DDDA User Guide

Version 1.5

Stanford Research Systems

email: DDDASupport@thinksrs.com

Table Of Contents

Chapter 1 DDDA

Introduction

What is DDDA?

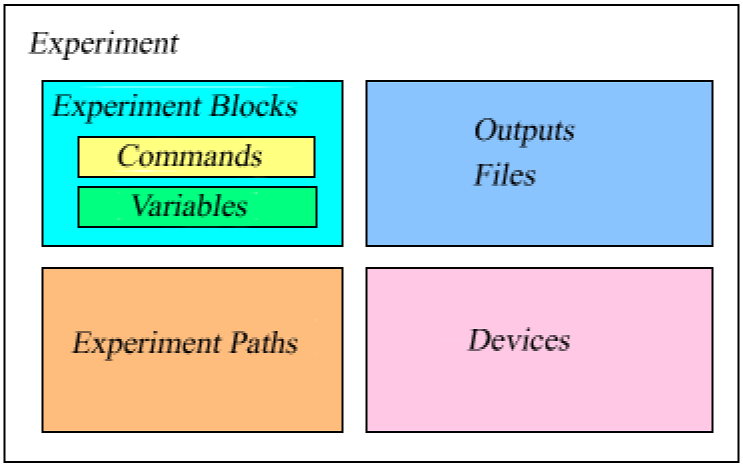
DDDA is a simple data acquisition program for Windows 95/98 and NT which allows you to:

Send and read data from an instrument using a GPIB or RS232 interface.
Graphically display the data.
Manipulate the data.
Control how and when the above are executed.

Why DDDA?

DDDA is for those "quick and dirty" data acquisition jobs. You've got an instrument, you want to talk to it or pull data off of it, and you want to do it quickly. DDDA lets you do all that conveniently and quickly by writing small, simple, data acquisition "programs" while shielding you from the complexity of GPIB or RS232 programming. DDDA lets you work right at the instrument command level. There's no "drivers" to install or debug. You send data to the instrument. You get data back and plot it. There's no fancy data analysis either -- but it's easy to exchange data with programs that do. And finally, you won't have to spend a year learning DDDA, in fact it pretty much can be mastered in an afternoon. All in all DDDA is the perfect choice for simple, fast data acquisition tasks - the kind you face all the time in the real world.

The DDDA Experiment



DDDA is used to create, edit and execute Experiments. An Experiment is a self-contained set of experiment components assembled by the user. There are four types of experiment components:

Experiment File: Text file of data collected in the experiment. There can be zero to many in any experiment. (New in version 1.5)
Output: Graphical representation of data collected in the experiment. There can be zero to many in any experiment.
Device: Electronic instruments that have been added to the Experiment. An instrument can be added to the Experiment if it can send and read bytes using a GPIB or RS232 interface. There can be zero to many in any experiment.
Experiment Path: Controls the execution order of the Experiment when it is run. One or more of these is mandatory if there is more than one Experiment Block.
Experiment Block: A self-contained set of block components that are put together by the user. There must be at least one Experiment Block in an experiment. There are two types of block components:

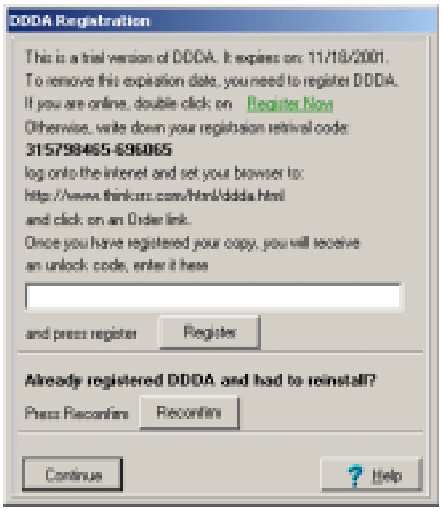
- **Variables**: A single value or an array of values used to store data in either string, integer, or float format. There can be zero to many in any block.
- **Commands**: Instructions to be executed when the Experiment is run. There can be zero to many in any block. There are three categories of commands:
  - **Instrument Commands**: Commands used to send bytes to and read bytes from a Device. (Please note, these are NOT the same as Device Commands.)
  - **Control Commands**: Commands used to control the execution of other commands. For example, one control command is Repeat, which allows for repeating a series of commands a user specified number of times.
  - **Variable Commands**: Commands which modify the value of a Variable or send the current value of a Variable to an Output.

When all the components, both block and experiment are assembled, the Experiment can be executed in DDDA.

Warning: Before attempting to run DDDA, please make sure your keyboard layout is set to English (United States). If is not, DDDA may not be able to read previous experiments.

Registering your copy of DDDA

After you install DDDA on your computer and run it for the first time you will see the Registration screen:



This is a trial version of DDDA. It expires on: 11/18/2001.
To remove this expiration date, you need to register DDDA.
If you are online, double click on Register Now
Otherwise, write down your registration retrival code:
315798465-696065
log onto the internet and set your browser to:
http://www.thinksrs.com/html/ddda.html
and click on an Order link.
Once you have registered your copy, you will receive
an unlock code, enter it here

and press register

Already registered DDDA and had to reinstall?

Press Reconfirm

You will also see this screen every subsequent time you run the program until you register it. You can also see this screen by selecting Help|Register from the DDDA Main Menu (if you have a registered copy, this menu item does not exist). Note that the expiration date is exactly 30 days from the date you run the program for the first time after installing. Once the expiration date arrives, you will no longer be able to run DDDA until you register your copy. There are two ways to register your copy of DDDA:

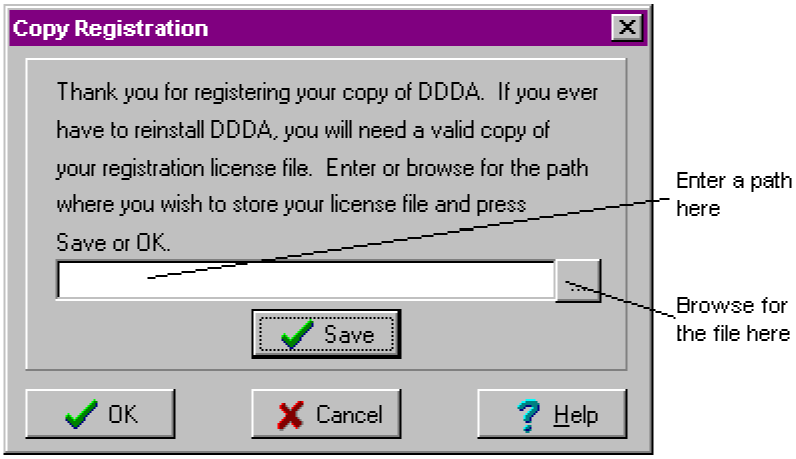
a) Using the registration retrieval code (First time users, requires you to purchase DDDA).
b) Using a previously saved Registration file (Reinstalling, costs nothing).

Warning: The registration retrieval code is generated THE FIRST TIME YOU RUN THE PROGRAM AFTER INSTALLING. If you uninstall DDDA and reinstall it, the registration retrieval code WILL CHANGE even if you have registered! The Unlock code you were given when you registered WILL NOT WORK in this case!

Using the registration retrieval code

Installing using the registration retrieval code requires you to purchase DDDA. If you are already online, you can purchase DDDA by double clicking on the "Register Now" link provided (Register Now). Using this link avoids the writing down of the registration retrieval code, it is automatically loaded into the registration web page. If you are not online, you can write down the registration retrieval code (in this case, it is 315798465-696065). When you log on to the internet, go to our web site at http://www.thinksrs.com/html/ddda.html and press the Register DDDA link.

Once you have successfully registered/purchased DDDA, the unlock code will also be emailed to you. This unlock code is good ONLY FOR THIS REGISTRATION RETRIVAL CODE. In other words, this unlock code will only work for THIS INSTALL OF THE PROGRAM. Enter the code in the space provided, and press Register. If the registration retrieval code and the unlock code are a matched set, you will see:



Thank you for registering your copy of DDDA. If you ever have to reinstall DDDA, you will need a valid copy of your registration license file. Enter or browse for the path where you wish to store your license file and press Save or OK.

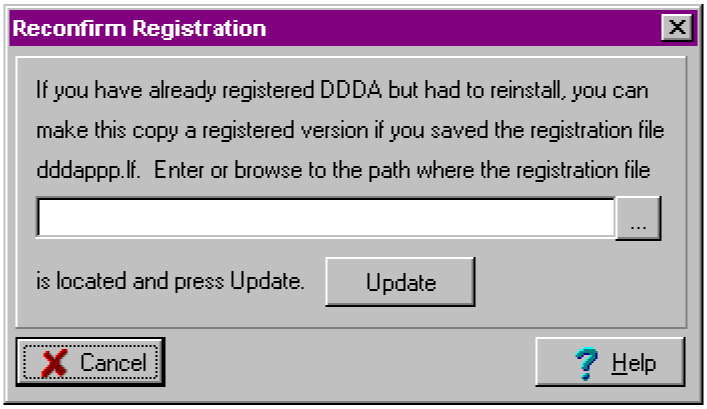
Enter a path here

Browse for the file here

Enter the path where you want to save the Registration File and press Save or OK. If you enter a filename, it will be ignored. The file must be called dddappp.lf. If you have not entered a path and press OK you will exit the form WITHOUT saving the file. Once you have successfully saved the file, a message will show telling you where the file has been saved. It is strongly recommended that you make a copy of your Registration file because it is the ONLY way to reinstall DDDA without purchasing it again. If you do not want to make the copy right now, you can always make a copy by selecting Help|About in the DDDA Main menu and pressing the Copy Registration File button.

Using a previously save Registration File

If you have already purchased DDDA, successfully registered it and MADE A COPY OF YOUR REGISTRATION FILE, then you can register DDDA without having to purchase it again. This option is made available to you so that if the unthinkable happens (your hard drive crashes, or other such horror) you can reinstall your registered copy of DDDA. (Remember, each registered copy of DDDA is legal on one and only one computer.) To use your registration file, press Reconfirm on the Registration screen and you will see:

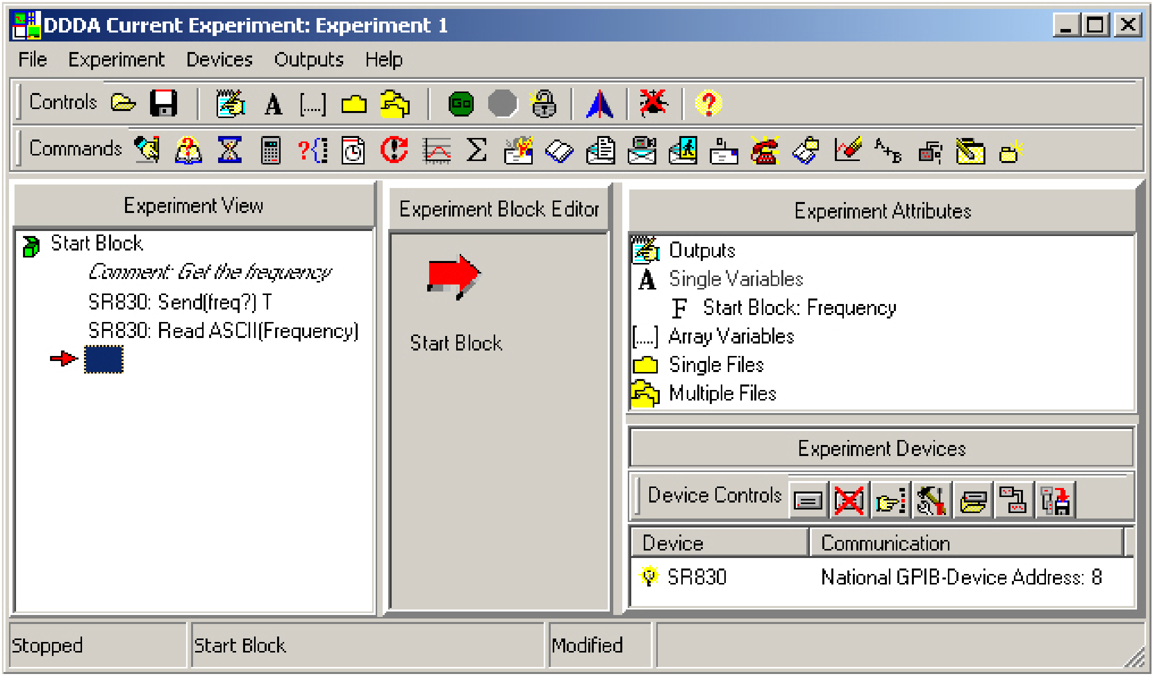


If you have already registered DDDA but had to reinstall, you can make this copy a registered version if you saved the registration file dddappp.lf. Enter or browse to the path where the registration file

is located and press Update.

Enter or browse to the path where your saved Registration file (dddappp.lf) is located and press Update. If the file is a valid Registration file, you will see a message stating that you are now registered

DDDA Main View



Start Block
Comment: Get the frequency
SR830: Send(freq?) T
SR830: Read ASCII(Frequency)

Start Block

Outputs
Single Variables
F Start Block: Frequency
[....] Array Variables
Single Files
Multiple Files

Device	Communication
SR830	National GPIB-Device Address: 8

Stopped	Start Block	Modified	

DDDA Main View Components

There are three main components in the DDDA Main View:

Experiment View: Leftmost Panel. This is where all the blocks and commands are listed. The block with the open box next to it is the current block. In order to add commands to the experiment, one block must be current. The arrow (and the blank box next to it) indicates where the next command will be added in the command list. All members of the Experiment View (including the blank box) have a popup menu associated with them. In addition, double clicking on any member of the Experiment View calls its editor view (except for the blank box, which has no editor). Commands are executed in the order they appear, however, blocks are not. Their order is determined in the:

Experiment Block Editor: Middle Panel. The first block to be run in an experiment is the Start Block (explained in Experiment Blocks and Paths). Once the Start Block is known, the path of the experiment can be followed by tracing the experiment paths (the lines with arrows). In the experiment shown above, the Start Block is the "Start Block" and the next block is "Block 1". See Experiment Blocks and Paths for more details. Right clicking anywhere in the Experiment Block Editor shows a popup menu.

Experiment Attributes: Rightmost Panel, upper half: The Experiment Attributes list, organizes the variables, outputs and files in the experiment. Attributes can be created, deleted and edited in this list. See the chapter on Experiment Attributes for more details.

Experiment Devices: Rightmost Panel, lower half: The Experiment Devices list allows the addition/removal of devices to the experiment. In addition the Experiment Devices section has tools for scanning for devices on a GPIB bus (assuming you have the right kind of GPIB card), interacting with devices without adding any commands to the experiment (this is useful to test that a good connection has been made with the instrument) and for saving a device to a DDDA readable file. Right clicking anywhere in the Experiment Devices list shows a popup menu. Double clicking on a device shows the Device Editor.

Status Bar: At the bottom of the DDDA form is a status bar that is divided into 4 sections. This left most section refers to the execution state of the experiment, running or stopped. The next section displays the currently active block. The third section refers to the state of the experiment. If the experiment has been changed in any way, the word Modified appears, otherwise it is blank. The forth and rightmost section is for hints. If you move your mouse over any speed button, a more detailed hint for that speed button will appear.

Speed Bars: At the top of the form, below the DDDA Main Menu are two speed bars, Controls and Commands. These provide shortcuts to the menu items found in the Main Menu and in various popup menus that appear in the Main View.

Reducing/Restoring the DDDA Main View: To reduce the DDDA Main View (which can be useful when running experiments) press the Reduce button in the Controls speed bar. When you reduce the DDDA Main View, only the Main Menu and the Controls speed bar will be visible and the Reduce button will become the Restore button. To show the DDDA Main view again, press the Restore button.

Launching Help: To get help on the DDDA Main view, press the Help button in the Controls Speed bar.

Exiting DDDA: To exit the program, either select File|Exit from the DDDA Main Menu or press the close button (the one with the X) in the corner of the DDDA Main View.

DDDA Files

There are 5 file extensions used by DDDA:

***.dae**: Experiment Files. These can be opened when DDDA is running by

- Selecting File|Open Experiment in the DDDA Main Menu and choosing a *.dae file.
- Selecting the file from a folder and dragging it into DDDA
- Double clicking on a *.dae file in a folder when DDDA is NOT running will launch the program with that experiment loaded.

***.blk**: Experiment Block Files. These can be opened when DDDA is running by

- Selecting File|Open|Experiment Block in the main DDDA Main Menu and choosing a *.blk file.
- Selecting Open Block from the Experiment Block popup menu (right click on a block).
- Selecting a *.blk file from a folder and dragging it into DDDA.

***.opt**: Inactive Output Files. These are outputs which contain previously recorded data. They cannot be used by a plot command. These can be opened when DDDA is running by

- Selecting File|Open|Output in the DDDA Main Menu and choosing a *.opt file.
- Selecting a *.opt file from a folder and dragging it into DDDA.

***.dev**: Device Files. These can be opened when DDDA is running by:

- Selecting File|Open|Device in the DDDA Main Menu and choosing a *.dev file.
- Selecting Open Device from the Device popup menu (right click on a device).
- Selecting a *.dev file from a folder and dragging it into DDDA.
- Clicking on the Open Device button in the Device Commands speed bar.

***.dcg**: Device Configuration Files. These can be opened when DDDA is running by:

- Selecting File|Open|Device Configuration in the DDDA Main Menu and choosing a *.dev file.
- Selecting Open Device Configuration from the No Device popup menu (right click anywhere in the Device list but on a device).
- Selecting a *.dcg file from a folder and dragging it into DDDA.
- Clicking on the Open Device Configuration button in the Device Commands speed bar.

Additional Help

If you can't find the answer to your questions in this manual (especially the FAQ's at the end of the manual) **the fastest way to get help is to email your question to dddasupport@thinksrs.com** (no, really, this is the fastest way!) If you are having problems with a particular experiment, block or device, save it and email the file as an attachment along with an explanation of the problem. If the problem is saving it, send a print out of the experiment (select Experiment|Experiment Report from the DDDA Main menu). Email that and we will take a look at it (see, this really is the best way!)

New in version 1.5

There are several new features in this version of DDDA:

Experiment Files: These are text files that can be written to during the course of an experiment. There are 2 types, Single File and Multiple File. Single Files maintain one filename throughout the experiment where Multiple Files can have more than one filename.

Output To File: This command sends the values of user specified variables to an Experiment File which then writes the values in ASCII format.

Create A New File: This command signals a Multiple File to create a new file.

In addition, the Wait command has been changed slightly. You will no longer have to wait for it to finish before aborting an experiment.

Chapter 2 DDDA Experiments

Introduction

The Experiment is what "runs" in both the DDDA Main View and the Debugger. An Experiment consists of one or more Experiment Blocks that are connected by Experiment Paths. Within an Experiment Block are Commands which are like lines of code in a traditional program. An Experiment can have one or more Devices which represent the instruments data is to be acquired from, Variables store data from a device, Outputs are used to display the variables and Experiment Files are used to write variable to ASCII text files.

Only one experiment can be open at a time. Experiments can be saved and opened from file (file extension *.dae). To save an experiment, select the File|Save menu item in the DDDA Main Menu. To save the experiment with a different name select File|Save As. An experiment can be opened in 3 ways:

1) The File|Open Experiment menu item in the DDDA Main View.
2) DDDA can be started with an experiment open by double clicking on a DDDA experiment file.
3) An experiment can be dragged into the DDDA Main View. If multiple experiments are selected, only the first one will be opened.

To create a new experiment, select File|New Experiment from the DDDA Main menu.
A list of recently opened and saved experiments can be found in the File|Reopen menu item in the DDDA Main Menu. This menu item is only enabled when experiment files have been opened or saved. To clear all the experiment files listed, select Clear from the Reopen menu. To open a file in the list, just select it.

Experiment Blocks and Paths

An experiment consists of one or more Experiment Blocks which are connected by Experiment Paths. It is the combination of blocks and paths the determines the course of the experiment. These blocks and paths are visually represented in the Experiment Block Editor.

Experiment Blocks

Each block consists of its own commands and variables. Experiment blocks can be saved independently. All its commands, variables, outputs, files and devices that its commands use are saved with the block. Variables can be read by other blocks but can only be written to (by a Read command, an Increment command, Serial Poll command, String Concatenate Command or a Calculate command) by commands that are within that block.

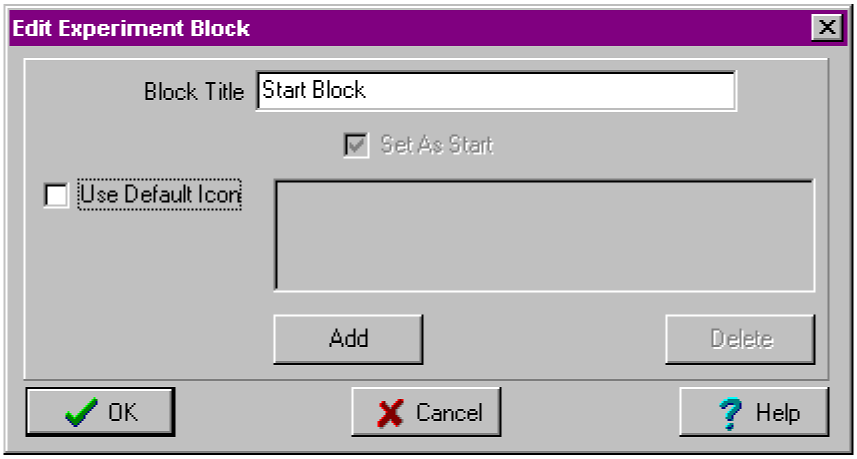
There are two types of Experiment Blocks, the Start Block and the Member Blocks. There must be one and only one Start Block per experiment (so if there is only one experiment block in the experiment, it is by default, the Start Block). There can zero to many Member Blocks. To determine/change the Start Block, right click on any block and Check/Uncheck the Start Block menu item. If there is only one block, unchecking the Start Block menu item does nothing.
To add a new block to an experiment do one of the following:

1) Right click anywhere there is not a block or path in the block editor and select the Add Block menu item.
2) Select File|New|Block from the DDDA Main Menu.

To read in a block from file, select File|Open|Block from the main menu or right click anywhere there is not a block or path in the block editor and select the Open Block menu item. Alternatively, a DDDA Experiment block file can be dragged into the DDDA main view and will be read and added to the experiment.

To delete an experiment block (remember, all the commands and variables disappear with it along with all paths entering it and all paths leaving it) right click on the block and select Delete from the menu. If there are no experiment blocks, no experiment can be run.

To edit the properties of the block, right click on the block in either the Experiment View or the Experiment Block Editor and select Edit Block from the popup menu. The following will show:


Edit Experiment Block

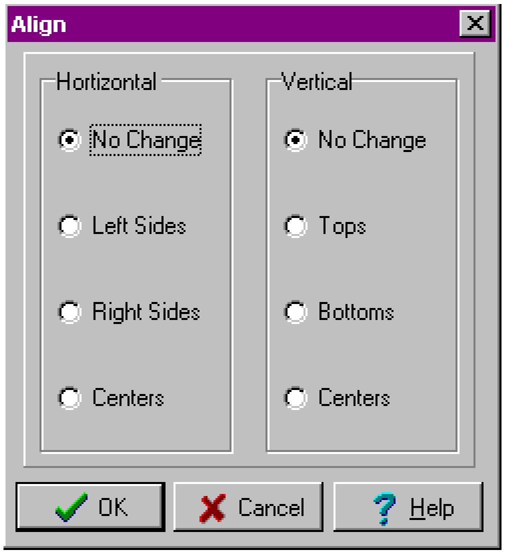
Block Title: Start Block

☑ Set As Start

□ Use Default Icon

The name is changed by changing the text in Block Title. The icon is changed by first unchecking the Use Default Icon check box. The icon list and Add and Delete buttons become visible (the Delete button is not enabled if there are no icons in the list). To add an icon, press the Add button and read in an icon from file (*.ico). Multiple icons can be read in and will be added to the list above the Add button. Select one of the icons and press OK. The icon then appears in the Block Editor. Icons can be removed from the list by selecting the icon and pressing Delete. The filenames of the icons are saved and will therefore reappear each time DDDA is started. The Set As Start check box is enabled if there is more one block in the experiment or if the block being edited is not the start block. Checking Set As Start and pressing OK, sets that block as the starting block in the experiment.

To align the blocks in the block editor, select the block you want to be the anchor block. While holding the Ctrl key, select (left click on) the other blocks that will be aligned with respect to the anchor. Release the Ctrl key and right click on any one of the selected blocks and the following form will show:


Align

Hortizontal
● No Change
○ Left Sides
○ Right Sides
○ Centers

Vertical
● No Change
○ Tops
○ Bottoms
○ Centers

Select how the blocks are to be aligned and press OK. The first block chosen is the anchor block (it does not move) the rest of the blocks align with respect to that one.

Experiment Paths

If there is more than one experiment block in an experiment, experiment paths connect the blocks. The path leaves one block (no arrow) and points, with an arrow, to the next block to be run in the experiment. The Start Block MUST have at least one path leaving it (assuming, of course, that there is more then one experiment block). Each block (Start or Member) can have up to two paths leaving it and many paths entering it. Only the Start Block can have 0 paths entering, all Member Blocks must have at least one path entering. If more that one path leaves any block, a Decision Command is added to that block and must be defined before the experiment can be run.

To add an experiment path, right click on the block that the path is leaving and select Connect Blocks from the menu (this menu item is disabled if there is only one experiment block). Now, when the mouse is moved, a line joins the block and the cursor. Click (either right or left) on the other end of the path. If there is no block or the original block is clicked on, no path is made.

To delete an experiment path, right click on the path and select Delete from the menu. Deleting the path does NOT delete the connected blocks.

Running Experiments

To run an experiment, either select Experiment|Start from the DDDA Main Menu, press the Go speed button in the Controls speed bar or press Ctrl G (the button and menu item are only enabled when the experiment is NOT running). To stop a running experiment, either select Experiment Stop from the DDDA Main Menu, press the STOP speed button in the Controls speed bar or press Ctrl S (the button and menu item are only enabled when the experiment IS running).

The Start and Stop speed buttons in the Controls speed bar can be "floated" on their own form by pressing the Unlock 🔓 speed button in the Controls speed bar. When this is done, the buttons will disappear from the Controls speed bar and the Unlock button will change to 🔒 the Lock button. To "anchor" them back onto the Controls speed bar, press the Lock speed button.

Experiment Report

This view is shown when the Experiment|Experiment Report menu item is selected in the DDDA Main Menu. This view produces a report of the current experiment based on your choices:

Command Listing: Lists all the commands as they appear in the Experiment View. The one exception is the Read Command. Underneath the command string, in italics, is a description of how the data is read (how many bytes, binary or ASCII, etc).

Output Listing: Lists all the outputs in the current experiment, their type and the number of commands that use them.

Single Listing: Lists all the single variables in the current experiment, their data type, the experiment block they belong to and the number of commands that use them.

Array Listing: Lists all the arrays in the current experiment, their data type, the experiment block they belong to, their size, and the number of commands that use them.

Single Files: Lists all the single files in the current experiment, their names, whether they close on write, whether existing files are overwritten, and the number of commands that use them.

Multiple Files: Lists all the multiple files in the current experiment, their names, whether they close on write, what is done when the filename already exists, the starting index, and the number of commands that use them.

Device Listing: Lists all of the devices along with how they communicate with the program.

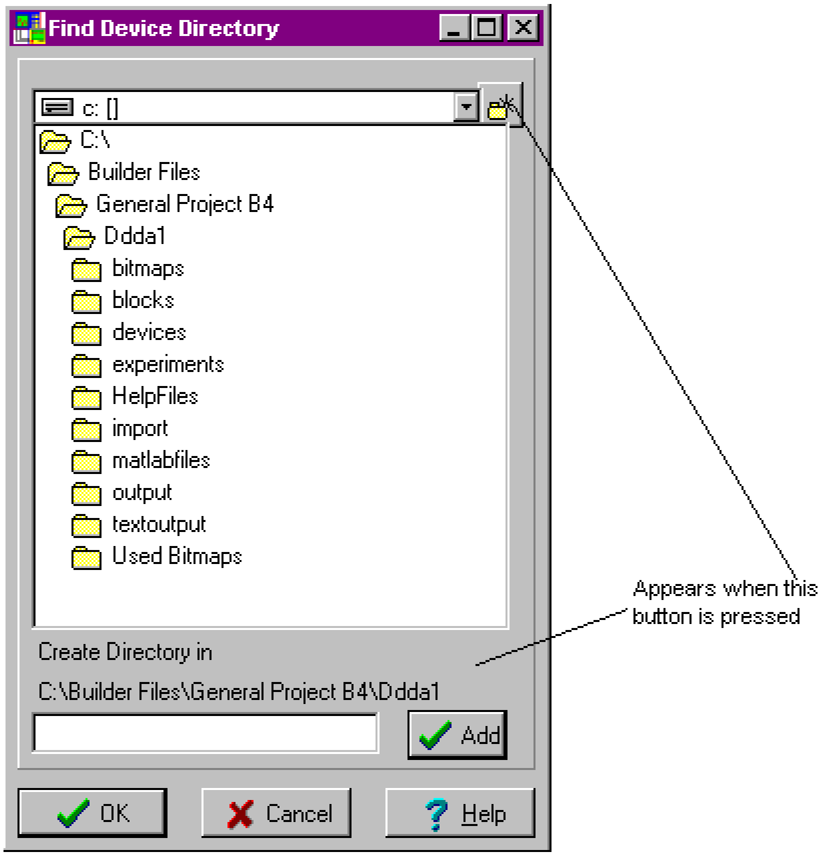
Device Commands: This option is only visible if Device Listings is checked. All the device commands and their parameters are added to the device listing.

Checking and unchecking choices, changes the report as it appears in the Memo field. There are 3 buttons at the bottom of the form for output of the report:

Print: Pressing the Print button shows a print dialog form where a printer can be chosen. Selecting **OK** sends the report to that printer.
Clipboard: Pressing the Clipboard button, sends all the text in the report to the Windows clipboard.
File: Pressing the File button shows a file save dialog form where the text file to save to can be chosen. If an *.rtf (rich text file) file is selected all formatting will be preserved. If any other file type is chosen, a plain text file will be saved (no formatting will be preserved).
Done: Closes the Experiment Report view.

Preferences

Selecting Experiment|Preferences from the DDDA Main Menu shows a dialog box that sets the default directories for experiments, outputs, devices and blocks. The directory can either be typed into the edit box or browsed for using the button to the right of the edit box. Only existing directories can be used. If a directory does not exist an error message is given. A directory can be created by pressing the Browse button:



C:\
Builder Files
General Project B4
Ddda1
bitmaps
blocks
devices
experiments
HelpFiles
import
matlabfiles
output
textoutput
Used Bitmaps

Create Directory in

C:\Builder Files\General Project B4\Ddda1

Appears when this button is pressed

The directory that is selected (highlighted) is the directory that will be used. To create a directory, use the new directory button at the top of the form. This shows the Create Directory panel. Select the parent directory for the new directory and press Add to create and select the new directory.

The browse dialog box is also shown when finding a directory for Multiple Files.

Chapter 3 Commands

Introduction

Commands are the instructions for the experiment. They exist within an experiment block and therefore a block must be selected before commands can be added to the experiment.

Adding Commands

There are two ways to add commands to an experiment:

1) Press on the speed buttons in the Commands tool bar at top of the Main Form.
2) Right click on the command below where you want to add a command. A popup menu will appear and one of the selections is Insert. Selecting Insert will reveal a submenu where you can select which command to add.
3) Right clicking on the blank command will reveal a popup menu with a list of commands.

Deleteing Commands

To delete a command in the command list, either

1) Select the command in the command list, keep your cursor over the command and press the delete button on your keyboard.
2) Select the Delete Command from the popup menu (right click on the command).

Editing Commands:

To edit a command, double click on the command you want to edit in the Experiment View or right click on the command and select Edit from the popup menu.

Copying, Pasting and moving commands:

To copy a command, right click on the command you want to copy, and select Copy from the popup menu. To paste the copied command, right click on the command below where you want to paste the command, and select Paste from the popup menu. Paste will be disabled if pasting the command is illegal. To move a command, select the command and drag it to its new position. The command will be inserted above the command it is dragged to. Dragging the command to an experiment block moves the command to the end of the command list for that block.

Commenting Out/Restoring Commands:

To comment out a command, right click on the command you want to comment out.   A popup menu will appear and one of the selections is Comment Out. Once you have selected Comment Out, the command string will become */*command*/.* When the program is run, the command will be skipped. To restore the command, right click on the command you want to restore.   A popup menu will appear and one of the selections is Restore. Once you have restored the command the italics will disappear along with the /**/.

Not all commands can be commented out. Comments, Labels without a repeat command and Decision commands cannot be commented out.   In addition, commenting out one command may cause other commands to be commented out:

1) If Then, Else and End If commands: If you comment out an If Then command, the corresponding Else and End If commands will also be commented out. If you comment out an End If command, the corresponding If Then and Else commands will be commented out. If you comment out the Else command, only the else command will be commented out. Restore follows the same rules.
2) Label and Repeat commands: If you comment out a label with a repeat command, the repeat command will be commented out as well. Restore follows the same rules.

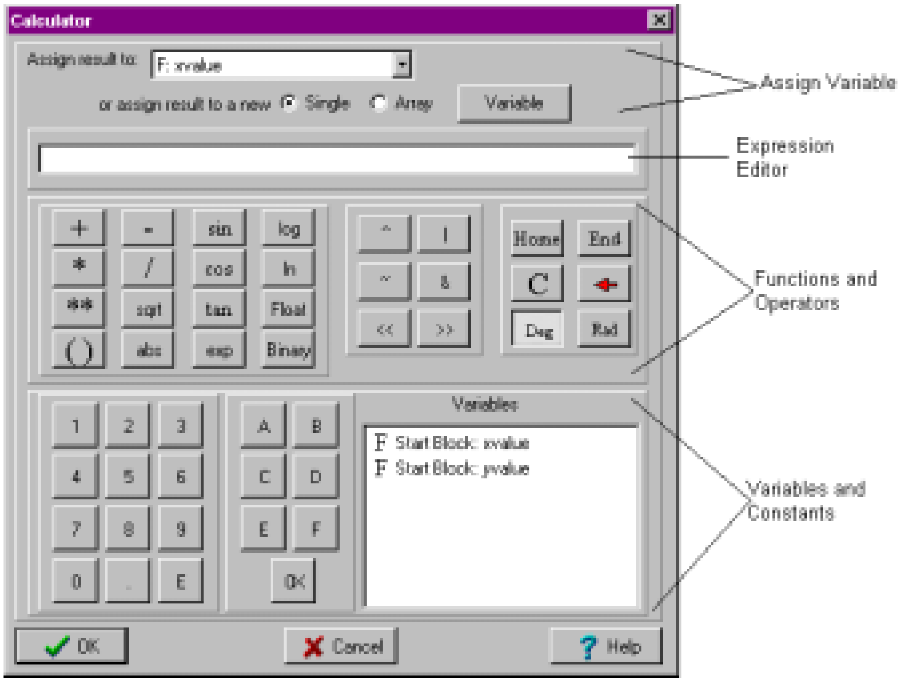
The rest of this chapter describes the commands and command editors available in DDDA.

Calculate Command

Description

This variable command allows you to perform mathematical calculations on float and integer Variables.

The Calculate Editor



Calculator

Assign result to: F: xvalue

or assign result to a new ● Single ○ Array

Variables

F Start Block: xvalue

F Start Block: yvalue

The Calculate editor is shown when:

- The Calculate speed button is pressed in the Commands tool bar at top of the DDDA Main View.
- The Calculate menu item is selected from the command popup.

How to use the Calculate editor:

In the Assign Variable area, you can either choose a variable to assign the result to, or you can select either Single or Array and then press Variable to add a new variable. Only float and integer variables can be used and if an array is chosen, an index must defined in the neighboring index combo box.

The Expression Editor is where the expression to be evaluated is defined. You can type in the expression or you can use the Function and Operators pad and the Variables and Constants pad.

On the left most pad of the Functions and Operators section are the decimal operators, add (+), subtract (-), multiply (*), divide (/). and power (**). Also included are the decimal functions sin, cos, tan, exp (inverse of natural log), log (log base 10), ln (natural log), square root (sqrt) and absolute value (abs). Decimal operators and functions can only be used with decimal (float and integer) constants and variables. On the middle pad of the same section are the binary operators, exclusive OR (^), Inclusive OR (|), NOT (~), AND (&)., Left Shift (<<) and Right Shift (>>). Binary operators can only be used with binary data and binary constants. The Float and Binary buttons on the leftmost pad are used to convert decimal and binary data.

Float takes binary data and interprets it as a float and Binary takes float data and treats it as binary data. For example, say you have serial polled a device and want to see if the device has requested service. In this case, you would take the resulting integer value from the poll, convert it to a binary and AND the value with 0x40: *Float(Binary(serialpollresult) & 0x40).* The Float function needs to be the outermost function because the result must be a float or integer(since only float and integer values can be used in Assign Variable).

The Home Button on the Functions and Operators rightmost pad takes the cursor in the Expression Editor and moves it to the beginning of the expression and the End Button moves the cursor to the end of the expression. C clears the Expression Editor and the Arrow Key is just like a backspace key in any text editor (it erases the character just to the left of the cursor). Deg and Rad only have meaning when using the trig functions sin, cos and tan. If the expression in the trig function is to be interpreted as radian, make sure Rad is selected. If the expression in the trig function is to be interpreted as degrees, make sure Deg is selected.

Variables can be added to the expression by double clicking on them in the Variables box or can be directly typed into the Expression Editor. If typing in, do not include the Block Name in the variable name. Remember that spelling is important but case is not. If an array is chosen it will appear in the Expression Editor as: *myarray[ ]* (if the user types it in, it must be in this form). The user needs to add the index between the brackets. Only a single integer variable or an integer can be between the brackets. Expressions are not allowed.

Integer and float constants can be entered by using the numeric pad (in Variables and Constants) or by directly typing them into the Expression Editor. If using binary constant, make sure to precede the number with *0x*.

Copying, Pasting, Moving and Comment Out Rules

This command can be copied. It can be pasted or moved anywhere within its parent block. This command can be commented out.

Debugger Output

In the debug output, the description of Calculate is the expression with the current values of all the variables. The result is the evaluation of the expression.

Additional Comments

If the result is assigned to an integer, any decimal places in the result are truncated (no rounding).

Clear Output Command

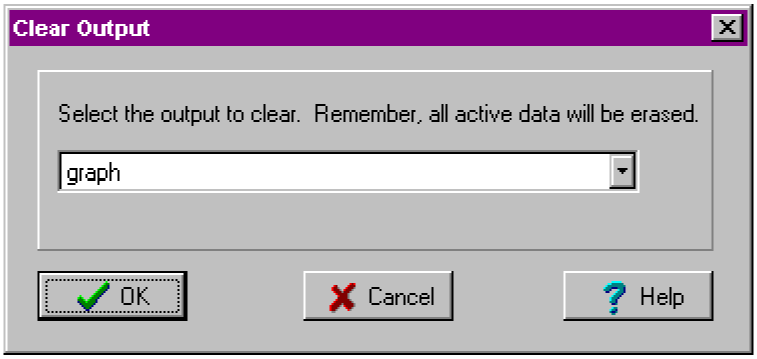
Description

This is a variable command which allows you to clear all or some of the data from the selected output. **Warning: The data is permanently lost when it is cleared.** What data is cleared depends on the type of output:

Text and BigNum Outputs: All data is cleared.

Table and Graph Outputs: All active data is cleared. Imported data is not cleared.

The Clear Output Editor



Select the output to clear. Remember, all active data will be erased.

graph

The Clear Output editor is shown when:

- The Clear Output speed button is pressed in the Commands tool bar at the top of the Main Form.
- The Clear Output menu item is selected from the Command popup.

How to use the Clear Output editor

Select the output to be cleared and press OK.

Copying, Pasting, Moving and Comment Out Rules

The Clear Output command can be copied. It can be pasted and moved anywhere. This command can be commented out.

Debugger Output

In the debug output the description of the Clear Output command is *Clearing output output name* where output name is the name of the output being cleared. There is no result.

Clear a Device's Read Buffer Command

Description

Clear a device's read buffer allows the user to clear the read buffer of a device. The command reads from the device until a timeout occurs. If there is only one device in the experiment, pressing the button in the speed bar or selecting the Clear Device Read Buffer menu item from the popup menu adds the command without showing an editor. If there is more than one device, an editor will show that allows the user to select the device whose read buffer is to be cleared.

Copying, Pasting, Moving and Comment Out Rules

This command can be copied. It can be moved or pasted anywhere in the experiment. This command can be commented out.

Debugger Output

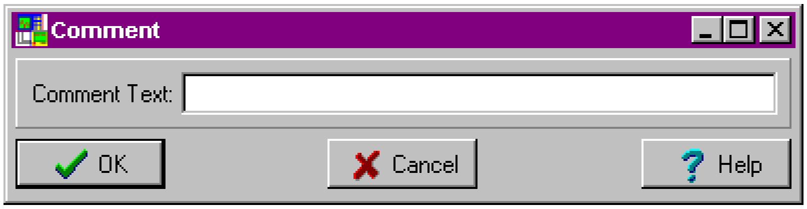
In the debug output, the description of Clear a Device's Read Buffer is just the phrase *Cleared buffer of device name*. There is no result.

Comment Command

Description

This is a general command which allows you to add a comment to the experiment. Comments do nothing when the experiment is run.

The Comment Editor



The Comment editor is shown when:

- The Comment speed button is pressed in the Commands tool bar at top of the Main Form.
- The Comment menu item is selected from the command popup.

How to use the Comment Editor

To define a comment, enter the comment text in the edit box (to the right of Comment Text).

Copying, Pasting, Moving and Comment Out Rules

Comments can be copied and moved anywhere. This command cannot be commented out since it is already a comment.

Debugger Output

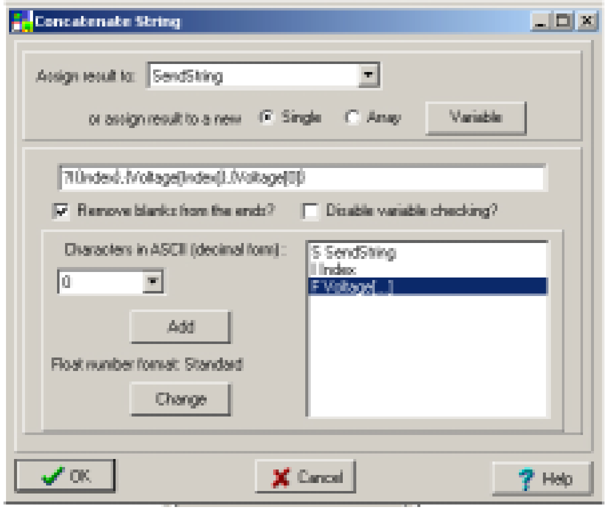
In the debug output, the comment has neither a description or a result.

Concatenate String

Description

Concatenate string allows the user to create a new string from other variables and user defined strings.

The Concatenate String Editor



Concatenate String

Assign result to: SendString

or assign result to a new ● Single ○ Array

?I{Index},{Voltage[Index]},{Voltage[0]}

☑ Remove blanks from the ends? ☐ Disable variable checking?

Characters in ASCII (decimal form) : 0

Float number format: Standard

S SendString
I Index
F Voltage[...]

The Concatenate Strings editor is shown when:

- The Concatenate String A+B speed button is pressed in the Command Speed Bar.
- The Concatenate String menu item is selected from the command popup (right click on a command in the Experiment View)

How to use the Concatenate Strings Editor
First, select the string variable which will hold the resulting string. One can be created by selecting the type (single or array) and pressing the Variable button.

Enter the string you wish to create in the edit box. The above example shows the string "?I{Index},{Voltage[Index]},{Voltage[0]})". This consists of text, a variable and an array, twice. Text is entered using either the keyboard or by selecting a character in the Characters in ASCII drop down box (the corresponding character will appear to the right) and pressing the Add button. A variable is entered using the format {*variable name*} where *variable name* is the name of the variable or by double clicking on the variable in the variable box. If the variable is an array, the format {*array name*[*array index*]} where *array name* is the name of the array and the *array index* is either an integer greater than 0 or the variable name of a single integer variable. Note, there are no {} around the *array index*. If you enter a variable index using the variable box, **you will need to remove the braces**.

The Change button is used to set the float number format when the value of a float variable is written to the result string. See the Number Format dialog box for details about the available formats.

If the Remove blanks from the ends? box is checked, any blanks that appear at the beginning or ending of the string in the edit box will be removed before the command is set.

If the Disable variable checking box? is checked, the program will not require that the string between the curly braces is a variable. If it is not a variable, it will just store it as a string. Uncheck this only if you need to send a {or a } to your instrument.

Copying, Pasting, Moving and Comment Out Rules

This command can be copied. It can be pasted or moved anywhere within the block it was created in. This command can be commented out.

Debugger Output

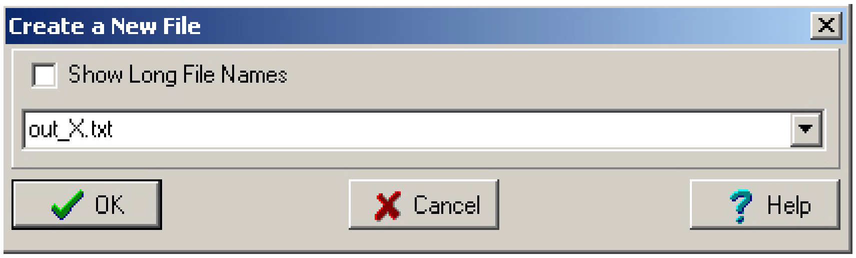
In the debug output, the description of Concatenate String is the original expression. The result is the result string.

Create A New File Command (New in version 1.5)

Description

Calling this command during an experiment closes the current Multiple File specified in the command (if it is open) and resets its filename to the next file

The Create A New File Editor



□ Show Long File Names

out_X.txt

The Create a New File editor is shown when:

- The **Create New Command** speed button is pressed in the **Command Speed Bar**.
- The **Create New Command** menu item is selected from the command popup (right click on a command in the **Experiment View**)

How to use the Create a New File Editor:
Select which Multiple file to use and press OK. Checking the **Show Long File Names** box shows the path to the file.

This editor cannot be shown when there are no Multiple File Attributes.

Copying, Pasting, Moving and Comment Out Rules

This command can be copied. It can be pasted or moved anywhere in the experiment. This command can be commented out.

Debugger Output

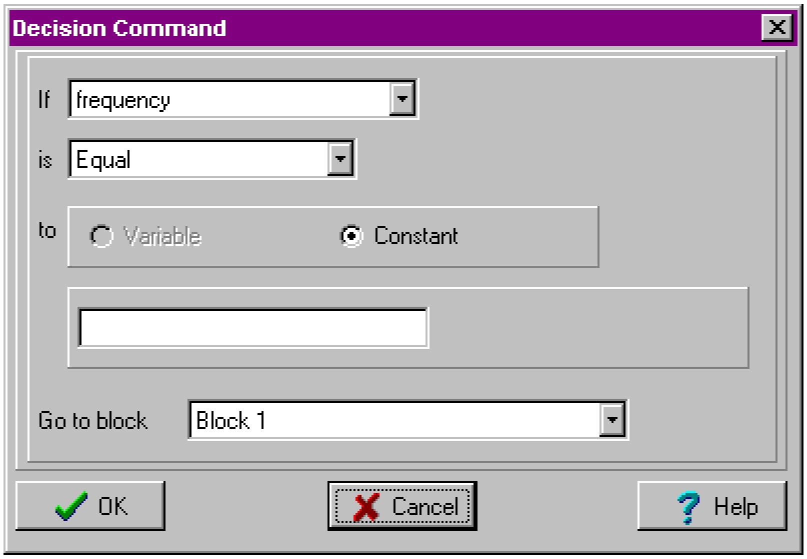
In the debug output, the description of Create A New File is the phrase *Creating new file* and the name (including path) of the file that will be written to the next time the Output To File command is run. There is no result.

Decision Command

Description

This is a control command that is automatically added to a block when it has 2 experiment paths leaving it. (It cannot be independently added by the user.) This command determines which block follows its parent block in the experiment.

The Decision Editor



Decision Command

If frequency

is Equal

to ○ Variable ● Constant

Go to block Block 1

The Decision editor is shown by double clicking on the Decision command in the Experiment View. The Decision Editor will only be shown if there is one or more variables already defined in the block

How to use the Decision Editor:

The top part of the editor is used to define a boolean expression. The lower part (Go to block) is used to select which of the two blocks the parent block has experiment paths to, will be the next block, if the boolean expression evaluates to true. The boolean expression is defined by comparing the current value of the selected variable (in the above example, frequency) to either another variable, if one exists, or a constant. In the above example, there are no other variables in the experiment so the Variable radio button is grayed out. String variables can only be compared using an equal expression as where other variables can be compared using equal, greater than, greater than or equal, less than and less than or equal (in the above example it is equal).

Copying, Pasting, Moving and Comment Out Rules

This command cannot be copied. It can be moved anywhere within its parent block. This command cannot be commented out.

Debugger Output

In the debug output, the description for the Decision command is *n condition x evaluates to true/false* where n is the current value of the variable to be compared and x is the current value to compare to n using the condition. The result is *Goto block blockname* where blockname is the block that follows its parent block.

Additional Comments

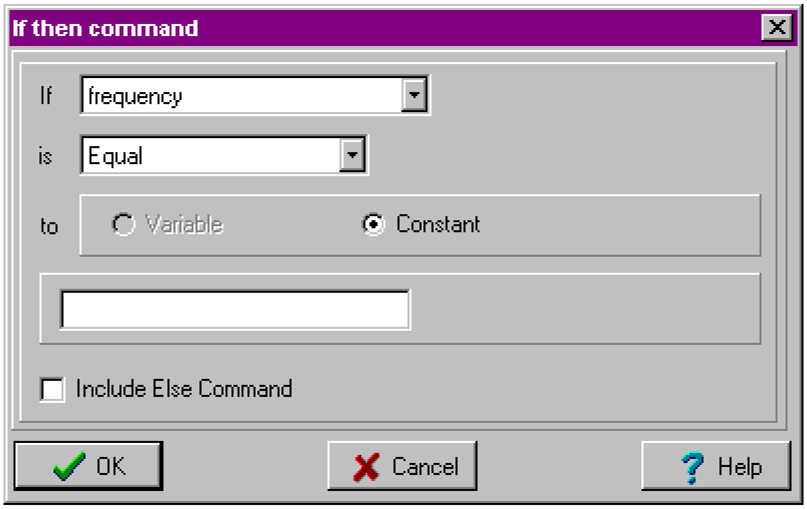
This command can only be removed by deleting one of the experiment paths.
In an experiment in which one or more blocks has a decision command, the experiment cannot be run until this command is defined.

If Then Else Command

Description

This control command allows you to execute different commands based on the outcome of a boolean expression.

The If Then Else Editor



If then command

If frequency

is Equal

to ○ Variable ● Constant

□ Include Else Command

The If Then Else editor is shown when:

- The If Then Else ?{ speed button is pressed in the Commands tool bar at top of the Main Form.
- The If Then Else menu item is selected from the command popup

The If Then Else Editor will only be shown if there is one or more variables already defined in the block

How to use the If Then Else Editor:

The boolean expression is defined by comparing the current value of the selected variable (in the above example, frequency) to either another variable, if one exists, or a constant. In the above example, there are no other variables in the experiment so the Variable radio button is greyed out. String variables can only be compared using an equal expression as where other variables can be compared using equal, greater than, greater than or equal, less than and less than or equal (in the above example it is equal). Checking Include Else Command will add an Else command between the If and EndIf commands.

Copying, Pasting, Moving and Comment Out Rules

If then else commands cannot be copied. They can be moved within certain restrictions. The If command cannot be moved below the Else command (if it exists) or below the EndIf command. The Else command (if it exists) cannot be moved below the EndIf command or above the If command. The EndIf command cannot be move above the Else command (if it exists) or above the If command.

If Then, Else and End If commands can be commented out. If you comment out an If Then command, the corresponding Else and End If commands will also be commented out. If you comment out an End If command, the corresponding If Then and Else commands will be commented out. If you comment out the

Else command, only the else command will be commented out. Restore follows the same rules.

Debugger Output

In the debug output, the description for the If statement is *n condition x evaluates to true/false* where n is the current value of the variable to be compared and x is the current value to compare to n using the condition. There is no result. If the statement evaluates to true, then the commands beneath the If command will be shown otherwise, the next command will be the else, if it has been defined, or endif if else has not been defined. If an else statement has been defined, then Else will appear. Else has no description or result.   If the commands beneath else have been executed, then they will appear, otherwise, the next command will be endif. Endif has no description or result.

Additional Comments

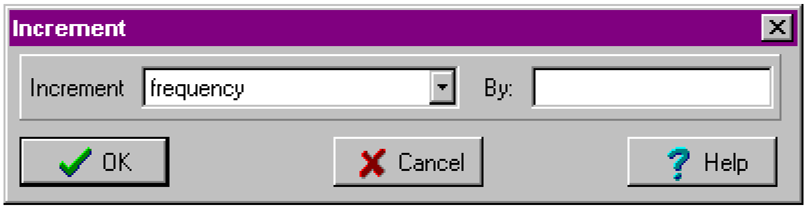
All commands between the If and the endif are indented. These commands are executed if the expression evaluates to true. Likewise, if the else command is included, all commands between the else and the endif command are indented. These commands are executed if the expression evaluates to false.

Increment Command

Description

This variable command is used to increment a float or integer variable by a specified amount.

The Increment Editor



Increment frequency By:

The Increment editor is shown when:

- The Increment Σ speed button is pressed in the Commands tool bar at top of the Main Form..
- The Increment menu item is selected from the command popup

How to use the Increment Editor:

At least one float or integer variable must already be defined in the block order create this command. Select the float or integer variable from the drop down list (to the right of Increment) and enter the amount you want to increment the variable by in the edit box (to the right of By).

Copying, Pasting, Moving and Comment Out Rules

This command can be copied. It can be moved or pasted within its parent block. This command can be commented out.

Debugger Output

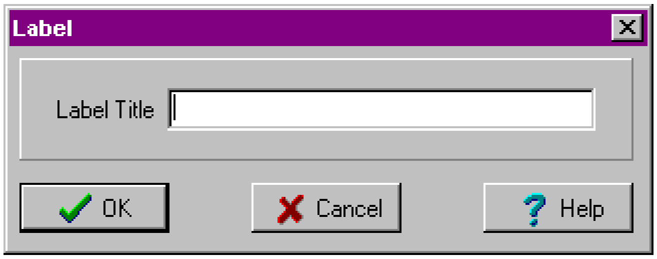
In the debug output, the description of the Increment command is *variable name=n+x* where n is the current value of the variable and x is the value incrementing by. The result is *Current value of counter=n* where n is the new value of the variable.

Label Command

Description

This can be a general command or a control command depending on how it is used. The Label command does nothing when it runs, it is just a label. However it is used by the Repeat and Repeat Until commands to form command control loops. All commands which use a Label must follow the Loop Rules.

The Label Editor



The Label editor is shown when:

- The Label speed button is pressed in the Commands tool bar at top of the Main Form..
- The Label menu item is selected from the command popup.

How to use the Label Editor

To define the label, just enter the title in the edit box (to the right of Label Title)

Copying, Pasting, Moving and Comment Out Rules

The Label command can be copied. It can be pasted anywhere (any Repeat or Repeat Until commands are NOT copied). If there is no Repeat or Repeat Until command associated with the Label, then it can be moved anywhere. Otherwise, it cannot be moved below the associated Repeat or Repeat Until command.

This command can be commented out **ONLY** if there is a repeat command associated with it. It will comment out the repeat command as well.

Debugger Output

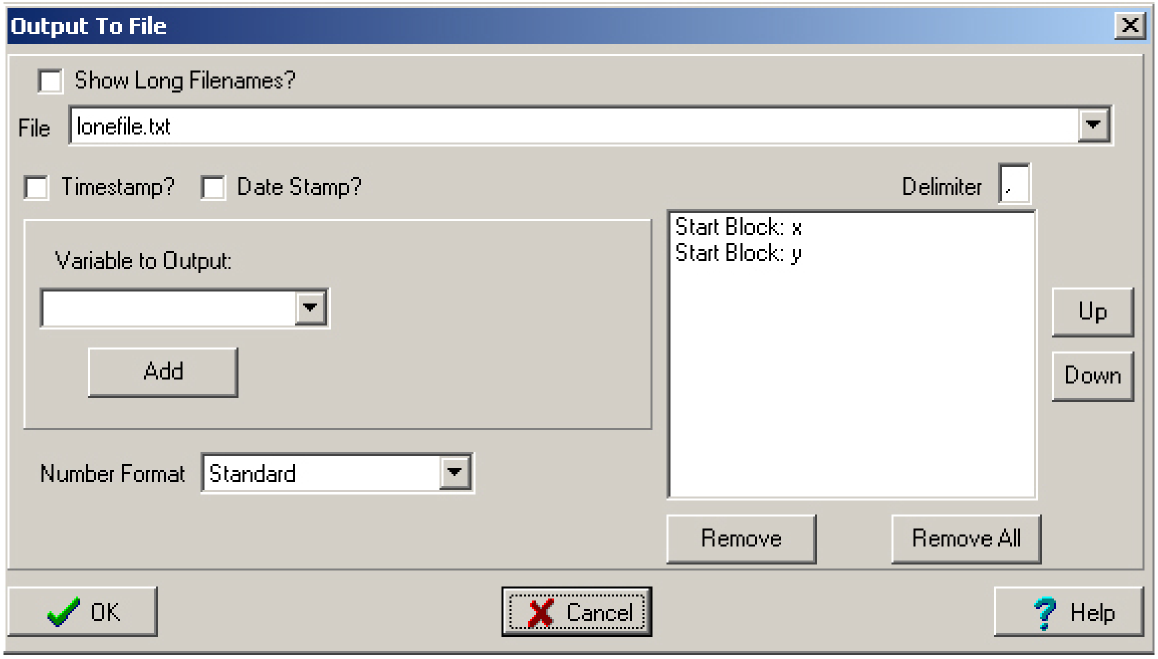
In the debug output, the description of the Label command depends on if there is a Repeat or Repeat until command that uses it. If there is a Repeat command, then the line *Repeating from Repeat n times* is written, where n is the number of times the commands are repeated. If there is a Repeat Until command then *Repeating from Repeat from label Until condition* where condition is the condition defined in the Repeat Until command is written. Otherwise, the line *Nothing repeats here* appears. There is no result.

Output To File Command (New in Version 1.5)

Description

This command writes a single line of text to the specified Experiment File.

The Output To File Editor



Output To File

□ Show Long Filenames?

File lonefile.txt

□ Timestamp? □ Date Stamp?

Delimiter ,

Variable to Output:

Number Format Standard

Start Block: x
Start Block: y

The Output To File Editor is shown when:

- The Output To File Command speed button is pressed in the Command Speed Bar.
- The Output To File Command menu item is selected from the command popup (right click on a command in the Experiment View)

How to use the Output to File Editor

First, select which Experiment file should receive the output. Then select the variable to output from the combo box and press the Add button. When this command is run, the values of the variables listed are written as one line to the file specified in File. The values are separated by the Delimiter (in the above example, it is a comma). The variables are written from left to right, where the top value is the leftmost value moving right, in order, to the bottom value. The order of the variables can be changed by selecting the variable to move and pressing the Up or Down button. If Timestamp and/or Date Stamp is checked, the current time and/or date are the first values written to the line.

Number Format is the same as that of the plots and the Send command. See the Number Format help topic for more information.

Remove, removes the selected variable from the list and Remove All removes all the variables from the list.

If an entire array is to be output, the array values are written in order, from left to right starting with the 0 index.

A Variable selected in the **Variable to Output Combo** box is not added to the list until the **Add** button is pressed.

This editor can only be shown when there are variables and experiment files.

Copying, Pasting, Moving and Comment Out Rules

This command can be copied. It can be moved or pasted only within its original block. It can be commented out.

Debugger Output

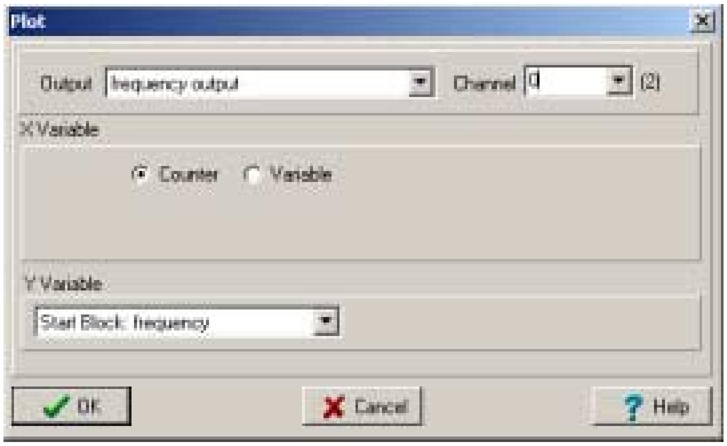
The description is just the values being written to the file. There is no result.

Plot Command

Description

This variable command sends the current value of up to two variables to an output.

The Plot Editor



Plot

Output frequency output Channel 0 (2)

X Variable

● Counter ○ Variable

Y Variable

Start Block: frequency

The Plot editor is shown when:

- The Plot speed button is pressed in the Commands tool bar at top of the Main Form.
- The Plot menu item is selected from the command popup.

How to use the Plot Editor:

The plot editor can only be called if there is at least one output and one variable already defined in the experiment. In addition, there must be at least one variable/output pair that is compatible. For example if there is only one output, a graph, and only one variable, a string, you cannot create a plot command since graphs cannot plot strings.

The channel box is only enabled if there is more than one channel in the output. In the above example, there are 2 (the "(2)" next to the channel box) channels. Enter either a integer variable or an integer that is greater than or equal to 0 but less the the number of channels in the output. Be careful when using a variable. If the value of the variable is less than zero or greater than the number of channels in the output, a experiment aborting error will result when running the experiment.

Depending upon the output, there can be up to three options for plotting:
Counter vs Variable: An internal integer counter is incremented every time the plot command is called during a running experiment. For Text and BigNum outputs the counter and variable values are delimited by a comma. For a table output, the counter and variable values are in separate columns, the counter being to the left. For a graph output, the counter is plotted on the x axis and the variable value plotted on the y axis. If the variable is an array, you can choose to plot each element in the array or you can choose to plot the entire array during one plot command. Choose the index from the Index Combo Box (Entire Array will be one of the drop down options). Plot entire array is not an option for BigNum outputs.

VariableX vs VariableY: This is similar to Counter vs Variable except instead of an internal counter, the current value of VariableX is plotted. Plotting the entire array is not an option for this case.
None vs Variable: This is only available to Bignum and Text outputs. Only the current value of the variable is plotted. Plotting the entire array is an option if the output is of type Text.

Copying, Pasting, Moving and Comment Out Rules

A Plot command can be copied. It can only be moved or pasted within its parent block. This command can be commented out.

Debugger Output

In the debug output, the description of the Plot command is just the value(s) it is plotting. There is no result.

Additional Comments

If an output's type is changed, all the plot commands are either changed to that output or are deleted if the variables they are sending to the output are incompatible.

More than one plot command can share an output.

Examples of creating plot commands are in the tutorials.

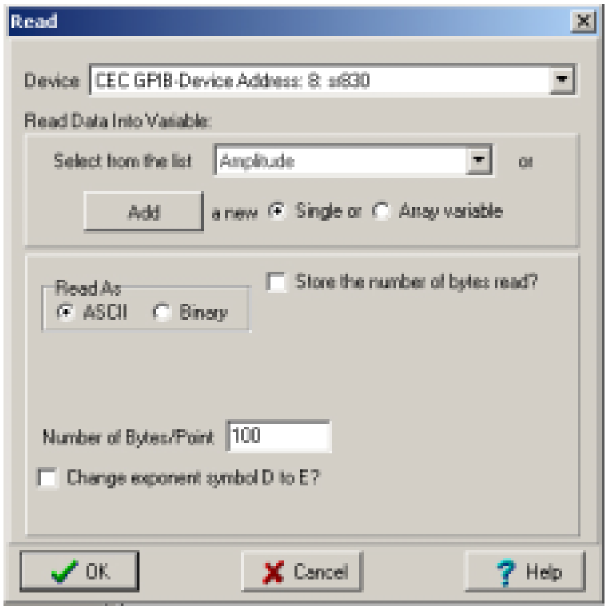
If you reduce the number of channels in the output, any plot command that plots to one of the removed channels will be deleted.

Read Command

Description

This instrument command reads bytes from an instrument, intreprets those bytes into one or more values and then writes those value(s) into a variable.

The Read Editor



Device CEC GPIB-Device Address: 8: sr830

Read Data Into Variable:

Select from the list Amplitude or

a new ● Single or ○ Array variable

Read As ● ASCII ○ Binary

□ Store the number of bytes read?

Number of Bytes/Point 100

□ Change exponent symbol D to E?

The Read editor is shown when:

- The Read speed button is pressed in the Commands tool bar at top of the Main Form..
- The Read menu item is selected from the command popup.
- The Commands|Read menu item is selected from the device popup.

How to use the Read Editor:

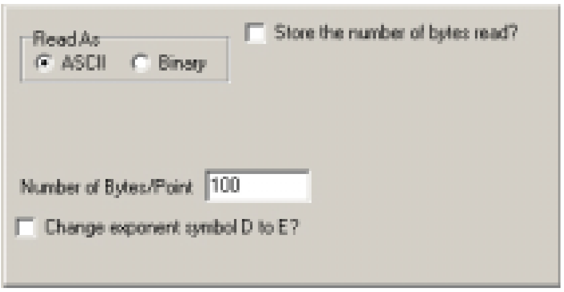
The Read editor always has two selections:

- The Device Selector where you can choose what device to read data from.
- The Variable Selector where you can either select or create a new variable in which to store the data read from the selected device.

The rest of the editor's appearance depends on 2 things:

- The number of values to be read in. If the variable selected is a single variable, then only one value can be read in. If however, that variable is an array, then you must define how many data points will be read into the array.
- How the data is to be read in, as ASCII or as binary.

For example, if you have selected a single variable to be read in as ASCII, the bottom third of the form will appear as:



Read As: ● ASCII ○ Binary

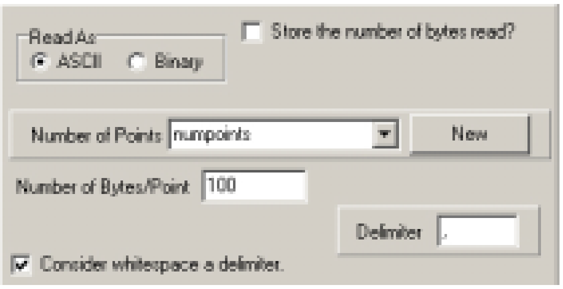
□ Store the number of bytes read?

Number of Bytes/Point 100

□ Change exponent symbol D to E?

Three things can be modified, the number of bytes/point that should be read (in this case, there is only one point so this is the total number of bytes read), whether or not to store the number of bytes read (explained later) and whether to whether to change the exponent symbol D to the accepted letter E. (New in version 1.4). There are some older instruments that use the symbol D, meaning double, instead of an E (1.0D-1 instead of 1.0E-1). This is only visible when reading into a float variable.

If however, you have chosen to read ASCII data in to a string array, the following would appear on the bottom third of the form:



Read As: ● ASCII ○ Binary

□ Store the number of bytes read?

Number of Points numpoints

Number of Bytes/Point 100

Delimiter ,

☑ Consider whitespace a delimiter.

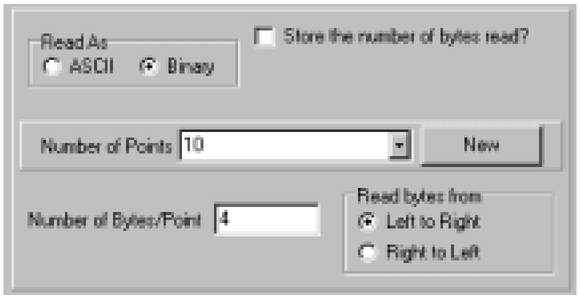
Three things have appeared, Number of Points, Delimiter, and Consider whitespace a delimiter:

- The **Number of Points** is the number of point to be read. This can be either an integer value or a integer variable (pressing New allows the user to add an integer variable). The number of points to be read cannot exceed the size of the array variable. Use caution when assigning a variable to this property. If the value of this variable exceeds the size of the array variable when running an experiment, the experiment will abort with an error.

- The **Delimiter** is a character that distinguishes one value from the next in the read array. For example, for the SR830, there is a command TCRA ? i, j, k which queries the points stored in the channel i, j and k buffer. The values are returned as ASCII float point numbers, multiple points being separated by commas. So for this case, the delimiter would be a comma.

- If **Consider whitespace a delmiter** is checked, characters separated by whitespace will be parsed into different values (see the section How DDDA Intreprets Data in Chapter 5). This checkbox only appears for string arrays and it is by default, checked. Float and integer arrays assume that whitespace is a delimiter. If whitespace is the only delimiter, as long as this is checked, any value can be put in the **Delimiter** edit box and the values will be parsed properly.

What if you need to read in a binary array? In that case, the following would appear:



Read As ○ ASCII ● Binary

□ Store the number of bytes read?

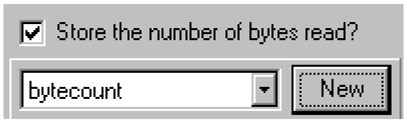
Number of Points 10

Number of Bytes/Point 4

Read bytes from
● Left to Right
○ Right to Left

Notice that now, the Number of Bytes/Point is 4 (the maximum number allowed for binary data) and the Delimiter box has been replaced with Read bytes from. Read bytes from indicates how the bytes are to be interpreted when storing them in *freqarray*. When the data is read, the program converts the bytes to, in the above example, a float value. The program needs to know if the most significant byte is sent first (read right to left) or the most significant byte is sent last (read left to right). The order depends on the instrument sending the data. If the number of bytes per point is one, this is ignored.

Now, about Store the number of bytes read. If this box is checked, the integer single variable combo box and the New button become visible:



☑ Store the number of bytes read?

bytecount

New allows the addition of a new integer variable which will store the number of bytes read. If Store Number of bytes read is checked, then when the experiment is run, the number of bytes actually read will be stored in the integer variable selected here (in the example, *bytecount*). There are advantages and disadvantages to doing this, see the section Sending and Reading Data in the Devices chapter of this manual.

Copying, Pasting, Moving and Comment Out Rules

The Read command can be copied. It can be pasted or moved anywhere within its parent block. This command can be commented out.

Debugger Output

In the debug output, the description of the Read command is *Reading from device name*. There are three to four result lines. The first line (and second in the case of CEC) is the exact command sent using the C library for the communication interface. The second line (third for CEC) is the status of the command (for GPIB it is a vendor dependent number in hex, check your GPIB manufacturer, CEC or National, for what the status numbers mean). The third line (fourth for CEC) if is present, is an error line and will appear in red.

Additional Comments

This command can also be defined using the Send and Read editor.
For detailed information on how DDDA sends data see the Communicating with Instruments section in the Devices chapter of this manual.

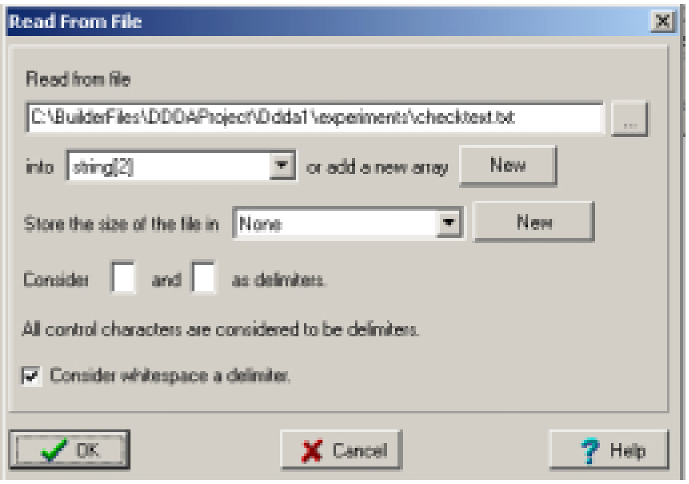
Read From File Command

Description

This variable command reads characters from an ASCII file, parses the characters and writes the whitespace delimited values to an array variable.

The Read From File Editor

The Read From File editor is shown when:



Read from file: C:\BuilderFiles\DDDAProject\Ddda1\experiments\checktext.txt

into: string[2] or add a new array

Store the size of the file in: None

Consider □ and □ as delimiters.

All control characters are considered to be delimiters.

☑ Consider whitespace a delimiter.

- The Read From File speed button is pressed in the Commands tool bar at top of the Main Form.
- The Read from file menu item is selected from the command popup

How to use the Read From File Editor:

At the top of the form, is an edit box for the filename. The filename must include the entire path. Any extension is allowed, however, this must be an ASCII file. To browse for the file, press the browse button next to the Read from file edit box.

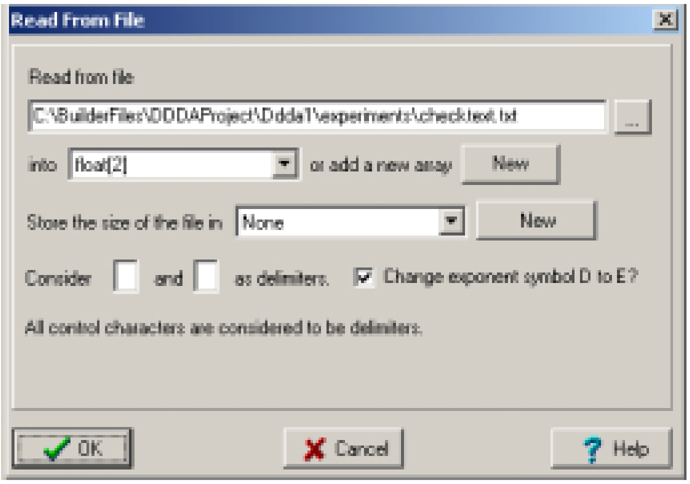
All data from the file must be read into an array. A new array can be defined by pressing the uppermost New button. If the size of the array is greater than the number of values read in, the Read From File command will terminate without error when the end of the file is reached. If the size of the array is less than the number of values read in, the Read From File command will terminate without error when the array is filled.

The number of data points read in from the file can be stored in a variable. This is defined in the Store the size of the file in combo box. Select None from the combo box if the number of data points should not be stored.

By default, all control characters are considered whitespace (they separate values). Two additional characters can be defined as whitespace by entering the character from the keyboard into the edit boxes. Only one character can be typed into each edit box.

If reading into a string array, the **Consider whitespace a delimiter** checkbox will be visible. If it is checked, strings separated by whitespace will be parsed into separate values. By default, it is checked.

Float and integer arrays assume that whitespace is a delimiter (see the section How DDDA Intreprets Data in Chapter 5).



Read From File

Read from file

C:\BuilderFiles\DDDAProject\Ddda1\experiments\checktext.txt

into float[2] or add a new array

Store the size of the file in None

Consider □ and □ as delimiters. ☑ Change exponent symbol D to E?

All control characters are considered to be delimiters.

If reading into a float array, the **Change exponent symbol D to E?** checkbox will be visible.   There are some older instruments that use the symbol D, meaning double, instead of an E (1.0D-1 instead of 1.0E-1). This is only visible when reading into a float array.

Example:
Reading in the file checktest.txt:
+123.4, -2.3E-12, -789.3,-2.3,-34.5678910

If the user defined "," as a delimiter, Read From File, would read 5 data points, all of which could be successfully written into either a string or float array of size 5 or greater.

In the experiment view, the Read From File command has the following format:

Read filename into array name SC(variable name)

if storing the number of values read or

Read filename into array name SCN

if not storing the number of values read.

Copying, Pasting, Moving and Comment Out Rules

The Read From File command can be copied. It can be pasted or moved anywhere within its parent block. This command can be commented out.

Debugger Output

In the debug output, the description of the Read From File command is *Storing Data Count in variable name* if storing the number of values read, otherwise it is *Not storing data count*. The result line is the *Number of values read/array size*. This is a ratio of the number of values written to the array over the array size.

Additional Comments

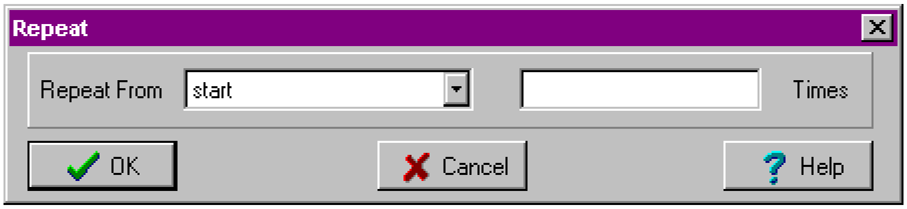
If the file Read From File is trying to read from does not exist when the experiment is run, a fatal (i.e. stops the running of the experiment) will occur.

Repeat Command

Description

This control command allows the user to repeat all the commands between a Label Command and this command a specified number of times. The repeat command follows the Loop Rules.

The Repeat Editor



Repeat

Repeat From: start

Times:

The Repeat editor is shown when:

- The Repeat speed button is pressed in the Commands tool bar at top of the Main Form.
- The Repeat menu item is selected from the command popup.

How to use the Repeat Editor

Select a label to repeat from from the drop down list (just to the right of Repeat From). Only valid Label commands will be listed. Valid Labels are those that are within the current loop (if there is one) and those that do not already have a Repeat or Repeat Until command using them. Enter a number of times to repeat in the edit box (just to the left of Times).

Copying, Pasting, Moving and Comment Out Rules

A Repeat command cannot be copied. It can moved within its parent block but not above its associated Label command. This command can be commented out. Its associated Label command will be commented out as well.

Debugger Output

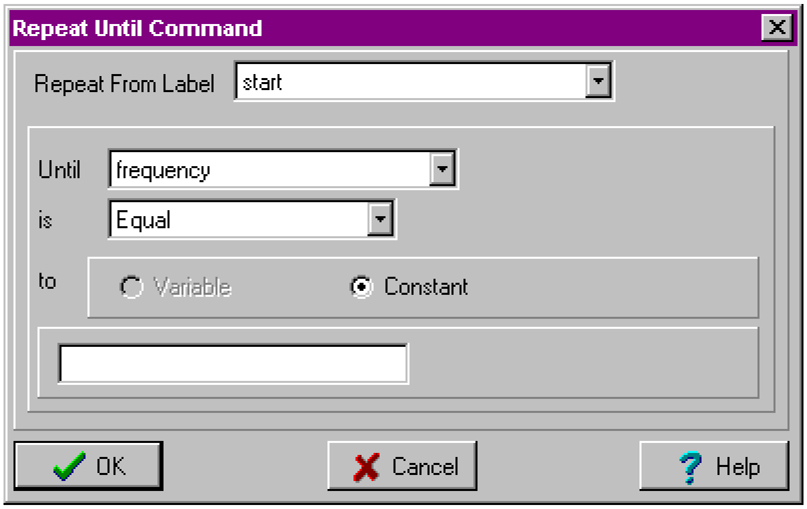
In the debug output, the description of the Repeat command is *Counter at n :Go to label* where n is the number of times the repeat command has been run and label is the next command it runs. If n is equal to the number of times it has to repeat, then the description is *Counter at n :Finished repeat.* There is no result.

Repeat Until Command

Description

This control command is a cross between the If Then Else command and the Repeat Command. All the commands between the corresponding Label and this command are repeated until the boolean expression defined evaluates to true.

The Repeat Until Editor



Repeat Until Command

Repeat From Label: start

Until: frequency

is: Equal

to: ○ Variable ● Constant

The Repeat Until editor is shown when:

- The Repeat Until speed button is pressed in the Commands tool bar at top of the Main Form.
- The Repeat Until menu item is selected from the command popup.

The Repeat Until Editor will only be shown if there is one or more variables already defined in the block

How to use the Repeat Until Editor

First, select the Label you wish to repeat from. Only valid Label commands will be listed. Valid Labels are those that are within the current loop (if there is one) and those that do not already have a Repeat or Repeat Until command using them.
The boolean expression is defined by comparing the current value of the selected variable (in the above example, frequency) to either another variable, if one exists, or a constant. In the above example, there are no other variables in the experiment so the Variable radio button is greyed out. String variables can only be compared using an equal expression as where other variables can be compared using equal, greater than, greater than or equal, less than and less than or equal (in the above example it is equal).

Copying, Pasting, Moving and Comment Out Rules

A Repeat Until command cannot be copied. It can be moved within its parent block but not above its associated Label command. This command can be commented out. Its associated Label command will be commented out as well.

Debugger Output

In the debug output, the description of the Repeat Until command is *condition evaluates to true/false* where condition is the condition defined in the Repeat Until command with the current values inserted. If the condition evaluates to true, the result is *GoTo: label* where label is the next command run, Otherwise, the result is *Repeat finished*.

Additional Comments

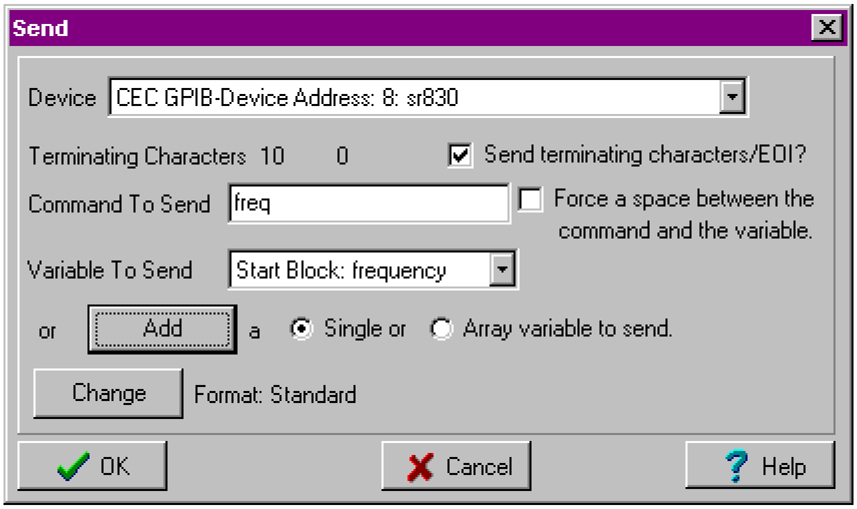
If the boolean expression never evaluates to true, an endless loop will result. Use care when creating the expression.

Send Command

Description

This instrument command is used to send data to a device. The data is sent as an ASCII string. The ASCII representation of the current value of a variable can also be sent. To send binary data see the Send Binary Command.

The Send editor



Send

Device: CEC GPIB-Device Address: 8: sr830

Terminating Characters 10 0 ☑ Send terminating characters/EOI?

Command To Send: freq □ Force a space between the command and the variable.

Variable To Send: Start Block: frequency

or Add a ● Single or ○ Array variable to send.

Change Format: Standard

The Send editor is shown when

- The Send speed button is pressed in the Commands tool bar at top of the Main Form.
- The Send menu item is selected from the command popup.
- The Commands|Send menu item is selected from the device popup.

How to use the Send Editor

In the above example, the string freq, the current value of *frequency* and a line feed will be sent when the Send command is run. If the variable is an array, an index combo box will appear next to the Variable To Send box. Because *frequency* is a float variable, the Change button and Format: label appear (these are not visible when the variable is a string or integer). Pressing the Change button, causes the Number Format dialog box to appear (see Chapter 4, BigNum output for details on the Number Format Dialog Box) where the format of the float variable can be customized. The line feed will be sent and EOI will be asserted (if defined in the device editor) because Send terminating characters/EOI? is checked. If it were not checked, the line feed would not be appended nor would EOI be asserted for this Send command ONLY. Unchecking this box is useful when sending an array of ASCII data to an instrument sending an ASCII array). This feature is not available when using the Send and Read editor.
The command that will show up in the Experiment View will be (after pressing OK):
Device Name: Send(*command string*) T/NT if none is selected in the variable combobox.
Device Name: Send(*command string variable name*) T/NT if a variable is selected in the variable combobox.

Device Name: Send(*variable name*) T/NT if the edit box for the command string is empty.
The T/NT refers to whether the selections in the device editor regarding terminating characters being appended to the end of the command string and EOI being asserted when data transmission is completed are active. If terminating characters/EOI are to be sent a T (terminator) will appear otherwise, an NT (no terminator).
In the above example the command in the Experiment View will be:
sr830: Send(freq frequency) T
By default, DDDA makes sure there are no spaces between the command to be sent and the variable value (if there is one). If the instrument requires that there be a space, check "Force a space between the command and the variable." This will force one space between the command and the variable value when the string is sent to the instrument. This has no effect if there is no variable sent or if there is no command string sent.

Sending an array of ASCII data to an instrument:

To send an ASCII array of data to an instrument you need to send each data string along with its delimiter separately, turning off terminating characters and EOI. For example, the following code would send the first 20 values in the float array floatdata to the SR785:

```
Comment: 1 is the trace and 10 is the number of complex points
Comment: A complex point consists of 2 floats
sr785: Send(tasc? 1,10) T
Comment: tasc returns a 1 if can accept all the points,  0 if not
sr785: Read Binary(check)
If check>=1
    Comment: send the first half of the first complex point
    sr785: Send(floatdata[0]) NT
    Calculate(index=1)
        Comment: send the rest of the points delimiting with a comma
        sendloop
                sr785: Send(, floatdata[index]) NT
                Increment index by 1
        Repeat from sendloop Until index==19
        Comment: send the last half of the last point with a comma and
        Comment: send the terminating characters/EOI
        sr785: Send(, floatdata[19]) T
End If
```

Copying, Pasting, Moving and Comment Out Rules

The Send command can be copied and it can be moved or pasted anywhere in the experiment. This command can be commented out.

Debugger Output

In the debug output, the description of the Send command is *Sending to device name*. There are two to three result lines. The first line is the exact command sent using the C library for the communication interface. The second line is the status of the command just sent (for GPIB it is a vendor dependent number in hex, check your GPIB manufacturer, CEC or National, for what the status numbers mean.) The third line, if it is present, is an error line and will appear in red.

Additional Comments

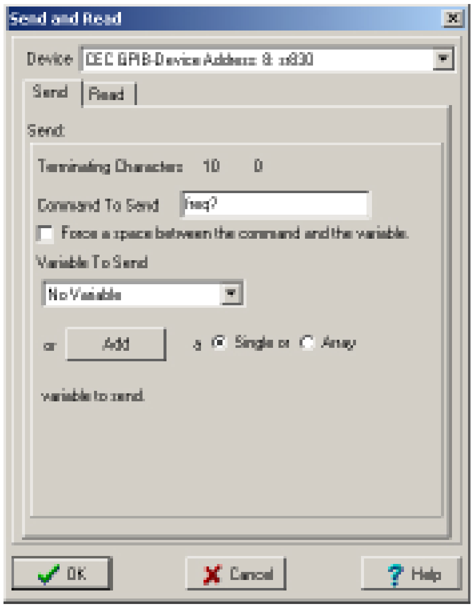
The Send command can also be defined by using the Send and Read editor.
For detailed information on how DDDA sends data see the Sending and Reading Data section of the Devices chapter in this manual.

Send and Read Editor

Description

Send and Read creates both a Send and Read command using one editor.

The Send and Read editor



Device: CEC GPIB-Device Address: 8: sr830

Send:

Terminating Characters 10 0

Command To Send freq?

□ Force a space between the command and the variable.

Variable To Send

No Variable

or Add a ● Single or ○ Array

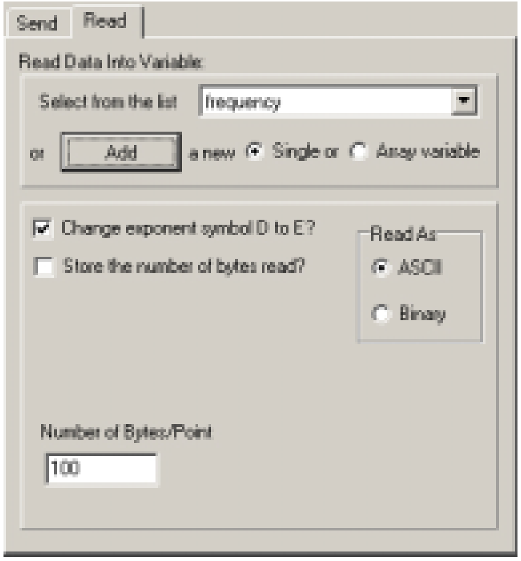
variable to send.

The Send and Read editor is shown when:

- The Send and Read speed button is pressed in the Commands tool bar at top of the Main Form.
- The Send and Read menu item is selected from the command popup.
- The Commands|Send and Read menu is selected from the device popup.

Notice that the first page is similar to the Send editor except that you cannot turn off terminating characters and EOI (terminating character will be appended and EOI will be asserted if they are defined in the Device Editor).

The second page is just like the Read editor.



Read Data Into Variable:

Select from the list frequency

or Add a new ● Single or ○ Array variable

☑ Change exponent symbol D to E?

□ Store the number of bytes read?

Read As

● ASCII

○ Binary

Number of Bytes/Point

100

Pressing OK adds both a Send and a Read Command to the Experiment View.

Additional Comments

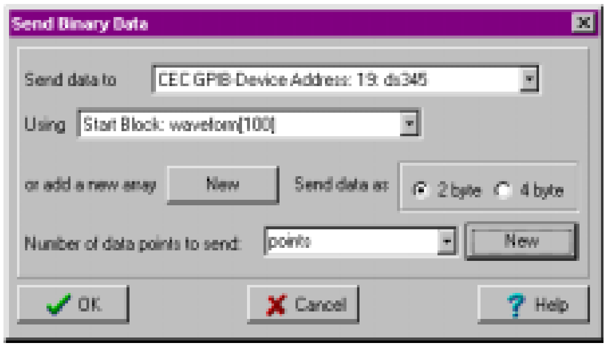
Please note that all required parameters (a command to send and a variable to read into) must be defined or an error will result when pressing OK.

Send Binary Data Command

Description

This instrument command is used to send binary data to an instrument. Only integer and float arrays can be sent using this command.

The Send Binary Data editor



Send data to: CEC GPIB-Device Address: 19: ds345

Using: Start Block: waveform[100]

or add a new array

Send data as: ● 2 byte ○ 4 byte

Number of data points to send: points

The Send Binary Data editor is shown when:

- The Send Binary Data speed button is pressed in the Commands tool bar at top of the Main Form.
- The Send Binary Data menu item is selected from the command popup.
- The Commands|Send menu item is selected from the device popup.

How to use the Send Binary Data Editor:

In the above example, the integer array *waveform* as 2 byte (16 bit) integers will be sent to the device ds345. The number of data points to send is defined in the integer variable *points*. No terminating characters are appended, however, if EOI is enabled, it will be sent upon completion of the transfer. If you send a float array, it will be sent in 4 byte IEEE floating point format. You cannot send a string array as binary.
The command that will show up in the Experiment View will be (after pressing OK):

```
ds345: SendBinary(waveform[points], 2 byte integer)
```

Copying, Pasting, Moving and Comment Out Rules

The SendBinary command can be copied and it can be moved or pasted anywhere in the experiment. This command can be commented out.

Debugger Output

In the debug output, the description of the Send Binary command is *Sending array name to device name*. There is one result line, *Sent number of points, number of bytes*.

Additional Comments

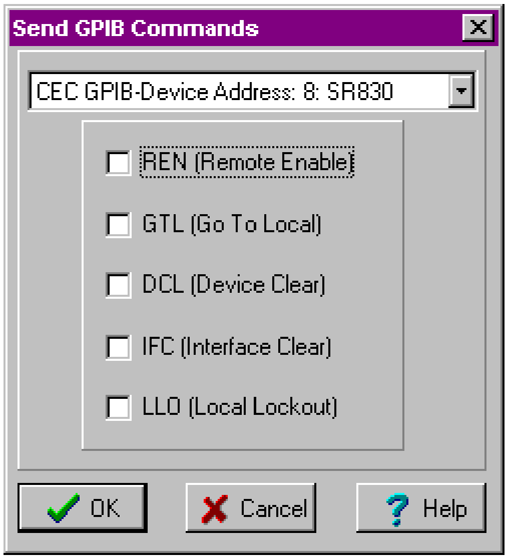
For detailed information on how DDDA sends data, see the section Sending and Reading Data in the Devices chapter of this manual.

Send GPIB Command

Description

This instrument command sends GPIB commands to an instrument. Obviously, this is only valid for instruments connected via GPIB. One or more GPIB commands can be sent at the same time.

The Send GPIB Editor



Send GPIB Commands

CEC GPIB-Device Address: 8: SR830

□ REN (Remote Enable)

□ GTL (Go To Local)

□ DCL (Device Clear)

□ IFC (Interface Clear)

□ LLO (Local Lockout)

The Send GPIB editor is shown when:

- The SendGPIB speed button is pressed in the Commands tool bar at top of the Main Form.
- The SendGPIB menu item is selected from the command popup.
- The Commands|SendGPIB menu item is selected from the device popup.

How to use the Send GPIB Editor

Check which GPIB commands you wish to send to the instrument and press OK. If no commands are selected, a error will result.

Copying, Pasting, Moving and Comment Out Rules

A Send GPIB command can be copied. It can be moved or pasted anywhere in the experiment. This command can be commented out.

Debugger Output

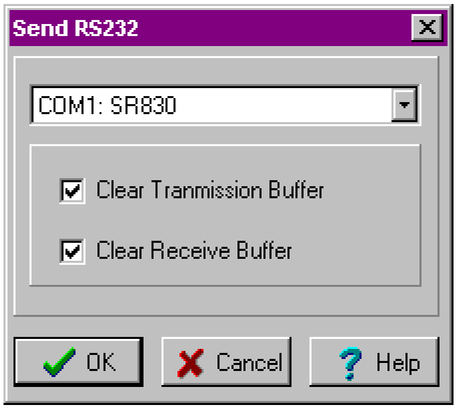
In the debug output, the description of SendGPIB command is *Sending to device name*. There are two to three result lines. The first line is the exact command sent using the C library for the communication interface. The second line is the status of the command just sent. It is a vendor dependent number in hex. Check your GPIB manufacturer, CEC or National, for what the status numbers mean. The third line, if present, is an error line and will appear in red.

Send RS232 Command

Description

This instrument command allows for clearing of the transmit and receive buffers of the RS232 line. Obviously this is only valid for instruments connected via RS232. Both buffers can be cleared using one Send RS232 command.

The Send RS232 Editor



COM1: SR830

☑ Clear Tranmission Buffer

☑ Clear Receive Buffer

The Send RS232 editor is shown when:

- The SendRS232 speed button is pressed in the Commands tool bar at top of the Main Form.
- The SendRS232 menu item is selected from the command popup.
- The Commands|SendRS232 menu item is selected from the device popup.

How to use the Send RS232 editor

Check the buffers you wish to clear and press Ok.

Copying, Pasting, Moving and Comment Out Rules

The Send RS232 can be copied. It can be moved or pasted anywhere in the experiment. This command can be commented out.

Debugger Output

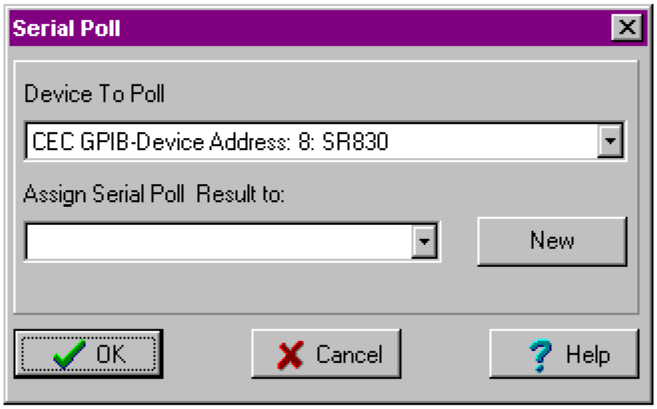
In the debug output, the description of SendRS232 command is *Clearing buffer name*. There is no result.

Serial Poll Command

Description

This instrument command is used to Serial Poll a device on a GPIB bus.

The Serial Poll Editor



Device To Poll: CEC GPIB-Device Address: 8: SR830

Assign Serial Poll Result to:

The Serial Poll editor is shown when:

- The Serial Poll ☎ speed button is pressed in the Commands tool bar at top of the Main Form.
- The Serial Poll menu item is selected from the command popup (right click on a command in the Experiment View)
- The Commands|Serial Poll menu item is selected from the device popup (right click on a device in the Experiment Devices list)

How to use the Serial Poll Editor

Select one of the GPIB devices from the drop down box. Select or add a single integer variable to hold the result in the drop down box just below the Assign Serial Poll Result to label.. In order to check individual bits in the result, use the Calculate command and cast the integer variable that holds the result to a binary.

Copying, Pasting, Moving and Comment Out Rules

The Serial Poll command can be copied. It can only be moved or pasted within the block it was created in. This command can be commented out.

Debugger Output

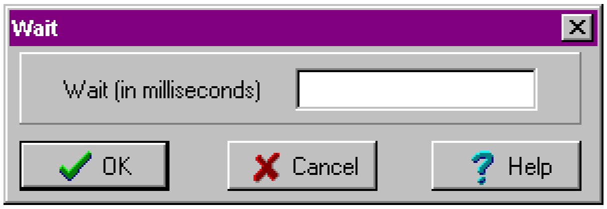
In the debug output, the description of SerialPoll command is *Serial polling: device name*. There are one to two result lines. The first line is *Poll result: hex number*. To interpret the hex number, see the manual for the device. The second line, if it is present, is *Read Command timed out.* Since this is an error, it is in red.

Wait Command (Changed in Version 1.5)

Description

This general command suspends the execution of the program for a specified number of milliseconds

The Wait Editor



The Wait editor is shown when:

- The Wait ⌛ speed button is pressed in the Commands tool bar at top of the DDDA Main View.
- The Wait menu item is selected from the command popup.

How to use the Wait Editor

Enter the amount of time you wish to suspend the execution of the program in the edit box (just to the right of the Wait (in milliseconds)

Copying, Pasting, Moving and Comment Out Rules

A Wait command can be copied. It can be moved or pasted anywhere in the experiment. This command can be commented out.

Debugger Output

In the debug output, the description of Wait is *Slept x ms* where x is the number of milliseconds the program waits before going to the next step. There is no result.

Additional Comments

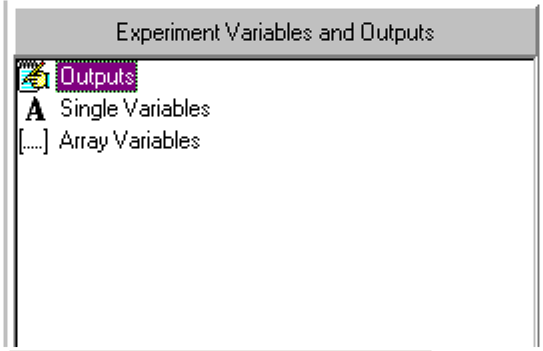
This command should NOT be used as a wait to trigger devices. The time that actually passes may be greater than the time defined.

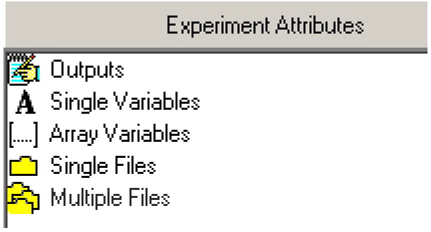
(New in Version 1.5) If the time is greater than 500ms, the Wait command will cycle through the time in increments of 500ms. This prevents you from having to wait for the Wait command to finish before aborting a running experiment.

Chapter 4 Experiment Attributes

Introduction

Variables are used to store one or more user generated values. Outputs provide a graphical representation of the values of these variables. Experiment Files store values of the variables directly in an ASCII text file. The Experiment Attributes section of the DDDA Main form maintains a list of all the variables, outputs, and experiment files used in a particular experiment:





This chapter describes what attribute types are available in DDDA and how they can be created, deleted and edited.

Experiment Variables

Introduction

There are two types of experiment variables: single variables which hold a single value and array variables which hold more than one value. Experiment Variables are owned by a particular experiment block and they are used by the commands in a block. Variables can be read in from a device or ASCII file, sent to a device, calculated and incremented. They are also used in Repeat, Repeat Until, If Then and Decision commands to define a boolean expression. The value of a variable can be sent to an output via a plot command and to an Experiment File using the Output To File Command. The current value of any variable, single or array, can be accessed at any time by selecting the Show Current Value menu item in the variable popup menu (this can be accessed by right clicking on any single variable or array variable in the Experiment Attributes list). The current value of the variable is then shown in a list box just below Experiment Attributes. Variables can also be made Read Only (inaccessible to Read, Increment, Serial Poll, or Calculate commands, their initial value is never changed) by checking the Read Only check box in the Variable Editor (shown any time a variable is added or edited).

Creating, Editing and Deleting Variables

A single variable can be added to an experiment by either double clicking on the Single Variable category in the Experiment Attributes list or by right clicking on the Single Variable category in the Experiment

Attributes list and selecting Add or Add Many from the popup menu. An array variable can be added to an experiment by either double clicking on the Array Variable category in the Experiment Attributes list or by right clicking on the Array Variable category in the Experiment Attributes list and selecting Add or Add Many from the popup menu. A variable can be edited by either double clicking on the variable in the Experiment Attributes list or by right clicking on the variable in the Experiment Attributes list and selecting Edit from the popup menu. Use caution when editing a variable. Any command that depends on the variable type will be deleted if the type is no longer compatible. For example, a variable called *counter* is created as an integer. Subsequently, an increment command is created that uses it. If *counter* is changed to a string, the increment command will be deleted since you cannot increment a string. A variable can be deleted by either selecting the variable in the Experiment Attributes list, keeping the mouse over the variable and pressing delete, or by right clicking on the variable in the Experiment Attributes list and selecting Delete from the popup menu. Remember, deleting a variable deletes all the commands that use it. The one exception may be the Output To File Command. If more than one variable is being output, deleting one of the variables in the list will NOT delete the command. Deleting all the variables in the command will remove it.

Variable Types

A variable can store data in one of three formats:

String: ASCII representation of the data. This cannot be calculated nor can it be used in a calculation. It can only be plotted to Text and BigNum outputs.

Integer: An integer representation of the data. This can be calculated, used in a calculation and plotted by any output.

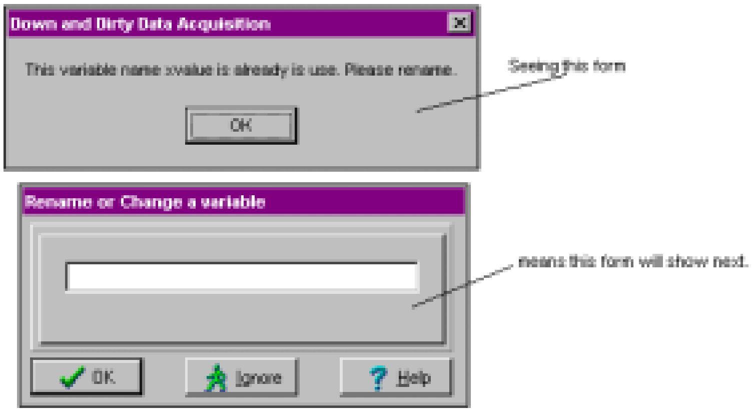
Float: The float representation of the data. This can be calculated, used in a calculation and plotted by any output.

Care should be taken when using one of the latter data formats, integer and float. If the data is read in, make sure only valid characters are stored in the variable. Valid characters are 0-9 and e/E (scientific notation) for integers and floats. A decimal point is also valid for float values. If invalid characters are present, a "Read" error will occur. Make sure the character to terminate read is set correctly for the device (See the Device Editor section in Chapter 5 for more details).

Variables saved/read in an Experiment Block

When Saving an Experiment Block, all variables that are read or written to are saved in the *.blk file whether they are owned by the block or not.

When a block is read into the experiment and it has a variable with the same name as a variable already in the experiment, the following will be shown:



This variable name xvalue is already is use. Please rename.

Seeing this form

means this form will show next.

The form requires that the variable be renamed or in some cases, it can be reassigned to a variable in another block (variables that are read). In this case, the Ignore button will be disabled.

Outputs

Introduction

Each Plot Command sends the current value of up to two variables to an output. An output is a separate window, owned by the experiment, that can be shared by more than one experiment block. More than one plot can send variables to a single output.

Creating, Editing and Deleting Outputs

An output can be added to an experiment by either double clicking on the Outputs category in the Experiment Attributes list or by right clicking on the Outputs category in the Experiment Attributes list and selecting Add or Add Many from the popup menu. A output can be edited by either double clicking on the output in the Experiment Attributes list or by right clicking on the output in the Experiment Attributes list and selecting Edit from the popup menu. Use caution when changing the type of an output. For example if the type of a Text output is changed to that of a Table, all plot commands that sent string variables to that Text output will be deleted. An output can be deleted by either selecting the output in the Experiment Attributes list, keeping the mouse over the output and pressing delete, or by right clicking on the output in the Experiment Attributes list and selecting Delete from the popup menu. Remember, deleting an output deletes all the plot commands that use it.

Output States

There are two states to any output: Active and Inactive. An active output takes data from a plot command and can show imported data. An inactive plot shows previously collected data or imported data, it cannot take data from a plot command. An active output can be made inactive by selecting offline from the Plot menu. An new active output will then replace that output in the output list. Once an output is made inactive, it cannot be made active again.

Reading in Saved Outputs

To read in saved output files (they are, by default, inactive), select File|Open|Output from the DDDA Main Menu and select the output file (*.opt) you wish to open. If an inactive output is hidden, you can show it by selecting Outputs|Opened Outputs from the DDDA Main Menu, select the output you wish to show and select Show from the menu. You can also remove the outputs from DDDA by selecting Delete in that same menu (this does NOT delete them from your disk). Bring To Front brings the inactive output in front of all other forms. The menu Outputs|Opened Outputs is disabled (grayed out) if there are no inactive output files opened.

Output Types

There are four types of outputs:

Text: A memo field shows the current value of any type of variable in ASCII form. The results of each plot command are separated by new lines.

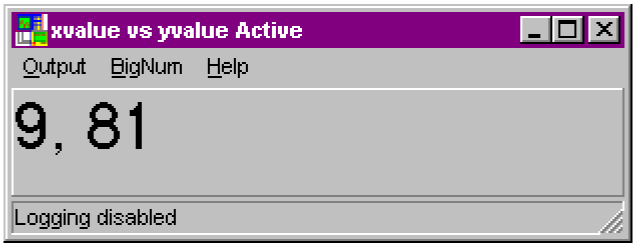
BigNum: Like text, this shows the current value of any type of variable in ASCII form. In this case, however, the value is overwritten with each plot command.

Table: This output can only plot float and integer variables. Each plot command feeds values into two columns. Columns can be hidden/shown by the user.

Graph: Like the table, this output can only plot float and integer variables. Each plot command is a separate trace on the graph.

Each of these types is described in detail in the following sections.

BigNum Output



9, 81

Logging disabled

Description

The BigNum output shows the current value of a variable in ASCII form. In this case, however, the value is overwritten with each plot command. Plot commands only need to send one variable to the BigNum output (None will be enabled in the Plot editor). Data cannot be imported or exported from the output nor can it transfer data to any other output.

Saving Data

Inactive Output

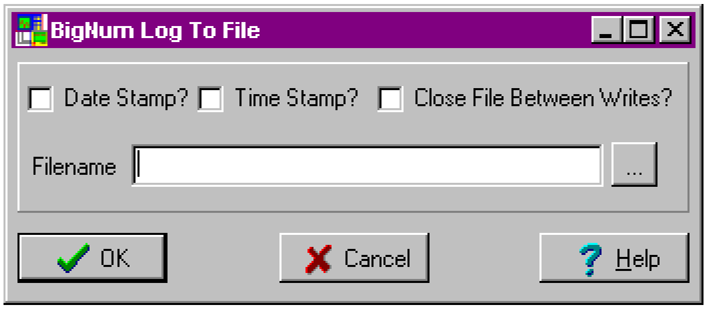
The BigNum output can be saved as an inactive output by selecting Output|Save or Output|Save As from the output's main menu.

Copy To Clipboard

The data showing in the output can be copied to the clipboard as text by selecting Output|Copy to Clipboard from the output's main menu.

Logging To File
BigNum has one advantage over the other outputs, namely, when the output is active, data can be directly written to file while the experiment is running. If this feature is enabled, every data point displayed on the BigNum output (and if selected, the date and time as well) is written as a separate line in a user named text file. When logging is enabled, the status bar on the output will say "Logging to file:" with the path and name of the log file.

To enable logging, select Bignum|Log Data to File|Enable Logging from the output's main menu. The following form will show:



□ Date Stamp? □ Time Stamp? □ Close File Between Writes?

Filename

If Date Stamp is checked, then when a data point is written to file, the date, in the format month/day/year will be written on the line preceding the data point. The date and the data point will be separated by a comma.

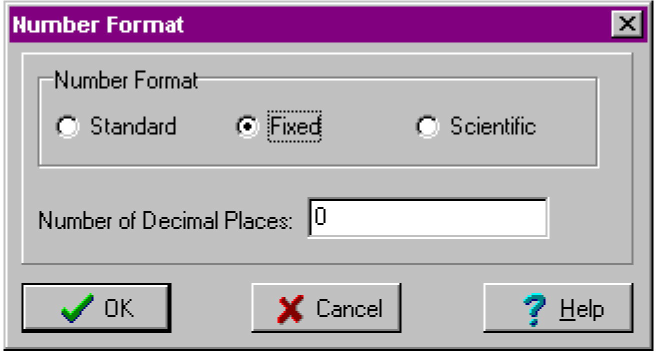
If Time Stamp is checked, then when a data point is written to file, the time, in the format hour:minute:second:millisecond will be written on the line preceding the data point. The time and the data point will be separated by a comma.
If Close File Between Writes is checked then the file will be closed after each data point is written. If this is not checked, then the file will be opened when the first data point is written and closed when the experiment finishes (or aborts, if there is an error). There are advantages and disadvantages to closing the file between writes. The disadvantage is that it takes longer since the file has to be opened and closed with each write. The advantage is that if the computer freezes or the power goes out, the data up to that point is saved in the file.

The filename can be typed into the filename box or can be browsed for using the ... Browse button. If the path for the file does not exist, an error will show when the OK button is pressed.
These options can be modified by selecting BigNum|Log Data To File|Edit Log File Parameters (this menu item is disabled when logging is turned off).

Warning: Do NOT use the same filename for BigNum logging as you do for an Experiment File in the same experiment. This can cause a serious conflict which can in program termination.

Appearance Options

Number Format: The format of the numbers that show in the BigNum output can be changed by selecting BigNum|Number Format in the main menu of the output:



Number Format

Number Format
○ Standard ● Fixed ○ Scientific

Number of Decimal Places: 0

As shown in the form, there are three types of formatting available:

Standard: Floating point numbers are given as many decimal places as the computer chooses to give them. Integer numbers have no decimal places (the editbox next to the Number of Decimal Places label as well as the label are not visible when Standard is selected).
Fixed: Floating point numbers are given as many decimal places as the user specifies in the edit box next to the label Number of Decimal Places.
Scientific: Float point numbers are represented in scientific notation, that is one digit to the left of the decimal place, as many digits to the right of the decimal places as the user specifies in the edit box next to the label Number of Decimal Places followed by the base 10 exponent.

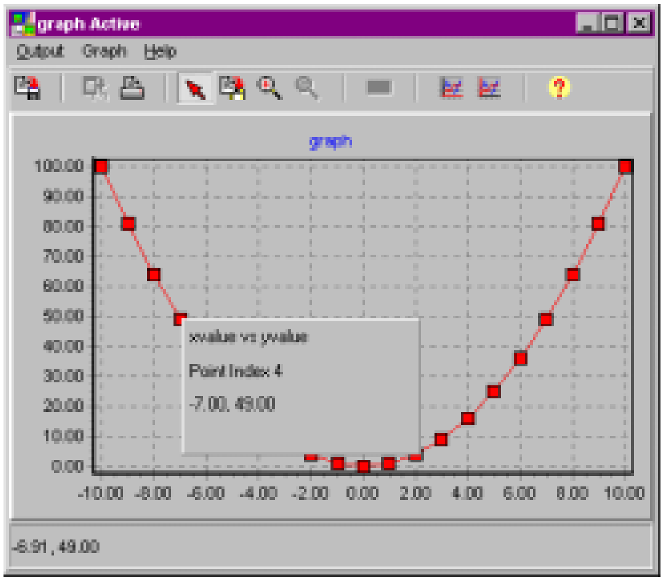
For example, the number 10.00 would be displayed as:
Standard: maybe 10.00, maybe 10, maybe 9.9999999, depends on the computer
Fixed (1 decimal place): 10.0
Scientific (1 decimal place): 1.0E+1
The number format changes when the experiment is run and new data is plotted.
Font: The font can be changed by selecting BigNum|Font from the output's main menu. This shows a font dialog box where fonts valid for your computer can be selected.
Title: The title of the output (in both the output form and in the Experiment Attributes list) can be changed by selecting BigNum|Title from the output's main menu. A form appears which has an edit box where you can enter a new title. The change occurs immediately after the Ok button in the form is pressed.

Additional Features

Clearing the current data can be done by selecting Output|Clear from the output's main menu.

Offline: An active output can be taken offline (remember this is permanent, once an output is taken offline, it cannot be made active again) by selecting Output|Offline from the output's main menu.

Graph Output



xvalue vs yvalue

Point Index 4

-7.00, 49.00

-6.91 , 49.00

Description

The Graph output is like the table in that it can only plot float and integer variables. The number of traces on the graph reflects the number of channels in the Output. Plot commands need to send either two variables or a counter and a variable to the graph output (None will be NOT be enabled in the Plot editor). The data in a graph output can be copied directly into a Table output, a Graph output, or a Text Output. A Graph Output can accept data from a Graph or another Table Output.

There can be two types of series in an active graph, active and inactive (inactive graphs have only inactive series). Active series are ones that get their data from a plot command. Inactive series are ones that have data that been imported or pasted into the graph. Active series can only be deleted by deleting the source plot command. Inactive series can be deleted by the user at any time (see the section on Appearance Options, Plot Appearance).

Saving Data

Inactive Output

The graph can be saved as an inactive output by selecting Output|Save or Output|Save As from the output's main menu.

ASCII Data

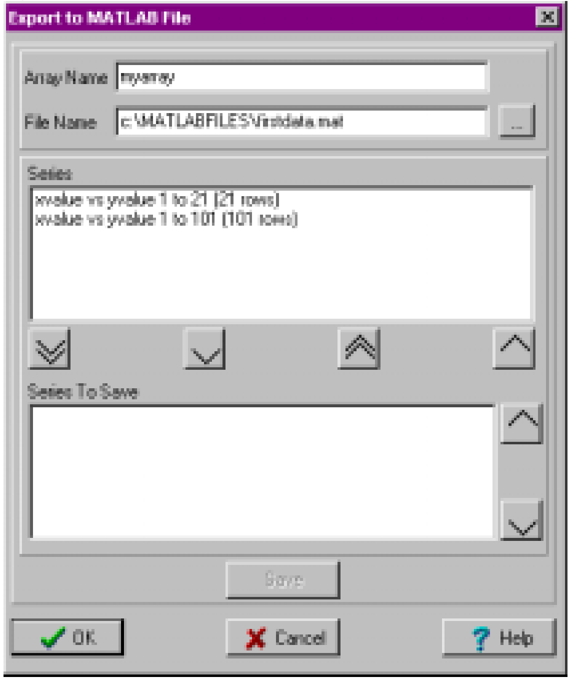
The data in the graph can be saved as comma delimited text, with values in the same row delimited by commas, by selecting Output|SaveAs Type|Comma from the output's main menu. The data can also be saved as tab delimited text, with values in the same row delimited by tabs, by selecting Output|Save As Type|Tab from the output's main menu.

Graphic

The graph itself can be saved as either a bitmap, a Windows metafile or as an enhanced metafile by selecting Output|Save As Type|Graphic from the output's main menu.

MATLAB® File

Selected data in the graph can be saved as a MATLAB® version 5.3 array by selecting Output|Save As Type|MATLAB from the output's main menu. This shows the Export MATLAB Data Form:



Array Name: myarray

File Name: c:\MATLABFILES\firstdata.mat

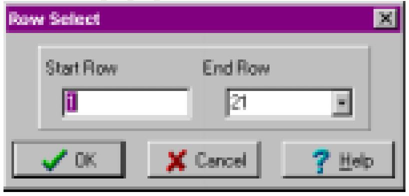
xvalue vs yvalue 1 to 21 (21 rows)
xvalue vs yvalue 1 to 101 (101 rows)

The name of the array should be entered in Array Name (in the above example, *myarray*) and the filename can be entered into File Name or can be browsed for by pressing the Browse button ... (in the above example the file is *c:\MATLABFILES\firstdata.mat*). If the file already exists, the array will be added to the file. If the array already exists in the file, the array will be overwritten.

When there is one or more series in the Series To Save box, the Save button will be enabled. Pressing Save, saves the matix at the named array in the named file.

All of the series that can be added will be listed in the Series box. All of the series selected to be added to the named array will be listed in the Series To Save box. Series can be moved from one box to other by pressing the arrow buttons that are between the boxes. Please remember that MATLAB arrays are rectangular matricies. All of the series MUST have the same number of rows. Therefore, when attempting to move a series to the Series To Save box, it must have the same number of rows as the series that are already in the box. In the above example, both of these series cannot be added as is to the Series To Save box because they differ in the number of rows.

The first series added to the Series To Save box determines the row count for the array. All series subsequently added must have the same number of rows. Changing the row count for a series before adding to it the Series To Save Box can be done by right clicking on the series and selecting Edit Points from the popup menu. The Row Select form will show:



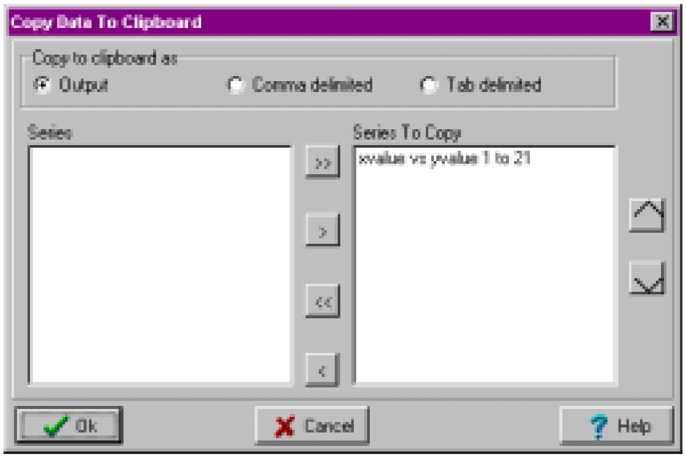
Start Row: 1

End Row: 21

Changing the starting row for a series already in the Series To Save box can be done by right clicking on the series and selecting Edit Points from the popup menu. The Row Select form will show. If this is not the only series in the box, the End Row box will be disabled in the Row Select form. The end row will be the selected starting row plus the row count for the array.

Copy Data to Clipboard

The data in the graph can be copied to the clipboard as text by selecting the Output|CopyToClipboard|Text menu item in the output's main menu or by selecting the Copy Button in a the graph output's speed bar. This shows the Copy Data Form:



Copy to clipboard as: ● Output ○ Comma delimited ○ Tab delimited

Series To Copy: xvalue vs yvalue 1 to 21

The Copy Data View copies selected series to the clipboard as a text. Data sets from the Series are added to the Series To Copy by either double clicking on the dataset or selecting one or more data sets and pressing the > button. By right clicking on a data set in the Series To Copy, the Copy Row Select form appears which allow the user to choose which rows to copy. Please note, the row numbers are NOT the x axis/xcolumn values, they are the index of the point in the output.   Pressing the up and down keys when one of the data sets in the Series to Copy box is selected, moves that data set up or down, respectively, in the list. The datasets are copied to the clipboard in order from the top data set to the bottom data set. There are three formats to choose from:

Output: This is text format with additional information that allows it to be directly imported into another output (pressing paste in another output does not result in the Import Text form being shown, described in the next section on Importing data). The data is just imported).

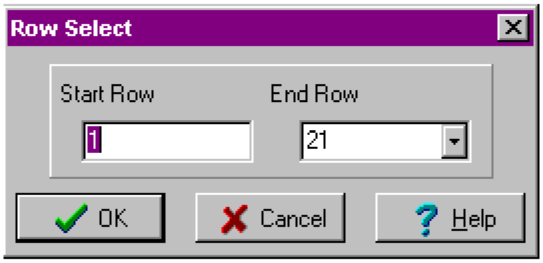
Comma: Data in the same row is delimited by commas.

Tab: Data in the same row is delimited by tabs.

This form is only shown if there is data in the output. If there is no data a warning message is shown instead.

Please note, only 64K of data can be placed on the clipboard at one time. If more than that is selected, a warning message will show after OK is pressed and nothing will be copied.

To change the rows to be copied to the clipboard, right click on the series in the Series To Copy box and select Edit Points from the popup menu. The Row Select form will show:



Start Row: 1

End Row: 21

Copy Graph to Clipboard

There are three ways the graph picture can be copied to the clipboard:

- As a bitmap, by selecting Output|Copy To Clipboard|Bitmap from the output's main menu.
- As a windows metafile by selecting Output|Copy To Clipboard|Metafile Window from the output's main menu
- As an Enhanced Metafile by selecting Output|Copy To Clipboard|Metafile Enhanced from the output's main menu

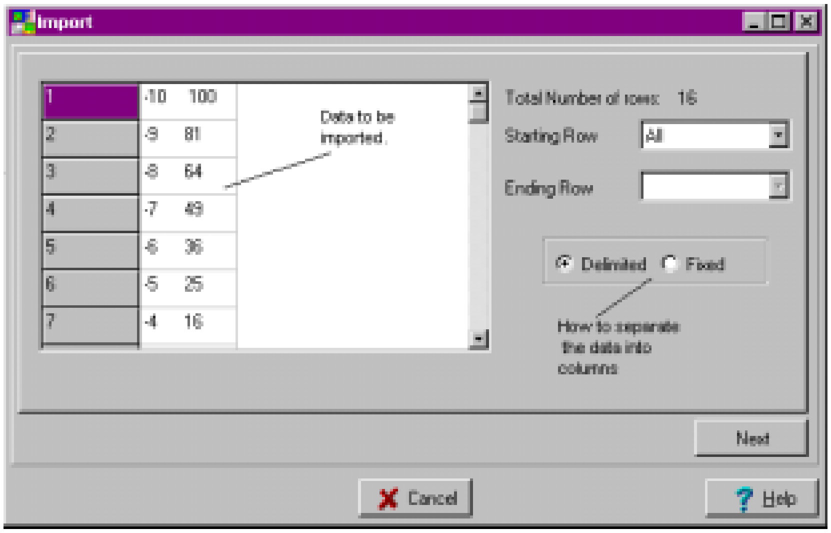
In all cases, these graphics can then be pasted into another windows program.

Importing Data into a Graph

Both an active an inactive graphs can accept data from other outputs, ASCII text, the clipboard or from a MATLAB® array.

Importing Text

ASCII data can be imported from a file by selecting Output|Import|Text from the output's main menu. ASCII data in the clipboard (if the values are floats or integers) can be pasted into the graph by selecting the Paste speed button in the graph speed bar. If a file is opened, or data pasted, the following form is shown:



Import

1	-10 100
2	-9 81
3	-8 64
4	-7 49
5	-6 36
6	-5 25
7	-4 16

Total Number of rows: 16

Starting Row: All

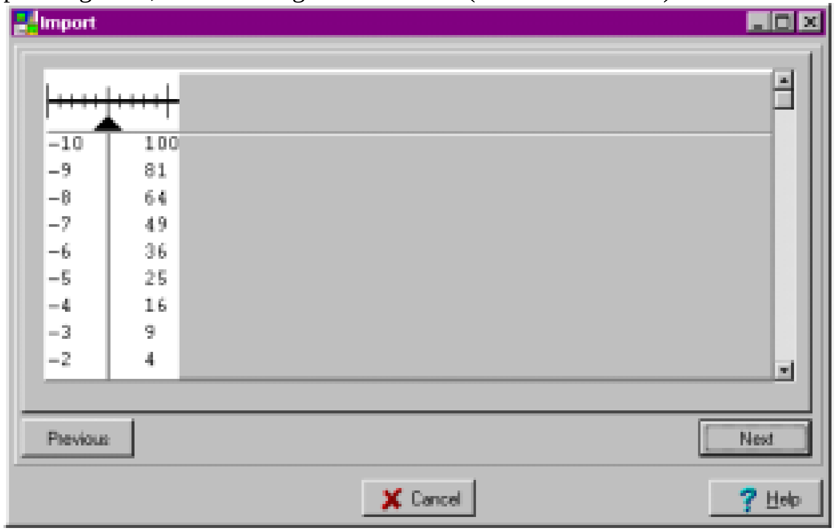
Ending Row:

● Delimited ○ Fixed

Start Row refers to the first value in the column that will be imported. The first option in the Start Row combo box is All. In that case, all the data from the column is used. If All is selected, then the End Row combo box is not enabled.

End Row refers to the last row that will be imported. The first option in End Row is End. In that case, the end row will be the last row in the table.

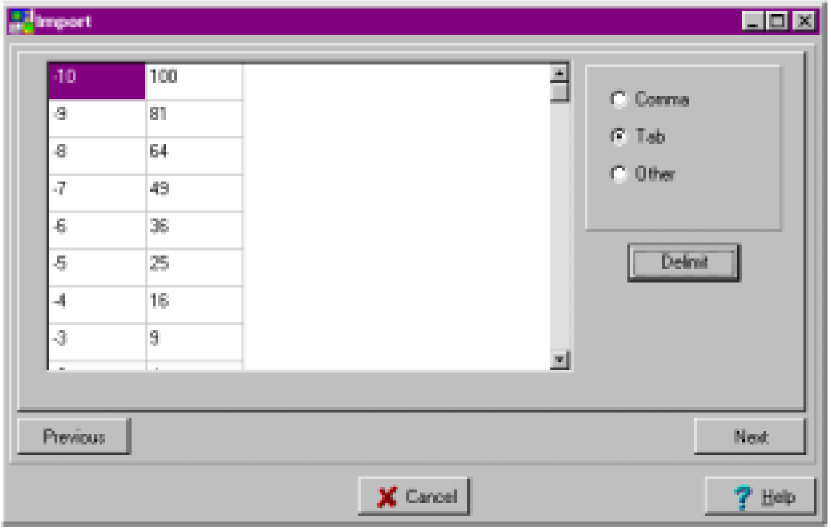
There are 2 ways to separate the rows into columns, using a delimiter (like a comma or a tab) or by defining a length for each column. In the example above, the latter is the case. By selecting Fixed and then pressing Next, the following will be shown (without the arrow):



```
-10   100
-9    81
-8    64
-7    49
-6    36
-5    25
-4    16
-3    9
-2    4
```

Left clicking on the space between the columns shows an arrow which defines where the columns will be. The arrow can be dragged (left click and drag) and removed (double click anywhere on the arrow).

If instead Delimited is checked on the first page of the Import view, with, of course a file where a delimiter was used, the following is shown:

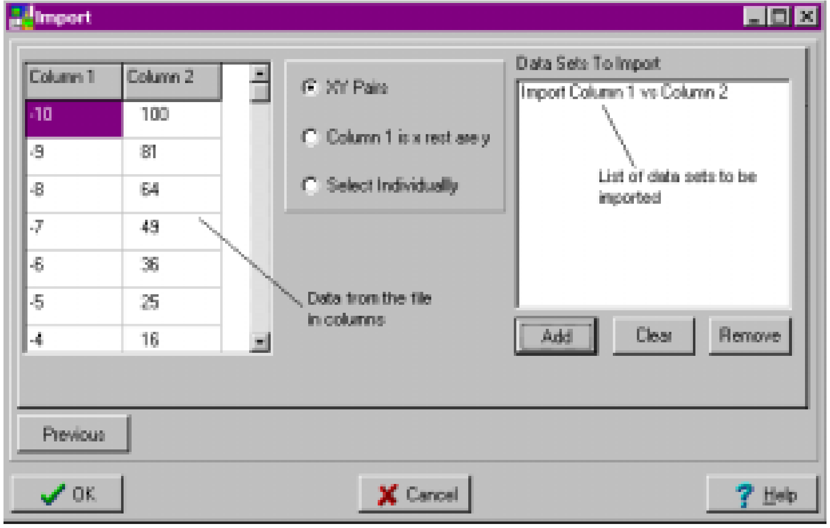


-10	100
-9	81
-8	64
-7	49
-6	36
-5	25
-4	16
-3	9

○ Comma
● Tab
○ Other

Selecting the delimiter character (if **Other** is choose, a text box appears where a single character can be entered) and pressing **Delimit** shows the data in its columns. **Warning**, if you don't press the **Delimit** button, the data will not be separated into columns.

In either case, pressing **Next** will show:



Column 1	Column 2
-10	100
-9	81
-8	64
-7	49
-6	36
-5	25
-4	16

● XY Pairs
○ Column 1 is x rest are y
○ Select Individually

Data Sets To Import
Import Column 1 vs Column 2

List of data sets to be imported
Data from the file in columns
Add
Clear
Remove
Previous
OK
Cancel
Help

Add Button: Adds the selected data sets to the Data Sets To Import box. What is added depends on what is selected in the radio box:

XY Pairs: All the data will be imported where x values are in the odd numbered rows and the y values are in the even numbered rows. This selection is only enabled when there are an even number of columns.

Column 1 is x rest are y: All the data will be imported where Column 1 will contain the x values for all data sets and the rest of the columns will contain y values.

Select Individually : X columns and Y columns are selected individually. A column selection box will appear when this radio button is selected. Each data set has to be added individually (press the Add Button after each x and y column selection).

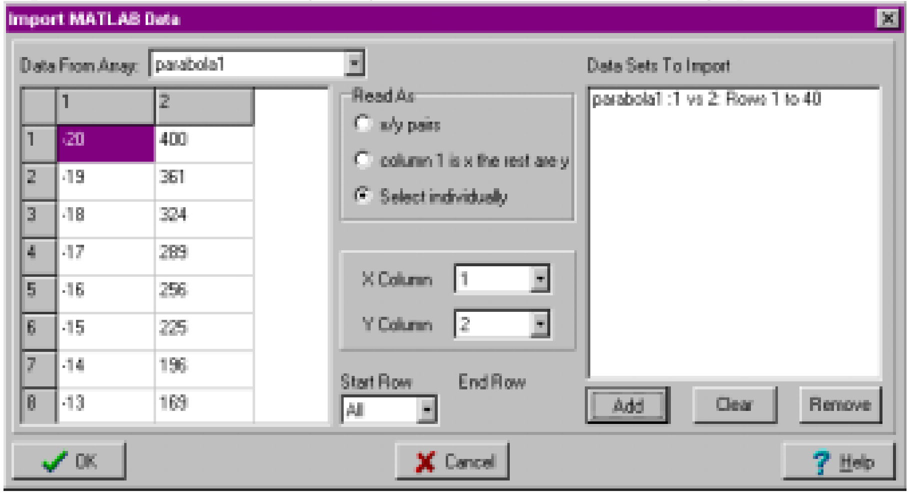
Clear Button: Clears all the data sets in the Data Sets To Import list box.

Remove Button: Removes the selected data set in the Data Sets To Import list box. If no data set is selected, pressing the Remove button does nothing.

OK Button: Imports all the data sets in the Data Sets To Import list box into the output. If no data sets are selected, a warning message is shown that no data sets will be imported.

Importing a MATLAB® file

A MATLAB array can be imported into a graph by selecting Output|Import|MATLAB File from the graph output's main menu. If a file is choose, the Import MATLAB Data form is shown where the user can import selected data from any array in the MATLAB file into the graph:



Data From Array: parabola1

	1	2
1	-20	400
2	-19	361
3	-18	324
4	-17	289
5	-16	256
6	-15	225
7	-14	196
8	-13	169

Read As
○ x/y pairs
○ column 1 is x the rest are y
● Select individually

X Column 1
Y Column 2

Start Row All
End Row

Data Sets To Import
parabola1 :1 vs 2 Rows 1 to 40

Read As: There are three ways to read in the data from a MATLAB array:

x/ypairs: Read in all the data in the array as x and y pairs, x values being in the odd numbered columns, y values being in the even numbered columns. In this case, the XColumn/Y Column panel is not shown. Pressing the Add button adds all the pairs to the Data Sets To Import list.

column 1 is x the rest are y: Read in all the data assuming the first column holds the x values and the rest of the columns are y values. In this case, the X Column/Y Column panel is not shown. Pressing the Add button adds all the data as pairs (column 1 being x, other columns being y) to the Data Sets To Import List.

Select individually: Select each X and Y pair individually. In this case, the X Column/Y Column panel is shown. Pressing the Add button adds just that XColumn and YColumn pair to the Data Sets To Import list.

Start Row refers to the first value in the column that will be imported. The first option in the Start Row combo box is All. In that case, all the data from the column is used. If All is selected, then the End Row combo box is not shown.

End Row refers to the last row that will be imported. In the example shown above, the Start Row is *4* and End Row is *25*, therefore rows 4 to 25 will be imported. The first option in End Row is End. In that case, the end row will be the last row in the table.

Add Button adds the selected data to the Data Sets To Import list.

Clear Button clears all the data in the Data Sets To Import list.

Remove Button removes the selected Data Set in the Data Sets To Import list. Pressing this button has no effect if nothing is selected.

OK Button imports all the data in the Data Sets to Import list into the output. If there is no data in the list, you are warned that there is no data in the list and are advised to either select data or press Cancel.

Note: For memory reasons, all of the data in the selected array may or may not appear in the grid below Data From Array. All of the selected data, however, WILL be imported into the output regardless of what shows in the grid.

Printing the Graph

Selecting Output|Print from the output's main menu or pressing the Print 🖶 speed button in the graph output's speed bar (at the top of the graph) sends the graph to a user specified printer.

Appearance Options

Number Format

The format of the numbers that will show in the Graph output can be changed by selecting Graph|Number Format in the main menu of the output. The same form that shows in BigNum, Number Format section shows in this case. The only difference is the number format changes immediately upon acceptance of the new format.

Graph Title and Axes Labels

The title that shows at the top of the graph and the axes labels can be modified by selecting the Graph|Graph Titles menu item in the main menu of the graph output. Selecting this item: shows the Graph Titles view. There are three sections:

The Graph Title:

Graph Title**:** The title of the graph can have multiple lines, the first line being used to identify the graph in the Output Section of the Main DDDA form.

Alignment: This is how the title will be aligned in the page at the top of the graph.

Font Button: Changes the font used to display the graph title.

and two axes sections. Each axis section consists of:

Axes Title Text: The title for the axes. It can have only one line.

Orientation: The allowed angles for displaying the axes title.
Font Button: Changes the font used to display the axes title.

Graph Range and Scaling

The Graph Range and Scaling can be changed by selecting Graph|Graph Attributes from the graph output main menu. Selecting this item shows the Graph Attributes view. There are 2 sections, one for the x axis and one for the y axis. Each section has 4 parameters:

Automatic Minimum: If this is checked, the graph determines the minimum axis value based on the data. If Automatic Minimum is not checked, a text box appears to the right of the check box where the minimum value for the graph can be entered. If there is no data in the graph, this value should not be greater than 0 if Automatic Maximum is checked. No such limit exists if there is data in the graph.

Automatic Maximum: If this is checked, the graph determines the maximum axis value based on the data. If Automatic Maximum is not checked, a text box appears to the right of the check box where the maximum value for the graph can be entered. If there is no data in the graph, this value should not be less than 0 if Automatic Minimum is checked. No such limit exists if there is data in the graph.

Automatic Increment: If this is checked, the graph determines the minimum step between axes labels. If Automatic Increment is not checked, a text box appears to the right of the check box where the increment can be entered. The increment must be a positive value greater than 1.0E-12.

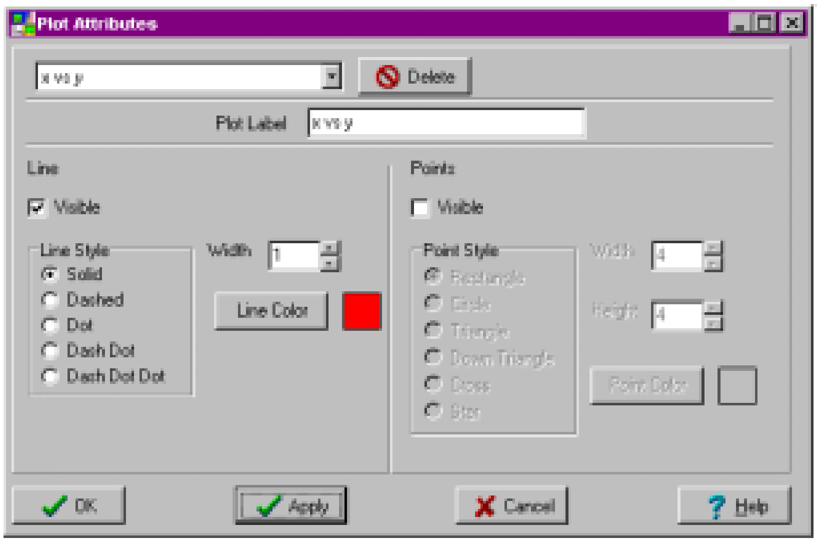
Log Scale?: If this is checked, the axis is converted to log scale. If any part of the axis is less than 0, this does nothing.

There are restrictions on the range of the graph. The total range (maximum-minimum) cannot be less than 1.0E-12. In addition, the maximum must be greater than or equal to the minimum value (in other words, no reverse axes).

If both Automatic Minimum and Automatic Maximum are checked, all data is shown. During the run of an experiment, the graph will resize itself to accommodate new data. This resizing does slow graphing down a little bit.

Plot Appearance

The line color and style and the point color and style can be modified for each trace (or plot) in the graph. Selecting Graph|Plot Attributes from the graph output's main menu shows the Plot Attributes view:



x vs y

Plot Label: x vs y

Line

☑ Visible

Line Style: ● Solid ○ Dashed ○ Dot ○ Dash Dot ○ Dash Dot Dot

Width: 1

Points

☐ Visible

Point Style: ● Rectangle ○ Circle ○ Triangle ○ Down Triangle ○ Cross ○ Star

Width: 4

Height: 4

The series to be modified can be selected from the combo box at the top of the form. If the series has been imported, i.e. it is not an active series, the Delete button is enabled (in order to delete an active series, the Plot command that creates it must be deleted) To hide a series in the graph, uncheck Visible under Line and uncheck Visible under Points.

Apply Button: Applies the changes to the series made in the Plot Attributes view.
OK Button: Applies the changes to the series made in the Plot Attributes view and hides the view.

Additional Features

Finding data points in the graph:

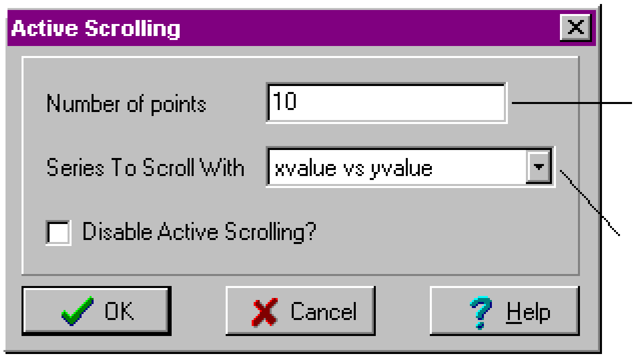
Mousing on a data point (easiest to do if points are showing, see Plot Appearance) shows the x and y values the point index and the series name of the data point in a box. The box shows for about 2-3 seconds and then disappears.

Graph Status Bar

The status bar at the bottom of the graph shows the x and y coordinates of the mouse. This shows all the time, whether or not the mouse if over a point.

Active Scrolling

While and experiment is running, it may be convenient to scroll the data in the graph instead of showing the entire range or making the graph constantly increase its y range. Selecting Graph|Active Scrolling from the graph output's main menu shows the Active Scrolling view where the parameters for scrolling while running an experiment can be set. This is only enabled in an active graph output:



Active Scrolling

Number of points: 10

Series To Scroll With: xvalue vs yvalue

□ Disable Active Scrolling?

Maximum Number of points which will be visible on the graph.

Series that will be used to determine the number of points

For example, using the parameters above, when 10 points of series xvalue vs yvalue have been added to the graph, the x axis range will change such that when the next point is added, that point will be plotted in the middle of the graph. 5 points will be to the left and the remaining 4 points will be added on the right. Only 10 points of x value vs y value will be visible at any one time. Setting these properties enables the **Active Scrolling** button and activates active scrolling (see Graph for more details).

To disable active scrolling after it has been turned on, check **Disable Active Scrolling?** and press OK. The Graph|Active Scrolling menu item will now be unchecked.

Since this is only used for graphs acquiring data, this feature is only enabled in active (See Outputs) graphs. In order to set these properties in an active graph, there must be at least one plot command using this graph as an output.

Graph Legend

To show a legend for the graph, check the Graph|Showing Legend menu item in the output's main menu. To hide the legend, uncheck this item. If there is only one series in the graph, the individual data points will scroll as data is collected.

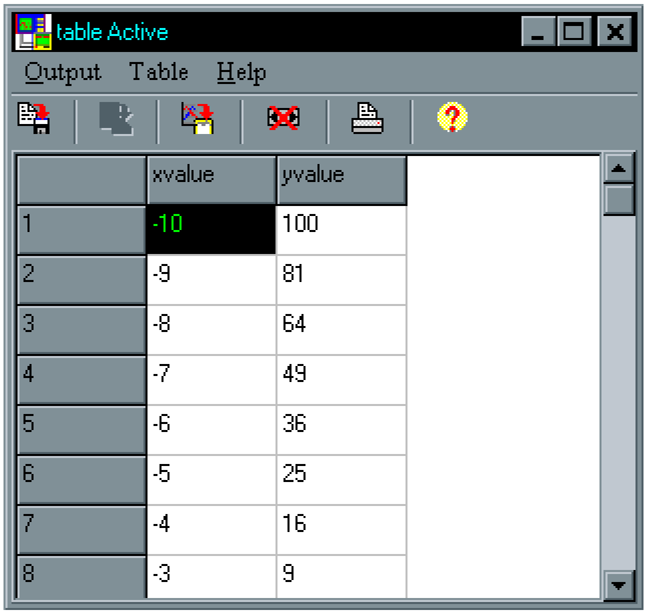
Clear

Clearing the current data can be done by selecting Output|Clear from the output's main menu.

Offline

An active output can be taken offline (remember this is permanent, once an output is taken offline, it cannot be made active again) by selecting Output|Offline from the output's main menu.

Table Output



	xvalue	yvalue
1	-10	100
2	-9	81
3	-8	64
4	-7	49
5	-6	36
6	-5	25
7	-4	16
8	-3	9

Description

This output can only plot float and integer variables. Each channel in the output is represented by a pair of columns in the table. Plot commands need to send either two variables or a counter a variable to the table output (None will be NOT be enabled in the Plot editor). The data in a table output can be copied directly into a Table output, a Graph output or a Text Output. A Table Output can accept data from a Graph or another Table Output.

Saving Data

Inactive Output

The table can be saved as an inactive output by selecting Output|Save or Output|Save As from the output's main menu.

ASCII Data

The data in the table can be saved as comma delimited text, with values in the same row delimited by commas, by selecting Output|SaveAs Type|Comma from the output's main menu. The data can also be saved as tab delimited text, with values in the same row delimited by tabs, by selecting Output|Save As Type|Tab from the output's main menu.

MATLAB ® File

Selected data in the table can be saved as a MATLAB® version 5.3 array by selecting Output|Save As Type|MATLAB from the output's main menu. This shows the Export MATLAB Data Form as previously described in the Graph Output, Saving Data, MATLAB file section.

Copy To Clipboard

The data in the table can be copied to the clipboard as text by selecting the Output|CopyToClipboard|Text menu item in the output's main menu or by selecting the Copy Button in a the table output's speed bar.

This shows the Copy Data Form as previously described in Graph Output, Saving Data, Copy To Clipboard section.

Importing Data into a Table

Both active and inactive tables can accept data from other outputs, ASCII text, the clipboard or from a MATLAB® array.

Importing Text

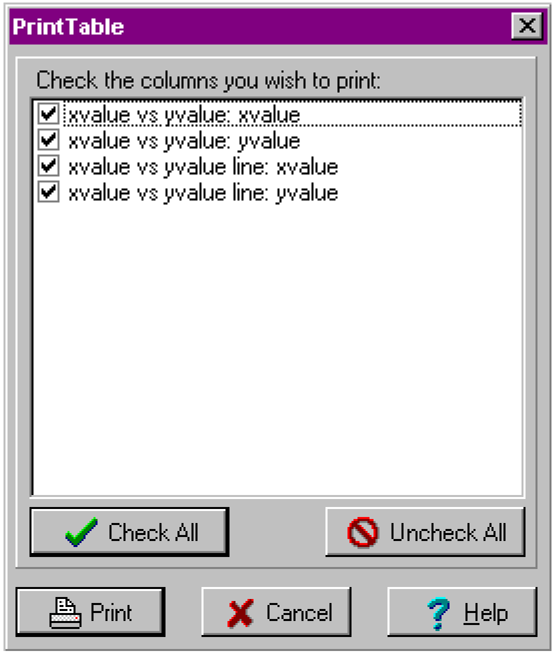
ASCII data can be imported from a file by selecting Output|Import|Text from the output's main menu. ASCII data in the clipboard (if the values are floats or integers) can be pasted into the table by selecting the Paste speed button in the table speed bar. If a file is opened, or data pasted, the form as previously described in Graph Output, Importing Data into a Graph, Importing Text section.

Importing a MATLAB® file

A MATLAB array can be imported into a table by selecting Output|Import|MATLAB File from the table output's main menu. If a file is choose, the Import MATLAB Data form (previously described in the Graph Output, Importing Data into a Graph, Importing a MATLAB file section) is shown where you can import selected data from any array in the MATLAB file into the table.

Printing a Table

Selecting Output|Print from the output's main menu or press the Print speed button in the table's speed bar shows the Print Table dialog where you can select one or more columns of data to send to the printer:



PrintTable

Check the columns you wish to print:

- ☑ xvalue vs yvalue: xvalue
- ☑ xvalue vs yvalue: yvalue
- ☑ xvalue vs yvalue line: xvalue
- ☑ xvalue vs yvalue line: yvalue

Check the columns to print (series: column) and press Print. All columns are automatically selected when the form is shown.

The columns are printed from left to right starting at the top of the list shown in the Print Table dialog box. Columns that do not fit on the page will not be printed (no wrapping). The width of the columns printed depends on the width of the columns in the Table that called this dialog box.

Appearance Options

Number Format

The format of the numbers that will show in the Table output can be changed by selecting Table|Number Format in the main menu of the output. The same form that shows in BigNum, Number Format section shows in this case. The only difference is the number format changes immediately upon acceptance of the new format.

Table Labels

The labels that show in each column of the table as well as the title of the table can be modified by Selecting the Table|Labels menu item from the table output main menu. Selecting this item shows the Table Labels form:

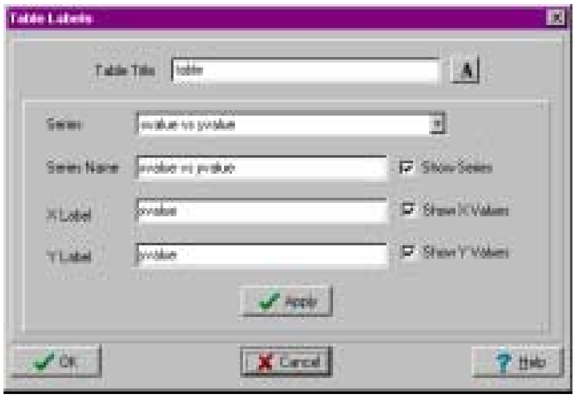


Table Labels

Table Title: table

Series: xvalue vs yvalue

Series Name: xvalue vs yvalue ☑ Show Series

X Label: xvalue ☑ Show X Values

Y Label: yvalue ☑ Show Y Values

If there are no plots, then just the Table Title and Font button are visible. Changing the font (pressing the font button and selecting a new font) affects the entire table. To change a series title, select the series from the Series combo box and its name, x column header and y column header appear in the edit boxes below. Changing the text in the edit boxes and pressing Apply changes the text in the table. To hide the entire series, uncheck the Show Series check box. To hide one of the columns, uncheck the Show X/Y Values for the column you wish to hide.

Additional Features

Auto Scrolling

This feature only has meaning for an active graph when the experiment is running. If Auto Scroll is checked, when new data is added, the table is scrolled vertically to make sure the newest data is always visible.

Hiding and Showing Columns

Columns can be hidden/shown by the user by either selecting the Table|Labels menu item or right clicking on the column to hide and selecting Hide from the popup menu. Imported data columns can also be moved. Right click on the column to be moved and select Move from the popup menu. Then click on the column where you want to move to. Only imported columns can be moved and they cannot be moved to where active columns are.

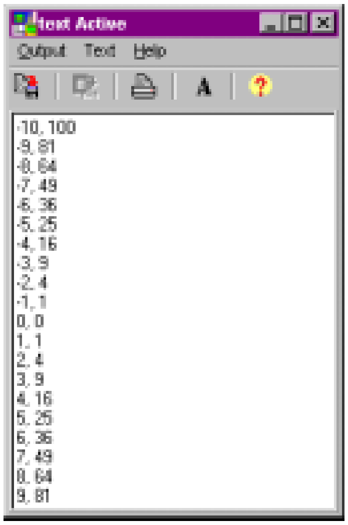
Offline

An active output can be taken offline (remember this is permanent, once an output is taken offline, it cannot be made active again) by selecting Output|Offline from the output's main menu.

Clear

Clearing the current data can be done by selecting Output|Clear from the output's main menu.

Text Output



```
-10, 100
-9, 81
-8, 64
-7, 49
-6, 36
-5, 25
-4, 16
-3, 9
-2, 4
-1, 1
0, 0
1, 1
2, 4
3, 9
4, 16
5, 25
6, 36
7, 49
8, 64
9, 81
```

Description

A text output shows the current value of any type of variable in ASCII form. The results of each plot command are separated by new lines. Plot commands only need to send one variable to the text output (None will be enabled in the Plot editor). Data from either a Table or a Graph can be transferred into a Text Output, however, data from a Text Output can only be directly transferred to another Text Output.

Saving Data

Inactive Output

The text output can be saved as an inactive output by selecting Output|Save or Output|Save As from the output's main menu.

ASCII Data

The data in the text output can be saved as text (*.txt file) by selecting Output|Save As Text from the output's main menu.

Copy To Clipboard

The selected data in the output can be copied to the Windows clipboard as ASCII text by choosing Output|Copy To Clipboard from the output's main menu or by clicking the Copy Button in a the text output's speed bar. This data can then be pasted into other Windows programs. Selecting Output|Select All Data from the text output's main menu selects all the data in the text window. This menu item is only enabled if there is data to select.

Importing Data into a Text Output

Both active and inactive text outputs can accept data from other outputs, ASCII text, the clipboard or from a MATLAB® array.

Importing Text

ASCII data can be imported from a file by selecting Output|Import|Text from the output's main menu. ASCII data in the clipboard can be pasted into the text output by selecting the Paste speed button in the text speed bar.

Importing a MATLAB ® File

A MATLAB array can be imported into a table by selecting Output|Import|MATLAB File from the text output's main menu. If a file is choose, the Import MATLAB Data form (previously described in the Graph Output, Importing Data into a Graph, Importing a MATLAB file section) is shown where you can import selected data from any array in the MATLAB file into the text output.

Printing a Text Output

Selecting Output|Print from the text output's main menu sends all the data in the Text Output to a user specified printer.

Appearance Options

Number Format

The format of the numbers that will show in the text output can be changed by selecting Text|Number Format in the main menu of the output. The same form that shows in BigNum, Number Format section shows in this case. Like the BigNum output, the number format changes when the experiment is run and new data is plotted.
Font: The font can be changed by selecting Text|Font from the output's main menu. This shows a font dialog box where fonts valid for your computer can be selected.
Title: The title of the output (in both the output form and in the Experiment Attributes list) can be changed by selecting Text|Title from the output's main menu. A form appears which has an edit box where you can enter a new title. The change occurs immediately after the Ok button in the form is pressed.

Additional Features

Clear

Clearing the current data can be done by selecting Output|Clear from the output's main menu.

Offline

An active output can be taken offline (remember this is permanent, once an output is taken offline, it cannot be made active again) by selecting Output|Offline from the output's main menu.

Experiment Files (New in version 1.5)

Introduction

Experiment Files are text files that can be written to during the course of an experiment. One to many variables can be output to the file at one time. Each call to write to the file results in one line being written in ASCII text format. There are two types, Single and Multiple. A Single file's name cannot be changed during the experiment as where a Muliple file's can be. There are 2 properties common to both file types, **Write filename at start** and **Close file between writes**. If **Write filename at start** is selected, the filename, (including path) being written to will be the first line of the file. If **Close file between writes** is selected, the file will be closed after every write to file. The Output to File command is used to write to the file.

The Single file has two additional properties, **Filename** and **Overwrite if exists**. The **Filename** is the path and name of the file to write to. The file does not have to exist but the directory does. An experiment error will result if you try to run an expeirment that has a Single file without a valid directory. The **Overwrite if exists** property determines what the experiment program does the first time it tries to write to the file and the file already exists. If **Overwrite if exists** is selected, the existing contents of the file are erased otherwise, all data is appended.

The Multiple file has 5 additional properties. The first 4 are used to create filenames. The **Directory** is the directory that will store the files. The directory must exist. Do NOT append a file name to the **Directory**. The **Base File Name** is the base name of the file to be written to. The **Extension** is the 3 letter extension to be added to the end of the created filename. The **Starting File Number** is the first number to be used when the experiment asks for a new file. This is done when the experiment is first run and when the Create a New Filename command is called. The filename for the file is *Directory\Base File Name_filenumber.Extension*. The remaining property has to do with existing files. One of three things can happen if the new filename created refers to an existing file:

Make a new file: the program will search for an unused filename. It will increment the filenumber 50 times before asking if you want to quit looking for a new filename. When an experiment is run, the starting filenumber will be changed to the next unused number at the end of the experiment.

Append to the file: when Output to file is called, data will be appended to the end of the existing file. When an experiment is run the starting filenumber is left unchanged.

Overwrite the file: the first time the experiment writes to the file, the contents are erased. When an experiment is run the starting filenumber is left unchanged.

Creating, Editing and Deleting Experiment Files

A Single Experiment File can be added to an experiment by either double clicking on the Single File category in the Experiment Attributes list or by right clicking on the Single File category in the Experiment Attributes list and selecting Add or Add Many from the popup menu. A Multiple Experiment File can be added to an experiment by either double clicking on the Multiple File category in the Experiment Attributes list or by right clicking on the Multiple File category in the Experiment Attributes list and selecting Add or Add Many from the popup menu. A file can be edited by either double clicking on the file in the Experiment Attributes list or by right clicking on the file in the Experiment Attributes list and selecting Edit from the popup menu. A file can be deleted by either selecting the file in the Experiment Attributes list, keeping the mouse over the file and pressing delete, or by right clicking on the file in the Experiment Attributes list and selecting Delete from the popup menu. Remember, deleting a file deletes all the commands that use it.

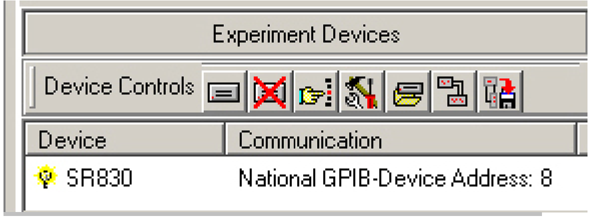
Experiment files saved/read in an Experiment Block

When saving an experiment block, all the Experiment files used by the block are saved.
When a block is read into the experiment and it has an Experiment File with the same filename (including path) as an existing Experiment File, all commands that use the read Experiment File are reassigned to the existing Experiment File.

Chapter 5 Experiment Devices

Introduction

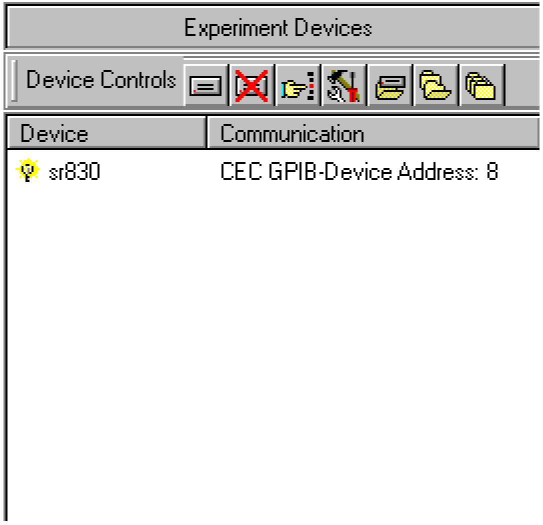
Any instrument that can interpret ASCII commands using GPIB or RS232 can be controlled with DDDA. All the devices defined in an experiment are listed in the Experiment Devices list in the DDDA Main View:



Experiment Devices

Device Controls

Device	Communication
SR830	National GPIB-Device Address: 8



Experiment Devices

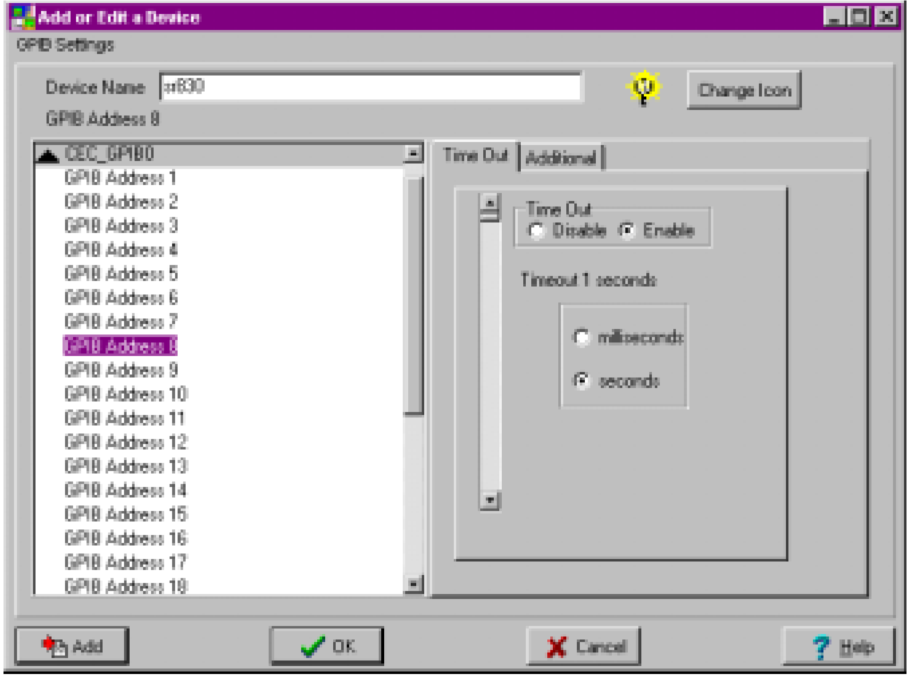
Device Controls

Device	Communication
sr830	CEC GPIB-Device Address: 8

Devices are defined in an Experiment using a device name and a device address(RS232 COM port or GPIB address). All instrument commands depend on devices being defined in the experiment. Send GPIB and Serial Poll commands require a GPIB device while Send RS232 require an RS232 interface.

Creating and Editing Devices with the Device Editor

The device editor allows the user to add devices to the experiment and to edit devices that are already defined. To add a new device, press the Add Device speed button above the Experiment Devices list or right click in any blank space and select Add Device from the popup menu. To edit an existing device, double click on the device in the Experiment Devices list or right click on the device Edit from the menu. Either of these actions (adding or editing) will show the Device Editor form:



Add or Edit a Device

GPIB Settings

Device Name: sr830

GPIB Address 8

CEC_GPIB0
GPIB Address 1
GPIB Address 2
GPIB Address 3
GPIB Address 4
GPIB Address 5
GPIB Address 6
GPIB Address 7
GPIB Address 8
GPIB Address 9
GPIB Address 10
GPIB Address 11
GPIB Address 12
GPIB Address 13
GPIB Address 14
GPIB Address 15
GPIB Address 16
GPIB Address 17
GPIB Address 18

Time Out: ○ Disable ● Enable

Timeout 1 seconds

○ milliseconds
● seconds

The Main View

In order to create a device, the following must be entered:
Device Name: The device must have a name. This name will appear in the Experiment Devices list and in all Send and Read commands. It is advised (but not mandatory) that each device has a unique name.
Device Address: The device must have a communication interface and device address. The Communication Interface lists any serial ports and GPIB cards present in your system (if you have a CEC card, please read the note about CEC cards in the Frequently Asked Questions section). Clicking on the left side (the up/down arrow) of the communication interface (RS232, GPIB0, etc) expands/collapses all the available device addresses. One of these device addresses must be selected in order to complete the device definition. If an address is being used by another device, the text for that address will read "IN USE BY" and the device name. If that address is chosen, the old device will be deleted and the device being defined will occupy that device address.

There are other properties of the device that can be changed. Once a device address is chosen, the tab sheet next to the Communication Interface list becomes enabled. For all communication interfaces, 2 tabs will be visible:

Time Out Page

The amount of time Send and Read commands wait before timeout out. The default is 1s. This may need to be lengthened if large amounts of data are read.

Additional Page

This controls terminating characters for the Send and Read commands.

Enable EOI: This checkbox is only enabled when a GPIB communication interface is being used. When checked, an EOI signal is sent after every Send command string.

First and Second characters to send after every send: These refer to ASCII characters that should be appended to every Send Command string. For example, some instruments require a carriage return, line feed combination at the end of a command string to indicate that the command is complete. In this case, the First Character would be 13 (\r) and the Second Character would be 10 (\n).

First and Second characters to terminate read: This refers to the string that is read in by a Read command. What this does depends on whether GPIB or RS232 is the communication interface. For example, lets say we are reading from an instrument that appends a carriage return line feed combination at the end of any value it sends back to the PC. The instrument returns the value "1.0324\r\n" and we have set the First Character to Terminate Read at carriage return, \r, and the Second Character to Terminate Read to line feed , \n. When a Read command is executed, one of the following will happen

GPIB: The entire string of "1.0324\r\n" is read in, but the string is truncated at \r (the first character to terminate read) so only "1.0324" is passed to the variable being read into.

RS232: . RS232 terminates the read when either all the bytes have been read in, the second (or first if second is not defined) terminating character is in the read string or a timeout occurs. In this case, "1.0324\r\n" would be read in (terminate at the second read character, \n), the string is truncated at \r (first character to terminate read) so "1.0324" would be passed to the variable being read into.

Decimal and Hexadecimal refer to how the numbers in the combo boxes for the terminating characters should be displayed. If Decimal is chosen, then the characters are defined in decimal, if Hexadecimal, the characters are defined in hex.

RS232 Page

If the communication interface is RS232, the RS232 tab is visible. This sheet allows the modification of the baud rate, the frame and the handshaking:

This program supports baud rates up to 115, 200 baud. The frame is how the data bits are organized. A frame consists of a

Start Bit: The first bit sent for each unit of data. This cannot be changed.

Data Bits: The data to be transferred, an ASCII character. The program only supports 8 data bits.

Parity: Used for error checking. This program supports none, even, odd, mark and space parity.

Stop Bits: The last bit(s) sent for each unit of data. This program supports one or two stop bits.

Handshaking controls the flow of data. Both hardware (RTS, DTR, CTS, and DSR) and software (XON/XOFF) protocols are supported by DDDA as well as no handshaking protocol. Receiver handshaking refers to the handshaking mode for received data and Transmitter handshaking refers to the handshaking mode for transmitting data.

Instruments are typically very unforgiving with respect to the RS232 settings, if they are not exactly correct, the instrument will not respond.

At the very bottom of the Device Editor there can be up to 4 buttons:

Add: Adds the device to the Experiment Devices list and clears the view for the next new device. If editing a device, this button will not be visible.

OK: Adds the device to the Experiment Devices list if is is new, updates is it was being edited, and then hides the Device Editor.

Cancel: Cancels the addition of a new device or the editing of an old device.

Help: Shows you this screen.

At the very top of the Device Editor is a lone menu item, GPIB Settings. GPIB Settings refers to the GPIB communication interface, see GPIB Settings for details on this option.

GPIB Settings

DDDA automatically scans for 1 National GPIB card and 1 CEC GPIB card upon startup of the program unless the properties in this form are changed. Usually, allowing DDDA to automatically scan for a card works, however, there is one notable exception:

CEC cards can double as National Cards, and as a result, when only a CEC card is installed, the Device Editor can show both a CEC card and a National Card. DDDA explicitly uses the manufacturer designed interface for each card, so a CEC card should never be used as a National Card (see Frequently Asked Questions for more details)

Setting the Number of Cards present in the system need only be done once if the program is properly shut down. The number of cards is saved when the program exits.

Device Files: Devices and Device Configurations

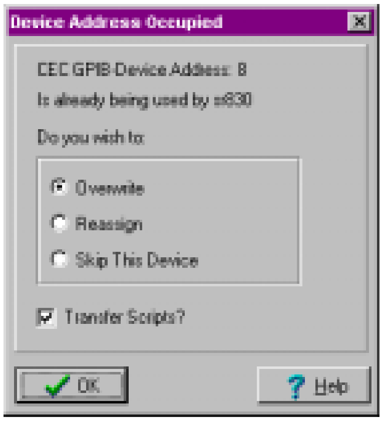
A device file (*.dev) contains a single device whereas a device configuration file (*.dcg) contains one to many devices. Each of the types of files can be saved and read in the Experiment Devices list of the Main DDDA form.

There are four ways to open device and device configuration files in DDDA:

Reading Devices and Device Configurations from file

- DDDA device files (*.dev) and DDDA Device Configuration files (*.dcg) can be dragged into the DDDA Main View.
- Selecting File|Open|Device or File|Open|Device Configuration menu items in the DDDA Main menu
- Selecting the Open Device or Open Device Configuration speed buttons in the Device Controls speed bar (at the top of the Experiment Devices list).
- Clicking anywhere in the Experiment Devices list where there is no device. Right click and select the Open Device or Open Device Configuration menu items.

When a device or device configuration is read and there is already a device in the experiment that is using the device address of the read device, the Device Occupied form is shown:



Device Address Occupied

CEC GPIB Device Address: 8

Is already being used by sr830

Do you wish to:

● Overwrite

○ Reassign

○ Skip This Device

☑ Transfer Scripts?

There are three options:

Overwrite: The current device is overwritten with the read device. In this case, there is a option to transfer scripts, meaning that any commands in the experiment using the current device will be transferred to the read device.

Reassign: Calls the Reassign Device Form, in which the read device can be given a new device address. There will be one drop down list for each communication interface available on your system. Select the new device address from one of the combo boxes. Only available addresses will be listed. Pressing OK reassigns the read device, pressing Cancel skips the reading in of the device.

Skip: Does not read in the device. This is especially useful when reading in a device configuration where all the devices will not be used. Note: If you are reading in a device from a block and select this, any commands that use this device will be deleted.

Pressing **OK** will perform the selected action. There is no cancel, use Skip to avoid reading in the device.

If a device is read in and the saved communication interface does not exist on the computer, the Reassign device form described in the previous section will be displayed.

Saving Devices and Device Configurations from file

Devices can be saved by selecting the device, right clicking on it and selecting Save Device from the popup menu.

Device configurations can be saved by Clicking on the Save Device Configuration button in the Device Controls speed bar at the top of the Experiment Devices list.

Finding and testing devices

Scan For Devices

Scan for Devices allows the user to scan for all the listeners connected to a GPIB interface card. In order for this feature to work, the card must support this feature and all the attached devices must be powered up. To check if a GPIB card supports this feature perform the following test:
CEC Card (NOT in National mode, see Advanced Communications): Run trtest. Choose the function Feature (F). Type in 0 (to check for Listener Present Feature). Press Return. If the return value is 0, you CANNOT scan for devices using this GPIB card. If the return value is 1, you CAN scan for devices using this GPIB card.
National Card: It must recognize the ibln command.

If there are no GPIB cards which can scan for listeners, the message "No GPIB board have been detected or the GPIB board does not support device scan" will appear when the Scan for Devices ☞ button in the Experiment Devices section is pressed. Otherwise, the Scan for Devices view will appear. Select the GPIB card from the combo box at the top and press Scan to detect listeners. If there are no listeners, the message "No devices found. Make sure the devices are powered up." will appear. Otherwise, the devices will be listed in the list box. Check all that should be added to the experiment and press either Apply or OK.

Testing Devices

Before running an experiment, it may be useful to see if all the devices in the experiment are responding properly. This can easily be done with Interact With Device. Right clicking on a selected device, pressing the Interact with Device ☞ speed button accesses the view or selecting Devices|Interact With Device from the DDDA Main View main menu.
To interact with a device:

1) Select the device (if it is not already selected) from the drop down list at the top of the form.
2) Select the Device Tab. To send a command make sure the Send check box is checked and enter a command in the edit box next to the check box. DO NOT include terminating characters. These should be set in the device editor. If the Send check box is not checked, this edit box will be disabled. To read a response from a device make sure the Read check box is checked and enter the number of bytes to read. This number must be an integer value greater than 0. If the Read check box is not checked, the edit box for entering the number of bytes will be disabled.
3) Press the Send button. All results will appear in the table. If a command has been sent, the command string will appear in the table under Action and whether or not it timed out will appear under Result. If a response has been read, the number of bytes to be read in will appear under Action in the table and the response will appear under Result. The Send button will then become disabled until all the commands are sent.

If the device does not respond properly, first make sure it is powered up and then check the terminating characters (see device editor).
The Properties menu has 2 menu items:
Show All Interactions: If this is checked, the exact string for the particular communication interface that is being used is shown. For example, for CEC GPIB interface, each send and read pair results in 2 strings being sent and one string being read. The first send sends the command. The second send tells the device it is the talker and to start sending data. The read command reads the data from the device.
Colors: This shows the Interact Color view which allows the colors of the Interact With Device form to be changed. The following colors can be changed by selecting the item from the list and color from the pallet:
Backgound1 and Background 2: Consecutive presses of the Send Command are distinguished in the Interact With Device table by alternating these 2 background colors.
Action: Changes the color of the text in the **Action** column. Make sure this does not match either Background 1 or Background 2 or the text will be invisible.

Send: Changes the color of the text indicating strings which are sent to a device in the **Result** column. Make sure this does not match either Background 1 or Background 2 or the text will be invisible.

Read: Changes the color of the text indicating strings which are read from a device in the **Result** column. Make sure this does not match either Background 1 or Background 2 or the text will be invisible.

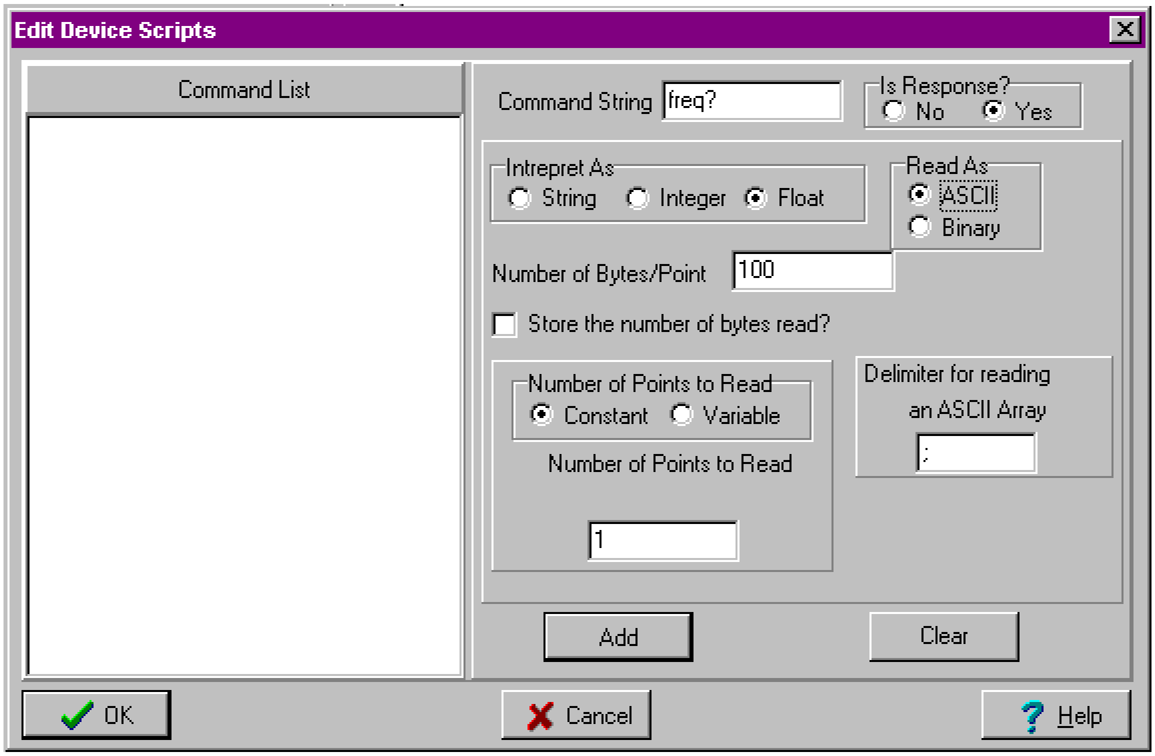
GPIB: Changes the color of the text indicating GPIB commands which are sent to a GPIB device in the **Result** column. Make sure this does not match either Background 1 or Background 2 or the text will be invisible.

Error: Changes the color of the text indicating an error when sending or reading from a device in the **Result** column. Make sure this does not match either Background 1 or Background 2 or the text will be invisible.

In addition to commands, GPIB commands can be sent to devices using GPIB interface. Select the GPIB tab. If the device selected using GPIB interface, all of the check boxes will be enabled (otherwise, they will not be). Check the GPIB commands to send and press the Send button. The GPIB command sent will appear under Action in the table and whether the device timed out will appear under Result.

Device Commands

A Device Command is a shortened version of the Send and Read commands used in DDDA, that is associated with a particular device. They can be created and edited using the Edit Device Commands form and are saved when the device is saved. To add these commands to a device, select the Add/Edit Device Commands menu item from the device popup menu (right click on the device you wish to add or edit commands):



Command String: freq?

Is Response? ○ No ● Yes

Intrepret As: ○ String ○ Integer ● Float

Read As: ● ASCII ○ Binary

Number of Bytes/Point: 100

□ Store the number of bytes read?

Number of Points to Read: ● Constant ○ Variable

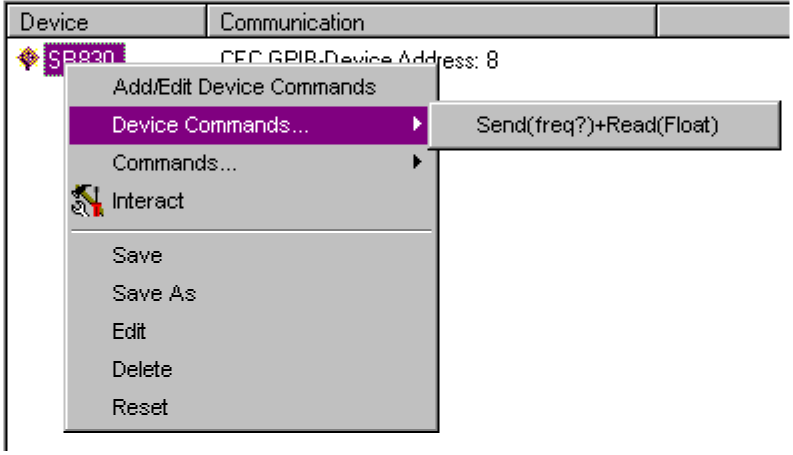
Number of Points to Read: 1

Delimiter for reading an ASCII Array: ;

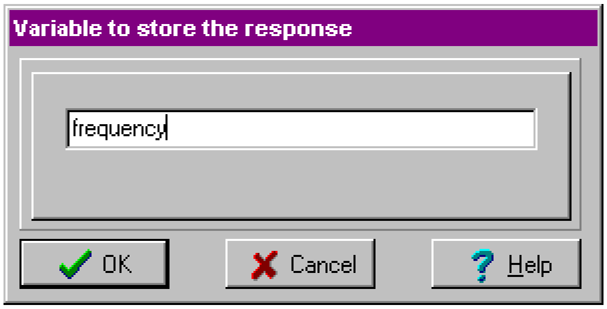
1) Enter the command string to be sent in the Command String box.
2) If the instrument does not send a response to this command select No skip to 8. If the instrument does send a response to this command select Yes. The remainder of the page will now become enabled.
3) Select how the data is to be interpreted, String, Integer or Float.
4) Select how to read in the data, as ASCII or Binary.
5) If the number of points to be read in is constant, select Constant and enter the number of points in the Number of Points To Read edit box.
6) If the number of points to be read in is greater than one and the data is being read in as ASCII, enter a delimiter in the Delmiter for reading and ASCII Array.
7) Check the Store the number of bytes read check box to store the number of bytes read in a integer variable.
8) Press Add. The Send and, if defined, a Read command, will be added to the Command List.

To edit a command, double click on the command or right click and select Edit from the popup menu. When finished editing, make sure to press the Update button or the command will not be changed. Pressing OK does NOT save changes to a device command. To delete a command, right click and select Delete from the popup menu.

All the Device Commands defined for the the device are listed in the Device Command menu item of the device popup menu (right click on a device in the Experiment Devices list, lower right hand corner of the Main DDDA View). :



In the above example, when the Device Command Send(freq?)+Read(Float) is selected, the Change Variable view is shown:


Variable to store the response

frequency

The Change Variable view is shown because of Read(Float) in the Send(freq?)+Read(Float) Device Command.   The float value being read needs to be assigned to a single variable. In this case, there are no single variables already in the selected block, so a new one has to be created. Pressing OK results in a Send and Read command being added to the Experiment View and a float single variable *frequency* being added to the Experiment Attributes list.

How DDDA communicates with instruments

This topic assumes that you are familiar with Capital Equipment Corporation's CEC488 version 5.05 functions, National Instruments NI-488 functions, and RS232 communication. Throughout this topic, the Arial regular font refers to library commands of either CEC or National Instruments.

Sending ASCII data

Capital Equipment Corporation: CEC488

DDDA uses the Transmit command to send data to a device. The END transmit command is set using SetOutputEOS with the first and second terminating characters defined by the user in the Device Editor. The timeout is set using the SetTimeout function. The instrument is set as a listener and the DATA command is followed by the string from the DDDA Send command.

National Instrument: NI-488

DDDA uses the ibwrt command to send data to a device. The terminating characters defined by the user in the Device Editor are appended to the end of the string from the DDDA Send command.

RS232

The transmission buffer is flushed. The terminating characters defined by the user in the Device Editor are appended to the end of the string from the DDDA Send command and the string is transmitted.

Sending binary data

Capital Equipment Corporation: CEC488

DDDA uses the TArray command to send binary data to a device. The timeout is set using the SetTimeout function. The instrument is set as a listener before the TArray command is sent.

National Instrument: NI-488

DDDA uses the ibwrt command to send binary data to a device. The terminating characters defined by the user in the Device Editor are not appended to the end of the data buffer.

RS232

The transmission buffer is flushed. The data buffer is sent over the RS232 line. The terminating characters defined by the user in the Device Editor are not appended to the end of the data buffer.

Reading data from an instrument

Capital Equipment Corporation: CEC488

DDDA uses the Transmit command followed by the rarray command to read data from a device. The transmit command is used to signal the instrument that it is the talker. The rarray command receives binary data (no characters are ignored) and terminates when either the specified number of bytes has been read in, an EOI signal is received or a timeout occurs.

National Instrument: NI-488

DDDA uses the ibrd command to read data from a device. The ibrd command receives binary data (no characters are ignored) and terminates when either the specified number of bytes has been read in, an EOI signal is received or a timeout occurs.

RS232

Bytes are read from the receive buffer until either the number of bytes is read in, the terminating read character defined by the user in the Device Editor is read in or when a timeout occurs. If there are 2 terminating read characters, the read will stop when the second character is read. The first character will be removed when the data is stored in the variable. NOTE: If you are reading in binary data the terminating characters you have defined will be ignored.

For all cases, if a timeout occurs, it means that all the data was not read or all the data was not sent in the amount of time specified by timeout.

When timeout is an error

If a timeout occurs during the execution of a Send command, it is always a "Send Error" that aborts the DDDA experiment.
If a timeout occurs during the execution of a Read command, it will abort the DDDA experiment with a "Read Error" if the number of bytes read in is NOT stored (the store number of bytes read in check box in the Read Editor and Send and Read Editor). If, however, the number of bytes read in is stored, the experiment will continue (no "Read Error") even if 0 bytes are read in. Be very careful with this. If 0 bytes are read in, and you try and plot the resulting value, you may or may not get a DDDA plot error which aborts the experiment. You will, however, get bogus data. It is therefore strongly suggested that if you store the number of bytes read, you actually check that value before attempting to do anything with data read from the device.

How DDDA intreprets data

All binary data is read and intrepreted "as is". No filtering is done. ASCII data, however needs to be filtered, whether it is removing terminating characters from the string or parsing a string array.

Intrepreting Single Values

All strings will first pass through the terminating character filter. The string that is read in from the instrument is tranferred to a new string that ends at the specified terminating character. If there is more than one, it stops at the first one. For example:

read string: "Stanford_Research_Systems.SR830.s//n12345.ver1.01 /n/r..st a reminder...."
terminating character: "/n" (not //n)
filtered string: "Stanford_Research_Systems.SR830.s//n12345.ver1.01"

Any remaining control characters and whitespace are then removed from the ends. If the value is to be intrepreted as a string, nothing more is done. If, however, the value is to be an integer or a float, a second filter is used.

Single Float Values

There are 6 acceptable float formats:

±n	±nE±n
±n.n	±n.nE±n
±.n	±.nE±n

Where the sign (±) is optional, n is a digit between 0 and 9 and E can be either an E or an e. Leading and trailing invalid characters are ignored. For example:

Example 1 (terminating character is "/n"):

Type of string	String	What was done
Read String	",-1.936E-004/n"	NA
Filter 1	",-1.936E-004"	The terminating character was removed
Filter 2	"-1.936E-004"	The comma was removed

Example 2 (terminating character is "/n"):

Type of string	String	What was done
Read String	"-1.936E-004,2.14E-005/n"	NA
Filter 1	"-1.936E-004,2.14E-005"	The terminating character was removed
Filter 2	"-1.936E-004"	The string was truncated at the comma

Example 3 (terminating character is "/n"):

Type of string	String	What was done
Read String	"-1.936 E-004/n"	NA
Filter 1	"-1.936 E-004"	The terminating character was removed
Filter 2	"-1.936"	The string was truncated at the whitespace.

After passing through the 2 filters, the value is converted to an IEEE floating point value. This will almost always be valid, so be careful.

Single Integer Values

There are 2 valid integer formats, ±n and n where n is a digit between 0 and 9. Like the float values, leading and trailing invalid characters are ignored. For example:

Example 1 (terminating character is "/n"):

Type of string	String	What was done
Read String	",-340/n"	NA
Filter 1	",-340"	The terminating character was removed
Filter 2	"-340"	The comma was removed

Example 2 (terminating character is "/n"):

Type of string	String	What was done
Read String	"-1.936E-004/n"	NA
Filter 1	"-1.936E-004"	The terminating character was removed
Filter 2	"-1"	The string was truncated at the decimal point, this is an integer.

Intrepreting Array Values

Once the entire array is read in, it is parsed. In addition to using the user specified delimiter, all control characters are automatically considered to be delimiters. For example:

read string: "1.234E-005,5.678E-005,9.012E-005\n"
The user specified that the delimiter is ","
The string will be parsed into three strings:
"1.234E-005"
"5.678E-005"
"9.012E-005"

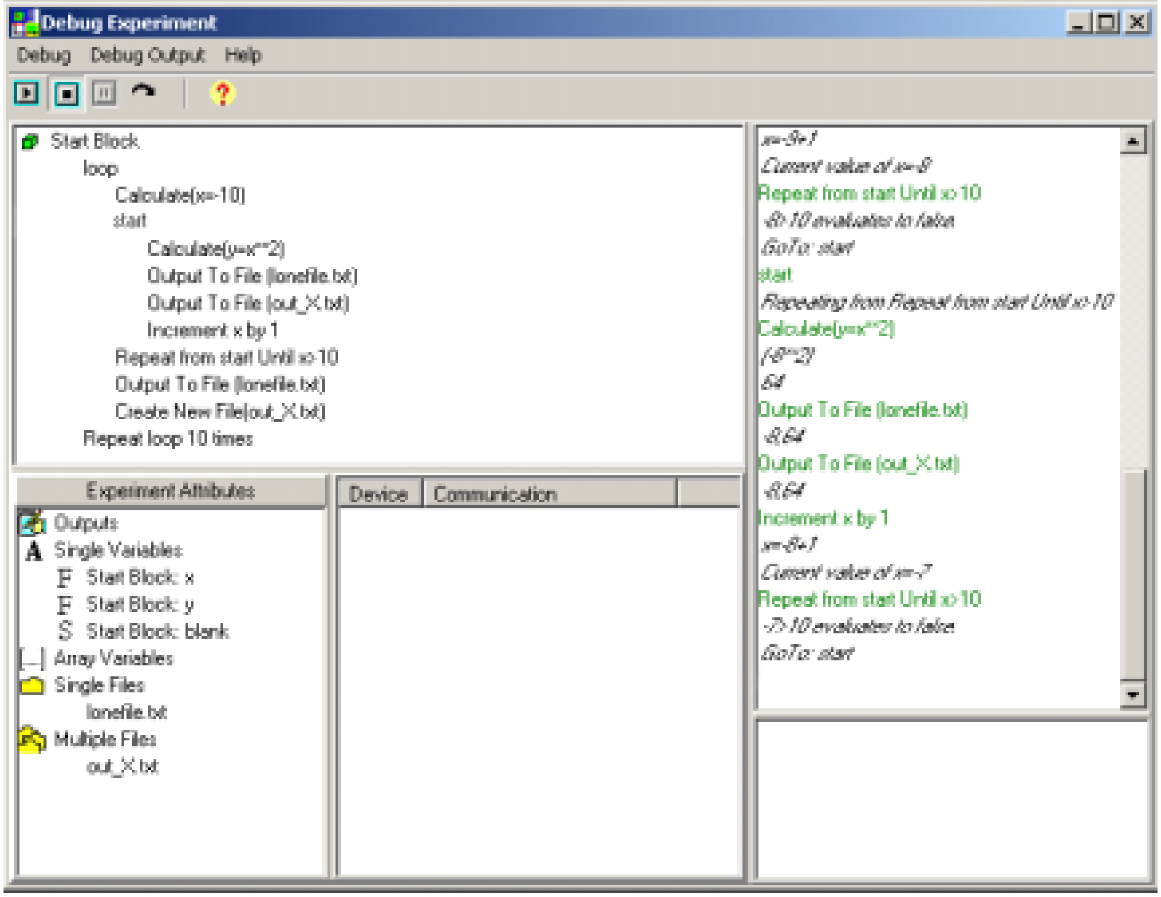
If these strings are to be intrepreted as integers or floats, they will pass through the appropriate above described filters before being converted.

Whitespace is always considered to be a delimiter if the value to be intrepreted is an integer or a float. If it is a string, however, the user can specify whether to consider it as a delimiter (the default) or not (see the topics Send and Read, Read and Read From File). For example, the string "This is my instrument" would be viewed as one string if whitespace is NOT considered a delimiter, but would be parsed into 4 strings if whitespace IS considered a delimiter.

Chapter 6 Debugging DDDA Experiments

The Debugger

Clicking Debug in the main tool bar of DDDA brings up the Debug Form:



```
Start Block
    loop
        Calculate(x=-10)
        start
            Calculate(y=x**2)
            Output To File (lonefile.txt)
            Output To File (out_X.txt)
            Increment x by 1
        Repeat from start Until x>10
        Output To File (lonefile.txt)
        Create New File(out_X.txt)
    Repeat loop 10 times
```

```
Outputs
Single Variables
    Start Block: x
    Start Block: y
    Start Block: blank
Array Variables
Single Files
    lonefile.txt
Multiple Files
    out_X.txt
```

```
x=-9+1
Current value of x=-8
Repeat from start Until x>10
-8>10 evaluates to false
GoTo: start
start
Repeating from Repeat from start Until x>10
Calculate(y=x**2)
(-8**2)
64
Output To File (lonefile.txt)
-8,64
Output To File (out_X.txt)
-8,64
Increment x by 1
x=-8+1
Current value of x=-7
Repeat from start Until x>10
-7>10 evaluates to false
GoTo: start
```

There are two halves, the experiment side, on the left, and the debug output on the right. On the Experiment Side, there are three sections:

Experiment View: Top, experiment side. A listing of the Experiment View just as it appears in the DDDA Main View. In the debugger, commands can only be edited and deleted. New commands cannot be added. Experiment blocks cannot be edited, deleted or added in the debugger. To edit a command either double click on it or select Edit from the popup menu (right click on the command and the popup menu will appear). To delete the command select Delete from the popup menu.

Experiment Attributes: Bottom left, experiment side. A listing of the Experiment Attributes just as it appears in the DDDA Main View. In the debugger, outputs, variables and files can be edited but new ones cannot be added nor can they be deleted. To edit an experiment attribute, double click on the attribute or select Edit from the popup menu (right click on the attribute and the popup menu will appear). The popup menu that appears when an output is selected has an additional menu item, Show, which shows the output window. The popup menu that appears when a variable is selected has an additional menu item, Trace. If Trace is checked when the experiment is run in the debugger, the name of the variable, the new value of that variable and the command that has updated its value is shown in the trace window. In addition, if Trace is checked the name of variable in the Experiment Attributes list appears in italics.

Devices: Bottom right, experiment side. A listing of devices just as it appears in the DDDA Main View. Devices can be edited and interacted with but they cannot be added or deleted. To edit a device, double click on the device or select Edit from the Device Popup menu (right click on a device). To interact with a device, select Interact With Device from the device popup menu.
On the Debug Output Side there are two sections:
Output Window: Shows the debug output for each script, see Debug Output.
Trace Window: Shows all the variables that have their Trace menu item checked. To clear the traces or to clear the trace window, right click on the trace window and select the corresponding menu item.

There are five buttons at the top of the debug form:
▶**Start**: Starts the experiment running from either a paused or stopped state.
■**Stop**: Stops the experiment. The text "Experiment Stopped" will appear at the bottom of the debug output. When Start is pressed, the experiment will start from the first command.
⏸**Pause**: Pauses the experiment. The command that it is paused on will be highlighted. When start is pressed, the experiment will start at the command highlighted.
↷**Step**: Runs the experiment one command at a time. This can be used at the beginning of an experiment as well as after pausing an experiment.
?**Help**: Shows the help file for the debugger.

The Menu items are as follows:
Debug:
Start, Stop, Pause, Step: same as the speed buttons (see above).
Scroll Output: If this is checked, the debug output is scrolled so that new lines are always visible.
Close: Closes the Debugger.
Debug Output: The following menu items are only visible if there is output in the Debug Output window.
Copy To Clipboard: Copies all the text in the Debug Output and puts it in the Window's Clipboard.
Save To File: Copies all the contents of the Debug Output window to a text file.
Print: Copies all the contents of the Debug Output window to a printer.
Clear: Clears all text from the Debug Output. This is done automatically at the start of every experiment.

Debug Output

The debug output consists of the following:

Block Name in blue, underlined
Command string, in green
Description of the command, in italics
Result of the command, in italics, if there is one. If the result is in error, it appears in bold red.

For example, lets say an sr830 is attached via CEC GPIB address 8 and the following exercise is run:

```
Start Block
sr830:Send(*IDN?)
sr830:Read(name)  name is a string variable
nameoutput: Plot(name)
```

If the experiment runs correctly, then the following will be listed in the debug output:

Start Block

```
sr830: Send(*IDN?)
Sending to sr830
transmit(UNT UNL MTA LISTEN 8 DATA '*IDN?' EOI END)
Status is 0x0000
sr830: Read(name)
Reading from sr830
transmit(UNT UNL MLA TALK 8)
rarray(response, 101)
Status is 0x0020
textname: Plot(name)
Stanford_Research_Systems,SR830,s/n22626,ver1.01
Experiment Finished
```

If, however, the instrument was not on, the following would result:

```
Start Block
sr830: Send(*IDN?)
Send Command timed out
Experiment Aborted
```

The following is a list of commands, their descriptions and possible results:

Calculate: The description is the expression with the current values of all the variables. The result is the evaluation of the expression.
Clear Output: The description of Clear Output is *Clearing output output name*. There is no result.
Comment:The description of Comment is just the comment. There is no result.
Concatenate Strings: The description of Concatenate String is the original expression. The result is the result string.
Create A New File: The description of Create A New File is the phrase Creating new file and the name (including path) of the file that will be written to the next time the Output To File command is run. There is no result.
Decision: The description for the Decision command is *n condition x evaluates to true/false* where n is the current value of the variable to be compared and x is the current value to compare to n using the condition. The result is the name of the block that will be next.
Increment: The description is *variable name=n+x* where n is the current value of the variable and x is the value incrementing by. The result is *Current value of counter=n* where n is the new value of the variable.

If Then Else: The description for the If statement is *n condition x evaluates to true/false* where n is the current value of the variable to be compared and x is the current value to compare to n using the condition. There is no result. If the statement evaluates to true, then the commands beneath the If command will be shown otherwise, the next command will be the else, if it has been defined, or endif if else has not been defined. If an else statement has been defined, then Else will appear. Else has no description or result. If the commands beneath else have been executed, then they will appear, otherwise, the next command will be endif. Endif has no description or result.

Label: The description depends on if there is a repeat or repeat until command that uses it. If there is a repeat command, then the line *Repeating from Repeat n times* is written, where n is the number of times the commands are repeated. If there is a repeat until command then *Repeating from Repeat from label Until condition* where condition is the condition defined in the Repeat Until command is written. Otherwise, the line *Nothing repeats here* appears. There is no result.

Output To File: The description is just the values being written to the file. There is no result.

Plot: The description of plot is just the value(s) it is plotting. There is no result.

Read: The description of Read is *Reading from device name*. There are three to four result lines. The first line (and second in the case of CEC) is the exact command sent using the C library for the communication interface. The second line (third for CEC) is the status of the command (for GPIB it is a vendor dependent number in hex, check your GPIB manufacturer, CEC or National, for what the status numbers mean.). The third line (fourth for CEC) if is present, is an error line and will appear in red.

Read From File: The description of the Read From File command is *Storing Data Count in variable name* if storing the number of values read, otherwise it is *Not storing data count*. The result line is the *Number of values read/array size*. This is a ratio of the number of values written to the array over the array size.

Repeat: The description is *Counter at n :Go to label* where n is the number of times the repeat command has been run and label is the next command it runs. If n is equal to the number of times it has to repeat, then the description is *Counter at n :Finished repeat.* There is no result.

Repeat Until: The description is *condition evaluates to true/false* where condition is the condition defined in the Repeat Until command with the current values inserted. If the condition evaluates to true, the result is *GoTo: label* where label is the next command run, Otherwise, the result is *Repeat finished*.

Send: The description of Send is *Sending to device name*. There are two to three result lines. The first line is the exact command sent using the C library for the communication interface. The second line is the status of the command just sent (for GPIB it is a vendor dependent number in hex, check your GPIB manufacturer, CEC or National, for what the status numbers mean.) The third line, if it is present, is an error line and will appear in red.

Send Binary Data: The description of the Send Binary command is *Sending array name to device name*. There is one result line, *Sent number of points, number of bytes.*

Send GPIB: The description of SendGPIB is *Sending to device name*. There are two to three result lines. The first line is the exact command sent using the C library for the communication interface. The second line is the status of the command just sent. It is a vendor dependent number in hex, check your GPIB manufacturer, CEC or National, for what the status numbers mean. The third line, if it is present, is an error line and will appear in red.

Send RS232: The description of SendRS232 is *Clearing buffer name*. There is no result.

Serial Poll: The description of SerialPoll is *Serial polling: device name*. There are one to two result lines. The first line is *Poll result: hex number.* To interpret the hex number, see the manual for the device. The second line, if it is present, is *Read Command timed out.* Since this is an error, it is in red.

Wait: The description is *Slept x ms* where x is the number of milliseconds the program thread slept before going to the next step. There is no result.

Chapter 7 Tutorials

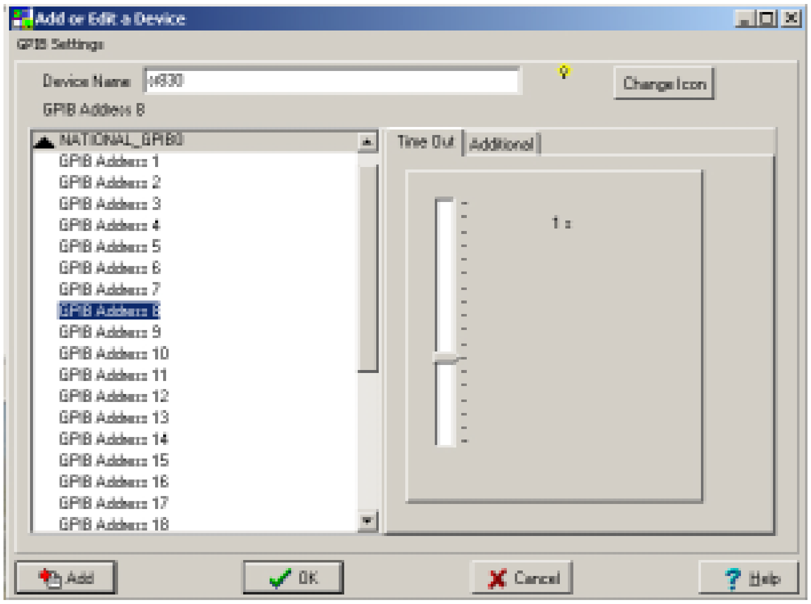
*Tutorial 1: *IDN? The Hello World of Data Acquisition - GPIB*

In this tutorial you will learn how to

- Add an instrument using a GPIB interface to a DDDA experiment.
- Send a command string to a device.
- Read a response from a device.
- Plot the response to a text output.

If you are using an IEEE 488.2 compliant device, then *IDN? should be in its command list. Before running this tutorial, connect your device with a GPIB cable and check the address of the device. (If you are actually testing this on an SR830, make sure you change the interface to GPIB.) This command is supposed to send back a string which identifies the manufacturer of the device, its name, and its serial number. To test this command using DDDA:

1) **Create the device**. Select the Add Device button in the Experiment Devices section or right click anywhere (other than on a device) in the Experiment Devices list box and select Add Device from the popup menu. You will get the Device Editor View:



Add or Edit a Device

GPIB Settings

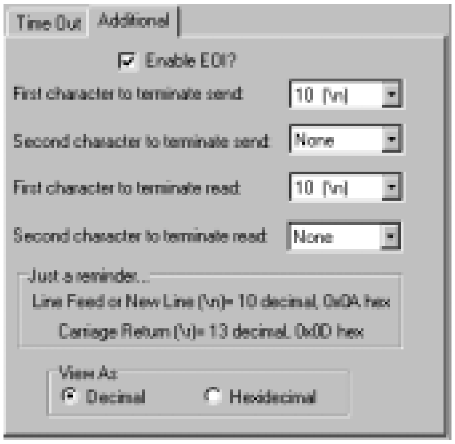
Device Name: sr830

GPIB Address 8

in the Experiment Devices section or right click anywhere (other than on a device) in the Experiment Devices list box and select Add Device from the popup menu.

Enter the name of the device, select the communication interface (click on the triangle on the left) and the address of the device. In the above example, the device is an sr830 connected via CEC_GPIB0 (communication interface) at GPIB address 8(device address).

Click on the Additional tab:



☑ Enable EOI?

First character to terminate send: 10 [\n]

Second character to terminate send: None

First character to terminate read: 10 [\n]

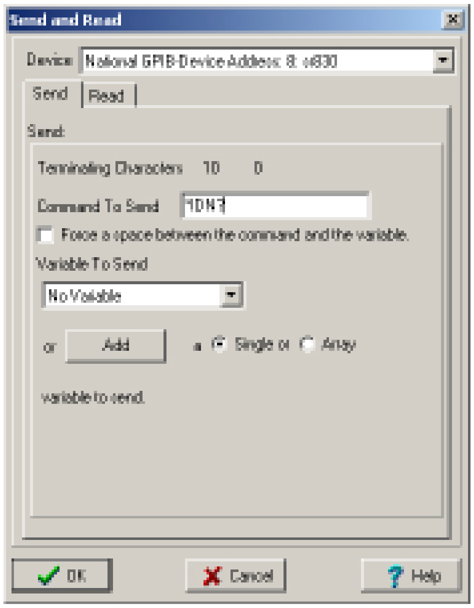
Second character to terminate read: None

Just a reminder...
Line Feed or New Line (\n)= 10 decimal, 0x0A hex
Carriage Return (\r)= 13 decimal, 0x0D hex

View As
● Decimal ○ Hexidecimal

Then make sure the First and Second character to send after every send are correct (the SR830 wants a line feed or EOI to signify the end of a command string), the First and Second character to terminate read are correct (the SR830 appends a line feed at the end of the return string), and press OK . This adds the device to the Experiment Devices list.

2) Set the Send and Read Commands: Select Send and Read from the Commands tool bar (at the top of the DDDA Main view) or right click on the blank command (the arrow points to it) in the Experiment View section and select Send and Read from the popup menu. You should see the Send and Read Editor:



Device: National GPIB Device Address: 8: sr830

Send:

Terminating Characters 10 0

Command To Send *IDN?

□ Force a space between the command and the variable.

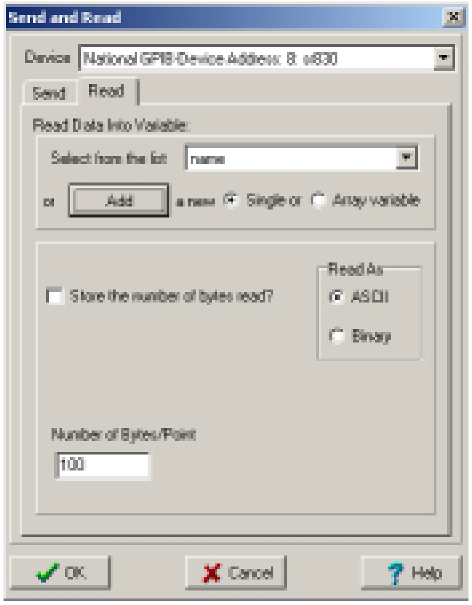
Variable To Send

No Variable

or Add a ● Single or ○ Array

variable to send.

The device should be selected (since it is the only device in the experiment). The Send tab should be selected. Enter the command **IDN?* in the Send Text Box. *No Variable* should be selected in the Send Variable Combo Box (this is for sending the current value of a variable, see Send for more details on this feature). Select the Read tab and you should see:



Device: National GPIB-Device Address: 8: sr830

Select from the list: name

or Add a new ● Single or ○ Array variable

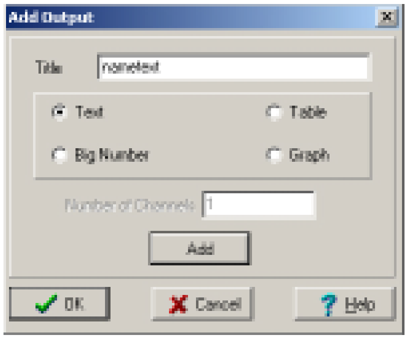
□ Store the number of bytes read?

Read As: ● ASCII ○ Binary

Number of Bytes/Point: 100

There should be no variables listed in the Read Combo Box (since there are no variables in the experiment yet) so select the Single radio button and then press Add. The Add Single view will appear and give the variable the name *name* and its type should be *String* (the default). Press OK and the *name* variable should appear selected in the Read Combo Box. Now the box below the read box should appear. Keep Read As selected as *ASCII* and the number of bytes/point to *100* Press OK and this form will disappear and 2 commands will appear in the Experiment View Section under *Start Block*.

3) Create the Output: Double click on Outputs in the Experiment Attributes list box or right click on Outputs and select Add Output from the popup menu. You should see the Add Output view:



Title: nametext

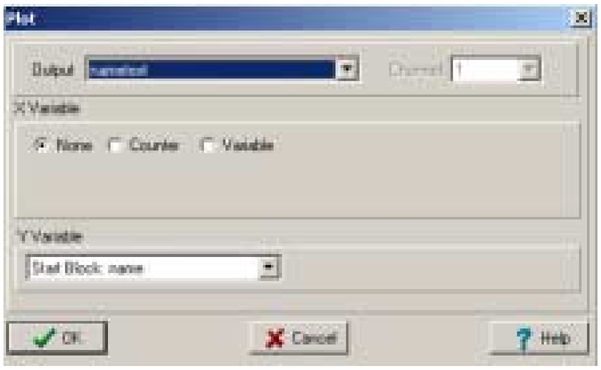
● Text ○ Table

○ Big Number ○ Graph

Number of Channels: 1

Give the output the Title *nametext* and select Text as the type (if it is not already selected) and press OK. There should now be a Text output called *nametext* under Outputs in the Experiment Attributes list box.

4) Add the Plot Command: Select Add Plot from the Commands tool bar (at the top of the DDDA Main view) or right click on the blank command in the Experiment View and select Plot from the popup menu. You should see the Plot Editor View:



Plot

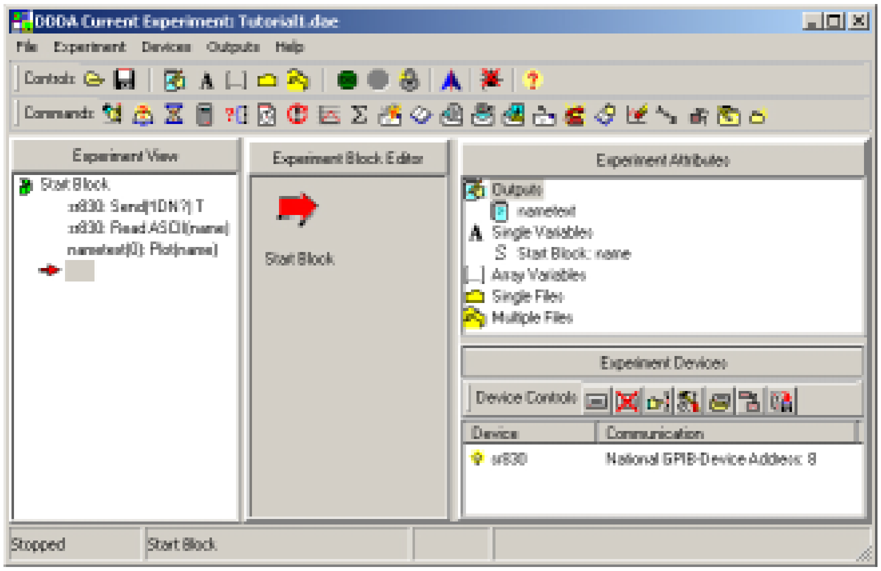
Output: nametext

Channel: 1

X Variable: ● None

Y Variable: Start Block: name

The *nametext* output, XVariable None and YVariable *name* should already be selected. Press OK and the plot command will be added to the command list under the *Start Block* in the Experiment View. At this point, the DDDA Main form should look like:



DDDA Current Experiment: Tutorial1.dae

Experiment View:

Start Block
sr830: Send(*IDN?) T
sr830: Read ASCII(name)
nametext(0): Plot(name)

Experiment Block Editor: Start Block

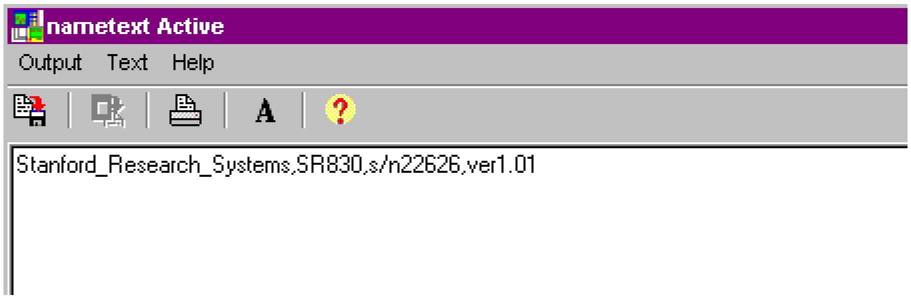
Experiment Attributes:

Outputs
nametext
Single Variables
S Start Block: name
Array Variables
Single Files
Multiple Files

Device	Communication
sr830	National GPIB-Device Address: 8

Stopped | Start Block

Double click on the output *nametext*, under Outputs in the Experiment Attributes box, to show the output. Press the Go button on the Controls bar in the DDDA Main form and you should see the following (for your device) in the output *nametext*:



```
Stanford_Research_Systems,SR830,s/n22626,ver1.01
```

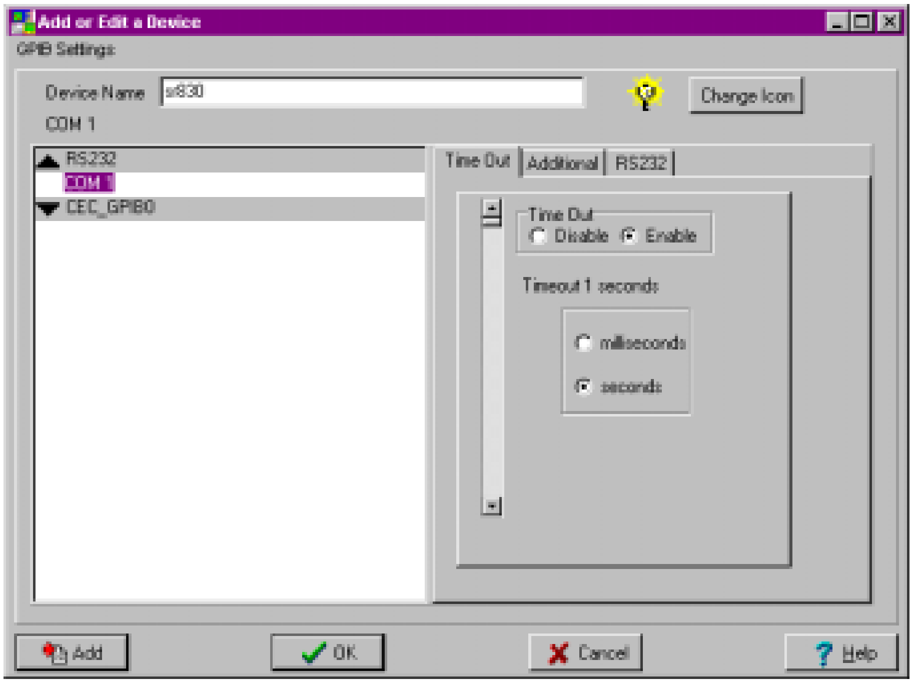
*Tutorial 2: *IDN? The Hello World of Data Acquisition - RS232*

In this tutorial you will learn how to

1) Add a device using an RS232 interface to an experiment.
2) Send a command string to a device.
3) Read a response from a device.
4) Plot a response to a text output.

If you are using an IEEE 488.2 compliant device, then *IDN? should be in its command list. This command is supposed to send back a string which identifies the manufacturer of the device, its name, an its serial number. Before continuing with this tutorial, connect to your device with an RS232 cable (If you are actually testing this on an SR830, make sure you change the interface to RS232). To test this command using DDDA:

1) Create the device. Select the Add Device button in the Experiment Devices tool bar or right click anywhere (other than on a device) in the Experiment Devices list box and select Add Device from the popup menu. You will get the Device Editor View:



Device Name sr830

COM 1

Time Out ○ Disable ● Enable

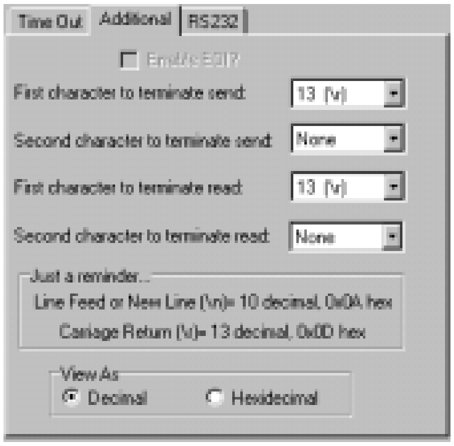
Timeout 1 seconds

○ milliseconds

● seconds

Enter the name of the device, select the communication interface (click on the left side of it) and select the address. In the above example, the device is an sr830 connected via RS232 (communication interface) at COM1(device address).

Click on the Additional tab:



Time Out | Additional | RS232

□ End/ic EOI?

First character to terminate send: 13 (\r)

Second character to terminate send: None

First character to terminate read: 13 (\r)

Second character to terminate read: None

Just a reminder...

Line Feed or New Line (\n)= 10 decimal, 0x0A hex

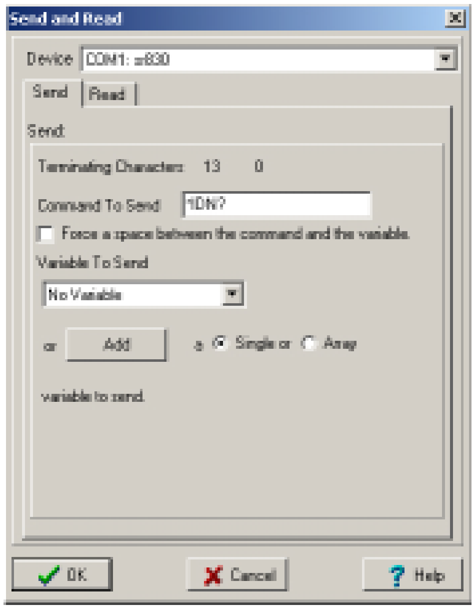
Carriage Return (\r)= 13 decimal, 0x0D hex

View As

● Decimal ○ Hexidecimal

Then make sure the First and Second character to send after every send are correct (the SR830 wants a carriage return (\r) or line feed (\n) to signify the end of a command string), the First and Second Character to terminate read are correct (the SR830 appends a carriage return at the end of the return string), and press OK . This adds the device to the Experiment Devices list.

2) Set the Send and Read Commands: Select Send and Read from the Commands tool bar (at the top of the DDDA Main view) or right click on the blank command (the arrow points to it) in the Experiment View section and select Send and Read from the popup menu. You should see the Send and Read Editor View:



Send and Read

Device: COM1: sr830

Send | Read

Send

Terminating Characters: 13 0

Command To Send: *IDN?

□ Force a space between the command and the variable.

Variable To Send

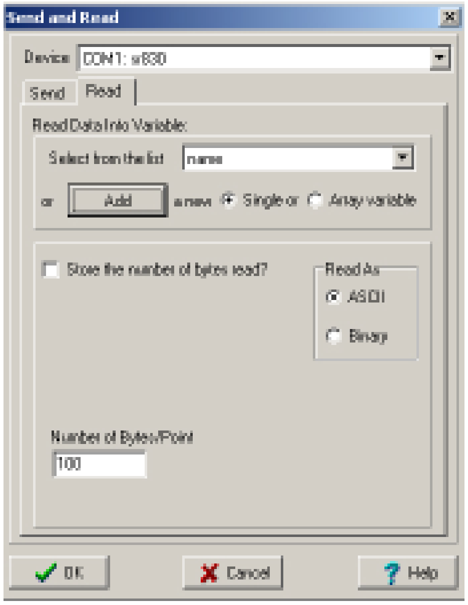
No Variable

or Add a ● Single or ○ Array

variable to send.

✓ OK ✗ Cancel ? Help

The device should be selected (since it is the only device in the experiment). Enter the command **IDN?* in the Send Text Box and *No Variable* should be selected in the Send Variable Combo Box (this is for sending the current value of a variable, see Send for more details on this feature). Select the Read tab and you should see:



Device: COM1: sr830

Read Data Into Variable:

Select from the list: name

or Add as new ● Single or ○ Array variable

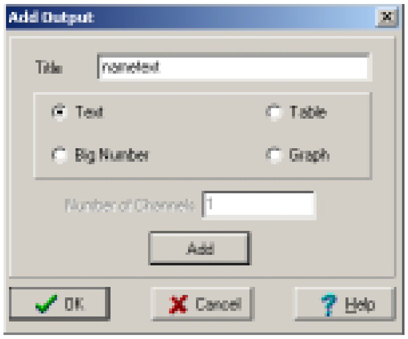
□ Store the number of bytes read?

Read As: ● ASCII ○ Binary

Number of Bytes/Point: 100

There should be no variables listed in the Read Combo Box (since there are no variables in the experiment yet) so select the Single radio button and then press Add. The Add Single form will appear and give the variable the name *name* and its type should be String (the default). Press OK and the *name* variable should appear selected in the Read box. Now the box below the Read box should appear. Keep Read As selected as *ASCII* and the Number of Bytes/Point to *100* . Press OK and this form will disappear and 2 commands will appear in the Experiment View under *Start Block*.

3) Create the Output: Double click on Outputs in the Experiment Attributes list box or right click on Outputs and select Add Output from the popup menu. You should see the Add Output view:



Title: nametext

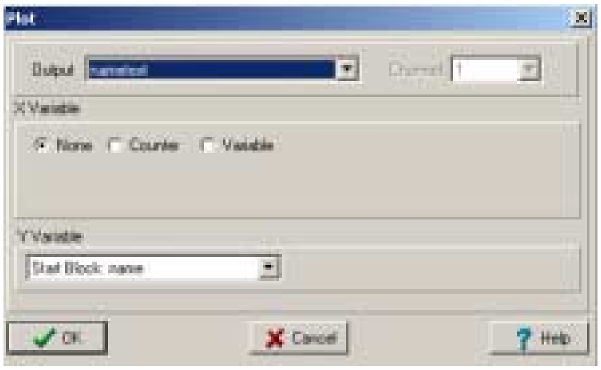
● Text ○ Table

○ Big Number ○ Graph

Number of Channels: 1

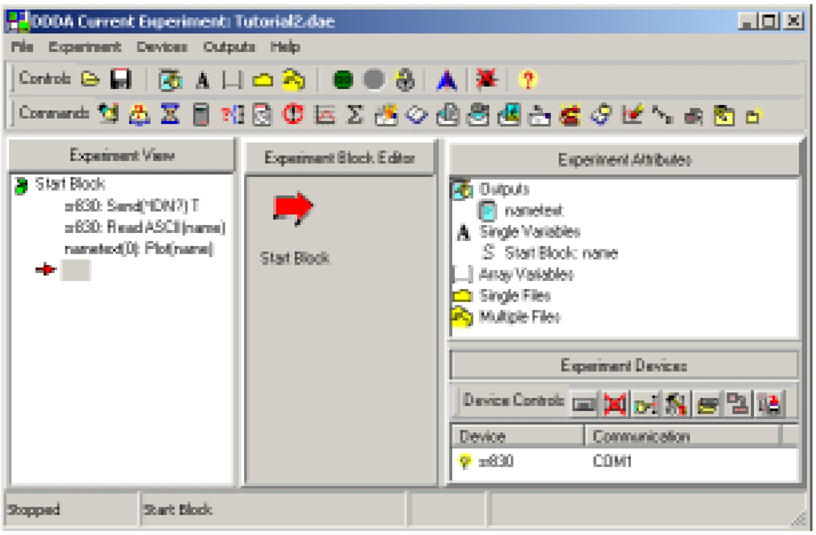
Give the output the name *nametext* and select Text as the type (if it is not already selected) and press OK. There should now be a Text output called *name* text under Outputs in the Experiment Attributes list box.

4) Add the Plot Command: Select Add Plot from the Commands tool bar (at the top of the DDDA Main view) or right click on the blank command in the Experiment View and select Plot from the popup menu. You should see the Plot Editor view:



Output: nametext
Channel: 1
X Variable: ● None ○ Counter ○ Variable
Y Variable: Start Block: name

The *nametext* output, XVariable None and YVariable *name* should already be selected. Press OK and the plot command will be added to the command list under the *Start Block* in the Experiment View. At this point, the DDDA Main form should look like:



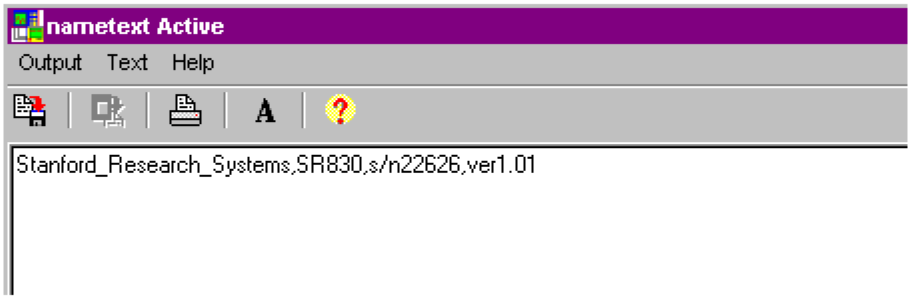
Start Block
sr830: Send(*IDN?) T
sr830: ReadASCII(name)
nametext(0): Plot(name)

Start Block

Outputs
nametext
Single Variables
Start Block: name
Array Variables
Single Files
Multiple Files

Device	Communication
sr830	COM1

Double click on the output *nametext*, under Outputs in the Experiment Attributes box, to show the output. Press the Go Go button on the Controls bar in the DDDA Main form and you should see the following (for your device) in the output *nametext*:



```
Stanford_Research_Systems,SR830,s/n22626,ver1.01
```

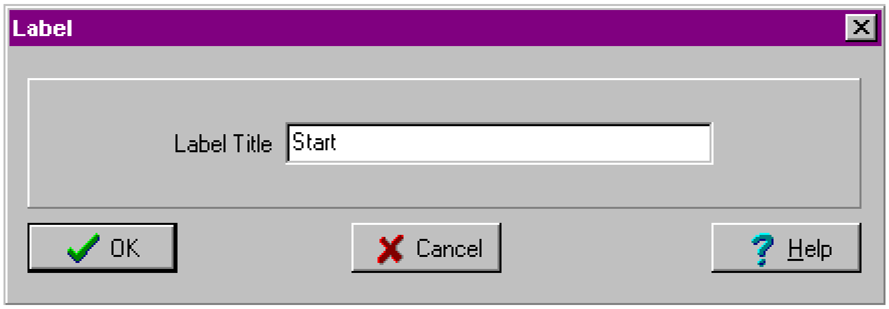
Tutorial 3: Parabola anyone?

In this tutorial you will learn how to

- Calculate a value using the calculator command.
- Use a Repeat Until command to control the number of times a series of commands is executed.
- Plot the calculated value to a graph output.
- Change the color of a trace in a graph output.
- Print and save a graph output.
- Save an experiment.

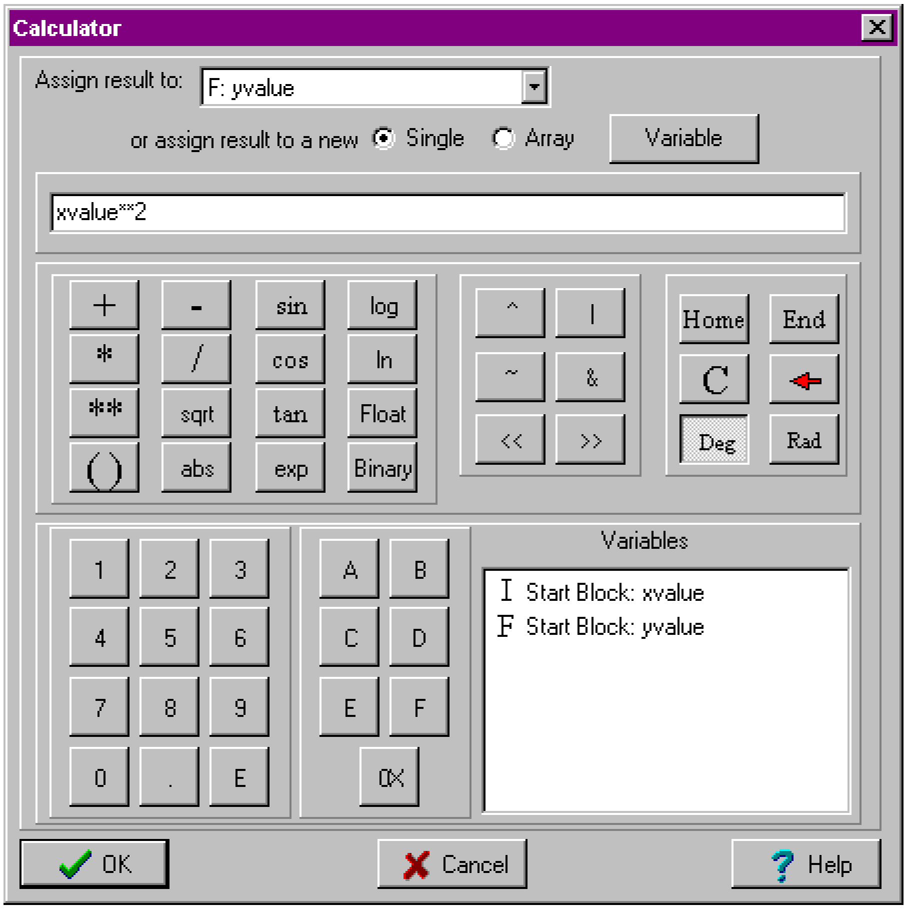
This tutorial will demonstrate the Repeat Until loop and the Calculate command by calculating a parabola and plotting it on a graph. No devices will be needed. Begin with a new experiment either by starting the program or by selecting File|New Experiment from the main menu in the Main DDDA form.
Add the following single variables (double click on Single Variables in the Experiment Attributes section):
xvalue: An integer variable with a starting value of -10;
yvalue: A float variable, no need to define a starting value.

Add the following output (double click on Outputs in the Experiment Attributes section):
Parabola: A graph output.
Add the following commands in order:
Label Command. Select Add Label from the Commands tool bar (at the top of the DDDA Main view) or right click on the blank command (the arrow points to it) in the Experiment View section and select Label from the popup menu. You will see the Label Editor View:


Label Title: Start

Enter *start* for the label title and press OK.

Calculate Command: Select Calculate from the Commands tool bar (at the top of the DDDA Main view) or right click on the blank command (the arrow points to it) in the Experiment View section and select Calculate from the popup menu. You will see the Calculate Editor View:



Calculator

Assign result to: F: yvalue

or assign result to a new ● Single ○ Array

xvalue**2

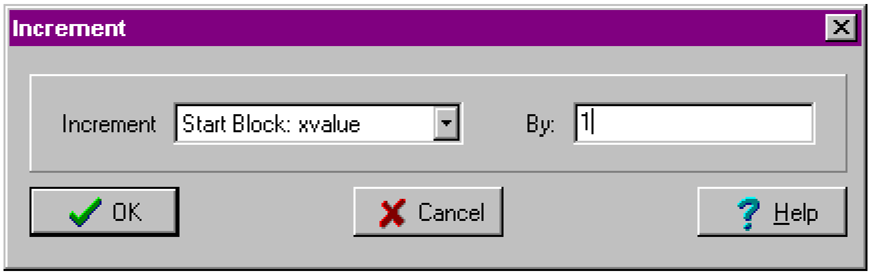
Variables

I Start Block: xvalue
F Start Block: yvalue

Assign the result to *yvalue*, enter *xvalue**2* in the Expression Editor and press OK.

Plot Command: Add a plot command using *Parabola* as the output (there should be no others in the list), *xvalue* as the XVariable and *yvalue* as the YVariable (See Tutorial 1 or Tutorial 2 for a more detailed explanation of adding a plot command)

Increment Command: Select Increment Σ from the Commands tool bar (at the top of the DDDA Main view) or right click on the blank command (the arrow points to it) in the Experiment View section and select Increment from the popup menu. You will see the Increment Editor View:



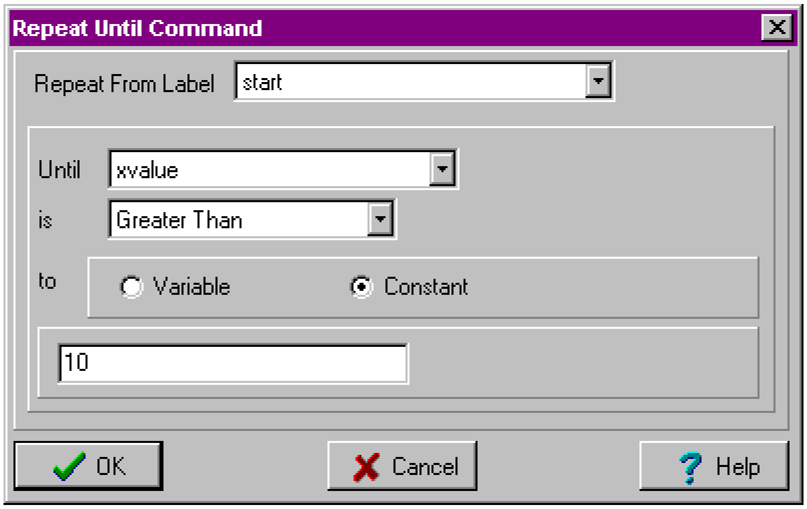
Increment

Increment: Start Block: xvalue

By: 1

Select *xvalue* as the variable to increment and enter *1* in the increment by edit box and press OK.

Repeat Until: Select Repeat Until from the Commands tool bar (at the top of the DDDA Main view) or right click on the blank command (the arrow points to it) in the Experiment View section and select Repeat Until from the popup menu. You will see the first page of the Repeat Until Editor View:



Repeat Until Command

Repeat From Label: start

Until: xvalue

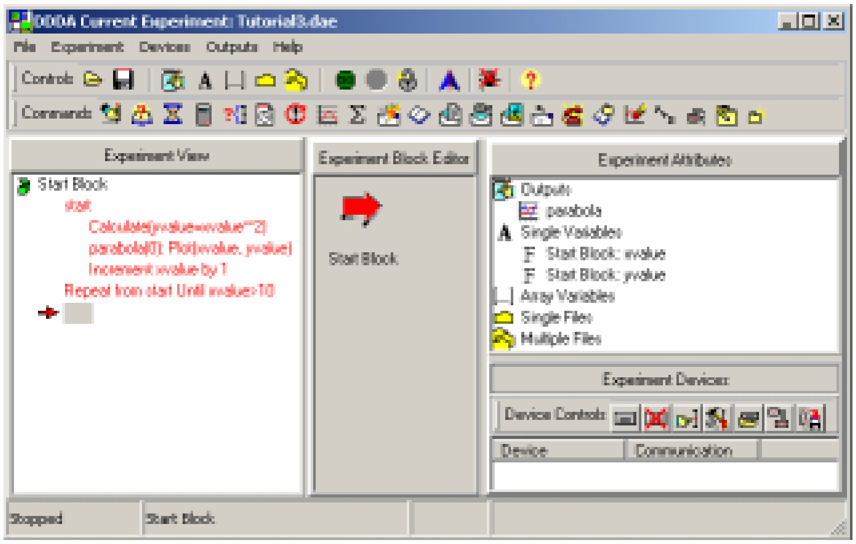
is: Greater Than

to: ○ Variable ● Constant

10

Select *Start* as the label to repeat from, then *xvalue* from the Until combo box, *Greater Than* from the Is combo box, select the *Constant* button, and then enter *10* in the edit box.   Press OK.

Once this is completed, the DDDA Main form should look like:



```
Start Block
    start
        Calculate(yvalue=xvalue^2)
        parabola[0]: Plot(xvalue, yvalue)
        Increment xvalue by 1
    Repeat from start Until xvalue>10
```

Start Block

Outputs
- parabola

Single Variables
- Start Block: xvalue
- Start Block: yvalue

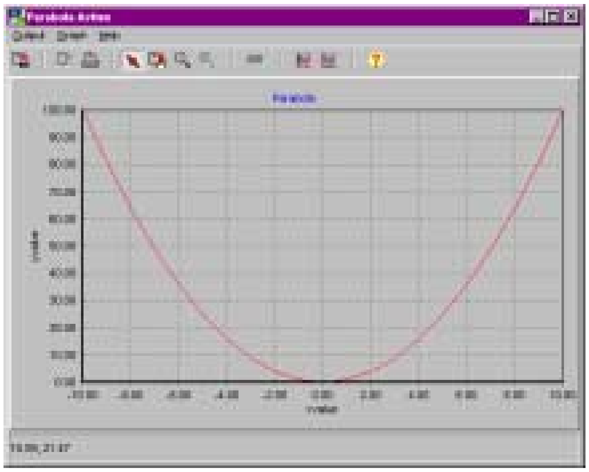
Array Variables

Single Files

Multiple Files

Stopped | Start Block

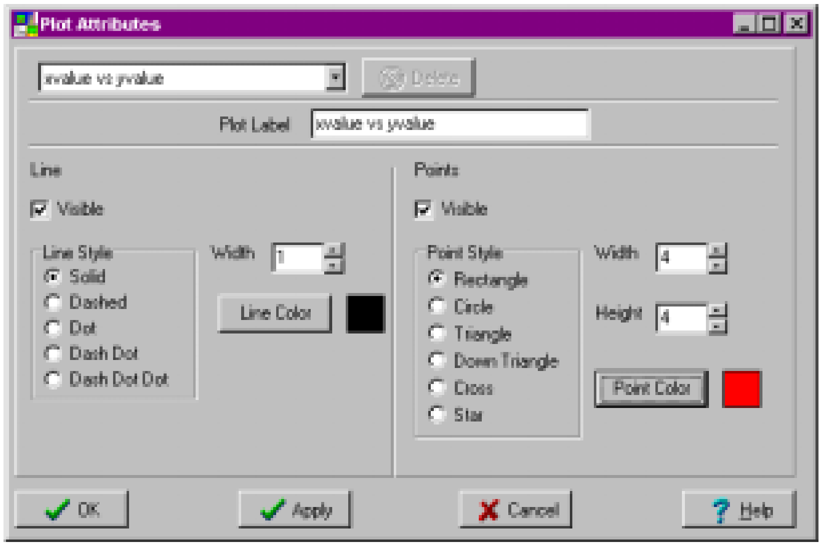
Double click on the output *parabola*, under Outputs in the Experiment Attributes box, to show the output. Press the Go Go button on the Controls bar in the DDDA Main form and you should see the following in the output *parabola*:



10.59, 21.47

Notice that two number appear at the bottom of the graph 10.59, 21.47. When you drag the cursor over the graph, the point (x,y) in the graph the cursor is over, appears here. To see the value of a particular data point in the graph, you have to click on the data point. The easiest way to see the points is to show them. How to do that is described next.

Changing the trace color: To change the trace color, select **Graph|Plot Attributes** from the *Parabola* output main menu. This shows the Plot Attributes View:



xvalue vs yvalue

Plot Label: xvalue vs yvalue

Line

☑ Visible

Line Style: ● Solid ○ Dashed ○ Dot ○ Dash Dot ○ Dash Dot Dot

Width: 1

Points

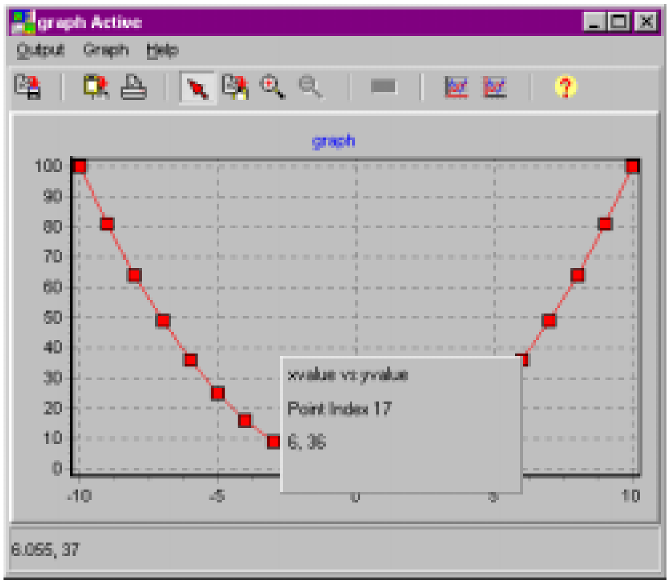
☑ Visible

Point Style: ● Rectangle ○ Circle ○ Triangle ○ Down Triangle ○ Cross ○ Star

Width: 4

Height: 4

The plot *xvalue vs yvalue* should already be selected as the plot to modify. Press **Line Color**. Select black from the color dialog box. Check the **Visible** box in the **Points** section. Everything in the **Points** section will now become enabled. Press **Point Color**. Select red from the color dialog box. Press **OK** and your *Parabola* graph will now look like:



xvalue vs yvalue

Point Index 17

6, 36

6.065, 37

Notice the box that says 6.00, 36.00. As stated above, if you click on a data point, the x and y values the point index and the series name will appear in a box near the point. This box will persist for a couple of seconds and then disappear.

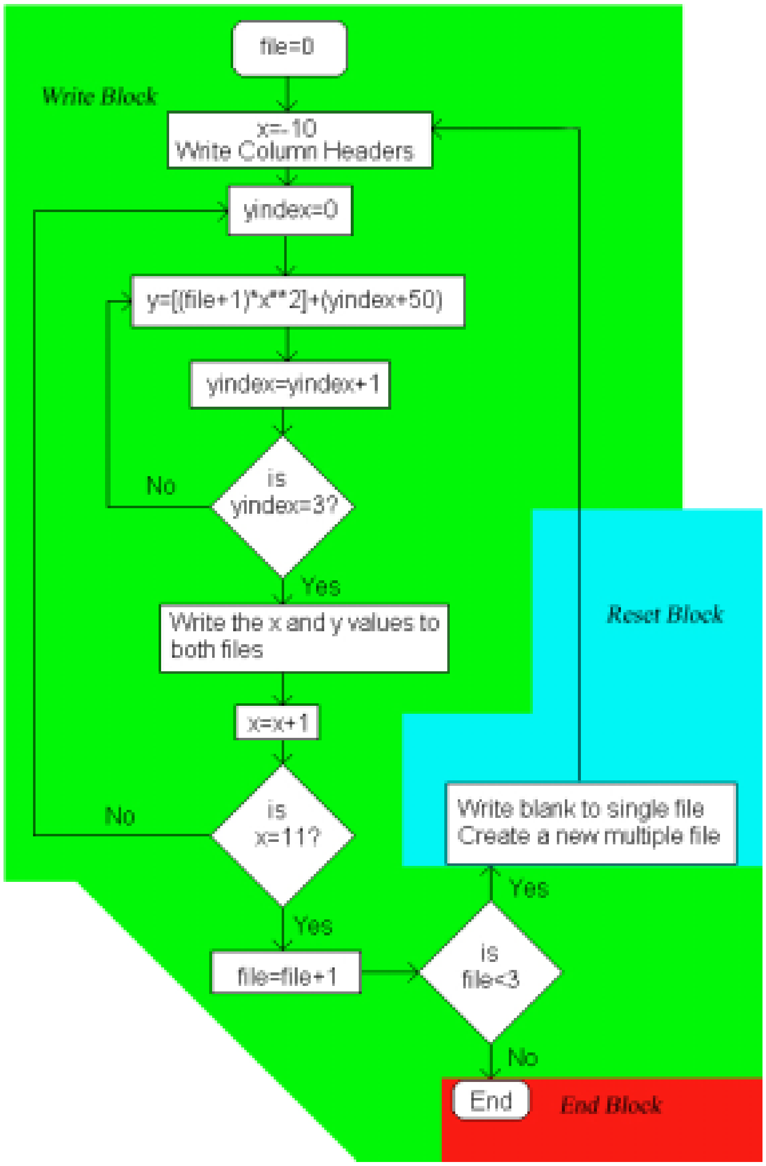
Printing the Plot: Select **Output|Print** from the *Parabola* output main menu. This shows the familiar windows print dialog box. Select which printer to print to and *Parabola* will be printed. If printing to a color printer, the colors will be preserved.

Saving the Plot: Select **Output|Save** from the *Parabola* output main menu. Enter a filename in the save dialog box. The file will be saved with the extension *.opt. This is a binary file that can be read by DDDA. All of the data as well as plot attributes, graph attributes, titles. etc. will be saved. To open a saved graph, select **File|Open|Output** from the DDDA Main Menu. Select a *.opt file using the open file dialog box. The opened file's filename will then appear as a menu item under **Output|Opened Outputs** in the DDDA Main menu. The output will not be active, in other words, it cannot be used in a plot command.

Saving the Experiment: Select **File|Save Experiment** from the DDDA Main Menu. Enter a filename in the save dialog box. The file will be saved with all the experiment blocks, variables, outputs and devices. The configuration of the output is saved with the experiment, but not the data.

Tutorial 4: Files and Parabolas (New in Version 1.5)

This is the most complicated of the Tutorials. It is a multiblock experiment that writes to a total of four files. In the first block, the column labels are set. In the remaining 3 blocks, the following is done:



Four files are written to, single.txt, multi_0.txt, multi_1.txt and multi2.txt. Single.txt contains 3 sets of data separated by blank spaces. Each of the multiple files contain one set of data.

The data should be as follows:

Multi_0.txt	Multi_1.txt	Multi_2.txt	Single.txt
X,y0,y1,y2	X,y0,y1,y2	X,y0,y1,y2	X,y0,y1,y2
-10,100,150,200	-10,200,250,300	-10,300,350,400	-10,100,150,200
-9,81,131,181	-9,162,212,262	-9,243,293,343	-9,81,131,181
-8,64,114,164	-8,128,178,228	-8,192,242,292	-8,64,114,164

-7,49,99,149	-7,98,148,198	-7,147,197,247	-7,49,99,149
-6,36,86,136	-6,72,122,172	-6,108,158,208	-6,36,86,136
-5,25,75,125	-5,50,100,150	-5,75,125,175	-5,25,75,125
-4,16,66,116	-4,32,82,132	-4,48,98,148	-4,16,66,116
-3,9,59,109	-3,18,68,118	-3,27,77,127	-3,9,59,109
-2,4,54,104	-2,8,58,108	-2,12,62,112	-2,4,54,104
-1,1,51,101	-1,2,52,102	-1,3,53,103	-1,1,51,101
0,0,50,100	0,0,50,100	0,0,50,100	0,0,50,100
1,1,51,101	1,2,52,102	1,3,53,103	1,1,51,101
2,4,54,104	2,8,58,108	2,12,62,112	2,4,54,104
3,9,59,109	3,18,68,118	3,27,77,127	3,9,59,109
4,16,66,116	4,32,82,132	4,48,98,148	4,16,66,116
5,25,75,125	5,50,100,150	5,75,125,175	5,25,75,125
6,36,86,136	6,72,122,172	6,108,158,208	6,36,86,136
7,49,99,149	7,98,148,198	7,147,197,247	7,49,99,149
8,64,114,164	8,128,178,228	8,192,242,292	8,64,114,164
9,81,131,181	9,162,212,262	9,243,293,343	9,81,131,181
10,100,150,200	10,200,250,300	10,300,350,400	10,100,150,200
			X,y0,y1,y2
			-10,200,250,300
			-9,162,212,262
			-8,128,178,228
			-7,98,148,198
			-6,72,122,172
			-5,50,100,150
			-4,32,82,132
			-3,18,68,118
			-2,8,58,108
			-1,2,52,102
			0,0,50,100
			1,2,52,102
			2,8,58,108
			3,18,68,118
			4,32,82,132
			5,50,100,150
			6,72,122,172
			7,98,148,198
			8,128,178,228
			9,162,212,262
			10,200,250,300*t*
			X,y0,y1,y2
			-10,300,350,400
			-9,243,293,343
			-8,192,242,292
			-7,147,197,247
			-6,108,158,208
			-5,75,125,175
			-4,48,98,148
			-3,27,77,127
			-2,12,62,112
			-1,3,53,103

			0,0,50,100
			1,3,53,103
			2,12,62,112
			3,27,77,127
			4,48,98,148
			5,75,125,175
			6,108,158,208
			7,147,197,247
			8,192,242,292
			9,243,293,343
			10,300,350,400

The best way to understand what is going on in this tutorial is to try different file properties to see what happens to the output.

Chapter 8 Frequently Asked Questions

When running in Windows 95, the program comes up just fine, but there are no pictures on the buttons! Why and how do fix it?
Between versions of Windows 95, Microsoft changed some things with their common controls dll. To fix this, you need to update your comctrl32.dll. The safest way to do this is to go directly to the Microsoft site and download the fix:
http://www.microsoft.com/msdownload/ieplatform/ie/comctrlx86.asp
We have done this with several computers with no problems.

When running in Windows 95, the program looks fine, but when I pass the cursor between regions in the main form, the cursor disappears? How do I fix this?
There seems to be a problem with the common controls dll for Win95 and the splitter cursor. For some reason, when the cursor passes over a a place that allows you to change the size of an area, the cursor disappears. It still works, if you drag the invisible cursor, the area will resize.

I just installed the program in Windows NT and it doesn't run. What's wrong?

You need to restart the computer before attempting to run DDDA on Windows NT.

I just installed and ran DDDA and I can't open any saved experiment files, blocks or devices. What is wrong?
Check your keyboard settings. DDDA MUST have the keyboard set to English (US). Periods must be interpreted as decimal points, and commas must be interpreted as thousands delimiters.

A CEC GPIB card is installed on the computer, however, when in the Device Editor, both a National and CEC GPIB communication interfaces are present. Why? Can I use either communication interface?
The answer to the last question, can I use either communication interface, is NO! Don't do that. The confusion arises because CEC cards can double as National cards so if you are using programs that need a National Card, the CEC card will work. DDDA explicitly addresses each card using the manufacturer designed interface so we are not using that CEC feature.
There are 2 solutions to this problem:

1) There is a dll called gpib-32.dll which is located in the Windows or Windows\System directory of your computer. If you quit DDDA, and then rename gpib-32.dll to something like gpib-32.old and then restart DDDA only the CEC communication interface will show up. If you are using programs which explicitly need a National GPIB card to work, then don't do this (you will break those programs), instead choose option number 2....
2) In DDDA Device Editor, select **GPIB** from the **Advanced** Menu (at the top of the form). Uncheck **Automatic** and enter *0* for **Number of National Cards** and *1* for **Number of CEC Cards**. Press **OK**. The National communication interface should disappear.

My Read command keeps timing out when I run the Experiment but it works just fine when I use the Debugger, what's wrong?
Some devices, especially older ones, assume a certain amount of time between send and read requests. DDDA is a fast program, especially when running on one of today's machines. As a result, the send and read requests may be coming too rapidly for the device. If the program runs in the Debugger, then this almost surely the problem. The solution is simple, put a Wait command between the Send and Read commands. Extend the amount of the wait until the problem goes away.

I see that I can have more than one experiment block in an experiment, but why would I ever need more than one?

You might not. However, you may have a set of commands that you always have to do. In that case, if you put all those commands into one block, you can just save the block. Then whenever you create a new experiment, you just add that block to it, without having to write all the commands again. For example, the instruments you are using each have an initialization process If you put all those commands into one block and save it, whenever you write a new experiment, you just add that block as the Start Block, no initialization commands to rewrite.

What is a BigNum output?
A BigNum output is just a way of monitoring a particular variable. It is shown in large size so it can be easily seen by the investigator. In addition, data output to a BigNum can be directly logged to a file. Please see BigNum Log To File for more information about this feature.

I have read in binary data and now I need to change it to a signed integer value. There are two data bytes, with the data in bits 1-11 and the sign at bit 12. What do I do?
There are five steps to this:

1) Get the Data: Create a read command where the data is read into an integer variable as binary (Read As selection to Binary in the Read Editor). Set the Advanced Property Number of Bytes to 2 and set how the bytes are laid out (left to right or right to left). For the rest of this example, the variable holding the read data will be called RawData.
2) **Get the Sign**: Create a calculate command. Assign the end result to an integer (called sign for the rest of this example) . To determine what is in bit 12, we need to AND the value with the hex value 0x800 (1000 0000 0000 in binary). If the result is 0, then bit 12 is 0 and the value is positive. If the result is 1, then bit 12 is 1 and the value is negative. In the Expression Editor enter the following:

float(binary(RawData) & 0x800)
This will change the integer value of RawData in to binary value and then AND it with the binary value of 0x800. The result will be changed to a float and then assigned to the integer variable sign. sign will be either a 0 or a 1.

3) **Get rid of the sign bit**: Create another calculate command. Assign the end result to the integer you read in, RawData for this example. To change what is in bit 12 to a 0, we need to AND the value with the NOT of 0x800 (0111 1111 1111 in binary). In the Expression Editor enter the following:

float(binary(RawData) & (~0x800))
This will change bit 12 to a zero, regardless of what it started as, a 0 or 1.

4) **Change the sign:** Create an if then command. Set it up to set for sign being greater than 0 (not greater than or equal to, be careful). You do not need an else.
5) **Set the sign if necessary**: In the if loop, add a calculate command. This command will only be carried out if sign is 1, therefore, the value should be negative. Assign the end result to Raw Data. In the Expression Editor enter the following:

0-RawData

When you have completed the four steps, you should have the following commands in the Experiment View:

```
sr245: Read(RawData)
Calculate(sign=float(binary(RawData)&0X800))
Calculate(RawData=float(binary(RawData)&(~0X800)))
If sign>0
   Calculate(RawData=0-RawData)
End If
```

I have selected Show Current Value for a variable, yet when I run the experiment, the value showing in the Current Value list doesn't change. What's wrong?
In short, nothing is wrong. Show Current Value does not trace the current value of the variable throughout the experiment. If you want to trace the value of a variable during an experiment, you need to use the Debugger and select Trace from the variable popup in the Debug View. Show Current Value, does exactly

what it says it does, it shows you the current value of the variable when you select this menu item from variable popup menu in the Main View.

Can I write to file while I am taking data?
Yes, but only when using the BigNum output.   Please see BigNum Log To File for more information about this feature.

How many data points can I collect in the various outputs?
That depends on how much memory you have on your computer and on the output. The more memory, the more points. The outputs dependencies are as follows:
The BigNum output has no limit since it only takes one data point at a time. If logging to file is enabled, then the data points can be saved.
The Text output has the smallest limit since it can plot both strings, integers and floats. It is recommended that you keep the total number of points output to a Text to less than 10,000.
The Graph and Table outputs have been known to accept up to 1,000,000 points. Remember, please, the more points, the slower the graph will be drawn.
The windows clipboard only holds 64K. Therefore the amount of copying you can do depends on that limit.

Can I get a printout of my experiment? How?
Yes you can. Select Experiment|Experiment Report from the DDDA Main Menu.

How can I output a blank line to a Single or Multiple file?
Create a string Single Variable and leave the initial value blank. Then add a Output To File command using that string variable. That will result in a blank line when the command in run in the experiment. See tutorial 4 for an example.

Glossary

Array Variable: An array variable is an experiment variable that holds more than one value. Each value in the array has a unique array index, beginning with zero. When creating or editing a command that uses variables, if an array is selected from the list box, a combo box will appear to the right of the array name. The combo box lists all possible integer variables that can be used as indices as well as a list of possible integer indices. Choosing one of these values sets the index for the array. Be aware that if the size of the array is changed at a later time and the index assigned here is no longer valid, an invalid index error will occur when the experiment is run.

Blank Command: The rectangle with an arrow pointing to it in the Experiment View. There will be only one in any experiment and it will disappear when the experiment is run. It indicates where new commands will be added when added using the Commands speed bar at the top of the DDDA Main View. It has a command popup (further down in the Glossary) menu associated with it.

Command: One of the predefined DDDA commands.

Command Popup: If you right click on any command but the Blank Command, a popup will appear that allows you to edit, delete, move, etc., the command. In addition there is an Insert menu item where other commands can be inserted above the one that calls the popup menu. The Blank Command has a popup that allows you to move the Blank as well as add a particular command where the Blank Command resides.

Communication Interface: GPIB or RS232.

Device: An electronic instrument that has been recognized by DDDA, in other words it appears in the Experiment Devices list (see DDDA Main View).

Device Popup: Right click on any device in the Experiment Devices list to see this popup.

Device Address: An RS232 port or GPIB address which the device uses to communicate with the program.

Device Configuration: A set of devices.

Experiment: A self contained set of experiment components assembled by the user (see Introduction). It is what DDDA "runs".

Index Combo Box: This combo box appears in any view where a single element in an array variable can be chosen The drop down list will contain any single variables already defined in the experiment that can be used as an index. You can also enter the index number (only integer values are allowed) or the name of the single integer variable into the text box. If the index is a number that is greater than the size of the array or is less than zero, an error will occur when you try to exit the dialog box using the Ok button. Use caution when assigning a single variable as the array index. If the value of the array index at run time is greater than the size of the array or less than zero, a fatal run time error will occur (fatal meaning that the experiment will abort).

Loop Rules:

1) Label commands cannot be shared, each repeat command must have its own Label.
2) Loops can be nested (loop inside a loop).
3) Loops cannot cross.
4) The label command must precede the repeat command and must be in the same block.

Multiple File: A text file that can be output to during the course of an experiment. Its filename can change during the experiment.

No Device Popup: Right click anywhere in the Experiment Devices BUT on a device list to see this popup.

Output: A graphical representation of the data. All variables appear in the Experiment Attributes list.

Parent Block: The experiment block that a particular command was created in and will be executed in.

Variable: The data represented as a value or array of values. All variables appear in the Experiment Attributes list.

Single File: A text file that can be output to during the course of an experiment. Its filename does not change during the experiment.

Single Variable: An experiment variable that holds only one value.

Index