge 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 to RTS. Some serial devices use DTR and DSR as signals to confirm that a device is connected and is turned on. WinWedge 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.

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 crosses the receive and transmit lines so that the transmit line on one end is connected to the receive line 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.

INDEX

D

E