QUATTRO™

Reference Guide

QUATTRO™: THE PROFESSIONAL SPREADSHEET

Borland's No-Nonsense License Statement!

This software is protected by both United States copyright law and international treaty provisions. Therefore, you must treat this software *just like a book*, with the following single exception. Borland International authorizes you to make archival copies of the software for the sole purpose of backing-up our software and protecting your investment from loss.

By saying, "just like a book," Borland means, for example, that this software may be used by any number of people and may be freely moved from one computer location to another, so long as there is **no possibility** of it being used at one location while it's being used at another. Just like a book that can't be read by two different people in two different places at the same time, neither can the software be used by two different people in two different places at the same time. (Unless, of course, Borland's copyright has been violated).

WARRANTY

With respect to the physical diskette and physical documentation enclosed herein, Borland International, Inc. ("Borland") warrants the same to be free of defects in materials and workmanship for a period of 60 days from the date of purchase. In the event of notification within the warranty period of defects in material or workmanship, Borland will replace the defective diskette or documentation. **If you need to return a product, call the Borland Customer Service Department to obtain a return authorization number.** The remedy for breach of this warranty shall be limited to replacement and shall not encompass any other damages, including but not limited to loss of profit, and special, incidental, consequential, or other similar claims.

Borland International, Inc. specifically disclaims all other warranties, expressed or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose with respect to defects in the diskette and documentation, and the program license granted herein in particular, and without limiting operation of the program license with respect to any particular application, use, or purpose. In no event shall Borland be liable for any loss of profit or any other commercial damage, including but not limited to special, incidental, consequential or other damages.

GOVERNING LAW

This statement shall be construed, interpreted, and governed by the laws of the state of California.

First Edition
Printed in U.S.A.
9 8 7 6 5 4 3 2 1

QUATTRO®

Reference Guide

This manual was produced in its entirety with Sprint®: The Professional Word Processor, available from Borland.

Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066

10 9 8 7 6 5 4 3 2 1

Table of Contents

How to Use This Manual

This reference guide is an alphabetical look-up intended for quick reference to Quattro; it assumes a general knowledge of Quattro procedures and concepts. Once you're well-acquainted with Quattro, this will probably be the book you keep by your computer and refer to the most.

This book is divided into four chapters:

- **Chapter 1, "Quattro Menu Commands,"** briefly describes each of Quattro's menu commands and how to use them. The chapter is divided into six major categories for easy reference: Installation, Working with the Spreadsheet, Files, Printing, Graphs, and Macros. Within these categories, commands are listed alphabetically. The fold-out menu trees on the inside front cover of this manual include page numbers showing where each command is covered. Each section includes abbreviated, step-by-step information about the command. More detailed information is given in the *Quattro User's Guide.*
- **Chapter 2, "@Function Commands,"** gives a detailed description of each @function command. The commands are listed alphabetically, and a table at the beginning lists them by group and gives brief descriptions and syntax.
- **Chapter 3, "Macro Commands,"** gives detailed descriptions of all macro commands, special commands used in Quattro macros. The commands are listed alphabetically, with a table summarizing them by group at the beginning of the chapter.
- **Chapter 4, "Menu-Equivalent Commands,"** defines the commands used to represent menu commands in macros and transcripts. It includes three tables. The first defines the commands and gives their menu equivalents. The second arranges the commands by Quattro menus. The third arranges the commands by Lotus 1-2-3-compatible menus.

Trademarks

Within this book, references are made to the following products:

- Compaq is a registered trademark of Compaq Computer Corporation.
- dBASE is a registered trademark of Ashton-Tate.
- Epson is a registered trademark of Epson Corporation.
- Hercules is a registered trademark of Hercules Computer Technology.
- IBM, PC/XT, XT, AT, Proprinter Portable, and 3270 are registered trademarks of International Business Machines Corp.
- Lotus 1-2-3 and Symphony are registered trademarks of Lotus Development Corp.
- Microsoft and MS-DOS are registered trademarks of Microsoft Corporation.
- Paradox is a registered trademark of Ansa Software.
- Postscript is a registered trademark of Adobe Systems.
- Reflex is a registered trademark of Borland/Analytica, Inc.
- SideKick and Sprint are registered trademarks of Borland International, Inc.
- Wordstar and MailMerge are registered trademarks of MicroPro International Corporation.

C H A P T E R

1

Quattro Menu Commands

This chapter describes each of the major Quattro commands and operations, from saving a file to loading a different menu tree. Each section briefly describes the function performed by a command and gives the step-by-step procedure for using it. For detailed information on any of the operations covered here, refer to the *Quattro User's Guide.*

This chapter is divided into six major sections:

- **Installation** covers loading Quattro and several defaults that you will probably need to set only once, such as spreadsheet colors and currency symbol.
- **Working with the Spreadsheet** covers Quattro commands that have to do with general spreadsheet work, from inserting a column to performing regression analysis. It also tells you how to work with Quattro add-ins.
- **Files** covers commands that deal with data files, from retrieving a spreadsheet file to parsing an imported text file.
- **Printing** covers all spreadsheet print commands. (Graph printing is covered in the *Graph* section.)
- **Graphs** covers all graphing commands, including printing a graph.
- **Macros** covers basic macro functions, such as recording, executing, and deleting.

Within each category, operations are listed alphabetically, either by command name or by function. For example, "Label Alignment" and "Erase a Block of Data" are under *Working with the Spreadsheet.* When there's a question about how an item should be listed, it is cross-referenced.

This organization makes it easy to look up an operation you want to perform, even if you don't know the name of the command used to perform it.

If you want to look up the function of a specific menu command, use the menu tree on the inside front cover. Next to each major command is listed the page on which it is covered.

Installation

Quattro requires very little installation. The type of screen you're using is detected automatically. If you're using a floppy-drive system, you need to use a simple command that sets Quattro's directory defaults for floppy disks.

There are also several defaults that you may want to set when you first start the program, such as printer specifications. This section includes information on those defaults and also tells you how to load and install Quattro.

For more details on what to do before loading Quattro for the first time (such as copying the disks), see Chapter 2 of *Getting Started with Quattro.*

Alternate Menu Tree

See "Menu Trees" on page 21.

Autoload Defaults for Installation

Each time you load Quattro, it looks for a file with the default autoload name (initially QUATTRO.WKQ). If found, it retrieves it automatically. You can change the name of the file that Quattro looks for with the **D**efault **S**tartup **A**utoload File command (see "Autoload File" on page 72).

You can also specify up to eight add-ins to be loaded automatically when you start Quattro (see "Default Add-Ins" on page 30).

Whenever you retrieve a file, Quattro looks for another autoload default—a macro with the name specified as the startup macro. Initially, the macro name is \0, but you can change it with the **S**tartup Macro command on the Default Startup menu (see page 125).

Automatic Versus Manual Recalculation

See "Recalculation Defaults" on page 26.

Beep

Whenever Quattro detects a mistake in your work, it causes the computer to beep. You can turn off this beep, if you find it annoying, with the **Beep** default.

To set the **Beep** default:

1. **Press** */DSB* to select **Beep** from the Default Startup menu.
2. **Select** ***No*** to keep Quattro from beeping on errors. **Select Yes** to return error beeps.
3. If you want the new setting to be permanent, **press** *Esc* to return to the Default menu and **select** ***U*pdate**.

Clock Display

The status area at the bottom of your Spreadsheet Screen includes the current date and time. Initially, the date and time are displayed in the Standard clock format (DD-MMM-YY and HH:MM AM/PM). You can change this to display date and time in the current Long International format (initially MM/DD/YY and HH:MM:SS) or, if you prefer, remove the date and time display.

To change the spreadsheet's date and time display:

1. **Press** */DFC* to select **Clock** from the Default Formats menu.
2. **Select an option** from the displayed menu. **International** displays the date and time in the Long International format; **None** removes them from display.
3. **Press** *Esc* to exit the Default Formats menu.
4. **Press** *U* to select **U**pdate from the Default menu.

The date and time are immediately displayed in the new format.

You can change the date and time to formats that are standard for other nations with the Default International command (see "International Defaults" on page 17).

For more details, see "Default Clock Display" in Chapter 5 of the *Quattro User's Guide.*

Colors

If you have a color monitor, you'll find that Quattro uses a variety of colors to display your spreadsheet. You can alter the default colors with the Default Colors command. If you have a monochrome monitor, you can use this command to customize the spreadsheet display, using normal, bold, underlined, and reverse video characters.

When you select the Default Colors command (*/DC*), Quattro displays a menu listing four different areas you can change the colors of:

- **Menu** affects the colors of menus.
- **Spreadsheet** affects the colors of the overall spreadsheet, such as the indicators and borders.
- **Conditional** affects the colors of data in the spreadsheet. For example, you can select different colors for labels or ERR cells. You can even specify different colors for values that fall within, above, or below a specified range.
- **Help** affects the colors in the help screens that are displayed when you press *F1*.

A fifth menu choice, Palettes, lets you reinstate the factory default colors for either a color or monochrome screen.

After you've selected the colors you want, select Update from the Default menu (*/DU*) to make the changes permanent.

The following subsections describe how to change the colors used to display each of these areas, and how to reinstate default colors.

To change the colors of a displayed graph, use the Graph Customize Colors command (see "Colors of a Graph" on page 89). To change the colors of a printed graph, use the Graph Print Colors command (see "Print a Graph" on page 109).

Conditional Colors

The Default Colors Conditional command (/DCC) lets you change the colors used to display different types of data, such as labels and ERR values. You can also specify a range of "normal" values. Values in the spreadsheet above or below that range can be displayed in different colors.

To vary the colors used for values outside and inside a certain range, you first need to set up a conditional range:

1. **Select Smallest Normal Value** from the Conditional Colors menu (*/DCCS*).
2. **Enter the lowest value** you want included in the range.
3. **Select Greatest Normal Value** from the Conditional Colors menu (press *G*).
4. **Enter the highest value** you want included in the range.

To change the colors used to display conditional values:

1. **Select On/Off** and choose Enable.
2. **Select the type of value** you want affected from the Conditional Colors menu.
3. **Select the color combination** you want to use from the displayed color palette (or menu, if you have a monochrome screen).
4. To change the color used for another type of data, select the type from the Conditional Colors menu and choose a color from the palette or menu.
5. If you want to use the new colors next time you work with Quattro, **press *Esc* twice and press *U*** to select Update from the Default menu.

The new conditional colors are used immediately.

For more details, see "Conditional Colors" in Chapter 5 of the *Quattro User's Guide.*

Help Colors

The Default Colors Help command *(/DCH)* affects the colors of help screens displayed when you press *F1.*

To change any of the help colors:

1. **Press** */DCH* to select Help from the Default Colors menu.
2. **Select the particular area** you want to adjust.
3. **Select the color combination** you want to use from the displayed color palette (or menu, if you have a monochrome screen).
4. **Press** *Enter.*
5. To adjust other help colors, repeat the steps above.
6. If you want to use the new colors next time you work with Quattro, **press *Esc* twice and press *U*** to select Update from the Default menu.

The new help colors are used immediately.

For more details, see "Help Colors" in Chapter 5 of the *Quattro User's Guide.*

Menu Colors

The **D**efault **C**olors **M**enu command (*/DCM*) affects the menu colors.

To change any of the menu colors:

1. **Press** */DCM* to select **M**enu from the Default Colors menu.
2. **Select the particular area** you want to adjust.
3. **Select the color combination** you want to use from the displayed color palette (or menu, if you have a monochrome screen).
4. To adjust other menu colors, repeat the steps above.
5. If you want to use the new colors next time you work with Quattro, **press** *Esc* **twice and press** *U* to select **U**pdate from the Default menu.

The new menu colors are used immediately.

For more details, see "Menu Colors" in Chapter 5 of the *Quattro User's Guide.*

Palettes—Reinstating Default Colors

After you've made changes to any of the colors used, you can reinstate the original default colors for either monochrome or color screens.

To reinstate default colors:

1. **Press** */DCP* to select **P**alette from the Default Colors menu.
2. **Select your screen type. M**onochrome uses two colors (such as black and green) plus special effects, such as highlighting and reverse video. **C**olor uses the range of colors available with your screen.
3. If you want to use the default colors from now on (and erase any changes you made with the Colors menu), **press** *Esc* **and select** ***U*pdate** from the Default menu.

Note: If you have a black and white monitor with a color graphics card, you may want to switch to the monochrome palette for a better display.

For more details, see "Reinstating Default Colors" in Chapter 5 of the *Quattro User's Guide.*

Spreadsheet Colors

The **D**efault **C**olors **S**preadsheet command (/DCS) allows you to change the color scheme of your spreadsheet.

To change spreadsheet colors:

1. **Press** */DCS* to select Spreadsheet from the Default Colors menu.
2. **Select the particular area** you want to adjust.
3. **Select the color combination** you want to use from the displayed color palette (or menu, if you have a monochrome screen).
4. To adjust other spreadsheet colors, repeat the steps above.
5. If you want to use the new colors next time you work with Quattro, **press** *Esc* **twice and press** *U* to select **U**pdate from the Default menu.

The new color combination is immediately reflected in the spreadsheet.

For more details, see "Spreadsheet Colors" in Chapter 5 of the *Quattro User's Guide.*

Compatibility Defaults

Quattro offers several options that allow for compatibility with other products.

To set compatibility defaults, select **C**ompatibility Options from the Default Startup menu (*/DSC*). This displays a menu that lets you:

- **Change the way menus work:** whether or not Quattro highlights the last item you selected in a menu and whether or not Quattro includes settings in the menus (see page 22).
- **Change when Quattro asks for confirmation:** any time you erase the spreadsheet or exit Quattro, or only when you might lose data by doing so (see page 11).
- **Change how macros are recorded:** either as exact keystrokes or as menu-equivalent commands that can be used with any menu tree (see page 21).

For more details, see "Setting Compatibility Options" in Chapter 5 of the *Quattro User's Guide.*

You can also move the descriptor line at the bottom of the screen to the top, if this is the layout you're used to. Just select **D**escriptor Line from the Layout menu (*/LD*). (See "Descriptor Line" on page 13 for details.)

Confirmation Defaults

Normally, if you select **E**rase (to clear the spreadsheet) or **Q**uit (to exit Quattro), and you have made changes to the spreadsheet that haven't been saved, Quattro displays a confirmation menu, asking if you're sure you want to do that. You can then cancel the command and save your work, if you want.

Lotus 1-2-3 displays confirmation prompts any time you use one of these commands, regardless of whether or not there is data that will be lost. If you like, you can tell Quattro to ask for confirmation as Lotus does (rather than as Borland does). (Quattro also confirms when you reset block names, and Lotus does not.)

To change when Quattro asks for confirmation:

1. **Press** */DSCB* to select **B**orland Style from the Default Startup Compatibility Options menu.
2. **Press** *N* to select **N**o.
3. If you want to save the new confirmation setting, **press** *Esc* to return to the Default menu and **select *U*pdate.**

To return to Quattro's original style of confirmation, set the **B**orland Style confirmation default to **Y**es.

Currency Format

See "International Defaults" on page 17.

Date: International Default

See "International Defaults" on page 17.

Defaults

There are several defaults that you may want to change when you first begin working with Quattro:

- **Beep** determines whether or not Quattro beeps when you make a mistake (see page 6).
- **Clock Display** determines how the date and time are displayed at the bottom of your screen (see page 6).
- **Colors** determines the colors used to display your spreadsheet, menus, values, and help screens (see page 7).
- **Confirmation** determines when Quattro displays confirmation menus (see page 11).
- **Currency** determines the format for displaying currency (see page 17).
- **Descriptor Line** lets you move the descriptor line from the bottom of the screen to the top, or vice versa (see page 13).
- **Directories** determines the directories that contain your Quattro program and data files (see page 13).
- **Extension** lets you assign a different file-name extension to be used as the default when none other is specified (see page 14).
- **Help Access Method** lets you speed up access to the Quattro help file if you have a hard disk (see page 16).
- **International** lets you change the international format used for dates, time, and punctuation (see page 17).
- **Keep Wide** determines whether Quattro displays settings on all menus or just the ones you expand (see page 23).
- **Macro Recording** lets you specify recording exact keystrokes as macros, instead of commands (see page 21).
- **Menu Trees** lets you load optional menu trees for Quattro to use (see page 21).
- **Printers** lets you specify special information about the printer(s) you're using (see page 24).
- **Recalculation** determines recalculation mode, order, and number of iterations performed (see page 26).
- **Remember** determines whether Quattro highlights the first options on menus, or the one you used last (see page 23).
- **Screen** lets you specify special information about the screen you're using (see page 28).

Once you've changed the settings, you can store the new values in the Quattro defaults file (QUATTRO.DEF) by selecting **U**pdate from the Default menu.

For detailed information on defaults, see Chapter 5 of the *Quattro User's Guide.*

Descriptor Line Display

Initially, the Quattro descriptor line is displayed at the bottom of the Spreadsheet Screen. You can reposition it at the top of the screen, if you prefer.

To reposition the descriptor line:

1. **Press** */LD* to select **D**escriptor Line from the Layout menu. The line is immediately moved to the opposite side of the screen.
2. To make the new position permanent, **select *U*pdate** from the Default menu (press *Esc,* then *DU*).

Directory Defaults

Quattro recognizes two different types of directories: data and resource. The data directory contains your spreadsheet files. The resource directory contains program files found on your Resource Disk.

When you first load Quattro, both default directories are set to the disk drive and/or directory you've loaded the program from. You can assign different locations to either directory.

To specify new default directories:

1. **Press** */DD* to select **D**irectories from the Default menu.
2. **Select *D*ata** if you want to change the data directory.
3. **Enter the location** of the directory containing most or all of your Quattro spreadsheet files.
4. **Select *R*esource** if you want to change the directory containing your program resource files.
5. **Enter the location** of your Quattro resource files.
6. **To save the new directories** as defaults, press *Esc,* then select **U**pdate from the Default menu.

Note: The resource directory refers to the location of the files that came on your Quattro Resource Disk. This disk contains files needed to access graph printing and help information. If you're using a system with two floppy drives (and no hard disk), you'll probably want to set this directory to Drive B. You can then keep the Resource Disk in Drive B and use Drive A to load the program (with the System Disk), and then to access spreadsheet files (replacing the System Disk with a work disk).

For more information on default directories, see "Directory Defaults" in Chapter 5 of the *Quattro User's Guide.*

You can always override the default data directory by specifying a path name with the file you're accessing. If you want to do several file operations within a directory that is different from the default, you can specify a "temporary default" with the **D**irectory command on the File menu (*/FD*). This command temporarily resets the default data directory.

To specify a temporary data directory default:

1. **Select *D*irectory** from the File menu (*/FD*).
2. **Enter the path name** of the directory you want to access temporarily.

Quattro now assumes the files you want are in the temporary directory, unless you specify otherwise. This directory remains in effect until you exit Quattro or specify a different directory with the **D**efault **D**irectories **D**ata command (*/DDD*).

For more information on using a temporary directory, see "Specifying a Temporary Data Directory" in Chapter 6 of the *Quattro User's Guide.*

File-Name Extension

Quattro initially uses the file name extension .WKQ for spreadsheets. When you save a spreadsheet, it automatically tacks on this extension (unless you specify a different one). When you retrieve a spreadsheet without specifying an extension, it looks for files with the .WKQ extension. You can change the default extension used for spreadsheets to anything you like.

To specify a different default file-name extension:

1. **Press** */DSE* to select **E**xtension from the Default Startup menu.
2. **Enter a three-letter extension** preceded by a period.
3. **If you want to use the new extension** from now on, press *Esc* and select **U**pdate from the Default menu.

Quattro assumes the new default extension immediately.

If you retrieve a file with any of the following extensions, it will be translated automatically into Quattro spreadsheet format:

.DB	Paradox
.DBF	dBase
.WRK	Symphony
.WK1	Lotus 1-2-3, version 2.0
.WKS	Lotus 1-2-3, version 1A

If you save or extract a file using any of these extensions, Quattro translates the file into the format used by that program.

For more information on translating files, see "Translating Files" on page 79. For further details, see "Translating Files" in Chapter 6 of the *Quattro User's Guide*.

Floppy-Drive Installation

If you're using Quattro on a floppy-drive system, you need to install Quattro for floppy-drive use before loading Quattro the first time.

To install Quattro for floppy-drive use:

1. **Put the Quattro Help Disk** in Drive A.
2. **Put the Quattro System Disk** in Drive B.
3. **Make sure you're logged on to** Drive A. (If not, type `A:` and press *Enter.*)
4. **Type** `FLOPPY` and press *Enter.*
5. **When prompted, replace the System Disk** in Drive B with the Resource Disk.
6. **When prompted,** remove both disks.

The FLOPPY command initiates a batch file that sets your default directories appropriately for a floppy-drive system. You can then load Quattro correctly (see page 19).

Once you've installed Quattro for a floppy-drive system, you shouldn't have to do it again. If you should switch to a hard disk system, however, you'll need to set up Quattro for running from a hard disk before copying the files to your hard disk.

To install Quattro for a hard disk system:

1. **Put the Quattro Help Disk** in Drive A.

2. **Put the Quattro System Disk** in Drive B.
3. **Make sure you're logged on to** Drive A. (If not, type `A:` and press *Enter.*)
4. **Type** `HARD` and press *Enter.*
5. **When prompted, replace the System Disk** in Drive B with the Resource Disk.
6. **When prompted,** remove both disks.

You can then copy the Quattro disks safely onto your hard disk.

Hardware Defaults

The **D**efault **H**ardware command (*/DH*) lets you specify information about the type of hardware equipment you're using: your computer's screen(s) and the printer you use for printing spreadsheets.

Normally, Quattro automatically detects the screen you're using and acts accordingly. If you're using a nonstandard display screen, or if you're using two screens—one for spreadsheets and one for graphs—you need to specify this with the **H**ardware **S**creens command (see page 28).

If you're going to print spreadsheets on something other than a parallel printer attached to your computer's first parallel port, or if your printer has automatic line-feed or uses single sheets of paper, you need to specify this with **H**ardware **T**ext Printer command (see page 25).

Before you print a graph, you need to specify further information about your graphics printer. This is done from the Graph Print menu (see page 24).

Help Access Method

When you press *F1* to get help, Quattro accesses a program help file, called QUATTRO.HLP, and displays information from that file. When you exit the help function, Quattro closes the help file, removing it from system memory. This avoids any problems that might arise should you remove the Help Disk from its drive.

If you have a floppy-drive system, you must have the Help Disk in Drive B before you can display on-screen help. If you have a hard disk, you can speed up help access by specifying **I**nstant as the help access method. This allows the help file to stay in system memory once retrieved, displaying instant help when you press *F1.*

To change the help access method:

1. **Press** */DSH* to select **H**elp Access Method from the Default Startup menu.
2. **Select the help access method** that reflects your system (**R**emovable for floppy drives, **I**nstant for hard disks).
3. **Press** *Esc* to return to the Default menu.
4. **Select** ***U*pdate** to store the method as the new default.

If the help access method has been changed to **I**nstant and you use the program on a floppy-based system, be sure to change the default back to **R**emovable.

Caution: Do not set the **H**elp Access Mehtod default to **I**nstant if you have a floppy-drive system.

Install Quattro for Floppy-Drive Systems

See "Floppy-Drive Installation" on page 15.

International Defaults

The International defaults determine national standards for displaying currency, punctuation, date, and time. Initially, these defaults reflect United States standards. You can change them to use the standards of other countries.

The International defaults let you change the way

- time is displayed
- dates are displayed
- currency is displayed
- punctuation is used for numeric values and to separate arguments in functions

The following subsections describe each of these options.

Currency Format

When you use the Currency display format, either as the default format or as the format for a specific block, values are displayed as dollars, with a

preceding dollar sign. You can change the currency format to display other types of currency with the International command (*/DI*). This command lets you change the character(s) used as a currency symbol, and lets you position the character(s) after or before the currency value.

To specify a different currency symbol:

1. **Press** */DIC* to select **C**urrency from the Default International menu.
2. **Enter the character(s)** you want to use. To specify an ASCII character not on your keyboard, hold down the *Alt* key and use the numeric keypad to enter the ASCII code for that character. (See Appendix C for ASCII character codes.)
3. **Select an option** from the displayed menu. **P**refix displays the symbol *before* the currency value. **S**uffix displays the symbol *after* it.

International Date

When you select the Date display format, either as the default (*/DFDD*) or for a block (*/BDD*), you're given a choice of five different date formats. The last two listed are the Short and Long International formats, which are determined by the International Date default.

Initially, the International Date default is set to a standard United States 24-hour format: mm/dd/yy (long) and mm/dd (short). You can change this default to reflect the standards of other countries.

To change the International Date format:

1. **Press** */DID* to select **D**ate from the Default International menu.
2. **Select the format** you want to use as the International Date format. The short format will be the same as the long, without the year.

For more details on international date formats, see "International Date Format" in Chapter 5 of the *Quattro User's Guide.*

You can also change the clock display on the Spreadsheet Screen to reflect the International default (see page 6).

International Time

When you select the Time display format, either as the default (*/DFDDT*) or for a block (*/BDDT*), you're given a choice of four different time formats. The

last two formats listed are the Short and Long International formats, and are determined by the International Time default.

Initially, the International Time default is set to a standard United States format: hh:mm:ss (long) and hh:mm (short). You can change this default to reflect the standards of other countries.

To change the International Time format:

1. **Press** */DIT* to select **T**ime from the Default International menu.
2. **Select the format** you want to use as the International Time format. The short format will be the same as the long, without the seconds.

For more details on international time formats, see "International Time Format" in Chapter 5 of the *Quattro User's Guide.*

You can also change the date display on the Spreadsheet Screen to reflect the International default (see page 6).

Punctuation Defaults

Normally, a period is used for a decimal point, and commas are used to indicate thousands in numeric values and to separate arguments in @function statements and macro commands. You can change the punctuation character used for any of these functions.

To change the punctuation characters used in Quattro:

1. **Press** */DIP* to select **P**unctuation from the Default International menu.
2. **Select the combination** of punctuation characters you want to use.
3. **To use the new punctuation from now on,** press *Esc* to return to the Default menu, then select **U**pdate.

For more details, see "Punctuation" in Chapter 5 of the *Quattro User's Guide.*

Load Quattro

To begin working with Quattro, you must first load it into your computer's memory.

Note: Before you can work with Quattro, you must copy the Quattro files onto your hard disk (see Chapter 2 of *Getting Started with Quattro*), or install Quattro for a floppy-drive system (see page 15).

To load Quattro from a hard disk:

1. **Go to the directory** that contains your Quattro files.
2. **Type** *Q*.
3. **Press** *Enter*.

To load Quattro on a floppy-drive system:

1. **Place your Quattro System Disk** in Drive A and the Resource Disk in Drive B. If your screen doesn't show the A> prompt, type `A:` and press *Enter*.
2. **Type** `QF` and press *Enter*.
3. **Remove the Quattro System Disk** from Drive A and move the Resource Disk from Drive B to Drive A.
4. **In Drive B,** put the Help Disk (if you want to access on-screen help) or a data disk for storing and retrieving your spreadsheets.

Note: If you're experienced with Lotus 1-2-3 and prefer to use the commands you already know, you can load the alternate 123.RSC menu tree and set several defaults to be Lotus-compatible at the same time as loading Quattro. Just type `Q123` on the DOS command line and press *Enter*.

If you have a hard disk, you can also automatically retrieve a spreadsheet when you load Quattro. Just specify the file name after QUATTRO on the DOS command line. For example,

```
QUATTRO FILENAME Enter.
```

To execute a macro in the spreadsheet at the same time, include the macro name after the file name on the command line, for example,

```
QUATTRO FILENAME MACRO
```

If you don't include a file name when you load Quattro, Quattro looks for a file with the autoload file name (initially QUATTRO.WKQ) and retrieves it automatically. To specify a different autoload file name, use the **A**utoload File command on the Default Startup menu (see page 72).

You can also specify up to eight program add-ins to be loaded automatically with Quattro. Use the **D**efault Add-Ins command on the Default Startup menu (see page 30).

Macro Recording

Normally, Quattro translates your recorded actions into *menu-equivalent commands* that can be used with any Quattro menu tree, as well as with Lotus 1-2-3. (See Chapter 4 for tables listing these commands.) If you prefer to record actual keystrokes instead (for compatibility with other programs), set the **M**acro Recording default to **K**eystroke:

1. **Press** */DSCM* to select **M**acro Recording from the Default Startup Confirmation Options menu.
2. **Select *K*eystroke** from the displayed menu.
3. If you want to use Keystroke recording mode from now on, **press** *Esc* **twice** to return to the Default menu and **select *U*pdate**.

For more details about macros, see the *Macros* section beginning on page 120.

Menu Trees

A menu tree is the command structure through which you interact with a program. Quattro includes a special add-in, called the Menu Builder, which lets you alter the standard Quattro menu tree to your own liking (see Chapter 13 of the *Quattro User's Guide*). You can use this add-in to rename, rearrange, and generally restructure the Quattro menu tree. You can save the changed menu trees in separate files, which you can then load to use use with Quattro.

Quattro includes two optional menu trees that were created with the Menu Builder: one that is compatible with the commands accepted by Lotus 1-2-3 (with Quattro additions) and one that contains only basic Quattro commands for novice users. You can use either of these by loading them as the current main menu tree. You can also load an alternate menu tree, which you can then switch to from within the program.

To change the main menu tree used by Quattro:

1. **Press** */DSM* to select **M**enu Tree from the Default Startup menu.
2. **Select *M*ain Menus**.
3. **Select the menu tree** you want to use from the list displayed.
4. **Press** *Esc* to exit the Startup menu.

5. **Press** *U* to select **U**pdate from the Default menu. This stores the new menu tree as the permanent default.
6. To establish the new menu structure, **press** *Esc* **then** */Q* to exit Quattro, then load Quattro again. The new menus will then be in effect.

Note: If you want to use the Lotus-compatible menu tree *and* set Quattro defaults to be compatible with the way Lotus operates, just type `Q123` *Enter* to load Quattro. This reconfigures Quattro to use the Lotus-compatible menu tree and loads a file that sets Lotus-compatible default settings.

To specify an alternate menu tree:

1. **Press** */DSM* to select **M**enu Tree from the Default Startup menu.
2. **Select *A*lternate Menus.**
3. **Select the menu tree** you want to use from the list displayed.
4. **Press** *Esc* to exit the Startup menu.
5. **Press** *U* to select **U**pdate from the Default menu. This stores the new menu tree as the permanent default.

To switch to the alternate menu tree:

1. **Press** */DSM* to select **M**enu Tree from the Default Startup menu.
2. **Select *S*witch Menus.**
3. **Choose the menu tree** you want to use: Main or Alternate. Quattro switches to the alternate menu tree as soon as you return to the spreadsheet.

To return to the main menu tree, select **S**witch Menus again (*/DSMS*).

For more information on using different menu trees, see "Using a Different Menu Tree" in Chapter 5 of the *Quattro User's Guide.*

To make certain menu commands more accessible, you can assign them to specific *Ctrl* key combinations (see "Shortcuts" on page 62).

Menus

All Quattro commands are contained in a series of menus, all accessed through one main menu. To display the main menu, press the slash key (/). To select a command from a menu, use the *Up arrow* and *Down arrow* keys to highlight the command and press *Enter*, or press the first letter of the command.

There are two system defaults that affect Quattro menus:

- **Keep Wide** tells Quattro whether or not to display settings on all menus, or just the ones you expand.
- **Remember** tells Quattro whether or not to highlight the last command you used in a menu or the menu's first command.

To change the colors of the menus, use the **D**efault **C**olors command (see "Colors" on page 7).

For more details on using Quattro menus, see "Using the Quattro Menus" in Chapter 1 of the *Quattro User's Guide.*

Keep Menus Wide

When you first display a Quattro menu, only the menu commands are shown. You can expand the menu to show command settings as well by pressing the EXPAND key (the grey plus on the numeric keypad). This widens the menu and displays current settings (such as default display format) on the right. The menus remain wide until you press the CONTRACT key (the grey minus on the numeric keypad), or exit the menus. When you return to menus, they will be narrow again.

If you'd prefer to keep the menus wide (until explicitly narrowed with the minus key), you can specify this with the **K**eep Wide command on the Default Startup menu.

To change the default menu width:

1. **Press** */DSCK* to select **K**eep Wide from the Default Startup Compatibility menu.
2. **Select *Yes*** to include settings with the menus as the default.
3. **Select *No*** to remove settings from the menus unless specifically widened with the plus key.
4. **If you want to save the new default,** press *Esc* to return to the Default menu and select **U**pdate.

Menu Memory

Normally, Quattro keeps track of the last command you used on each menu. When you next use that menu, it highlights that command automatically. This is especially useful when you're several levels deep into the menus and want to return to the spreadsheet momentarily (with *Ctrl-*

Break). To return to your place in the menus, you just press the slash key, then hit *Enter* repeatedly until you reach your place.

If for any reason you'd rather Quattro not remember your place in the menus, you can turn off the menu memory function with the Remember command on the Default Startup menu.

To alter menu memory:

1. **Press** */DSCR* to select **R**emember from the Default Startup Compatibility menu.
2. **Select *No*** to turn off menu memory. **Select *Yes*** to return menu memory.
3. **If you want to save the new default,** press *Esc* to return to the Default menu and select **U**pdate.

Printers

There are two printer commands that let you specify information about the printer(s) you're using:

- **T**ext Printer on the Default Hardware menu (*/DHT*) lets you change any of the assumptions Quattro makes about the printer you're using for spreadsheets.
- **P**rinters on the Graph Print menu (*/GPP*) lets you give Quattro the more specific printer information it needs to print graphs. You can specify information for two different printers, if you like, then switch between the two.

The following subsections describe using each of these commands.

Graphics Printer(s)

To accurately print your graphs, Quattro needs to know exactly what type of printer you're using, including make, model, and mode. You can specify information for up to two printers, then indicate to Quattro which one you're using with the **P**rint **D**estination command.

To specify your first (default) graphics printer:

1. **Select *P*rinters** from the Graph Print menu (*/GPP*).
2. **Select *1*st Printer.**
3. **Select *T*ype of Printer.**

4. **Select the make** of your printer from the displayed list (or the one closest to it).
5. **Select the model** of your printer.
6. **Select the mode** of your printer.
7. **Press** *D* to select **D**evice. Then select the device connection to which this printer is attached.
8. **If you selected a serial device,** you may need to specify baud rate, parity, and stop bits. By default, Quattro uses the settings established by DOS for these commands. If you want to change any of them, select the command and enter a new value.
9. **To specify information about a second printer,** press *Esc* and select **2**nd Printer. Repeat the last six steps.
10. **Press** *Esc* **twice** to return to the Graph Print menu and select **U**pdate.

The **D**estination default sends your graph to the first printer when you print. To use the second printer, select **2**nd Printer from the Print Destination menu *(/GPD2)*.

For more information on setting your graphics printer specifications, see "Defining the Printers" in Chapter 8 of the *Quattro User's Guide*.

Text Printer

For printing spreadsheets, Quattro assumes that you have a parallel printer connected to your computer's first parallel slot and that your printer doesn't have automatic line-feed and uses continuous paper. If any of these assumptions is wrong, you need to change the appropriate text printer defaults.

To change text printer defaults:

1. **Press** */DHT* to select **T**ext Printer from the Default Hardware menu.
2. **If you're using a serial printer** or a different computer slot, press *D* to select **D**evice, and select the appropriate device connection.
3. **If you selected a serial device,** you may need to specify baud rate, stop bits, and parity. By default, Quattro uses the settings established for DOS for these commands. If you want to change any of them, select the command and enter a new value. (See your printer manual for the correct settings.)
4. **If your printer has automatic line-feed** (if your printout appears double spaced), select **A**uto-LF and change the setting to Yes.

5. **If you're printing on single sheets** of paper, select **1** Sheet and change the setting to Yes.
6. **Select Quit.**

For more information on specifying text printer information, see "Text Printer Defaults" in Chapter 5 of the *Quattro User's Guide.*

Punctuation

See "International Defaults" on page 17.

Recalculation Defaults

The Recalculation defaults determine the way formulas in the spreadsheet are calculated. There are three Recalculation defaults: Iteration, Mode, and Order. Each is discussed in the following subsections.

Iteration

Iteration determines how many times formulas in the spreadsheet are recalculated before calculation is considered complete. Initially, the iteration default is one. If your spreadsheet contains circular references, or if you're using rowwisse or columnwise recalculation order, you may want to increase the number of iterations performed.

To change the Iteration default:

1. **Press** */DRI* to select **I**teration from the Default Recalculation menu.
2. **Enter the number** of iterations you want performed each time the spreadsheet is recalculated.
3. **If you want use the iteration count as the new default,** press *Esc* to return to the Default menu, then select **U**pdate.

For more information on recalculation iteration, see "Recalculation Defaults" in Chapter 5 of the *Quattro User's Guide.*

Mode

Normally, recalculation is set to automatic mode, which means that formulas are automatically recalculated when necessary. You can change recalculation mode to manual, during which spreadsheet formulas will only be recalculated when you press the CALC key (*F9*).

To set recalculation mode to manual:

1. **Press** */DRM* to select **M**ode from the Default Recalculation menu.
2. **Select *M*anual** from the displayed menu.
3. **If you want use manual mode as the new default,** press *Esc* to return to the Default menu, then select **U**pdate.

When recalculation is set to manual, you must press the CALC key (*F9*) in order to recalculate the spreadsheet's formulas.

To return recalculation mode to automatic, select **M**ode again and choose **A**utomatic (press */DRMA*).

Caution: If you're using Quattro in manual recalculation mode, you should press the CALC key (*F9*) frequently. This will both keep your values up to date and avoid a possible slowdown of the system.

For more information on recalculation mode, see "Recalculation Defaults" in Chapter 5 of the *Quattro User's Guide.*

Order

Normally, Quattro recalculates a spreadsheet in Natural order, which means that before a formula is calculated, each cell it references is recalculated first. You can also set recalculation order to Rowwise, which recalculates a spreadsheet downward, one row at a time, or Columnwise, which recalculates a spreadsheet from left to right, one column at a time.

To reset the recalculation order:

1. **Press** */DRO* to select **O**rder from the Default Recalculation menu.
2. **Select the order** in which you want your spreadsheet recalcuated.
3. **If you want use that order as the new default,** press *Esc* to return to the Default menu, then select **U**pdate.

For more information on recalculation order, see "Recalculation Defaults" in Chapter 5 of the *Quattro User's Guide.*

Screen Defaults

Quattro automatically detects the following screen types:

CGA
MCGA
VGA
EGA
Hercules
IBM 8514/A
IBM 3270 PC

If you're using a non-standard screen type, you may need to specify information about the screen with the **D**efault **H**ardware **S**creen command. You can also use this command to adjust the aspect ratio used to display graphs on your screen, to switch to a different screen resolution, or to specify a second screen to be used for graphs.

To specify screen information:

1. **Press** */DHS* to select **S**creen from the Default Hardware menu. Settings for the first four commands on the Screen menu are determined automatically by Quattro.
2. If you have a black and white monitor attached to a CGA card, Quattro will erroneously detect your screen as color. **Select Color** and specify **N**o as the setting. (If you have a Compaq computer, Quattro will correctly interpret your screen as a non-color screen.)
3. If you have a CGA card (that isn't a Compaq), Quattro assumes that your screen flickers (snows) when you scroll and works to prevent it. If you think your screen doesn't flicker when scrolling, you can save screen display time by overriding the screen snow autodetection. **Select Screen Snows** and choose **N**o. If your screen then begins to flicker, you'll need to set this default back to **A**utodetect.
4. **To use a special screen driver** (other than the one automatically loaded by Quattro), select **U**se Special Driver and choose **Y**es. Then select **D**river Name and enter the name of the driver.
5. **To use a second screen** for displaying graphics, select **G**raphic Screen and choose **Y**es. Quattro prompts you for the name of the screen driver to load for that screen. Enter the name of the screen driver file that corresponds to the screen.
6. **To change the resolution** used by the screen (and possibly the colors available), select **R**esolution and choose a mode from those displayed.

7. **To adjust the aspect ratio** of the screen (if pie charts displayed are not perfectly circular), select Aspect Ratio. Use the *Up arrow* and *Down arrow* keys to perfect the shape of the displayed circle, then press *Enter*.
8. **To store the changes** as the new defaults, press *Esc* twice to return to the Default menu, then select Update.

For more details, see "Screen Defaults" in Chapter 5 of the *Quattro User's Guide*.

Time: International Format

See "International Defaults" on page 17.

Update Defaults

Quattro defaults come preset to certain values. You can change these values temporarily by specifying new values with the menus. Some of these values, such as column width and display format, will be saved automatically with the current spreadsheet. If you want to use the new settings with all new spreadsheets, you can update them; they will be stored in the Quattro configuration file (QUATTRO.DEF).

To update defaults, select the Update command on Default menu (*/DU*). You can also update print layout defaults by selecting Update from the Print Page Layout menu (*/PPU*) and graph print defaults by selecting Update from certain graph menus. **Note:** If you select Update from either the Print Page Layout menu or a menu, any changes to defaults on the Default menu are also updated.

If you don't update changes to any of the defaults, the settings will revert to their previous values when you erase the spreadsheet or exit Quattro.

For more details, see "Updating System Defaults" in Chapter 5 of the *Quattro User's Guide*.

Working with the Spreadsheet

This section describes commands you can use to enter, change, and adjust the display of data in a spreadsheet.

Add-Ins

Quattro offers two program supplements, called *add-ins*, that extend the main Quattro program: the Menu Builder and Transcript.

The Menu Builder makes it possible for you to alter the Quattro menu tree to your own liking. You can change the existing menu or create your own menu trees.

Transcript records every step you take in Quattro in a log file. You can use it to restore lost work, undo mistakes, audit changes to a file, and create macros. (See Chapter 13 of the *Quattro User's Guide* for details on both the Menu Builder and Transcript.)

You can also purchase other compatible add-ins separately.

To use a Quattro add-in, you must first *load* it into the computer's temporary memory, then open, or *run* it.

For more details on add-ins, see Chapter 13 of the *Quattro User's Guide.*

Default Add-Ins

If you use an add-in consistently, you can have it loaded automatically each time you use Quattro.

To specify autoload add-ins:

1. **Press** */DSD* to select **D**efault Add-Ins from the Default Startup menu.
2. **Select the add-in** you want loaded automatically from the displayed list.
3. **To specify a second add-in**, select another. You can specify up to eight add-ins for automatic loading.
4. **Press** *Esc* to return to the Default menu.
5. **Select *U*pdate** to store the new defaults with the program.

Load an Add-In

To load a Quattro program add-in:

1. **Press** */ML* to select **L**oad (Add-In) from the Macro menu.
2. **Select the add-in** you want to load from the displayed list.

Quattro loads the add-in into memory. It will then be listed as an option when you select **R**un (Add-In) from the Macro menu.

When you load the Transcript add-in, it immediately begins recording your actions. To view, replay, or copy those actions, you must then run the program.

When you exit Quattro, all loaded add-ins are removed from memory. You'll have to reload them before they can be run.

You can also specify up to eight add-ins to be loaded automatically when you load Quattro (see "Default Add-Ins" on page 30).

Run an Add-In

Once a program add-in is loaded, you can access the add-in by *running* it.

To run a program add-in:

1. **Press** */MR* to select **R**un (Add-In) from the Macro menu.
2. **Select the add-in** you want to run from the displayed list.

Quattro runs the add-in immediately. To exit the add-in and return to the spreadsheet, press *Esc* from the first screen or menu of the add-in.

Unload an Add-In

Once you've loaded an add-in into system memory, it stays there until you exit Quattro, at which time it is discarded and must be reloaded again next time you want to run it.

Add-ins can take up quite a bit of system memory, which you may want to use for other things, such as RAM-resident programs. You can save system memory by unloading add-ins you're not using right now.

To unload a program add-in:

1. **Press** */MU* to select **U**nload (Add-In) from the Macro menu.
2. **Select the add-in** you want to unload from the displayed list.

Quattro unloads the add-in from memory. To run the add-in again, you must reload it with the Load (Add-In) command.

Advanced Block Commands

The **A**dvanced command on the Block menu displays a menu of more advanced block commands. With these commands, you can:

- **Copy the values** stored in a block, discarding any formulas used to calculate those values (see page 67).
- **Transpose** data in a block, so that the position of the data (in columns and rows) is switched (see page 66).
- **Protect** a block of cells from being edited, erased, or replaced (see page 57).
- **Remove protections** from a block of cells (see page 57).
- **Assign** a name to a block, so you can later reference the block's name instead of its coordinates. You can also create a table in your spreadsheet listing all block names and their coordinates (see page 34).

For more details, see "Advanced Block Commands" in Chapter 4 of the *Quattro User's Guide.*

Advanced Commands

Quattro offers several "advanced" spreadsheet commands, which you can access by selecting **A**dvanced from the main menu (press */A*). The Advanced commands are:

- ***D*atabase**, which offers commands for use with a spreadsheet set up like a database (see page 41). You can use them to search through and sort a database, and set up a form for easy data entry.
- ***W*hat-If**, which automatically creates a sensitivity analysis table, listing results for a range of variables (see page 68).
- ***R*egression**, which creates a regression analysis table, showing how one value is affected by other related values (see page 59).
- ***F*requency**, which creates a frequency distribution table, showing how often values within given ranges appear in a block (see page 50).

- ***Matrix***, which lets you multiply and invert matrices (see page 54).

For advanced block commands, see the following section.

For more details on Advanced commands, see Chapters 9 and 10 of the *Quattro User's Guide.*

Assign Names to Cells in a Database

The **A**ssign Names command automatically assigns block names to each cell in the second row of a given database using the field names in the first row.

When you use the **Q**uery command to search for data in a database, you need to specify search criteria, which include both the field to search through and the condition to search for.

You indicate the field to search by referencing the first cell underneath the field name. For example, C5>500 searches the field in column C for values greater than 500 (assuming that the field name is in cell C4).

You can make it easier to enter search criteria by naming the first data cell in a field according to its field name. Then, instead of referencing cell addresses, you can reference the field by name, for example, SALES>500.

The **A**ssign Names command on the Query menu does this for you automatically. It names each cell in the second row of your database block according to the label (field name) in the cell above it.

To assign names to the second row of your database, **press** */ADQA* to select **A**ssign Names from the Advanced Database Query menu. Quattro assigns names to each of the cells in the second row of the database indicated by the **Q**uery **B**lock command.

For more details, see "Naming the First Field Entries" in Chapter 9 of the *Quattro User's Guide.*

Block Commands

The **B**lock command on the main menu (*/B*) offers a variety of commands that work on a given block of cells. With these commands, you can:

- **Copy** data from one part of the spreadsheet to another (see page 40).
- **Move** data from one part of the spreadsheet to another (see page 55).
- **Erase** data from a block of cells (see page 47).

- **Change the format** in which numbers are displayed in a block of cells (see page 45).
- **Change the alignment** of labels (text) within a block of cells (see page 52).
- **Fill** a block of cells with a series of numbers (see page 48).
- **Search** through a block for an entry and change all instances to another entry (see page 57).
- **Reformat** a block of text to fill a specific block (see page 59).

Another command on the Block menu, Advanced, displays a menu of more advanced block functions (see page 32).

Block Names

For convenience and accuracy, you can assign any block of cells a name. That name can then be used instead of the block's address to refer to the block.

The following subsections describe how to assign a name to a block, delete block names, use labels to assign names, and create a table listing block names.

You can also use the Advanced Database Search Assign Names command to automatically assign names to the cells in the second row of a database using the field names in the first row (see "Assign Names to Cells in a Database" on page 33).

Create a Block Name

You can create any number of block names for a spreadsheet. The block names are stored with the spreadsheet in which they were created.

To assign a name to a block of cells:

1. **Press** */BAC* to select Create from the Block Advanced menu.
2. **Enter a block name.**
3. **Specify the block of cells** you want to name.

To change the coordinates assigned to an existing name, delete the block name (see the next subsection), then recreate the block name with the new coordinates.

To display a list of all existing block names for a spreadsheet, **press the NAMES key** (*F3*). Selecting a name from the list automatically enters it on the input line.

For more details, see "Naming a Block" in Chapter 4 of the *Quattro User's Guide.*

Delete Block Names

When you delete the contents of a block, only the data in the block is erased; any name assigned to the block remains intact. When you no longer use a block, it's a good idea to delete its name.

To delete a single block name:

1. **Press** */BAD* to select **D**elete from the Block Advanced menu.
2. **Select a block name** from the displayed list, or type in a name.

To delete *all* block names used in a spreadsheet:

1. **Press** */BAR* to select **R**eset from the Block Advanced menu.
2. **Press** *Y* to select **Y**es from the displayed warning menu.

All block names in the spreadsheet are immediately removed from memory.

Note: If **B**orland-Style confirmation is turned off (*/DSCBN*), you won't receive a confirmation menu.

For more details, see "Deleting a Block Name" and "Deleting All Block Names" in Chapter 4 of the *Quattro User's Guide.*

To delete the data stored within a block, use the **B**lock **E**rase command (see "Erase a Block of Data" on page 47).

Labels for Block Names

You can assign names to many single-cell blocks in one sweep with the **L**abels command. This command uses adjacent labels to automatically assign names to each cell in a block.

To use adjacent labels for block names:

1. **Press** */BAL* to select **L**abels from the Block Advanced menu.
2. **Select the option** that indicates the position of the *cells you want named* in relation to the labels. For example, if the cells you're naming are *below* the labels, select **D**own.
3. **Specify the block** containing the *labels* (not the cells to be named).

Only labels and string-value formulas can be used as block names. Any numeric values in the block will be ignored.

Block names assigned with the **L**abels command are not affected by any subsequent changes to the labels themselves.

For more details, see "Using a Label to Identify a Block" in Chapter 4 of the *Quattro User's Guide.*

You can also use the **A**dvanced **D**atabase **S**earch **A**ssign Names command to automatically assign names to the cells in the first row of a database using the field names (see "Assign Names to Cells in a Database" on page 33).

Make a Table of Block Names

To view a list of all named blocks, press the NAMES key, *F3.* Then press the EXPAND key (the grey plus key) to see the coordinates for each block. You can also create a table that lists the same information, and is displayed as a permanent part of your spreadsheet.

To create a table of block names in your spreadsheet:

1. **Move the cell selector** to the upper-left corner of the block you want to use for the block names table.
2. **Press** */BAM* to select **M**ake Table from the Block Advanced menu.
3. **Press** *Enter.*

Quattro instantly creates a two-column table, beginning in the specified cell. The first column lists block names alphabetically. The second column indicates the corresponding cell ranges.

A block name table is not automatically updated. If you add, change, or delete block names, you must re-create the table to reflect the changes.

For more details, see "Making a Table of Named Blocks" in Chapter 4 of the *Quattro User's Guide.*

Colors

See "Colors" in the *Installation* section on page 7.

Columns

The Quattro Spreadsheet consists of 256 columns, labeled A through IV. With commands on the Quattro menus, you can insert and delete columns; temporarily hide columns from view, then return them to view; and change the width of columns individually or globally.

Each of these operations is described in the following subsections.

Column Width

When you first display a blank spreadsheet, all of its columns are the same width—the width specified as the default. You can change the widths of columns individually, or change the widths of all columns at once by resetting the default.

Each of these possibilities is described in the following subsections.

Default Column Width

The default column width affects all columns except those specifically adjusted with the **C**olumn **W**idth command. Initially, this default is 9. You can change it to any value between 1 and 240.

To change the default column width:

1. **Press** */DFW* to select **W**idth of Col from the Default Formats menu.
2. **Type the value** you want to use as the default column width.
3. **Press** *Enter.*
4. If you want to save the new value as the permanent default (to be used with all subsequently displayed spreadsheets), **select *U*pdate** from the Default menu (*/DU*).

You can also change the default column width from within the Column menu:

1. **Press** */CS* to select **S**et Global from the Column menu.

2. **Enter the number** you want to use as the default column width.

The default column width is stored with the current spreadsheet, whether or not you update the default settings. If you update the default column width, it is used with each new spreadsheet you create.

The default column width does not affect columns that have been individually adjusted. To affect these columns, they must first be reset to the default width with the **C**olumn **R**eset command (see the following subsection).

For more details, see "Adjusting All Columns" in Chapter 4 of the *Quattro User's Guide* and "Default Column Width" in Chapter 5.

Individual Column Width

All columns in a spreadsheet start out the same width. You can change the width of any of the columns to best display the data contained in it.

To change the width of an individual column:

1. **Move the cell selector** to the column whose width you want to change.
2. **Press** */CW* to select **W**idth from the Column menu.
3. **Type a number** from 1 to 240, *or* press *Left arrow* or *Right arrow* to widen or shorten the current column on the screen, until it's the width you want.
4. **Press** *Enter.*

Columns whose widths have been adjusted with the **C**olumn **W**idth command are not affected by the default column width setting. To return them to the default setting, use the **C**olumn **R**eset command.

To reset a column's width to the default:

1. **Move the cell selector** to the column whose width you want to return to the default.
2. **Press** */CR* to select **R**eset from the Column menu.

For more details, see "Adjusting Individual Columns" and "Resetting Column Width" in Chapter 5 of the *Quattro User's Guide.*

Delete Columns

To delete a column (and any data it contains) from the spreadsheet:

1. **Move the cell selector** to any cell in the column you want to delete.
2. **Press** */CD* to select **D**elete from the Column menu.
3. **Press** *Enter* to delete the column containing the current cell, or enter a different cell block. To delete more than one column at a time, specify a block that extends across two or more columns.

For more details, see "Deleting Columns" in Chapter 4 of the *Quattro User's Guide.*

Hide and Expose Columns

With the **C**olumn **H**ide and **C**olumn **E**xpose commands, you can temporarily remove columns from the screen, leaving only those columns you want to view. When you're ready, you can then return one or more hidden columns to the screen.

To hide a column from view:

1. **Move the cell selector** to any cell in the column you want to hide.
2. **Press** */CH* to select **H**ide from the Column menu.
3. **Press** *Enter* to hide the current column. To hide more than one column at once, extend the block shown to include at least one cell in each column you want hidden.

The specified columns are immediately hidden from view. They will return temporarily (with asterisks next to their column letters) whenever you use a command that you might want to access them with, such as **M**ove or **C**opy.

To return a hidden column to the screen:

1. **Press** */CE* to select **E**xpose from the Column menu. All hidden columns temporarily return to the screen with asterisks (*) next to their column letters.
2. **Move the cell selector** to a cell in the column you want to redisplay (or enter its address). To redisplay more than one column, specify a block of cells that includes those columns.

The specified columns return to view immediately. All other hidden columns are removed from view again.

For more details, see "Concealing Columns from Display" in Chapter 4 of the *Quattro User's Guide.*

Insert Columns

To insert a blank column in your spreadsheet:

1. **Move the cell selector** to any cell in the column to the right of where you want to insert a column.
2. **Press** */CI* to select **I**nsert from the Column menu.
3. **Press** *Enter* to insert a single column to the left of the current cell, or enter a different cell block. To insert more than one column at a time, specify a block that extends across two or more columns. The columns selected will move right and the same number of columns will be inserted.

The new columns are inserted, and existing columns move to the right.

For more details, see "Inserting Columns" in Chapter 4 of the *Quattro User's Guide.*

Copy Data

With the **B**lock **C**opy command (*/BC*), you can copy data from one block of cells to another.

To copy a block of cells:

1. **Press** */BC* to select **C**opy from the Block menu.
2. **Specify the block** you want to copy.
3. **Specify the upper-left corner** cell of the block you want to copy the data to. To make multiple copies of the source, specify more than one cell (if the source block is a single cell, row, or column).

The data is immediately copied from the source to the destination. The source cells are unaffected.

Caution: Copying data has special effects on formulas. See "Copying Formulas" in Chapter 4 of the *Quattro User's Guide* for details.

To copy a block of data *and* transpose the columns and rows of the block, use the **B**lock **A**dvanced **T**ranspose command (see "Transpose Data" on page 66).

To copy a block of data *without* copying any formulas that calculated the data, use the **B**lock **A**dvanced **V**alues command (see "Values of Formulas" on page 67).

Data Tables

See "What-If Table" on page 68.

Database Commands

Quattro includes a set of commands that are intended for use with a spreadsheet set up like a database (with columns as fields and rows as records):

- **Query** lets you search through a database for records that contain specific information. You can optionally copy those records to a different part of the database, or delete them (see page 57).
- **Sort** lets you arrange records in your database by the entries of specified fields (see page 62).
- **Form Input** restricts movement of the cell selector to only those cells that are not protected (see page 49).

Although **D**atabase commands are designed with a database setup in mind, you can use them with any arrangement of data in the spreadsheet.

To access the **D**atabase commands, select **D**atabase from the Advanced menu (press */AD*).

For more details, see Chapter 9 of the *Quattro User's Guide.*

Date Values

To enter a date in the spreadsheet, press *Ctrl-D,* then type a date in any of the following formats:

- DD-MMM-YY (04-Jul-87)
- DD-MMM (04-Jul) (assumes the current year)

- MMM-YY (Jul-87) (assumes the first day of the month)
- The Long International date format—initially MM/DD/YY (07/04/87)
- The Short International date format—initially MM/DD (07/04)

Quattro displays the date in the current International Date display format (initially MM/DD/YY). To change this format, use the **D**efault **F**ormat **D**isplay command.

You can also enter dates values using any of the date @functions: @DATE, @DATEVALUE, @NOW, @TODAY, @DAY, @DAYOFWEEK, @MONTH, and @YEAR. You can precede the first four with *Ctrl-D* to generate the date format. Otherwise, the functions return a date serial number, which you then reformat as a date.

For more details on entering dates, see "Dates" in Chapter 3 of the *Quattro User's Guide.*

To change the format of a block to date:

1. **Press** */BD* to select **D**isplay Format from the Block menu.
2. **Press** *D* to select **D**ate.
3. **Select the format** you want.
4. **Specify the block** you want affected.

For more details, see "Changing Display Format for a Block" in Chapter 4 of the *Quattro User's Guide.*

To change the default display format to date:

1. **Press** */DFD* to select **D**isplay Format from the Default Formats menu.
2. **Press** *D* to select **D**ate.
3. **Select the format** you want to use for the entire spreadsheet.

For more details, see "Default Display Format" in Chapter 5 of the *Quattro User's Guide.*

You can also change the format used for Long and Short International dates (see "International Dates" on page 18).

For a table showing different available date formats, see "Display Format" on page 44.

Default Add-Ins

See "Default Add-Ins" under "Add-Ins" on page 30.

Defaults

Quattro uses two types of *defaults*: command defaults and system defaults. *Command defaults* are the standard settings used with menu commands. For example, the default setting for the **P**rint **D**estination command is Printer. *System defaults* are settings that affect how the program works, for example, whether labels are centered or aligned left.

You can change any of the system defaults with the **D**efault command (*/D*). When you select this command, the Default menu appears, listing the following items:

***H*ardware** lets you indicate special information about your printer or screen (see page 16).

***C*olors** lets you change the colors used in the program (see page 7).

***I*nternational** lets you change the format used to display currency, date, time, and punctuation (see page 17).

***F*ormats** lets you change display format, label alignment, zero display, column width, and clock display (see page 48).

***D*irectories** lets you specify default data and program directories (see page 13).

***R*ecalculation** lets you change the way formulas are recalculated (see page 26).

***P*rotection** lets you protect or remove protection from the entire spreadsheet (see page 57).

***S*tartup** lets you specify various defaults used by Quattro each time you load the program, for example, a file or add-ins to be loaded automatically and the default file-name extension (see page 64).

***U*pdate** lets you store any changes you've made to the system defaults as the new default values (see page 29).

For detailed information on system defaults, see Chapter 5 of the *Quattro User's Guide.*

Delete Data

To erase a block of spreadsheet data, see "Erase a Block of Data" on page 47.

To delete a file, see "Erase a File" on page 74.

To clear the spreadsheet of all data, without affecting stored files, see "Erase the Spreadsheet" on page 47.

To delete one or more columns, see "Delete Columns" on page 38.

To delete one or more rows, see "Rows: Insert and Delete" on page 60.

Display Format

Numbers can be displayed in the spreadsheet in a variety of formats—as currency, for example, or percentages. Quattro formats the numbers automatically, so that all you need to enter is the digits and decimal points.

There are two ways to change the format in which numbers are displayed:

- Use the **B**lock **D**isplay Format command to change the format used in a block of cells.
- Change the default display format, which affects all numbers in any spreadsheet displayed, *except* those that have been explicitly formatted with the **B**lock **D**isplay Format command.

Table 1.1 lists the various display formats available and shows examples and format codes.

Table 1.1: Display Format Options and Their Effects

Format	(Precision)	Code	Value
Fixed	(2)	F2	-1234.57
Scientific	(2)	S2	-1.23E+03
Currency	(2)	C2	($1,234.57)
,(Financial)	(2)	,2	(1,234.57)
General		G	-1234.567
+/-		+	
Percent	(2)	P2	-123456.70%
Text (Show formulas)		T	@NOW
Date 1 (dd-mmm-yy)		D1	03-Oct-87
Date 2 (dd-mmm)		D2	03-Oct
Date 3 (mmm-yy)		D3	Oct-87
Date 4 (Long Intn'l)		D4	10/03/87
Date 5 (Short Intn'l)		D5	10/03
Date - Time 1 (HH:MM:SS)		D6	09:20:09 PM
Date - Time 2 (HH:MM)		D7	09:20 PM
Date - Time 3 (Long Intn'l)		D8	21:20:09
Date - Time 4 (Short Intn'l)		D9	21:20

Block Display Format

The **B**lock **D**isplay Format command (*/BD*) changes the format used to display numbers within a specified block of cells.

To change the display format for a block:

1. **Press** */BD* to select **D**isplay Format from the Block menu.
2. **Select the format you want** to use for a specific block of cells.
3. **Specify the block of cells** you want affected.

All numbers in the specified block are displayed in the selected format. Labels in the block are affected by the **H**idden format only, which removes both values and labels from view.

To change the display format for all cells in a spreadsheet (except those formatted with **B**lock **D**isplay Format), set the default display format (see the following subsection).

To return a block's display format to the default format, set the display format for the block to **R**eset.

For more details, see "Changing Display Format for a Block" in Chapter 4 of the *Quattro User's Guide.*

Default Display Format

The format Quattro uses to display numbers is determined by the default display format. This default affects the format used for any number that hasn't been formatted with the **B**lock **D**isplay Format command.

Initially, the default display format is **G**eneral, which displays numbers exactly as you entered them. You should set this default to the format you expect to use the most, either within the current spreadsheet, or with all spreadsheets.

To change the default display format:

1. **Press** */DFD* to select **D**isplay from the Default Formats menu.
2. **Select the format you want** to use as the default.
3. If you want to use this format as the permanent default, **press** *Esc* to return to the Default menu and **select *U*pdate**. If you don't select **U**pdate, Quattro will use the previous display format the next time you load the program. The current default will be stored with the current spreadsheet, whether or not you update it.

The new display format takes effect immediately.

For more details, see "Default Display Format" in Chapter 5 of the *Quattro User's Guide.*

To change the display format used for a single cell or a block of cells, use the **B**lock **D**isplay Format command (see page 45).

DOS

See "Operating System Access" on page 56.

Edit a Cell Entry

To delete a cell entry, simply move the cell selector to it and press *Del.*

To replace a cell entry, simply select the cell and type in the new entry. The old entry is discarded.

To edit the existing contents of a cell, select the cell and press the EDIT key (*F2*). Quattro enters Edit mode and displays the current cell contents on the input line. Use the *Right* and *Left* arrow keys to position the edit cursor

where you want to make the change. Type in characters you want to insert. Use *Backspace* or *Del* to erase characters. When you've made the desired changes, press *Enter* to enter the corrected entry into the current cell.

For more details on editing, see "Editing Entries" in Chapter 3 of the *Quattro User's Guide.*

Erase a Block of Data

To erase data in a single cell, simply move the cell selector to the cell and press *Del.*

To erase all the data stored within a block of cells:

1. **Press** */BE* to select Erase from the Block menu.
2. **Indicate the cell block** you want to erase data from.

Quattro erases everything within the specified block (but leaves the block's format intact).

For more details, see "Erasing a Block" in Chapter 4 of the *Quattro User's Guide.*

Erase the Spreadsheet

To erase all data stored within the current spreadsheet:

1. **Press** */E* to select Erase from the main menu.
2. If any changes have been made to the spreadsheet since you last saved it, Quattro displays a warning menu asking, "Lose Your Changes?" **Select *Yes*** to erase the spreadsheet and your changes. **Select *No*** to cancel the operation.

For more details, see "Erasing the Spreadsheet" in Chapter 4 of the *Quattro User's Guide.*

Files

See the *Files* section beginning on page 72.

Fill a Block with Numbers

With the **B**lock **F**ill command (*/BF*), you can automatically fill a block of cells with sequential numbers of any range and increment you specify.

To fill a block with numbers:

1. **Press** */BF* to select **F**ill from the Block menu.
2. **Indicate the block** you want to fill with values.
3. **Enter the number** you want to start the sequence with.
4. **Enter a step value.** A step value of 1 enters every value in the sequence, 2 enters every other number, and so on.
5. **Enter an end value.** Quattro will enter all values in the series up to this number or until the cell block is filled.

Quattro fills the specified block with values in the given series.

Note: In response to any of the fill prompts, you can enter either a number or a formula calculating a number (such as @TODAY or @RADIANS(45)). You cannot, however, enter dates with the *Ctrl-D* prefix.

For more details, see "Filling a Block with Sequential Values" in Chapter 4 of the *Quattro User's Guide.*

Format Defaults

There are several default settings that deal with spreadsheet format. To access these settings, select **F**ormat from the Default menu. The Default Formats menu is then displayed, listing the following options:

- ***D*isplay** specifies the default format used to display numbers in the spreadsheet (see page 44).
- ***A*lign** affects how labels in the spreadsheet are aligned when no other alignment is specified (see page 53).
- ***H*ide Zeroes** determines whether or not zero values are displayed in the spreadsheet (see page 71).
- ***W*idth of Col** determines the width of all columns, except those specifically adjusted with the **C**olumn **W**idth command (see page 37).
- **Clock** lets you specify how the date and time are displayed at the bottom of the Spreadsheet Screen (see page 6).

Format Numbers

See "Display Format" on page 44.

Form Input

With the **F**orm Input command (*/ADF*), you can restrict cell selector access to only those cells in a block that have been unprotected with the **B**lock **A**dvanced **U**nprotect command. You might want to use this command in a macro to set up a form for easy data entry.

To set up a spreadsheet for data entry:

1. **Press** */DPE* to enable default spreadsheet protection.
2. **Use the *B*lock *A*dvanced *U*nprotect command** (*/BAU*) to remove protection from the cells you want access to.
3. **Press** */ADF* to select **F**orm Input from the Advanced Database menu.
4. **Indicate the block** you want to use for data entry.

Quattro restricts access to only those cells in the specified block that are unprotected. All cells outside this block are inaccessible.

For more details, see "Setting Up a Form for Data Entry" in Chapter 9 of the *Quattro User's Guide.*

Formulas

Formulas are an essential part of most Quattro spreadsheets. You can use formulas to calculate data in the spreadsheet for you, for example, to add all the figures in a column and display the sum.

A formula combines values and at least one operator to calculate a single end value. For example,

+A2 – B3

subtracts the value in cell B3 from the value in A2. (The plus sign preceding A2 prevents the formula from being interpretted as a label by Quattro.)

The result of a formula is displayed in the cell, just as if you had typed it in. The formula you entered appears on the descriptor line when the cell is selected, and on the input line when you're editing the cell.

Formulas can be up to 240 characters long and must begin with one of the following characters:

```
0 1 2 3 4 5 6 7 8 9 . + - ( @ # $ *
```

You can include spaces between values and operators, but Quattro will delete them.

For more details, see "Formulas" in Chapter 3 of the *Quattro User's Guide.*

Freeze Columns and/or Rows

See "Titles, Freeze" on page 65.

Frequency Distribution

By creating a frequency distribution table, you can see the number of times values in a block fall within given ranges, called a *bin*. The bin is a block of ascending values entered in the spreadsheet, indicating one or more ranges of values. For example, the bin:

```
  0
 50
100
```

would look for values within three ranges: between 1 and 50, between 51 and 100, and over 100. The number of times values were found within those ranges would be displayed to the right of the bin block.

To create a frequency distribution table:

1. **Create a bin of value ranges** in your spreadsheet. It must be entered in a single-column block with a column of blank cells to the right. The numbers must appear in ascending order.
2. **Press** */AF* to select **F**requency from the Advanced menu.
3. **Indicate the block of values** to analyze.
4. **Indicate the spreadsheet block** containing the bin values.

Quattro calculates how many values in the values block fall within the ranges given in the bin block and displays the results to the right of the bin ranges. Blank cells are interpreted as 0 values.

You can also use labels in frequency distribution tables. For example, the bin

Adam
Jennifer
Zoe

looks for all labels that fall alphabetically between each of the names.

For more details, see "Creating a Frequency Distribution Table" in Chapter 10 of the *Quattro User's Guide.*

Function Keys

The function keys at the top or left of your keyboard (labeled F1 through F10) perform special operations in Quattro. See Appendix A for full descriptions of each key.

Graphs

See the *Graphs* section beginning on page 88.

Help

To display a screen of helpful information about what you're doing in Quattro, press the HELP key (*F1*).

To display information about the help system, press *F1* from within a help screen.

Each help screen contains several selectable items, called *zoom boxes*, that contain different topics. To display additional information about a topic, select its zoom box. To back up to the previous help screen, press *Esc.*

To return to the main help screen, press *Alt-F1.* To return directly to the spreadsheet from within any level of help, select the *Back to Quattro* zoom box, press *Ctrl-Break,* or select **Q**uit from the main screen.

Once you're back in the spreadsheet, you can return directly to the last help screen displayed by pressing *Alt-F1.*

For more details, see "Getting Help" in Chapter 1 of the *Quattro User's Guide.*

Insert

To insert one or more blank rows in a spreadsheet, see "Rows: Inserting and Deleting" on page 60.

To insert one or more blank columns in a spreadsheet, see "Insert Columns" under "Columns" on page 40.

To insert data from another spreadsheet file, use the **F**ile **C**ombine command, see "Combine Spreadsheets" on page 73.

Justify Text

See "Reformat Text" on page 59.

Label Alignment

When you enter a label (text) in a spreadsheet cell, it is aligned according to the label alignment default (initially set to **L**eft). You can change this default to either **R**ight or **C**enter. You can override this default by preceding the label with a *label-prefix character*, which specifies the alignment of the label. You can also change the alignment of a block of labels with the **B**lock **L**abel Align command (*/BL*). Each of these possibilities is discussed in the following subsections.

Block Label Alignment

Once you've entered labels, you can change their alignment with the **B**lock **L**abel Align command (*/BL*). This command affects only labels that have already been entered.

To realign a block of labels:

1. **Press** */BL* to select **L**abel Align from the Block menu.
2. **Select the alignment** you want.
3. **Indicate the block** of cells you want to realign.

For more details, see "Aligning Labels in a Block" in Chapter 4 of the *Quattro User's Guide*.

Default Label Alignment

The label alignment default affects the alignment of all future labels not preceded by a label-prefix character. (Existing labels are not affected.) Initially, this default aligns labels left. You can change it to either **C**enter or **R**ight.

To change the label alignment default:

1. **Press** */DFA* to select **A**lign from the Default Formats menu.
2. **Select the alignment** you want to use as the default.
3. **If you want to save the new alignment** as the new default setting, press *Esc* to return to the Default menu and select **U**pdate.

For more details, see "Default Label Alignment" in Chapter 5 of the *Quattro User's Guide.*

Label Prefixes

To align a label differently than the default, you can include a label-prefix character when you enter the label. The label-prefix characters are interpreted as follows:

'	Left Align
"	Right Align
^	Center Align

If you don't include a label prefix, the prefix correlating to the default alignment is automatically inserted and can be seen when you edit the label. You can change the way a label is aligned by altering the prefix in Edit mode.

For more details, see "Aligning Labels in a Block" in Chapter 5 of the *Quattro User's Guide.*

Layout

The **L**ayout command (*/L*) lets you change the way the Spreadsheet Screen is presented. You can use commands on the Layout menu to:

- **Freeze rows and/or columns** on the screen as *titles*, which remain in place even when you scroll the rest of the screen (see page 115).

- **Split the screen** into two windows so you can view two different areas at once (see page 70).
- **Change the position** of the descriptor line (see page 13).

Load Quattro

See "Load Quattro" in the *Installation* section on page 19.

Macros

See the *Macros* section beginning on page 120.

Matrix Tables

Using the **M**atrix commands (*/AM*), you can create an inverted copy of a matrix, and multiply two matrices to create a third.

Invert a Matrix

To invert a matrix:

1. **Press** */AMI* to select **I**nvert from the Advanced Matrix menu.
2. **Indicate the block** that contains the matrix you want to invert.
3. **Specify the upper left cell** of the block to which you want to write the inverted matrix.

Quattro copies the inversion of the source matrix to the destination.

For more details, see "Matrix Inversion" in Chapter 10 of the *Quattro User's Guide.*

Multiply a Matrix

To multiply two matrices:

1. **Press** */AMM* to select **M**ultiply from the Matrix menu.
2. **Indicate the first** of the matrices you want to multiply.

3. **Specify the second matrix.**
4. **Enter the top left cell** of the area where you want the resulting matrix entered.

Quattro multiplies the two matrices and enters the resulting matrix in the destination block.

For more details, see "Matrix Multiplication" in Chapter 10 of the *Quattro User's Guide.*

Menu Trees

See "Menu Trees" in the *Installation* section on page 21.

Merge Spreadsheets

See "Combining Spreadsheets" on page 73.

Move Data

You can easily move data from one block of cells to another block in the spreadsheet.

To move data in a spreadsheet:

1. **Press** */BM* to select **M**ove from the Block menu.
2. **Indicate the block** containing the data you want to move.
3. **Specify the upper left corner** of the block you want to move the data to.

Quattro moves the data from the source block to the destination block, and automatically adjusts any formulas involved.

Caution: Moving data has special effects on formulas, blocks, and cell references. See "Moving a Block" in Chapter 4 of the *Quattro User's Guide* for details.

Multivariate Regression

See "Regression" on page 59.

Operating System Access

You can access your computer's operating system (DOS) without having to exit Quattro.

To access DOS:

1. **Press** */FO* to select **OS** from the File menu. The spreadsheet disappears and the DOS prompt is displayed.
2. **Work with DOS** for as long as you want.
3. **Type** `EXIT` on the DOS command line when you're ready to return to Quattro.

The spreadsheet reappears exactly as you left it.

For more details, see "Accessing the Operating System" in Chapter 1 of the *Quattro User's Guide.*

Point Out Cells

To specify a cell block, either within a formula or in response to a prompt, you can *point* it out with the cell selector.

To point out a single cell, move the selector to it and press *Enter.*

To point out a multiple-cell block:

1. **Move the cell selector** to one corner of the block you want to specify.
2. **Press the period key** (.) to *anchor* the block.
3. **Move the selector** to the opposite end of the block. The block is highlighted as you extend it.
4. **Press** *Enter* or enter an operator, comma, or close parenthesis.

For more details, see "Pointing Out Cells" in Chapter 3 of the *Quattro User's Guide.*

Print a Spreadsheet

See the *Print* section beginning on page 81.

Protection

To prevent data in your spreadsheet from being erased or overwritten, you can enable spreadsheet protection. You cannot delete, edit, or replace data in cells that are protected. You can, however, remove protection from individual cells or blocks to maintain access to those cells you expect to change.

To enable protection for all cells except those that have been explicitly "unprotected," use the **D**efault **P**rotection command:

1. **Press** */DP* to select **P**rotection from the Default menu.
2. **Select Enable.**

To disable protection for the entire spreadsheet (even those cell explicitly protected), set default protection to **D**isable (*/DPD*).

For more details, see "Protection Defaults" in Chapter 5 of the *Quattro User's Guide.*

To disable protection for a specific block of cells:

1. **Press** */BAU* to select **U**nprotect from the Block Advanced menu.
2. **Indicate the block** of cells you want to remove protection from.

When default protection is enabled, you'll still be able to change data in cells you've unprotected individually.

To remove the special "unprotected" status from a block of cells:

1. **Press** */BAP* to select **P**rotect from the Block Advanced menu.
2. **Indicate the block** of cells you want to return protection to.

The cells you unprotected will be protected again when default protection is enabled.

For more details, see "Protecting and Unprotecting Cell Blocks" in Chapter 4 of the *Quattro User's Guide.*

Query a Database

The **Q**uery command on the Advanced Database menu (*/ADQ*) lets you search through, or "query" a database for information. You specify the

information to look for by entering *criteria* to be met by records in the database.

To query a database:

1. **Press** */ADQ* to select **Q**uery from the Advanced Database menu.
2. **Select Block** and specify the block of data you want to search.
3. **If you want to reference field names** instead of addresses in your search criteria, select **A**ssign Names (press *A*). Quattro names each cell in the second row of your database according to the label above it.
4. **Specify your search criteria.** You can do this either by selecting **F**ormula Criterion and entering a search formula, or by setting up a criteria table in the spreadsheet and identifying it with the **C**riteria Table command.
5. **If you want to copy matched records** to another part of the spreadsheet, set up an output block, listing the fields you want included. Then select **O**utput Block from the Query menu (*/ADQO*) and specify the coordinates of the block.
6. **Finally, select the search operation** you want to perform from the Query menu. **L**ocate highlights each matching record in the database. **E**xtract copies matching records to the output block. **U**nique copies only unique matching records to the output block. **D**elete erases all matching records.
7. **Select Quit** when the operation is finished.

For more information on using **A**ssign Names to create block names, see "Assign Names" on page 33.

For detailed information on querying, see Chapter 9 of the *Quattro User's Guide.*

To search for and replace data in a spreadsheet, use the **B**lock Search/Replace command (see page 61).

Quit Quattro

To exit the Quattro program and return to DOS:

1. Select **Q**uit from the main menu (*/Q*). If you've made any changes to the spreadsheet since last saving it, Quattro displays a warning menu asking, "Lose Your Changes?"
2. Select Yes if you want to exit anyway. Select No if you want to stay in Quattro. You can then save your spreadsheet before exiting.

Note: You don't have to exit Quattro to access your computer's operating system (DOS). If you want to use DOS without leaving Quattro, use the OS command on the File menu *(/FO)* (see page 56).

Recalculation

See "Recalculation Defaults" in the *Installation* section on page 26.

Reformat Text

The **B**lock **R**eformat command rearranges text in your spreadsheet to fit the block you specify. You can use it to change the width of paragraphs (entered as long labels) in your spreadsheet.

To reformat text:

1. **Make sure the labels** you want to reformat all begin in the same column with no blank cells in between.
2. **Move the cell selector** to the cell containing the first label to be reformatted.
3. **Press** */BR* to select **R**eformat from the Block menu.
4. **Indicate the columns and/or rows** you want the reformatted text to be displayed in. If you specify columns within the current row only, the text will fill up as many rows as necessary to display the text within those columns. If you specify both columns and rows, the text will be reformatted only if it can fit within the block you specify.

Caution: If you specify columns within the current row only, be sure there are enough blank rows underneath for the reformatted text. Otherwise, it will be displaced.

For more details on reformatting text, see "Reformatting Text Entries in a Block" in Chapter 4 of the *Quattro User's Guide.*

Regression Analysis

The **R**egression command *(/AR)* creates a regression analysis table showing how a set of variables may be affected by other sets of variables.

To create a regression analysis table:

1. **Set up dependent variables** in one column and independent variables in one or more adjacent columns. Make sure all the columns are the same length.
2. **Press** */AR* to select **R**egression from the Advanced menu.
3. **Select *D*ependent** and specify the column that contains the data you think might be affected by other variables.
4. **Select *I*ndependent** and specify the column(s) containing the data you think might be affecting the dependent data.
5. **Select *O*utput** and specify the upper left cell of the block in which you want the regression information written.
6. **If you want to force the Y-intercept** value to zero, select **Y** Intercept, then **Z**ero.
7. **Select *G*o.**

Quattro builds a regression analysis table based on data in the independent and dependent blocks and displays it in the output block.

For more details, see "Performating Regression Analysis" in Chapter 10 of the *Quattro User's Guide.*

Rows: Insert and Delete

To insert one or more blank rows in the spreadsheet:

1. **Move the cell selector** to any cell in the row below where you want to insert the new row(s).
2. **Press** */RI* to select **I**nsert from the Rows menu.
3. **Press** *Enter* to insert one row above the current cell. To insert more than one, press the period key (.) to anchor the current cell, then move the selector up to include the number of rows you want to insert, and press *Enter* (or type in the row numbers).

All rows in the selected block and below will move down to make room for the new rows.

To delete one or more rows from the spreadsheet:

1. **Move the cell selector** to any cell in the row you want to delete. (If you want to delete several rows, select a cell in one of the outside rows.)
2. **Press** */RD* to select **D**elete from the Rows menu.

3. **Press** *Enter* to delete the current row. To delete more than one adjacent rows, press the period key (.) to anchor the current cell, then move the selector to include at least one cell in each row you want to delete, and press *Enter*.

For more details, see "Inserting and Deleting Rows" in Chapter 4 of the *Quattro User's Guide*.

Search a Database

See "Query a Database" on page 57.

Search and Replace Data

With the **B**lock **S**earch/Replace command (*/BS*), you can search through your spreadsheet for specific entries and change them automatically to something else.

To search and replace data:

1. **Press** */BS* to select **S**earch/Replace from the Block menu.
2. **Select *B*lock** and indicate the block of cells you want to search through.
3. **Select *S*earch String** and enter the value you want to search for.
4. **Select *R*eplace** and enter the value you want to replace the Search value with.
5. **Select *G*o**.

Quattro searches through all entries in the specified block. When it finds a match of the Search value, it pauses and displays a box indicating which character in the cell contains the search string.

A menu in the box asks, "Replace This String?" Select **Y**es to replace the string with the Replace value then continue. Select **N**o to leave the string as is and go to the next match. Select **A**ll to replace all matching strings automatically without prompting you. Select **Q**uit to halt the search.

For more details, see "Searching for and Replacing Data in a Block" in Chapter 4 of the *Quattro User's Guide*.

You can also use the **Q**uery command to search through a database spreadsheet for information (see page 57).

Sensitivity Analysis

See "What-If Table" on page 68.

Shortcuts

You can simplify access to a Quattro menu command by creating a *shortcut* to the command. A shortcut stores the command in an easy-to-remember key sequence. You can then initiate the command by pressing the key sequence.

To create a shortcut:

1. **Press** / and go to the menu containing the command you want to create a shortcut for.
2. **Use the arrow keys** to highlight the command on the menu, but don't select it.
3. **Press** *Ctrl* **and** *Enter* at the same time.
4. **Hold down the** *Ctrl* key and press the key you want to use to access the command.

To initiate the command you stored in a shortcut, press *Ctrl* and the key you assigned to the shortcut.

To save your shortcuts for future use, select **U**pdate from the Default menu (press */DU*).

Note: You cannot create a shortcut for a command on an overlapping menu (submenu), such as **G**raph **T**ypes, or a command that simply brings up another menu.

Sort a Database

With the **D**atabase **S**ort command (*/DS*), you can reorder the data in a block in ascending or descending order. The data doesn't necessarily need to be part of a database, but it will be sorted by rows within the block specified.

To sort data:

1. **Press** */ADS* to select **S**ort from the Advanced Database menu.

2. **Select *B*lock** and specify the block you want to sort. If you're sorting part of a database, make sure to include all columns of the database. *Do not* include column headings.
3. **Select *1*st Key** and specify any cell in the column you want to use as the primary sort key.
4. **Select the order** in which you want data sorted (**A**scending or **D**escending).
5. **If you want to specify further sort keys,** do so in the same way.
6. **Select *G*o.**

The data is sorted by row in the order specified.

For more details, see "Sorting a Database" in Chapter 9 of the *Quattro User's Guide.*

Sort Order

When you sort data in ascending order, Quattro places numbers first, then labels and uses dictionary sort order for all labels. You can change the sort order Quattro uses with the **S**ort Order command on the Sort menu.

To change the sort order:

1. **Select *S*ort Order** from the Sort menu.
2. To adjust the order in which labels are sorted, **select *L*abel Order.** Choose **A**SCII to sort labels in ASCII order. Choose **D**ictionary (the initial default) to sort in dictionary order.
3. To adjust the order in which numbers and labels are placed, **select Numbers before Labels,** then select Yes (to sort numbers before labels) or No (to sort labels before numbers.
4. To save the new sort order for future sorting, select **U**pdate from the Default menu (select **Q**uit, then press */DU*).

For more details, see "Changing Quattro's Sort Order" in Chapter 10 of the *Quattro User's Guide.*

Spreadsheet Colors

See "Colors" on page 7.

Startup Defaults

The Startup defaults specify information used to start up the Quattro program. When you select **Startup** from the Default menu (*/DS*), the Startup menu is displayed, listing ten default commands:

- **Autoload File** lets you specify a file to be loaded automatically each time you start up Quattro (see page 72).
- **Startup Macro** lets you specify a macro to be loaded automatically at startup (see page 125).
- **Extension** lets you specify a file name extension to be added automatically to spreadsheet files (see page 14).
- **Beep** lets you tell Quattro whether or not to beep when you make an error (see page 6).
- **Help Access Method** lets you use a fast method of accessing help files if you have a hard disk system (see page 16).
- **Default Add-Ins** lets you specify up to eight program add-ins to be loaded automatically each time you load Quattro (see page 30).
- **Menu Tree** lets you use a different menu structure with Quattro (see page 21).
- **Compatibility Options** lets you set options for compatibility with other products, such as macro recording, and whether or not menus display settings (see page 10).

To save the changes you've made to the startup defaults as permanent defaults, press *Esc*, then select **Update** from the Default menu.

Time Values

There are several @function commands you can use to display time in your spreadsheet: @TIME, @TIMEVALUE, and @NOW. These functions translate the specified time into a time serial number. If the display format for the cell containing the serial number is set to **Time**, it displays the serial number in regular time format.

To display a serial number as a time, you can change the display format of a block to **Time**, or you can change the display format default (used for the entire spreadsheet) to **Time**.

To change the format of a block to **Time**:

1. **Press** */BD* to select **D**isplay Format from the Block menu.
2. **Press** *D* to select **D**ate.
3. **Press** *T* to select **T**ime.
4. **Select the time format** you want.
5. **Specify the block** you want affected.

For more details, see "Changing Display Format for a Block" in Chapter 4 of the *Quattro User's Guide.*

To change the default display format (for the entire spreadsheet) to Time:

1. **Press** */DFD* to select **D**isplay from the Default Formats menu.
2. **Press** *D* to select **D**ate.
3. **Press** *T* to select **T**ime.
4. **Select the format** you want to use for the entire spreadsheet.

For more details, see "Default Display Format" in Chapter 5 of the *Quattro User's Guide.*

You can also change the format used for Long and Short International time (see "International Time" on page 18).

For a table showing different available time formats, see "Display Format" on page 44.

Titles, Freeze

The Titles command on the Layout menu (*/LT*) lets you freeze specific columns or rows on your screen. The unfrozen portion of the spreadsheet (to the right or beneath the frozen titles) can still be scrolled as usual.

To freeze spreadsheet titles on your screen:

1. **Scroll the spreadsheet** so that the column(s) or row(s) you want to use as titles are at the top or left of the screen.
2. **Move the cell selector** to the row just below or to the column just to the right of the section you want to freeze. To freeze both columns and rows, position the selector in the top-left cell of the part you want to remain scrollable.
3. **Press** */LT* to select **T**itles from the Layout menu.

4. **Select the appropriate option.** Both freezes both above and to the left of the selector, Horizontal freezes rows above the selector, and Vertical freezes columns to the left.

Quattro freezes the specified area as titles. You will not be able to move the selector into the frozen area except by using the GOTO (*F5*) key.

To unfreeze titles, press */LTC* with the selector anywhere in the spreadsheet to select Clear from the Titles menu.

For more details, see "Freezing Rows and Columns" in Chapter 4 of the *Quattro User's Guide.*

Transcript

Transcript is a Quattro add-in that, when loaded, records all your actions in a log file. You can then view your command history and replay back sections to undo mistakes and restore your work. You can also copy sections of the command history to your spreadsheet to create macros.

For detailed information on Transcript, see "The Transcript Add-In" in Chapter 13 of the *Quattro User's Guide.*

Transpose Data

With the Transpose command (*/BAT*), you can *transpose* a block of data. In other words, you can reverse the position of data in columns and rows.

To transpose a block of data:

1. **Press** */BAT* to select Transpose from the Block Advanced menu.
2. **Indicate the block** of data you want to transpose.
3. **Specify the upper left cell** of the block you want to copy the transposed data to. This must be a cell outside of the source block.

Quattro copies the data in the source block to the destination, switching the positions of columns and rows. Any data already in the destination block is overwritten.

Caution: If data exists in the block you specify as the destination block, it will be overwritten by the new data. Remember that the transposed data may be a different shape than the original, so you might want to copy it to an area far from other data, then move it again if necessary.

For more details, see "Transposing Columns and Rows" in Chapter 4 of the *Quattro User's Guide.*

Update Defaults

Quattro defaults come preset to certain values. You can change these values temporarily by specifying new values with the menus. Some of these values, such as column width and display format, will be saved automatically with the current spreadsheet. If you want to use the new settings with all new spreadsheets, you can update them; they will be stored in the Quattro configuration file (QUATTRO.DEF).

To update defaults, select the **U**pdate command on Default menu (*/DU*). You can also update print layout defaults by selecting **U**pdate from the Print Page Layout menu (*/PPU*) and graph print defaults by selecting **U**pdate from the Graph Print menu (*/GPU*).

Note: If you select **U**pdate from either the Print Page Layout menu or the Graph Print menu, any changes to defaults on the Default menu are also updated.

If you don't update changes to any of the defaults, the settings will revert to their previous values when you exit Quattro.

For more details, see "Updating the System Defaults" in Chapter 5 of the *Quattro User's Guide.*

Value-Dependent Colors

See "Colors" on page 7.

Values of Formulas

The **V**alues command (*/BAV*) lets you copy the values and labels contained within a block, discarding any formulas used to determine those values.

To copy only the resulting values of formulas:

1. **Press** */BAV* to select **V**alues from the Block Advanced menu.
2. **Indicate the block** of cells you want to copy.
3. **Specify the top left cell** of the block you want to copy values to.

Quattro copies the contents of the source block into the destination block. It copies resulting values of formulas only, discarding the formulas themselves. Any existing data in the destination block is overwritten.

For more details, see "Converting Formulas to Their Values" in Chapter 4 of the *Quattro User's Guide.*

What-If Table

The **W**hat-If command */AW* lets you see how varying one or two values in your spreadsheet affects the rest of your data. It creates a table of figures that shows the results of changing one or two variables.

You can create a what-if, or sensitivity, table based on one or two variables. If you use only one variable, you can use more than one formula for substituting values. If you use two variables, you'll be able to use only one substitution formula.

A One-Way What-If Table

A one-way what-if table substitutes a range of values, one at a time, in one or more formulas to create a table of results.

To create a one-way what-if table (using one variable):

1. **Create a column of substitution values**—numbers that you want to substitute into a cell to create a table of possible end results. (Be sure to leave a blank cell above the column.)
2. **Enter the formulas** you want to use to create the table on the row above the substitution values and beginning one column to the right. The formulas should directly or indirectly reference a cell, the *input cell,* into which the substitution values will be placed, one at a time, to create the table.
3. **Press** */AW1* to select **1** Variable from the What-If menu.
4. **Specify the block** you want to use as a data table, including formulas and substitution values.
5. **Specify the cell** to use as the input cell (into which substitution values will be placed one at a time).

Quattro places each value in the substitution column in the input cell, one by one, and computes the formula(s) each time. It then fills in the table with the results.

If you change any of the substitution values or any of the formulas in the first row, you'll need to reissue the **What-If** command to update the data.

For more details, see "Creating a One-Way Sensitivity Table" in Chapter 10 of the *Quattro User's Guide.*

A Two-Way What-If Table

A two-way what-if table substitutes two sets of values into a single formula to create a table of results.

To create a two-way what-if table (using two variables):

1. **Create a column of substitution values**—numbers that you want to substitute in a formula (via the first input cell) to create a table of possible end results. (Be sure to leave a blank cell above the column.)
2. **Create a row of substitution values**, above and one column to the right of the substitution column. These values will also be substituted in the given formula (via the second input cell).
3. **Enter the formula** you want to use to create the table in the top left cell of the table. The formula should reference either directly or indirectly two input cells: one for values in the substitution column (Input Cell 1) and the other for values in the substitution row (Input Cell 2).
4. **Press** */AW2* to select **2** Variables from the What-If menu.
5. **Specify the block** you want to use as a data table, including both substitution ranges.
6. **Specify the first cell** you referenced in the formula (Input Cell 1), to be replaced by values in the column of substitution values.
7. **Specify the second cell** you referenced in the formula (Input Cell 2), to be replaced by values in the row of substitution values.

Quattro places each value in the substitution column in Input Cell 1 and each value in the substitution row in Input Cell 2, pair by pair, and computes the formula each time. It then fills in the table with the results.

If you change any of the substitution values or the formula, you'll need to reissue the **What-If** command to update the data.

For more details, see "Building a Two-Way Sensitivity Table" in Chapter 10 of the *Quattro User's Guide.*

Windows

With the **W**indows command on the Layout menu (*/LW*), you can view two parts of your spreadsheet on the screen at once. To move the selector to the other window, press the WINDOW key (*F6*).

When you first open a second window, it is *synchronized* with the first, so that if you scroll in one, the other scrolls automatically to keep the same rows or columns visible. You can disable synchronization to scroll the windows independently.

To open a second window:

1. **Move the selector** to the row or column at which you want to split the screen.
2. **Press** */LW* to select **W**indows from the Layout menu.
3. **Select *V*ertical or *H*orizontal.** **V**ertical splits the screen at the current row. **H**orizontal splits it at the current column.

To synchronize or unsynchronize your windows:

1. **Press** */LW* to select **W**indows from the Layout menu.
2. **Select *U*nsync** to be able to scroll the windows individually. **Select *S*ync** to synchronize the windows so that when you scroll in one, the other scrolls automatically.

To remove the second (bottom or right) window from the screen:

1. **Press** */LW* to select **W**indows from the Layout menu.
2. **Select *C*lear.**

For more details, see "Using Two Windows to View a Spreadsheet" in Chapter 4 of the *Quattro User's Guide.*

Write-Protect

See "Protection" on page 57.

Zero Display

Normally, all values are displayed in the spreadsheet, even when they equal zero. Optionally, you can set up Quattro so that it doesn't display values of zero. The values still remain in the cell, they are just not visible.

To suppress display of zero values:

1. **Press** */DFH* to select **H**ide Zeroes from the Default Formats menu.
2. **Select Yes.**

To return zero values to the screen, select **H**ide Zeroes and specify **N**o (*/DFHN*).

For more details, see "Default Zero Display" in Chapter 5 of the *Quattro User's Guide.*

Files

Each spreadsheet you work with in Quattro is stored in a file on disk. This section describes commands that deal with spreadsheet files and files created with other programs. Each of these commands is listed on the File menu (*/F*). With them, you can:

- Save the current spreadsheet in a file (see page 79).
- Retrieve an existing spreadsheet file (see page 78).
- Specify a temporary data directory default (see page 14).
- Combine spreadsheet files (see page 73).
- Import data stored in a text file, automatically translating it to Quattro spreadsheet format (see page 76).
- Extract data from a spreadsheet into a separate spreadsheet file (see page 74).
- Delete any file from your disk (see page 74).
- "Parse" a file, breaking down long labels (usually created by importing a text file) into separate cell entries (see page 77).
- Access your computer's operating system (DOS), without exiting Quattro (see page 56).

To specify an *autoload* file (to be retrieved automatically when you start Quattro), use the Autoload File command on the Default Startup menu (see page 72).

Autoload File

Each time you load Quattro, Quattro looks in the default data directory for a file you've saved under the autoload file name. If it finds one, it retrieves it automatically.

Initially, the autoload file name is QUATTRO.WKQ. You can change it, however, to any file name you want. You can also tell Quattro to look in a different directory for the file by including a path name with the file name.

To autoload a different file:

1. **Press** */DSA* to select Autoload File from the Default Startup menu.
2. **Enter the name** of a file. Include the path name if the file's not in the default data directory.
3. **Press** *Esc* to exit the Startup menu.

4. **Press** *U* to select **U**pdate from the Default menu.

The file you specified will be retrieved each time you load Quattro.

You can also retrieve a file when you load Quattro by specifying it on the DOS command line with the Q command. For example,

Q SALES

loads Quattro and retrieves the file called SALES (with the default filename extension) at the same time. When you retrieve a file this way, the autoload file is not retrieved.

Combine Spreadsheets

With the File Combine command (*/FC*), you can copy all or part of an existing spreadsheet into the current spreadsheet. When you combine two files, you can replace existing values with the copied ones, add the copied values to existing ones, or subtract them from existing values.

To combine two spreadsheets:

1. **Move the cell selector** to the top left cell of the block in which you want to insert the data from the "combine file."
2. **Press** */FC* to select **C**ombine from the File menu.
3. **Select an option** from the displayed menu. **C**opy replaces existing values, **A**dd adds new values to existing ones, and **S**ubtract subtracts new values from existing.
4. **Select an option** from the new menu displayed. **F**ile copies all the spreadsheet's data into the current spreadsheet. **B**lock lets you copy a specific section of the spreadsheet.
5. **If you chose *B*lock**, specify a cell range or block name.
6. Select the file to combine from the displayed list, or type in the name.

Quattro copies the contents of the specified file or block into the current spreadsheet, beginning with the current cell. The new data assumes the default and format settings of the current spreadsheet.

For more details on combining files, see "Combining Files" in Chapter 6 of the *Quattro User's Guide*.

For more details, see "Combining Files" in Chapter 6 of the *Quattro User's Guide*.

Create a New Spreadsheet

To create a new spreadsheet, just enter data in the blank spreadsheet displayed when you load Quattro. Then save your work in a file using the File Save command (see page 79).

If the spreadsheet screen already contains information, you can clear it using the Erase command (press */E*). (Save the data in a file first if you don't want to lose it.)

Erase a File

With the File Erase command, you can delete any file existing on your disk.

To delete a file:

1. **Press** */FE* to select Erase from the File menu.
2. **Select the type of file** you want to delete (Worksheet, Printing, Graph, or Other).
3. **Select the file** to delete from the displayed list.

To delete a file in a directory other than the default data directory, specify the location with the Directory command (*/FD*) first.

For more details, see "Erasing a File" in Chapter 6 of the *Quattro User's Guide.*

Export Data

Quattro will automatically translate spreadsheet files into formats used by popular spreadsheet programs (Lotus 1-2-3, Paradox, Symphony, and dBASE). Simply attach the appropriate file-name extension when you save the file.

For a list of these extensions and more information on translating files, see "Translating Files" on page 79.

Extract Part of a Spreadsheet

With the File Xtract command, you can create a new file out of part of the current spreadsheet. It also gives you the option of saving the results of

formulas as values, discarding the formulas that calculated them. The original file remains intact.

To save, or extract, part of a file:

1. **Press** */FX* to select **X**tract from the File menu.
2. **Select an option** from the displayed menu. *Formulas* saves the block exactly as is. *Values* saves the resulting values of formulas instead of the formulas themselves.
3. **Enter the name** you want to give the extracted file. (If you enter the name of an existing file, Quattro displays an Overwrite Warning menu. Select Replace to overwrite the existing file, Backup to create a backup of the existing file, or Cancel to halt the command.)
4. **Indicate the block** you want to extract.

The specified block of data is stored in the file.

To store part of a spreadsheet in a file for use with another program, include the file-name extension used by that program. If it's one of the programs that Quattro has a translator for, Quattro will translate it automatically. (See "Translating Files" on page 79 for more information.)

For more details, see "Extracting Part of a Spreadsheet" in Chapter 6 of the *Quattro User's Guide.*

File-Name Extension

If you don't specify a file-name extension for a file, Quattro assumes it has the default extension, which is initially .WKQ. Unless you specify otherwise, it tacks on this extension when you save or extract a file and looks for this extension when it retrieves, combines, or imports a file.

If you specify a file-name extension that is used by one of the programs Quattro includes a translator for (Lotus 1-2-3, Paradox, Symphony, or dBASE), Quattro automatically translates the file.

If you intend to use a different program's format in most cases, you can change the default file-name extension to that used by the program. For example, if you always want to save your spreadsheets in Paradox format, so you can use them with either Quattro or Paradox, you could change the file-name extension to .DB.

To change the default file-name extension, use the Extension command on the Default Startup menu (see page 14).

For a list of translatable files, see "Translating Files" on page 79.

Import a File

The File Import command (*/FI*) lets you copy text files into a spreadsheet, automatically translating them into spreadsheet format.

To import a text file:

1. **Move the cell selector** to the upper left corner of the block in which you want to place the imported file.
2. **Press** */FI* to select Import from the File menu.
3. **Select the appropriate option.** Comma & "" Delimited File refers to text files that use commas and/or quotation marks to divide data into fields. All other text files are considered ASCII text files.
4. **Select the file** you want to import from the displayed list. If it's in a different directory, type in the file name, including its path.

Quattro copies the text file into the spreadsheet beginning with the current cell.

Most ASCII text files require some formatting after importing. Usually, you'll need to "parse" the data, to break it down into several different fields (see page 77).

To import data from another Quattro spreadsheet, use the File Combine command (see "Combining Spreadsheets" on page 73).

To work with a file created with another spreadsheet or database program, use the File Retrieve command. If it's one of the programs that Quattro has a translator for, Quattro will translate it automatically (see "Retrieving a Spreadsheet" on page 78).

For more details, see "Importing a File" in Chapter 6 of the *Quattro User's Guide.*

Merging Spreadsheets

See "Combining Spreadsheets" on page 73.

Parse Data

When you import an ASCII text file, all the text is stored in one column. Usually, you'll need to break down, or *parse*, the data into individual fields, or columns. You can also parse data in a regular spreadsheet, for example, to break down a Name column into First and Last columns.

To parse data:

1. **Position the cell selector** at the first cell containing data to be parsed.
2. **Press** */FPC* to select **C**reate from the File Parse menu. Quattro inserts a format line above the current cell. It contains parsing directions based on data in the first cell.
3. **Edit the format line,** if necessary, to reflect exactly how you want the data broken down. Use L to indicate a label, V for values, T for time, and D for date. The > character indicates continuation of the entry, and * indicates blank characters that may be filled in by longer characters underneath. Use S to tell Quattro to skip, or delete, the character underneath its position.
4. **Add more format lines,** if necessary, to change the way the data is broken down midstream.
5. **Press** */* to select **I**nput from the Parse menu.
6. **Indicate the block** containing the data you want to parse.
7. **Press** *O* to select **O**utput from the Parse menu.
8. **Indicate the upper-left cell** of the block to which you want to copy the parsed data.
9. **Press** *G* to select **G**o from the Parse menu.

Quattro breaks down the data in the input block as indicated by the format line(s) and writes it into the output block.

For more information on all aspects of parsing data, see "Breaking Down Long Labels" in Chapter 6 of the *Quattro User's Guide.*

Password Protection

You can protect a spreadsheet file from unathorized access by assigning a password to the file. Then, before you can retrieve the file, you must supply the correct password.

To assign a password to a file:

1. With the spreadsheet displayed, press */FS* to select **S**ave from the File menu.
2. **Type the name of the file,** followed by a space and the letter P.
3. **Press** *Enter.*
4. **Type the password** you want to assign to the file and press *Enter.*
5. **Enter the password again** for verification.

When you retrieve a file that's been encrypted with a password, Quattro asks you for the file name. You must supply the correct password before it is displayed.

For more details on passwords, see "Assigning a Password to a File" in Chapter 6 of the *Quattro User's Guide.*

Retrieve a Spreadsheet

To retrieve a spreadsheet you've stored in a file:

1. **Press** */FR* to select **R**etrieve from the File menu.
2. **Select the file** you want to retrieve from the displayed list, or enter a file name. (To display files in a different directory, type the directory's path name and press *Enter.*)

Caution: When you retrieve a file, Quattro erases any data currently displayed in your spreadsheet. If you don't want to lose the data, make sure to save it before retrieving a file.

Quattro automatically translates files created with several different spreadsheet and database programs. All you have to do is include the file's extension with the file name when you retrieve it. See "Translating Files" on page 79 for a list of program files that Quattro can translate.

To retrieve a text file for use with Quattro, use the File **I**mport command (see page 76).

To insert data from another file into the current spreadsheet, use the File **C**ombine command (see page 73).

For more details, see "Retrieving a File" in Chapter 6 of the *Quattro User's Guide.*

Save a Spreadsheet

To save spreadsheet data in a file:

1. **If you're using a floppy-based system,** make sure there's a work disk in the drive specified as the default data directory. (To check this, press */DDD* and press + if necessary to show the settings on the menu.)
2. **Press** */FS* to select **S**ave from the File menu.
3. If you're saving a spreadsheet for the first time, **enter a file name** and press *Enter*. If you're resaving a retrieved spreadsheet, press *Enter* to save it under the same name.
4. If the file name you specified already exists, Quattro displays an overwrite warning menu. Select **C**ancel to cancel the operation, **R**eplace to overwrite the existing file, or **B**ackup to make a copy of the existing file, then overwrite it.

Note: Backup files are assigned the file-name extension .BAQ. To retrieve a backup file, you need to include the extension with the file name.

For more details, see "Saving a Spreadsheet" in Chapter 6 of the *Quattro User's Guide.*

Text Files

To translate text files for use with Quattro, use the **F**ile **I**mport command (see "Import a File" on page 76).

Translate a File

Quattro includes several special translator files that save spreadsheets in formats for use with different programs and automatically translate files created with different programs when you retrieve them. It recognizes the format required by each program by the file's extension.

Table 1.2 shows the types of files Quattro can translate, the extensions use for each, and the translator files required for each.

Table 1.2: Files Quattro Can Translate

Extension	Program	Translator Files Required	
		To Save	To Retrieve
.DB	Paradox	FSDB.TRN	FRDB.TRN
.DBF	dBASE III	FSDBF.TRN	FRDBF.TRN
.DB2	dBASE II	FSDB2.TRN	
.WRK	Symphony	FSWRK.TRN	FRWRK.TRN
.WK1	Lotus 1-2-3, version 2.0	FSWK1.TRN	FSWK1.TRN
.WKS	Lotus 1-2-3, version 1A	FSWKS.TRN	FSWKS.TRN
.WKE	Lotus 1-2-3, Educational	FSWKE.TRN	FSWKE.TRN

To save or retrieve a file for use with one of the above programs, just include the appropriate extension when you specify the file name.

You can also combine files (see "Combine Spreadsheets" on page 73) and extract files (see "Extract Part of a Spreadsheet" on page 74) in different program formats.

In all cases, Quattro performs the translation automatically. You never even see it.

For more details, see "Translating Files" in Chapter 6 of the *Quattro User's Guide.*

Printing

The basic procedure for printing the current spreadsheet is as follows:

1. **Make sure the information** specified with with the Hardware Text Print command is correct (see page 25).
2. **Press** */P* to select **P**rint from the main menu.
3. **Press** *B* to select **B**lock and specify the block you want to print.
4. **Make any necessary changes** to the other print commands.
5. **Turn the printer on** and adjust the paper alignment (if necessary) with the **A**djust Printer commands.
6. **Select** *Go* to begin printing.

Quattro immediately prints the specified spreadsheet block.

There are several options for printing spreadsheet data. You can:

- Add headers and footers to the pages (see page 83).
- Add column and/or row headings to each page (see page 83).
- Adjust the margins and page length (see page 84).
- Change the format of the printout (see page 82).
- Insert page breaks, or print without page breaks (see page 84).
- Print the data to a disk file, instead of to the printer (see page 82).
- Send commands to the printer with setup strings (see page 87).
- Reset the printer command settings (see page 86).

Each of these options is discussed in the following subsections.

Adjust the Printer

To position the paper properly in your printer before printing, use the Adjust Printer (*/PA*) command. When you select this command from the Print menu, a menu with three options is displayed:

- ***S*kip Line** moves the paper in the printer forward one line.
- ***F*orm Feed** moves the paper in the printer forward to the top of the next page.
- ***A*lign** tells Quattro to assume that the current position of the paper in the printer is at the "Top of Page."

Use the Skip Line and Form Feed commands to adjust the position of the paper, then use Align to set the Top of Page position.

For more details, see "Adjusting the Printer" in Chapter 7 of the *Quattro User's Guide.*

Destination

Normally, Quattro prints your spreadsheet on your printer. You can optionally "print" data from your spreadsheet to a disk file for future use. The file includes all print information, such as headers and setup strings, so that it can be printed at any time in the future.

To specify printing to disk:

1. **Select *D*estination** from the Print menu (*/PD*).
2. **Select *F*ile** from the displayed menu.
3. **Enter the name** you want to give the file. Don't include an extension; .PRN is added automatically.

For more details, see "Printing to a Disk File" in Chapter 7 of the *Quattro User's Guide.*

Format

Instead of printing your data in regular spreadsheet format, you can optionally print it as a list of cell contents, including format information.

To tell Quattro to print a list of cell contents:

1. **Select *F*ormat** from the Print menu (*/PF*).
2. **Select Cell-Formulas.**

When you print the spreadsheet, cell contents will be listed, one by one, exactly as they appear on the status line when highlighted. For example:

```
A1: (W18)(C0) +A10*300
```

To return to printing in standard spreadsheet format, set the Format command to As Displayed (*/PFA*).

For more details, see "Changing the Print Format" in Chapter 7 of the *Quattro User's Guide.*

Headers and Footers

With the **Header** and **Footer** commands, you can print information at the top and/or bottom of each page.

To enter a header or footer:

1. **Select *Header* or *Footer*** from the Print Page Layout menu (*/PPH* or */PPF*).
2. **Type the text** you want to use.
3. **Press *Enter*.**

You can use three special characters as commands in headers or footers:

- # enters the current page number.
- @ enters the current date.
- | (vertical line) determines the position of the text. It works like a tab. One | centers the text, two | characters align it right.

For more details, see "Using Headers and Footers" in Chapter 7 of the *Quattro User's Guide*.

Headings

To include column and/or row headings on each page of your printout, use the headings commands:

1. **Select *Top* Headings** from the Print menu (*/PT*).
2. **Indicate the block** of cells you want printed across the top of each page. You need only specify one cell in each row you want printed.
3. **Press *L*** to select **L**eft Headings.
4. **Indicate the block** of cells you want printed along the left side of each page. You need only specify one cell in each column you want printed.

For more details, see "Including Headings" in Chapter 7 of the *Quattro User's Guide*.

Margins

The initial default margin settings leave a half-inch margin on all sides of a printed page. You can change these margins to position your spreadsheet anywhere on the page.

To change the margin settings:

1. **Select *M*argins** from the Print Page Layout menu (*/PPM*).
2. **Select the margin** you want to change.
3. **Enter the new value** for the margin.
4. **To change other margins,** select them from the menu and specify new values.
5. **Select *Q*uit** when you're finished.
6. **To store the new settings** as defaults (to be used from now on), select Update from the Page Layout menu.

For more details, see "Changing the Margins" in Chapter 7 of the *Quattro User's Guide.*

Page Breaks

When printing a spreadsheet, Quattro automatically inserts page breaks according to the Page Length setting. Normally, this setting is set to print on paper that's 11.5 inches long.

To print on paper of a different length, change the Page Length setting:

1. **Select *P*age Length** from the Margins & Page Length menu (*/PPMP*).
2. **Specify the length** of the paper you'll be printing on (in number of lines).

For more details, see "Changing the Margins" in Chapter 7 of the *Quattro User's Guide.*

You can also insert page breaks manually either within the spreadsheet or by using the print menus.

To insert a page break from within the spreadsheet:

1. **Move the selector** to the first cell in the row at which you want to begin a new page.

2. **Press** */RI Enter* to insert a blank row above the current cell.
3. **Type** |:: (a vertical line and two colons) and press *Enter*. This enters two colons at the beginning of the row, indicating a page break.

To insert a page break using the menus:

1. **Move the selector** to the first cell in the row at which you want to begin a new page.
2. **Press** */RI Enter* to insert a blank row above the current cell.
3. **Press** */PPC* to select **C**reate Break from the Print Page Layout menu.

For more details, see "Inserting Page Breaks" in Chapter 7 of the *Quattro User's Guide*.

You can print a spreadsheet with or without page breaks. To specify printing without page breaks:

1. **Select *B*reak Pages** from the Print Page Layout menu (*/PPB*).
2. **Select *N*o**.

When you print the spreadsheet, Quattro won't insert page breaks or print headers and footers. Page breaks that you entered in the spreadsheet with the |:: character sequence, however, will be recognized.

For more details, see "Printing without Page Breaks" in Chapter 7 of the *Quattro User's Guide*.

Page Layout

The **P**age Layout command on the Print menu (*/PP*) lets you adjust the way the printed spreadsheet appears. With commands on this menu, you can:

- **Add headers and footers** to the top or bottom of each printed page (see above).
- **Specify printing with or without page breaks** (see page 84).
- **Adjust margins and page length** (see page 84).
- **Insert a page break** in the current row of the spreadsheet (see page 84).
- **Specify a setup string** of special special commands to send to the printer (see page 87).

Page Layout settings are saved automatically with the current spreadsheet. If you want to use the new defaults with *all* spreadsheets, select the Update command that is also on this menu.

Print a Graph

See "Print a Graph" in the *Graph* section on page 110.

Printer Specifications

Unless given other instructions, Quattro assumes that you have a parallel printer attached to your computer's first parallel port, that the printer does not have automatic line-feed, and that it uses continuous paper. If this is not the case, you need to give Quattro some information about your printer before you can print. See "Text Printer" in the *Installation* section on page 25.

In order to print a graph, you must have a graphics printer, and you must specify information about that printer with the **P**rinters command on the Graph Print menu (*/GPP*). This command lets you indicate make, model, and mode of up to two graphics printers. You can also specify other printer information, such as device connection and stop bits. See "Graphics Printer" in the *Installation* section on page 24.

Reset the Print Commands

After you've made changes to any of the print settings, you can return them to their previous settings with the Reset command:

1. **Select *R*eset** from the Print menu (*/PR*).
2. **Select the area** you want to reset.

For more details, see "Resetting the Print Options" in Chapter 7 of the *Quattro User's Guide.*

Setup Strings

A *setup string* is a sequence of characters that is sent to your printer just before printing. The characters are translated into special commands. You can use setup strings to create special effects, for example, letter-quality printing. (For a table of setup string codes used by popular printers, see Appendix B, "Printer Setup Codes.")

To specify a printer setup string:

1. **Select Setup String** from the Print Page Layout menu (*/PPS*).
2. **Type the string** of setup codes you want to send to the printer. Type as many commands as you like (up to 39 characters). Do not use spaces to separate commands.
3. **Press** *Enter*.

For more details, see "Using Setup Strings" in Chapter 7 of the *Quattro User's Guide*.

Graphs

With Quattro, you can create up to 10 different kinds of graphs to illustrate your spreadsheet data. You can use graph customizing commands to alter the look and presentation of your graph. And you can save your graph for future use, or print it.

To create a Quattro graph:

1. **Press** */G* to select **G**raph from the main menu.
2. **Press** *G* to select **G**raph Type.
3. **Select the type** of graph you want to create.
4. **Press** *S* to select **S**eries Values.
5. **Press** *1* to select **1**st Series.
6. **Indicate the block** containing the series of values you want to graph first.
7. **Assign up to five more series of values** using **S**eries **V**alues commands. When you're done, **press** *Q* for **Q**uit.
8. **Label the *X*-axis,** if you like, using data from the spreadsheet. Press *X* to select **X**-Axis Values, and indicate the block containing data you want to use to define the X axis.
9. **Press** *V* (**V**iew) or *F10* to display the graph on the screen. When you're ready to return to the spreadsheet, **press** *Esc.*
10. **Use the graph customize commands** to perfect the graph display, viewing the graph between changes, if you like.
11. To print the graph, **press** *P* to select **P**rint, enter information about your print (if you haven't already), change any print options you want to adjust, and **press** *G* for **G**o.

There are two major exceptions to the procedure above. One is that pie charts can graph only one series of values; if more are assigned, they will be ignored. The other is XY graphs, which actually graph values assigned with the **X**-Axis Values command, rather than using the values as labels.

The last graph you create is saved with the spreadsheet. If you want to store more than one graph with a spreadsheet, use the **G**raph **N**ame command to store them under different names.

The following subsections describe each of the commands available with the Graph function, in alphabetical order. For details on building and customizing graphs, see Chapter 8 of the *Quattro User's Guide.*

Colors of a Graph

You can change any of the colors used to display your graph on the screen:

1. **Press** */GCC* to select Colors from the Graph Customize menu. (If necessary, you can press the EXPAND key (the grey plus key) to display the existing color settings.)
2. **Select the area** you want to change the color of.
3. **Select the color** you want to use for that area.

You can also change the colors of grids from the Grids menu (*/GCGC*) and the colors of individual series values from the 1st..6th Series menus (*/GCS#C*).

To change the colors used in a pie chart, use the **C**ustomize **P**ies **C**olors command (see the next section).

To change the colors used to print a graph (if you want different colors than those displayed on the screen), use the **P**rint **C**olors command (see page 110).

For more details, see "Changing Colors of the Graph" in Chapter 8 of the *Quattro User's Guide.*

Colors of a Pie Chart

To change the colors used for different slices of a pie chart:

1. **Select Colors** from the Customize Pies menu (*/GCPC*). (If necessary, you can press the EXPAND key (the grey plus key) to display the existing color settings.)
2. **Select the slice** you want to change the color of. (If your pie contains more than nine slices, the colors will be repeated in the pattern you set up for the first nine.)
3. **Select the color** you want to assign to the slice.
4. **Repeat the last two steps** for each pie slice you want to change the color of.

For more details, see "Changing the Color of Pie Slices" in Chapter 8 of the *Quattro User's Guide.*

Customize

The Graph Customize command (*/GC*) lets you adjust the graph display in numerous ways, from adding titles to changing the grid pattern.

When you select **C**ustomize from the Graph menu (*/GC*), the Graph Customize menu is displayed, listing the following options:

***S*eries** lets you change the way series of values are represented in the graph (see page 112).

***X*-Axis** lets you change the way the X axis is displayed (see page 118).

***Y*-Axis** lets you change the way the y-axis is displayed (see page 119).

***C*olors** lets you adjust the colors or hues used to display each section of the graph (see page 7).

***P*ies** lets you change the way pie charts are displayed (see page 109).

***R*esolution** lets you change the resolution used to display graphs on your screen (see page 111).

***G*rids** lets you change the pattern used to display grids on the graph (see page 102).

***V*iew** displays the current graph.

Customize by Series

When you select **S**eries from the Graph Customize menu (*/GCS*), the Customize Series menu is displayed. This menu lists graph characteristics you can change for each series of values plotted (see "Series Customizing" on page 112 for details).

The last six commands on the menu, 1st Series through 6th Series, let you customize the graph by series. This means that you can use one menu to view and change every characteristic of a given series.

When you select one of the numbered series (1st through 6th) from the Customize Series menu, the 1st through 6th Series Customize Menu appears. This menu duplicates commands available from other menus, but groups them together on one menu for a single series:

***B*lock** displays the cell block assigned to the series. It duplicates the **S**eries Values command on the Graph menu. You can use it to assign different values to the series (see page 113).

Legend lets you assign a title to series, which is then displayed in the legend (see page 104).

Interior Labels lets you assign a series of values on the spreadsheet to be displayed alongside corresponding points on the graph (see page 103).

Placement of Labels determines where the values assigned with the above command are placed in the graph (see page 103).

Color lets you change the color used to represent this series in the graph (see page 89).

Override Type lets you assign a type of graph to be used specifically by this series, overriding the type specified with the Graph Type command (see page 107).

Markers lets you change the symbol used to represent this series in "markers only" and "combined lines and markers" graphs (see page 105).

Fill Pattern lets you change the pattern used to fill the bars representing the series in bar graphs (see page 92).

Reset erases any changes made to the menu, returning settings to their default values.

View displays the current graph. Press *Esc* to return to the menu.

Quit exits the menu and returns you to the Graph Customize Series menu.

EPS (Postscript) Files

See "Write to EPS or PIC File" on page 117.

Explode Pie Chart Slices

To emphasize specific sections of a pie chart, you can *explode* slices, pulling them out slightly from the circle.

To explode pie slices:

1. **Select Explode** from the Customize Pies menu (*/GCPE*). (If necessary, you can press the EXPAND key (the grey plus key) to display the existing explode settings.)
2. **Select the slice** you want to explode. (The first slice begins at the 0 degree mark [3:00 on a clock] and proceeds counterclockwise. Only the first nine can be exploded.)

3. **Select *E*xplode** from the displayed menu.
4. **Repeat the last three steps** for each pie slice you want to explode.

To return a slice back to its original position, select Explode, select the slice number, and choose Don't Explode.

For more details, see "Exploding Pie Slices" in Chapter 8 of the *Quattro User's Guide.*

Fill Patterns

Bar and area graphs use fill patterns to distinguish between different series. Quattro uses a different fill pattern for each series of values plotted. You can change the pattern used for each series with the Patterns command (*/GCSP*).

There are two ways to change the fill patterns. You can

- select Patterns from the Customize Series menu, then select the series you want to change the fill pattern for.
- select the series you want to change the fill pattern for from the Customize Series menu, then select Fill Pattern.

The first method is best for changing the fill patterns of several series. The second is best when you're making other changes to the same series.

To change the patterns used for several series with the Customize Series menu:

1. **Select *P*atterns** from the Customize Series menu (*/GCSP*).
2. **Select the series** you want to change the patterns for.
3. **Select the pattern** you want to use.
4. **Repeat the last two steps** for each series you want to change the fill pattern for.
5. **To use the new fill patterns** as the default for all future graphs, press *Esc,* then select Update from the Customize Series Patterns menu.

To change fill patterns by series:

1. **Select the series** for which you want to change the fill pattern from the Customize Series menu (*/GCS#*). (If necessary, you can press the EXPAND key (the grey plus key) to display the existing fill patterns.)

2. **Press** *F* to select **F**ill Patterns.
3. **Select the fill pattern** you want to use for that series.
4. If you want to use this set of patterns for future graphs, **press** *Esc* **twice** to return to the Customize Series menu, **select *P*atterns, then *U*pdate.** This saves patterns changes to any of the series.

For more details, see "Changing Fill Pattern" in Chapter 8 of the *Quattro User's Guide.*

Fonts

See "Font of Titles" on page 116.

Graph Type

You can create up to ten types of graphs with Quattro:

- Line (see page 97)
- Bar (see page 94)
- 3-Dimensional Bar (see page 100)
- XY (see page 101)
- Stacked Bar (see page 99)
- Pie (see page 98)
- Area (see page 94)
- Rotated Bar (see page 98)
- Markers (see page 105)
- Combined Lines and Markers (see page 95)

To change graph type:

1. **Select Graph Type** from the Graph menu (*/GG*).
2. **Select the graph type** you want to use.

Each graph type is described on the following pages. See "Graph Types" in Chapter 8 of the *Quattro User's Guide* for more thorough descriptions of each graph type.

Area Graph

Description: Uses lines to graph values one series on top of another, showing cumulative values. The space underneath each series is filled with a different fill pattern and/or color. The first series is plotted on the bottom, with successive series stacked on top.

Used To: Show the totals created by the series involved, as well as to compare the progress of individual series.

Values: Up to six series of values can be plotted. X-axis values supply horizontal labels along the x-axis.

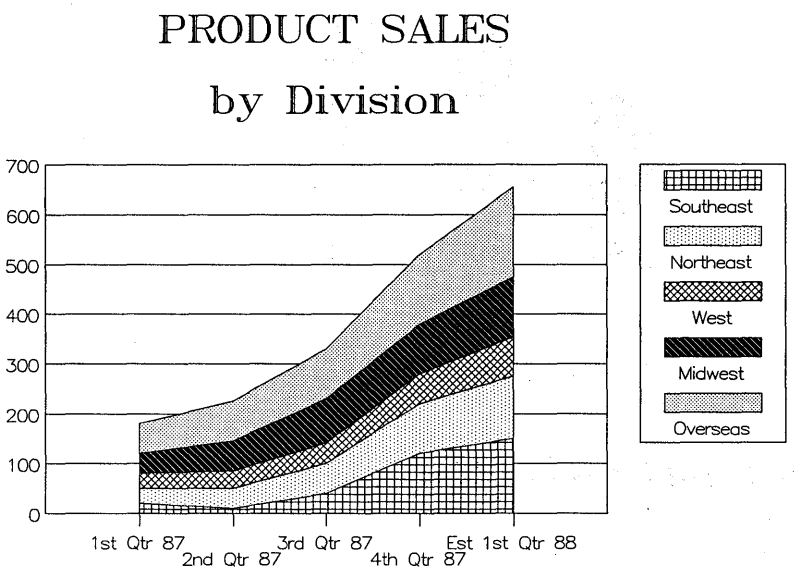


Figure 1.1: An Area Graph

Bar Graph

Description: Uses vertical bars, extending upward from the X-axis, to indicate values. The value of a bar is determined by its height, set according to the scale shown on the Y axis. When more than one series is plotted, bars for each are shown side by side to emphasize the difference, and each series is assigned a different fill pattern and/or color. The width of the bars is determined by the number of values plotted.

Used To: Compare and contrast values in a series, as well as like values in different series.

Values: Up to six series of values can be plotted. X-axis values supply horizontal labels along the x-axis.

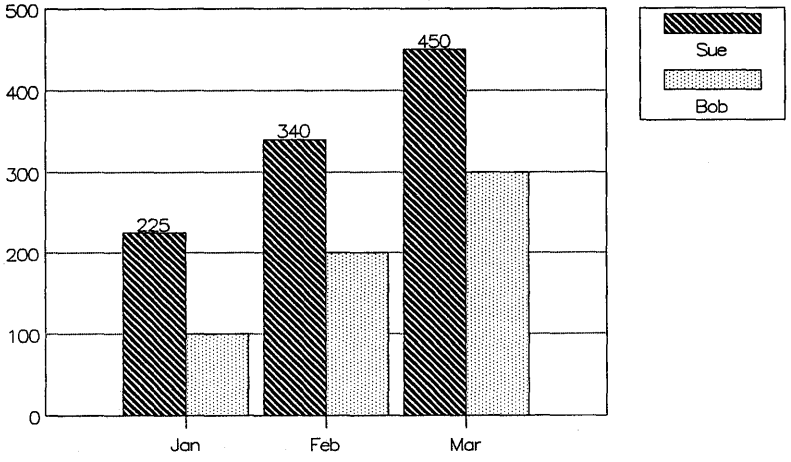


Figure 1.2: A Standard Bar Graph

Combined Lines and Markers Graph

Description: Exactly like a standard line graph except that symbols, or *markers*, are added to indicate values as positioned on the graph. A different marker symbol is used for each series of values plotted.

Used To: Show progression of values, but with more of a focus on individual values within each series.

Values: Up to six series of values can be plotted. X-axis values supply horizontal labels along the x-axis.

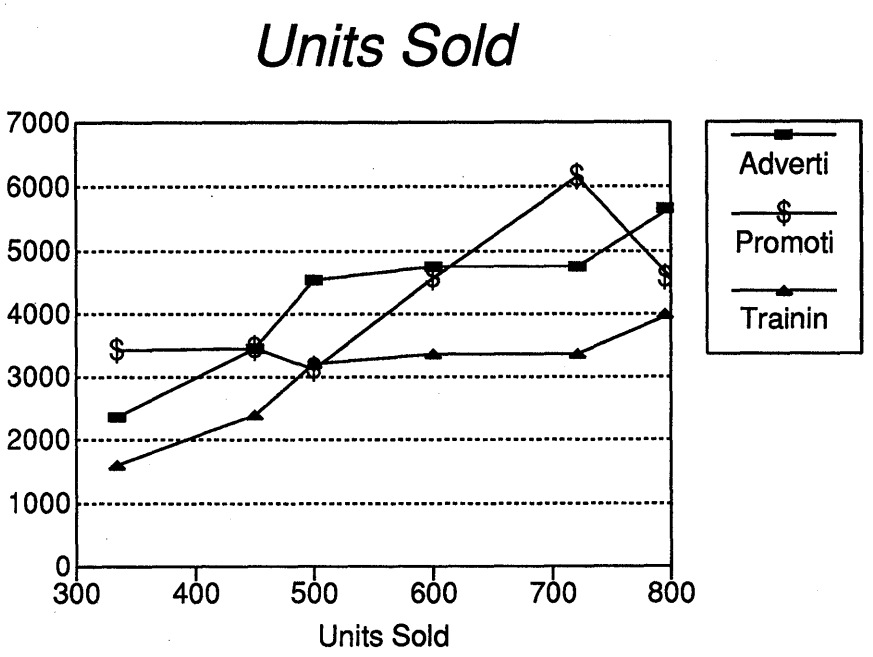


Figure 1.3: A Lines and Markers Graph

Line Graph

Description: Maps each series of values on the graph with a line. The line connects values on the graph in the sequence in which they appear in the series.

Used To: Show progression of values over time—revealing patterns and/or trends.

Values: Up to six series of values can be plotted. X-axis values supply horizontal labels along the x-axis.

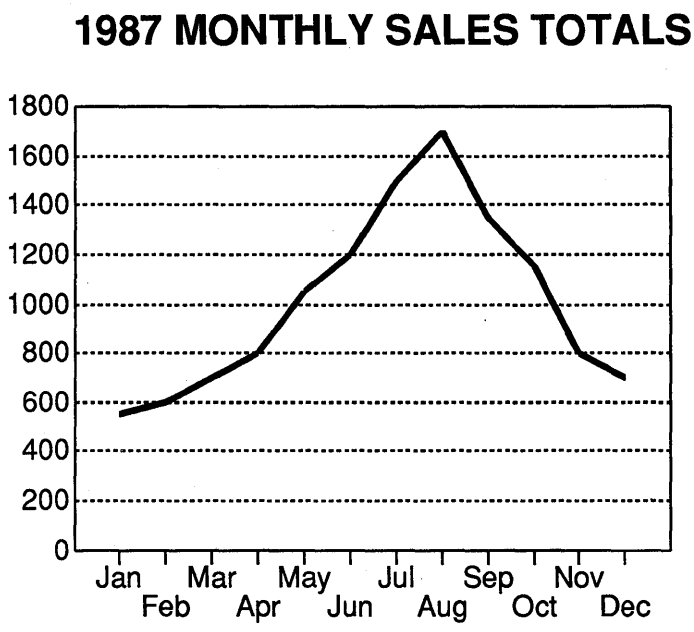


Figure 1.4: A Standard Line Graph

Markers Graph

Description: Uses markers to indicate values on a graph, yet unlike the Lines and Markers graph, does not connect all markers in a series with a line.

Used To: Focus on the positions of all values, regardless of the series they belong to, often revealing interesting clusters of values.

Values: Up to six series of values can be plotted. X-axis values supply horizontal labels along the x-axis.

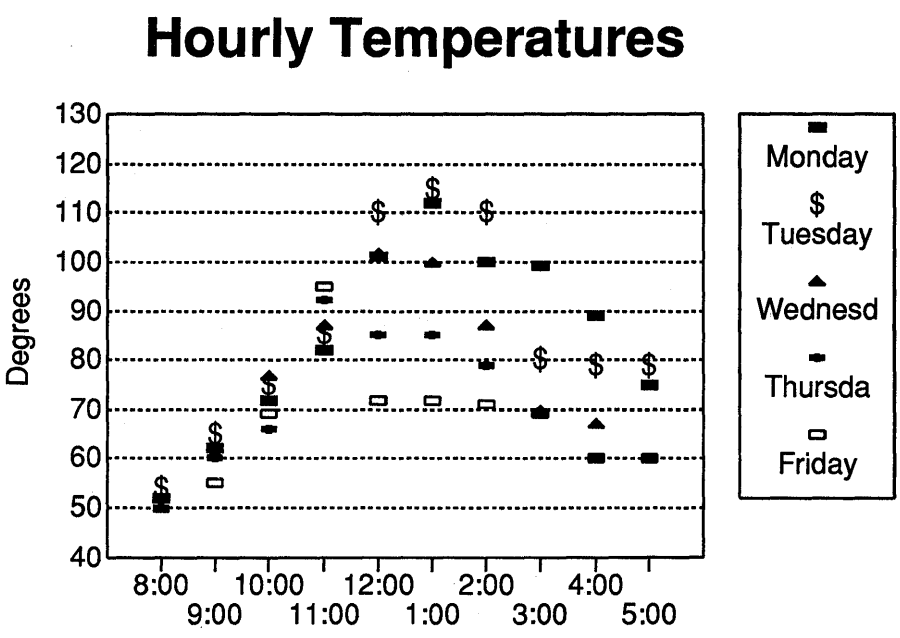


Figure 1.5: A Markers Only Graph

Pie Chart

Description: Shows each value in a single series as a wedge within a circle (a slice of a pie). The pie is split proportionally by the values being graphed, with each value shown as a percentage of the series' total.

Used To: Show the breakdown of values within a series.

Values: Only one series of values can be plotted. X-axis values label individual slices of the pie.

Rotated Bar Graph

Description: Exactly like a Standard Bar graph, except that the positions of the X and Y axes are reversed, extending the bars horizontally from left to right.

Used To: Compare and contrast values, and also give a more goal-oriented presentation.

Values: Up to six series of values can be plotted. X-axis values supply horizontal labels along the x-axis.

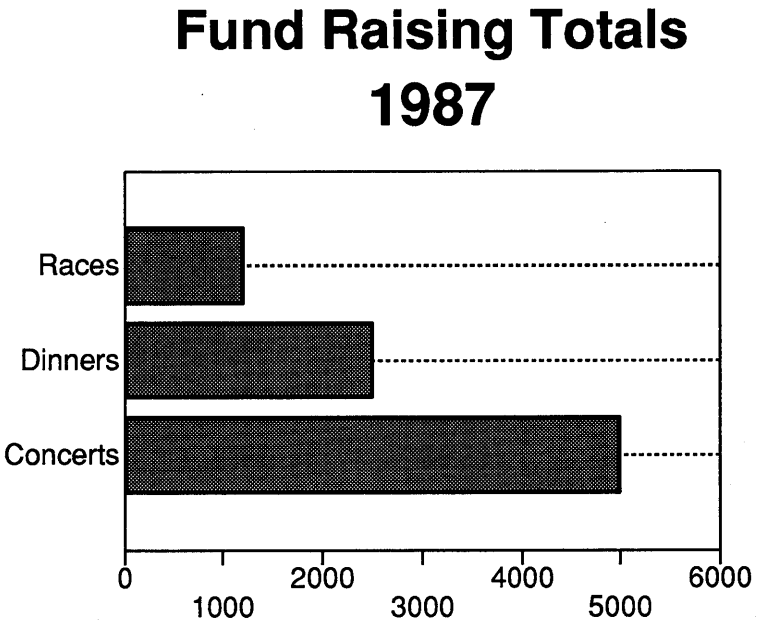


Figure 1.6. Rotated Bar Graph

Stacked Bar Graph

Description: Places related values of multiple series one on top of another instead of side by side. This shows cumulative values as well as individual. The first series of values is plotted on the bottom of the graph, and each progressive series is stacked on top of the previous one.

Used To: Show the relationship between individual values and the total.

Values: Up to six series of values can be plotted. X-axis values supply horizontal labels along the x-axis.

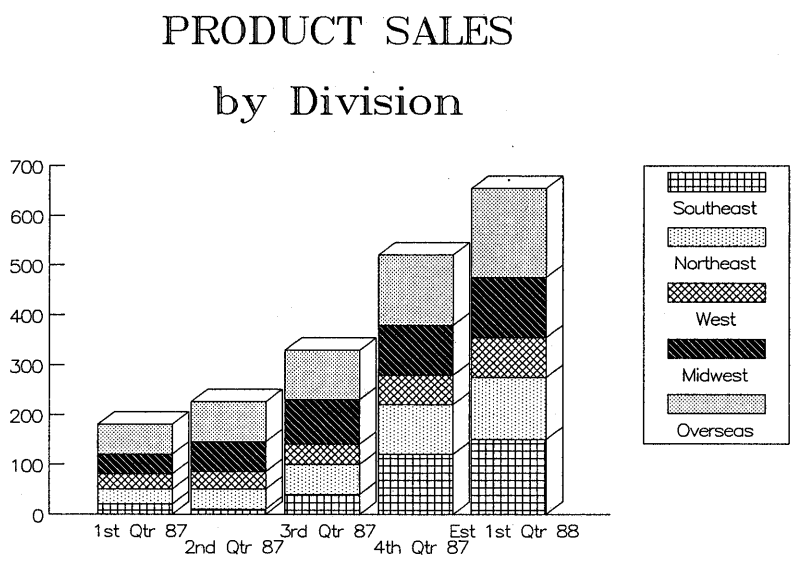


Figure 1.7: A Stacked Bar Graph

Three-Dimensional Bar Graph

Description: Differs from a standard bar graph only in that the bars are given "shadows" that make them appear three-dimensional. This is a costmetic feature only and works best with wider bars (which appear when fewer values are graphed).

Used To: Compare and contrast values in a series, and like values in different series.

Values: Up to six series of values can be plotted. X-axis values supply horizontal labels along the x-axis.

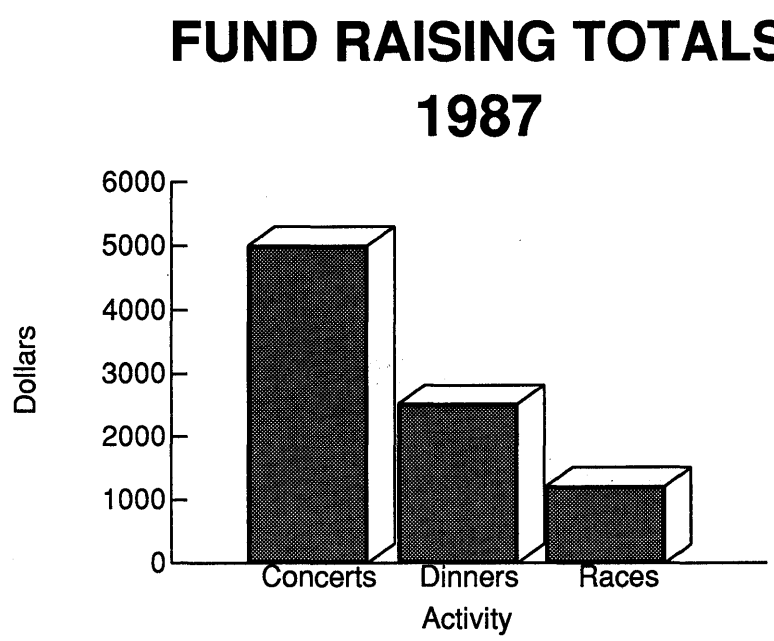


Figure 1.8: A Three-Dimensional Bar Graph

XY Graph

Description: Like a Standard Line graph, except that you can assign variables to both the X and Y axes. The values plotted show the relationship between the two variables.

Used To: Plot values in one series against those in another.

Values: Up to six series of values can be plotted. Values assigned to the first series are used as the Y-axis scale. X-axis values are used as the X-axis scale.

Figure 1.9: An XY Graph

Grids

Quattro uses a grid (a pattern of horizontal and/or vertical lines) to make it easier to locate values on a graph. You can change the pattern of the grid, assign a different color to a grid, or remove the grid completely from a graph. You can also frame the graph with a rectangular box.

To adjust a graph's grid:

1. **Press** */GCG* to select **G**rids from the Graph Customize menu.
2. To change the color of the grid, **select Color** and choose the color you want to use.
3. To change the pattern of the grid, **select Line Type**. A menu with eight different line types is shown (including | for vertical lines, + for both vertical and horizontal, **F**ull Frame for no grids and an enclosed box, and **N**one for no grids and a half box). Select the type you want to use, then select **Q**uit.
4. To add or remove a thin line boxing the entire graph, select Frame Graph, and select Yes or **N**o.

For more details, see "Adjusting Gridlines" in Chapter 8 of the *Quattro User's Guide.*

Interior Labels

To help define the individual points plotted on a graph, you can label each value within a series with the **I**nterior Labels command (*/GCSI*). For the labels, you can use any group of values in your spreadsheet—either text or numbers. The labels are assigned sequentially to the values in the specified series.

There are two ways to assign interior labels. You can

- select **I**nterior Labels from the Customize Series menu, then select the series you want to assign labels to.
- select the series you want to assign labels to from the Customize Series menu, then select **I**nterior Labels.

The first method is best for assigning inside labels to several series. The second is best when you're making other changes to the same series.

To assign inside labels with the Customize Series menu:

1. **Select *I*nterior Labels** from the Customize Series menu (*/GCSI*).
2. **Select the series** you want to label.
3. **Specify the cells** you want to use to label the points in the series. A menu appears listing placement options.
4. **Select the position** you want for the labels. (**N**one removes the labels from display.)
5. **Repeat these steps** for each series you want to assign labels to.

To assign interior labels by series:

1. **Select the series** you want to assign labels to from the Customize Series menu (*/GCS1...6*).
2. **Press** */I* to select **I**nterior Labels.
3. **Specify the cells** you want to use to label the points in the series. The labels will be placed above the individual points.
4. **To reposition the labels,** press *P* to select **P**lacement of Labels. Then select the position you want from the displayed menu. (Select **N**one to remove the labels altogether.)

For more details, see "Labeling Points on a Graph" in Chapter 8 of the *Quattro User's Guide.*

Label Format for Pie Charts

Each slice of a pie chart is automatically labeled with the percentage that slice contributes to the whole. You can change these labels to display currency or actual spreadsheet values, or remove them entirely.

To change the format of pie slice labels:

1. **Select Label Format** from the Customize Pies menu (*/GCPL*). (If necessary, you can press the EXPAND key (the grey plus key) to display the existing format settings.)
2. **Select the format** you want to use.

Note: To display both spreadsheet values and percentages, specify the spreadsheet values with the **X**-Axis Values command, and leave the Label Format command set to **%**.

For more details, see "Label Format" in Chapter 8 of the *Quattro User's Guide.*

Legends

When you first create a graph, the legend to the right of the graph defines the marker symbols, line color, or fill pattern used for each series of values. It labels each series with the corresponding series number. You can replace these series numbers with your own labels using the **L**egends command. You can also remove this legend from the graph completely.

There are two ways to define a legend. You can

- select Legends from the Customize Series menu, then select the series you want to define in the legend.
- select the series you want to assign labels to from the Customize Series menu, then select Legends.

The first method is best for defining all parts of a legend. The second is best when you're making other changes to the same series.

To define a legend with the Customize Series menu:

1. **Select Legends** from the Customize Series menu (*/GCSL*).

2. **To remove the legend** from the graph, select **O**n/Off, then select **D**isable. To return a hidden legend to the screen, select **O**n/Off, then Enable.
3. **To define the first series,** press *1* to select **1**st Series.
4. **Enter the name** you want to use to define the series.
5. **To define further series,** select the corresponding series commands and enter names for them.

To define a legend by series:

1. **Select the series** you want to define from the Customize Series menu (*/GCS1...6*).
2. **Press** *L* to select **L**egend.
3. **Enter the name** you want to use to define the series.

For more details, see "Defining the Parts of a Legend" in Chapter 8 of the *Quattro User's Guide.*

Markers

A "combined lines and markers" or "markers only" graph uses markers to indicate individual points on a graph. Quattro uses a different marker symbol for each series of values plotted. You can change the symbols with the **M**arkers command (*/GCSM*).

There are two ways to change marker symbols. You can

- select **M**arkers from the Customize Series menu, then select the series you want to change markers for.
- select the series for which you want to change the marker symbol from the Customize Series menu, then select **M**arkers.

The first method is best for reassigning marker symbols to several series. The second is best when you're making other changes to the same series.

To change marker symbols with the Customize Series menu:

1. **Select *M*arkers** from the Customize Series menu (*/GCSM*).
2. **Select the series** you want to change the markers for.
3. **Select the symbol** you want to use.
4. **Repeat the last two steps** for each series you want to change the markers for.

5. **To use the new marker symbols** as the default for all future graphs, select **U**pdate from the Markers menu.

To change the marker symbol for a single series:

1. **Select the series** you want to change from the Customize Series menu (*/GCS#*)
2. **Press** *M* to select **M**arkers.
3. **Select the marker symbol** you want to use for that series.
4. If you want to use the new marker symbol for that series as the default for all future graphs, **press** *Esc* **twice, then select** ***M*****arkers, then** ***U*****pdate.**

For more details, see "Changing the Marker Symbols" in Chapter 8 of the *Quattro User's Guide.*

Named Graphs

The last graph you created in a spreadsheet is saved automatically with the spreadsheet. If you want to store more than one graph with a spreadsheet, you can *name* the graphs. You'll then be able to call up each graph you've named within the spreadsheet in which you created them.

To name a graph:

1. **Select** ***N*****ame** from the Graph menu (*/GN*).
2. **Press** *S* to select **S**ave.
3. **Enter a name** for the graph. (If the name you gave already exists, Quattro replaces the old graph with the new one.)

To display a graph that you've previously named:

1. Display the spreadsheet in which the graph was created, then **select** ***D*****isplay** from the Graph Name menu (*/GND*).
2. **Select a graph name** from the displayed list, or type one in.
3. **Select** ***V*****iew** from a graph menu, or press *F10*, to display the graph.

To delete a previously named graph:

1. **Select** ***E*****rase** from the Graph Name menu (*/GNE*).
2. **Select the name** of the graph you want to delete, or type it in.

To delete *all* named graphs from a spreadsheet file, select **R**eset from the Graph Name menu (*/GNR*).

For more details, see "Storing and Displaying Graphs" in Chapter 8 of the *Quattro User's Guide.*

Override Graph Type

Quattro lets you override the chosen graph type by assigning a different graph type to an individual series. You can combine up to four different graph types in one graph this way.

The graph types that can be combined are

- Lines
- Markers
- Bar
- Combined Lines & Markers
- XY

In order to assign different graph types to individual series, the general graph type must be one of the above.

There are two ways to assign an overriding graph type. You can

- select **O**verride Type from the Customize Series menu, then select the series you want to assign a different graph type to.
- select the series you want to assign a different graph type to from the Customize Series menu, then select **O**verride Type.

The first method is best for assigning overriding graph types to more than one series. The second is best when you're making other changes to the same series.

To assign an overriding graph type to a series using the Customize Series menu:

1. **Set the general graph type** to one of the types listed above with the **G**raph Type command (*/GG*).
2. **Select *O*verride Type** from the Customize Series menu (/GCSO).
3. **Select the series** you want to display in a different graph type.
4. **Select the type of graph** you want to use.
5. **Repeat the last two steps** for each series you want to use a different graph type for.

To assign an overriding graph type by series:

1. Use the Graph Type command (*/GG*) to **set the general graph type** to one of the types listed above.
2. **Select the series** you want to display in a different graph type from the Customize Series menu (*/GCS#*).
3. **Press** */O* to select Override Type.
4. **Select the type of graph** you want to use.

Caution: When you change the general graph type (with the Graph Type command) to anything other than those types listed above, the Override Type settings are erased.

For more details, see "Overriding the Graph Type" in Chapter 8 of the *Quattro User's Guide.*

Patterns of Slices in a Pie Chart

Each slice of a pie chart is displayed with a different fill pattern on your screen. You can change the patterns used for individual slices.

To change the fill patterns used for pie slices:

1. **Select *P*attern** from the Pies menu (*/GCPP*). (If necessary, you can press the EXPAND key (the grey plus key) to display the existing pattern settings.)
2. **Select the slice** you want to change the pattern of. (If your pie chart contains more than nine slices, the patterns will be repeated.)
3. **Select the pattern** you want to fill the slice with.
4. **Repeat the last two steps** for each slice you want to change the pattern of.

For more details, see "Changing Fill Patterns for Pie Slices" in Chapter 8 of the *Quattro User's Guide.*

PIC (Lotus) Files

See "Write to EPS or PIC File" on page 117.

Pies

Pie charts are considerably different from other types of Quattro graphs. For this reason, they have their own set of customization commands.

To customize a pie chart, select **P**ies from the Graph Customize menu (*/GCP*). The Pies menu is displayed, listing the following commands:

***L*abel Format** lets you change the format used to label the pie slices (see page 104). ***E*xplode** lets you pull one or more slices out from the pie (see page 91).

***P*attern** lets you change the patterns used for pie slices (see page 108).

***C*olors** lets you change the colors used for pie slices (see page 89).

***V*iew** displays the current graph.

Each of these options is described (alphabetically) in the following subsections.

Postscript Files

See "Write to EPS or PIC File" on page 117.

Print Colors

When you print your graph on a color plotter, Quattro assumes you want to use the same colors as those used to display the graph on your screen. When the plotter's in **M**anual mode (set with the **G**raph **P**rint **P**rinters command), Quattro pauses between colors and prompts you to insert the next color. If you like, you can specify different colors for printing without changing the colors used on the screen. Quattro then prompts you for those colors.

To use different colors to print your graph:

1. **Select *C*olors** from the Graph Print menu (/GPC).
2. **Select the area** you want to change the color of.
3. **Select the color** you want to use.
4. **Repeat the last two steps** for each area you want to change the color of.
5. **Select *Q*uit.**

6. **If you want to use these colors** as the defaults for all your graphs, press *Esc*, then **U**pdate.

To revert back to using the same colors as those displayed on the screen, select **C**opy the Screen's from the Print Colors menu (/GPCC).

For more details, see "Changing the Colors Used to Print Your Graph" in Chapter 8 of the *Quattro User's Guide.*

Print a Graph

You can print the current graph on a printer, or "print" it to a disk file for future printing. You can specify the exact dimensions of the printed graph and print the graph sideways or straight up.

To print a graph:

1. **Select *P*rint** from the Graph menu (*/GP*).
2. If you haven't already specified information about your printer(s), **select *P*rinters** and select the correct printer make, model, and mode for up to two printers. Then return to the Print menu.
3. If you're using a different printer than before, or if you want to print your graph to a disk file instead of to a printer, **select *D*estination** and tell Quattro where to send your file.
4. To change the way the graph will be laid out on the page, **select *L*ayout** and make the adjustments you want. **L**eft Edge, **T**op Edge, **H**eight, and **W**idth let you specify different dimensions for the printed graph. **M**easurement lets you use centimeters instead of inches to specify dimensions. **O**rientation determines whether or not the graph is printed sideways, and **B**reak Pages determines whether or not the printer moves to the top of the next page after printing a graph.
5. If you're printing on a color plotter or printer and want to use colors different than those displayed on the screen, **select *C*olors** and specify the colors you want to use. Then return to the Graph Print menu.
6. **Select *G*o** when you're ready to begin printing.

For more information on specifying printer information, see "Graphics Printer" in the *Installation* section on page 24.

For details on changing the colors used to print a graph, see page 109.

To write a graph into an EPS (PostScript) or PIC (Lotus-compatible) file, use the **W**rite EPS/PIC command (see page 117).

For more details on printing graphs in general, see "Printing Graphs" in Chapter 8 of the *Quattro User's Guide.*

Reset a Graph

After you've made changes to settings on the graph menus, you can remove some or all of those changes with the Graph Reset command (*/GR*). This command lets you return all graph commands to their default settings, or remove the block assignments given any of the six series.

To reset a graph:

1. **Select *R*eset** from the Graph menu (*/GR*).
2. **Select the area** you want to reset. Graph resets the entire graph. The series commands remove the block assignments from those series. X-Axis removes the block assignment given with the X-Axis Values command.

Reset is also available on several individual graph menus. Use the command to reset settings on those menus.

For more details, see "Resetting a Graph" in Chapter 8 of the *Quattro User's Guide.*

Resolution

If you have a color monitor, you may be able to change the resolution used by your computer screen, and sometimes the colors available for graphics, with the Resolution command (*/GCR*). For example, if you have a CGA card, you can switch to a different color scheme, or use black and white display for better resolution.

To change the resolution used by your screen:

1. **Select *R*esolution** from the Graph Customize menu (*/GCR*).
2. **Select the mode** you want to use.

Quattro uses the new screen mode right away. You'll be able to see its effects by displaying a graph.

For more details, see "Changing Your Screen's Resolution" in Chapter 8 of the *Quattro User's Guide.*

Scale a Graph

Quattro automatically scales your graphs to best display your data. If you're working with an XY or rotated bar graph, however, you can rescale both the *x*-axis and *y*-axis of your graphs to suit your needs.

To rescale the X axis:

1. **Select *X*-Axis** from the Graph Customize menu (*/GCX*).
2. **Select *S*cale**, then select Manual from the displayed menu.
3. **Select *L*ower** and enter the first number you want to appear on the *x*-axis.
4. **Select *U*pper** and enter the last number you want to appear on the *x*-axis.
5. **Select *I*ncrement** and enter the number of the interval you want between tick marks on the axis. If you enter 0, Quattro adds tick marks automatically at regular intervals.

You can rescale the *y*-axis for any type of graph (except pie charts). Use the same procedure as above, except select **Y**-Axis from the Graph Customize menu (*/GCY*). To remove the scaling label on the *y*-axis (such as *Thousands*), select **D**isplay Scaling from the Y-Axis menu (*GCYD*) and choose No.

Note: On a rotated bar graph, the *x*-axis still refers to the horizontal axis, even though the *x*-axis labels are marking the vertical axis.

For more details, see "Adjusting the X-Axis Scale" in Chapter 8 of the *Quattro User's Guide*.

Series Customizing

You can change the way values in a series are displayed in a graph with the Graph **C**ustomize **S**eries command (*/GCS*). When you select this command, the Customize Series menu is displayed, with the following options:

- ***M*arkers** lets you change the markers used to identify values in a series (see page 105).
- ***P*atterns** lets you change the fill pattern used to identify a series (see page 92).
- ***L*egends** lets you define the markers, patterns, or colors included in a graph's legend, or lets you remove the legend from the graph (see page 104).

- **Interior Labels** lets you label the individual points in a graph (see page 103).
- **Override Type** lets you specify a different graph type to be used for one or more individual series (see page 107).
- **1st Series through 6th Series** offer the same customization as the first six commands, but they are grouped together for each of the six series (see page 90).

Series Values

You can plot up to six series of values on a graph. Each series is a specified block of data in your spreadsheet.

To designate a series of values to be plotted on a graph:

1. **Select Series Values** from the Graph menu (/GS).
2. **Select the series** you want to specify.
3. **Indicate the block** you want to assign to the series.

You can also set the series values from within the Customize Series menu:

1. **Select the series** you want to assign values to from the Customize Series menu.
2. **Select Block.**
3. **Specify the block** containing the values you want to use.

To clear values from the series values commands, use the **R**eset command (see page "Reset a Graph" on 111).

For more details, see "Defining Each Series of Values" in Chapter 8 of the *Quattro User's Guide.*

Ticks

Quattro marks off each incremental value on an axis with a tick mark. There are three ways you can change the display of tick marks on an axis:

- Display tick labels on two levels (*x*-axis only).
- Change the display format of tick labels.
- Replace some tick labels with minor tick marks.

Alternate Ticks

If your tick mark labels don't all fit across the *x*-axis, you can display them on two different levels, alternating between levels.

To display tick labels on alternate levels:

1. **Select Alternate Ticks** from the X-Axis menu (*/GCXA*).
2. **Select Yes** from the displayed menu.

To return tick labels to one level, select Alternate Ticks again and choose No.

For more details, see "Alternating Ticks" in Chapter 8 of the *Quattro User's Guide.*

Format of Ticks

The tick labels you assign with the **X**-Axis Values command are displayed across the *x*-axis in the format in which they are displayed in the spreadsheet. Values on the *y*-axis are displayed as integers and are automatically scaled if necessary. You can change the display format of either with the Format of Ticks command.

To change the format of tick mark labels:

1. **Select Format of Ticks** from the Customize X-Axis menu (*/GCXF*) or the Customize Y-Axis menu (*/GCYF*), depending on the axis you want to adjust.
2. **Select the format** in which you want to display the labels.

For more details, see "Format of Ticks" in Chapter 8 of the *Quattro User's Guide.*

Number of Minor Ticks

If all your tick labels don't fit across the *x*-axis, or if the *y*-axis seems cluttered, you can skip over some of the labels, replacing them with minor tick marks.

To skip over some of the tick labels:

1. **Select *No.* of Minor Ticks** from the Customize X-Axis menu (*/GCXN*) or the Customize Y-Axis menu (*/GCYN*), depending on the axis you want to adjust.
2. **Enter the number** of labels you want to skip. If you enter **1**, Quattro displays every other label. **2** displays every third label, **3** displays every fourth, and so on.

For more details, see "Number of Minor Ticks" in Chapter 8 of the *Quattro User's Guide.*

Titles

You can add up to three titles to a Quattro graph: a main title (with two lines), an *x*-axis title, and a *y*-axis title. You can change the size, color, and character font used for each.

Add Titles

To add titles to a graph:

1. **Select *T*itles** from the Graph menu (*/GT*).
2. **Select *1*st Line** to specify the first line of a main title. Then enter the text you want displayed in large letters above the graph.
3. **Select 2nd Line** to specify a second line for the main title. Then enter the text you want displayed in smaller letters between the first line of the main title and the graph.
4. **Select *X*** to title the *x*-axis; enter the text you want displayed below the *x*-axis.
5. **Select *Y*** to title the *y*-axis; enter the text you want displayed vertically to the left of the *y*-axis.

You can alter the size, color, and font of these titles (see the following subsections).

You can also add or change titles to the *x*-axis and *y*-axis using the **C**ustomize X-Axis *(/GCXT)* and **C**ustomize Y-Axis *(/GCYT)* commands. Simply select the command and enter a title. Entering a title with the **T**itles command affects the correlating title command on the Customize menu and vice versa.

For more details, see "Adding Titles" in Chapter 8 of the *Quattro User's Guide.*

Color of Titles

To change the colors used to display your graph's titles:

1. **Select Titles** from the Graph Customize Colors menu (*/GCCT*).
2. **Select the title** you want to change the color of.
3. **Select the color** you want to use for the title.
4. **Repeat the last two steps** for any other titles you want to change the color of.

To change the colors used to print your graph's titles, use the **Graph Print Colors** command (see page 109).

For more details, see "Changing Colors of the Graph" in Chapter 8 of the *Quattro User's Guide.*

Font of Titles

All graph titles are initially displayed in the default type style, or font, which uses plain block letters on the screen and prints using the Triplex font. If you like, you can choose a different font for displaying the main title.

To use a different font for your main title:

1. **Select Font** from the Graph Titles menu (*/GTF*).
2. **Select the font** you want to use for both lines of the main title.

For more details, see "Using a Different Font" in Chapter 8 of the *Quattro User's Guide.*

Size of Titles

The first line of the main titles is automatically sized depending on its length. The second line of the main title and the axes titles appear much smaller. You can adjust the size of each.

To change the size of your graph's titles:

1. **Select Size** from the Titles menu (*/GTS*). **Select the title(s)** you want to adjust.
2. **Select the size** you want for the title(s). **A**utosize adjusts the size according to the length of the title.

For more details, see "Adjusting Title Sizes" in Chapter 8 of the *Quattro User's Guide.*

View Graph

To view the current graph, use the **V**iew command. This command is available on the main Graph menu, as well as most other graphing menus.

You can also display the current graph from within the spreadsheet by pressing the GRAPH key (*F10*).

To remove the graph from the screen and return to the spreadsheet, press *Esc.*

Write to EPS or PIC File

You can save your Quattro graph in an EPS file (which can be incorporated by word-processing programs that use PostScript) or a PIC file (which can be printed with Lotus 1-2-3).

To save a graph in an EPS or PIC file:

1. **Select Write EPS/PIC** from the Graph Print menu (*/GPW*).
2. **Select the type** of file you want to create: EPS (PostScript) or PIC (Lotus-compatible).

Quattro writes the graph to the file immediately. There's no need to select **G**o.

For more details, see "Writing to an EPS or PIC File" in Chapter 8 of the *Quattro User's Guide.*

X-Axis Customizing

You can change the way the *x*-axis is displayed with the Customize **X**-Axis command (*/GCX*). This command displays the X-Axis menu, which allows you to

- add a title to the *x*-axis, or change the existing title (see page 115).
- assign a block of spreadsheet values to the *x*-axis (see page 118).
- rescale the axis (see page 112).
- change the way ticks and tick-mark labels are displayed (see page 113).

X-Axis Values

X-axis values are generally used to label the *x*-axis. (The exception to this is XY graphs, which are discussed in the next section.) *X*-axis values can be any block of cells in your spreadsheet—labels or numeric entries.

To assign *x*-axis values:

1. **Select *X*-Axis Values** from the Graph menu (*/GX*).
2. **Indicate the block** of cells you want to use for labels.

For more details, see "Labeling the *X*-Axis" in Chapter 8 of the *Quattro User's Guide.*

XY Graphs

Unlike other graphs, an XY graph plots values on both the *x*-axis and the *y*-axis. The values you assign to the first series are used to scale the *y*-axis. The *x*-axis values determine the scale of the *x*-axis. The graph then plots the relationship between the two values. If more than one series of values is used, they are plotted in relationship to the *x*-axis values.

To create an XY graph:

1. **Select *XY*** from the Graph Type menu (*/GGX*).
2. **Select *X*-Axis Values** and specify the spreadsheet block to use for the *x*-axis. (All entries in the block must be numeric values.)
3. **Select *S*eries Values, then *1*st** and specify the spreadsheet block to use for the *y*-axis.

4. **If you want to plot further series**, assign them with the Series Values 2nd through 6th commands. Press *Esc* when you're finished.
5. To title the *x*-axis values, **press** *TX* to select **X** from the Titles menu, then enter a title.
6. To title the *y*-axis values, **press** *Y* to select Y from the Titles menu, then enter a title.

For more details, see "XY Graphs: The Exception" in Chapter 8 of the *Quattro User's Guide.*

Y-Axis Customizing

You can change the way the *y*-axis is displayed with the **C**ustomize **Y**-Axis command (*/GCY*). This command displays the Y-Axis menu, which allows you to:

- Add a title to the *y*-axis, or change the existing title (see page 115).
- Rescale the axis (see page 112).
- Change the way ticks and tick-mark labels are displayed (see page 113).

Macros

A macro is a recorded sequence of keystrokes and/or commands that can be invoked with a single command. Quattro allows you to store an unlimited number of macros in each spreadsheet. Each macro is given a name, which you can then use to invoke the macro.

Although you can create a macro by typing keystrokes in a cell and assigning a block name to the cell, the most efficient way to create a macro is by recording it. In Macro Record mode, Quattro records each step you take and stores it in part of the spreadsheet. Those steps are then repeated each time you execute the macro.

The following subsections describe how to record, execute, debug, delete, and name a macro. For complete details on macros, see Chapter 12 of the *Quattro User's Guide*.

Abort a Macro in Debug Mode

When a macro is finished executing in Debug mode, the Debug Window disappears. To exit Debug mode, press the DEBUG key (*Shift-F8*) again.

To abort a macro being debugged before it's finished, press / and select Abort from the Debug menu. The Debug Window disappears. You can debug another macro, or press *Shift-F8* to exit Debug mode.

Auto-Execute Macro

See "Startup Macro" on page 125.

Breakpoints

When you're debugging a macro, Quattro executes the macro one step at a time. If you know that most of a macro is correct, you can use *breakpoints* to execute a macro at full speed up to a certain point, then begin Single-Step mode. There are two kinds of breakpoints you can set: *standard breakpoints* and *conditional breakpoints*. Standard breakpoints suspend execution when the breakpoint cell is reached. Conditional breakpoints suspend execution when the condition stored in the conditional breakpoint cell is reached.

To resume a suspended macro, press the space bar to continue in Single-Step mode, or press *Enter* to continue at full speed until the next breakpoint or until the end of the macro.

You can set up to four standard breakpoints and four conditional breakpoints per spreadsheet.

To set a breakpoint:

1. **Press the slash key** (/) from within the Debug Window.
2. **Press** *B* to select **B**reakpoint.
3. **Select the number of the breakpoint** you want to set.
4. **Select *B*lock** and specify the cell or block of cells at which you want the macro to stop.
5. If you want the macro to be executed more than once before stopping, **select *P*ass Count** and specify the number of passes you want the macro to run before stopping.
6. Select **Q**uit to return to the Debug Window.

To set a conditional breakpoint:

1. **Select Conditional** from the Debug menu.
2. **Select the number of the breakpoint** you want to set.
3. **Specify the cell** containing the condition.
4. **Select *Q*uit** to return to the Debug Window.

When you execute the macro, Quattro performs each step at full speed until the specified cell or condition is reached. It then suspends execution until you press the space bar or *Enter*.

To clear all breakpoints (and trace cells) you've set, select **R**eset from the Debug menu (*/R*). You can also clear breakpoints by selecting **C**lear Breakpoints from the Macro menu once you've exited Debug mode (*/MC*).

For more details, see "Suspending Macro Execution" in Chapter 12 of the *Quattro User's Guide*.

Call a Macro

See "Execute a Macro" on page 123.

Create a Macro with Transcript

An easy way to create macros is to copy sections of your command history recorded with Transcript. You can copy any marked-off section of the history directly into your spreadsheet, then assign the block a macro name.

For details on Transcript, see "The Transcript Add-In" in Chapter 13 of the *Quattro User's Guide*.

Debug a Macro

To isolate specific problem areas in a macro, you can use the macro debug function. To debug a macro, press the DEBUG key (*Shift-F8*), then execute the macro. A Debug Window appears in the bottom half of the screen, showing the macro you're debugging.

To execute the first step of the macro (the first keystroke or command), press the space bar. Press the space bar repeatedly to execute each step of the macro until you pinpoint the error. To execute the rest of the actions in the macro at full speed, press *Enter*.

To display the Debug menu, press the slash key (/) from within the Debug Window. The Debug Menu contains commands that let you

- **Set conditional or standard breakpoints** at which macro execution will be suspended (see page 120).
- **Set trace cells** whose contents will be displayed in the Debug Window while the macro is executing (see page 126).
- **Abort the macro** and exit Debug mode (see page 120).
- **Reset any breakpoints** you set for the macro (see page 125).
- **Edit a cell** without leaving Debug mode (see page 123).

For more details, see "Debugging a Macro" in Chapter 12 of the *Quattro User's Guide*.

Delete a Macro

To delete a macro, you need to delete both its name and its contents:

1. **Press** */MD* to select **D**elete from the Macro menu.
2. **Select the macro name** you want to delete. The name is deleted.

3. **To delete the *contents*** of the macro, press *Esc* to return to the main menu, then press *BE* to select Erase from the Block menu.
4. **Specify the block** containing the macro.

For more details, see "Deleting a Macro" in Chapter 12 of the *Quattro User's Guide.*

Edit a Macro

When you're debugging a macro, you can use the Edit a Cell command on the Debug menu to edit the macro without exiting Debug mode.

When you select Edit a Cell from the Debug menu (*/E*), Quattro prompts you for the address of the cell you want to edit. Type in the address (you can't point in Debug mode). Quattro then displays the contents of the cell on the input line. You can make changes to the cell just as you would in regular Edit mode. When you press *Enter*, Quattro enters the changes and returns you to the Debug Window.

For more details, see "Editing a Macro" and "Editing a Cell in Debug Mode" in Chapter 8 of the *Quattro User's Guide.*

Execute a Macro

If the macro was given an alphabetic name, such as \a or \f, you can execute it simply by holding down the *Alt* key and pressing the letter following the backslash. All other macros can be executed using the MACRO key or the Macro Execute command.

To execute a macro with the MACRO key:

1. **Press the MACRO key (*F8*).**
2. **Select the macro** you want to execute from the displayed list, or type it in.

To execute a macro with the Macro Execute command:

1. **Press** */ME* to select Execute from the Macro menu.
2. **Move the selector** to the first cell of the block containing the macro you want to execute, or type in the macro's name.
3. **Press** *Enter.*

For more details, see "Executing a Macro" in Chapter 8 of the *Quattro User's Guide*.

Keystroke Macro Recording

See "Macro Recording" in the *Installation* section on page 21.

Name a Macro

When you record a macro, Quattro prompts you for a macro name and automatically assigns that name to the block containing the macro. If you create a macro by entering keystrokes in a cell as a label, you'll need to name the block containing the macro before it can be executed.

To name a macro:

1. **Press** */MN* to select Name from the Macro menu.
2. **Enter the name** you want to give the macro. To be able to execute it using the *Alt* key, use an alphabetic name (a backslash followed by one letter).
3. **Indicate the first cell** of the block containing the macro.

For more details, see "Naming a Macro" in Chapter 8 of the *Quattro User's Guide*.

Record a Macro

To record a macro:

1. **Hold down the** *Alt* **key** and press *F8*.
2. **Enter a name** for the macro.
3. **Specify the cell** in which you want to store the macro. Make sure there are plenty of empty cells underneath it, to store the overflow. Quattro enters Record mode.
4. **Enter the exact keystrokes** and/or commands you want stored in the macro.
5. **When you've completed** the sequence you want recorded, press *Alt-F8* again. Quattro exits Record mode.

Note: Normally, Quattro translates your recorded actions into *menu-equivalent commands* that can be used with any Quattro interface, as well as with Lotus 1-2-3. (See Chapter 4 for tables listing these commands.) If you prefer to record actual keystrokes instead (for compatibility with other products), set the **M**acro Recording default to **K**eystroke (see "Macro Recording" on page 21).

For more details, see "Recording a Macro" in Chapter 8 of the *Quattro User's Guide.*

Reset Macro Breakpoints

To remove all breakpoints (standard and conditional) and trace cells set for the spreadsheet, select **R**eset from the Debug menu (press */R* while in the Debug Window).

You can also reset breakpoints and trace cells from within the spreadsheet (instead of the Debug Window). Just select **C**lear Breakpoints from the Macro menu (*/MC*).

For more details, see "Resetting Breakpoints and Trace Cells" in Chapter 12 of the *Quattro User's Guide.*

Startup Macro

If you create a macro in Quattro and give it the name \0, it will be executed automatically every time you retrieve the spreadsheet. This is called an *auto-execute*, or *startup*, macro.

To change the name of the macro that will be executed automatically:

1. **Press** */DSS* to select **S**tartup Macro from the Default Startup menu.
2. **Enter the name** of a macro.
3. **Press** *Esc* to exit the Startup menu.
4. **Press** *U* to select **U**pdate from the Default menu.

Each time you load a spreadsheet, Quattro looks for a macro with the name specified with the **S**tartup **M**acro command. If it finds one, it loads it automatically.

For more details, see "Autoload Macro" in Chapter 5 of the *Quattro User's Guide.*

Trace Cells

Macros often affect the contents of one or more specific cells. By monitoring the contents of these cells during debugging, you can see more clearly what the macro is doing.

Quattro lets you specify up to four *trace cells*, whose contents are shown during debugging in the Trace section of the Debug Window. The window is updated instantly as the contents of the trace cells change.

To set trace cells:

1. **Press** /while the Debug Window is displayed.
2. **Press** *T* to select **T**race Cells from the menu.
3. **Select the trace cell** you want to set.
4. **Specify the cell** you want to trace.
5. **Select Quit** to return to the Debug Window.

To clear all trace cells (and breakpoints) you've set, select **R**eset from the Debug menu (*/R*). You can also clear trace cells and breakpoints by selecting **C**lear Breakpoints from the Macro menu once you've exited Debug mode (*/MC*).

For more details, see "Setting Trace Cells" in Chapter 8 of the *Quattro User's Guide.*

C H A P T E R

2

@Function Commands

Functions by Type

Table 2.1 lists and briefly describes each of the @functions available within each category. For detailed descriptions of each @function, see "Function Descriptions" on page 133.

Table 2.1: Functions Listed by Category

Function	Returns
Mathematical Functions	
@ABS(*X*)	The absolute value of X
@ACOS(*X*)	The arc cosine of X
@ASIN(*X*)	The arc sine of X
@ATAN(*X*)	The arc tangent of X (2 quadrant)
@ATAN2(*X,Y*)	The arc tangent of X/Y (4 quadrant)
@COS(*X*)	The cosine of X
@DEGREES(*X*)	The number of degrees in X radians
@EXP(*X*)	e raised to the X power
@INT(*X*)	The integer portion of X
@LN(*X*)	The Log base e of X
@LOG(*X*)	The Log base 10 of X
@MOD(*X,Y*)	The remainder of X/Y
@PI	The value pi (3.14159..)
@RADIANS(*X*)	The number of radians in X degrees
@RAND	A random number between 0 and 1
@ROUND(*X,N*)	X rounded to the number of digits specified (up to 15)
@SIN(*X*)	The sine of X
@SQRT(*X*)	The square root of X
@TAN(*X*)	The tangent of X
Aggregation and Counting Functions	
@AVG(*List*)	The average of (List)
@COUNT(*List*)	A value equal to the number of non-blank cells in (List)
@MAX(*List*)	The maximum value in (List)
@MIN(*List*)	The minimum value in (List)
@STD(*List*)	The standard deviation of all non-blank values in (List)
@SUM(*List*)	The sum of values in (List)
@VAR(*List*)	The variance of all non-blank values in (List)

Table 2.1: Functions Listed by Category, continued

Function	Returns
String Functions	
@CHAR(*X*)	The ASCII character for code number X
@CLEAN(*String*)	Returns non-printable ASCII codes from a string
@CODE(*String*)	The ASCII code for the first character in String
@EXACT(*String1,String2*)	1 if String1 and String2 are identical; otherwise 0
@FIND(*SubString,String, StartNumber*)	The character position of the first SubString found in String
@HEXTONUM(*String*)	The numerical value of hexidecimal string
@LEFT(*String,N*)	The first N characters in String
@LENGTH(*String*)	The number of characters in String
@LOWER(*String*)	The lowercase value of String
@MID(*String, StartNumber,N*)	N characters of String, beginning with the StartNumber character position
@N(*Block*)	The numeric value of the upper-left cell in Block (0 if it's a label)
@NUMTOHEX(*X*)	The hexidecimal value of X
@PROPER(*String*)	The text in String with the first letter in each word capitalized
@REPEAT(*String,N*)	String, repeated N times
@REPLACE(*String, StartNum,N,NewString*)	Removes N characters from String, beginning with StartNum, then inserts NewString in its place
@RIGHT(*String,N*)	The last N characters in String
@S(*Block*)	The string value of the upper-left cell in Block (0 if it's a value entry)
@STRING(*X,N*)	Numeric value of X as a string, with N decimal places
@TRIM(*String*)	String without leading, trailing, or consecutive spaces
@UPPER(*String*)	String in upper case
@VALUE(*String*)	Numeric value of String

Table 2.1: Functions Listed by Category, continued

Function	Returns
Miscellaneous Functions	
@@(*Cell Address*)	The contents of the cell specified by Cell Address
@CELL(*Attribute,Block*)	The requested attribute of Block
@CELLINDEX(*Attribute, Block,Column,Row*)	The requested attribute of the cell in the offset position of Block
@CELLPOINTER(*Attribute*)	The requested attribute of the current cell
@CHOOSE(*Number,List*)	The value in List in the position of Number
@COLS(*Block*)	The number of columns in Block
@CURVALUE(*GeneralAction, SpecificAction*)	The current value of the given menu command
@ERR	The value ERR (error)
@HLOOKUP(*X,Block,Row*)	The contents of the cell Row number of rows beneath X in Block
@INDEX(*Block,Column,Row*)	The contents of the cell located at the specified column and row in Block
@MEMAVAIL	The amount of memory currently available
@MEMEMSAVAIL	The amount of external memory currently available
@NA	The value of NA (not available)
@ROWS(*Block*)	The number of rows in Block
@VLOOKUP(*X,Block,Column*)	The contents of the cell Column number of columns to the right of X in Block
Logical Functions	
@FALSE	The logical value 0
@FILEEXISTS(*FileName*)	1 if the given file name exists, otherwise 0
@IF(*Cond,X,Y*)	X if Cond is true, Y if Cond is false
@ISERR(*X*)	1 if X is ERR, otherwise 0
@ISNA(*X*)	1 if X is NA, otherwise 0
@ISNUMBER(*X*)	1 if X is a number, otherwise 0
@ISSTRING(*X*)	1 if X is a string, otherwise 0
@TRUE	The logical value 1

Table 2.1: Functions Listed by Category, continued

Function	Returns
Financial Functions	
@CTERM(*Int,Fv,Pv*)	The number of compounding periods
@DDB(*Cost,Salvage,Life, Period*)	Double-declining depreciation allowance
@FV(*Pmt,Int,Term*)	The future value of an annuity
@IRR(*Guess,Block*)	The internal rate of return
@NPV(*Int,Block*)	The present value of future cash flow
@PMT(*Prin,Int,Term*)	The payment amount for a loan
@PV(*Pmt,Int,Term*)	The present value of an annuity
@RATE(*Fv,Pv,Term*)	The periodic interest rate
@SLN(*Cost,Salvage,Life*)	Straight-line depreciation allowance
@SYD(*Cost,Salvage,Life, Period*)	Sum-of-the-years'-digits' depreciation allowance of an asset
@TERM(*Pmt,Int,Fv*)	The number of payment periods of an investment
Date & Time Functions	
@DATE(*Yr,Mo,Day*)	A date serial number
@DATEVALUE(*DateString*)	A date serial number
@DAY(*DateNumber*)	The day of the month (1-31)
@HOUR(*TimeNumber*)	The hour of the day (1-23)
@MINUTE(*TimeNumber*)	The minute of the hour (1-59)
@MONTH(*DateNumber*)	A month number (1-12)
@NOW	The current date/time serial number
@SECOND(*TimeNumber*)	A second number (1-59)
@TIME(*Hr,Min,Sec*)	A time serial number
@TIMEVALUE(*TimeString*)	A time serial number
@TODAY	The current date
@YEAR(*DateNumber*)	A year number (0-199)

Table 2.1: Functions Listed by Category, continued

Function	Returns
Database Aggregation Functions	
@DAVG(*Block,Column, Criteria*)	Average values in Column of Block that meet Criteria
@DCOUNT(*Block,Column, Criteria*)	Number of values in Column of Block that meet Criteria
@DMAX(*Block,Column, Criteria*)	Maximum value in Column of Block that meet Criteria
@DMIN(*Block,Column, Criteria*)	Minimum value in Column of Block that meet Criteria
@DSTD(*Block,Column, Criteria*)	Standard deviation of values in Column of Block that meet Criteria
@DSUM(*Block,Column, Criteria*)	Sum of values in Column of Block that meet Criteria
@DVAR(*Block,Column, Criteria*)	Variance of values in Column of Block that meet Criteria

Function Descriptions

This section describes each of the @functions available with Quattro, listed in alphabetical order. It shows format, argument requirements, and examples for each.

To see brief descriptions of each @function listed under each type of function, refer to the previous section, beginning on page 127.

@@

Format: @@(*Cell*)

Cell = a single cell address

@@ is used to reference a cell that contains another cell address or block name that is written as a label. @@ translates the label into a proper cell or single-cell block reference and returns the contents of that cell.

Examples

@@(A15) = the contents of the cell referenced in A15
@@("A15") = the contents of A15
@@("BLOCK_NAME") = the contents of the top-left cell in the block named BLOCK_NAME
@@(A3) = 50 if A3 contains the label 'A1 and cell A1 contains the value 50

@@(A3) = Total if A3 contains the label 'Block, which is the name of cell C9, which contains the label 'Total

@ABS

Format: @ABS(X)

X = a numeric value

@ABS returns the absolute (positive) value of *X*.

Examples

@ABS(-100) = 100
@ABS(100) = 100
@ABS(0) = 0
@ABS(-13.90%) = 0.139

@ABS(A2) = positive value of A2
@ABS(A2-10) = positive value of ten less than the value in A2

@ACOS

Format: @ACOS(X)

X = a numeric value between –1 and 1

@ACOS returns the arc cosine of angle *X*. The result is the angle (in radians) whose cosine is *X*. To convert radians to degrees, use the @DEGREES function (see page 150).

Examples

@ACOS(1) = 0
@ACOS(0.5) = 1.047198
@DEGREES(@ACOS(0.5)) = 60
@ACOS(@ABS(B10)) = the cosine of the positive value of B10
@ACOS(2) = ERR (X is greater than 1)

@ASIN

Format: @ASIN(X)

X = a numeric value between –1 and 1

@ASIN calculates the arc sine of angle *X*. The result is the angle (in radians) whose sine is *X*. To convert radians to degrees, use the @DEGREES function (see page 150).

Examples

@ASIN(1) = 1.570796
@ASIN(0.25) = 0.25268
@DEGREES(@ASIN(0.5)) = 30
@ASIN(-2) = ERR(X is less than –1)
@ASIN(-2) = ERR (X is less than –1)

@ATAN

Format: @ATAN(X)

X = a numeric value

@ATAN calculates the arc tangent of angle *X*. The result is the angle (in radians) whose tangent is *X*. To convert radians to degrees, use the @DEGREES function (see page 150).

Examples

@ATAN(0.5) = 0.463647
@ATAN(1) = 0.7853982
@DEGREES(@ATAN(1)) = 45

@ATAN2

Format: @ATAN(*X*,*Y*)

X = a numeric value
Y = a numeric value <> *X*

@ATAN2 calculates the arc tangent of the angle represented by the point with x/y coordinates *X* and *Y*. The result is the angle (in radians) whose tangent is y/x. To convert radians to degrees, use the @DEGREES function (see page 150).

Examples

@ATAN2(1,2) = 1.107148
@ATAN2(1,1) = 0.785398
@DEGREES(@ATAN2(1,1)) = 45

@AVG

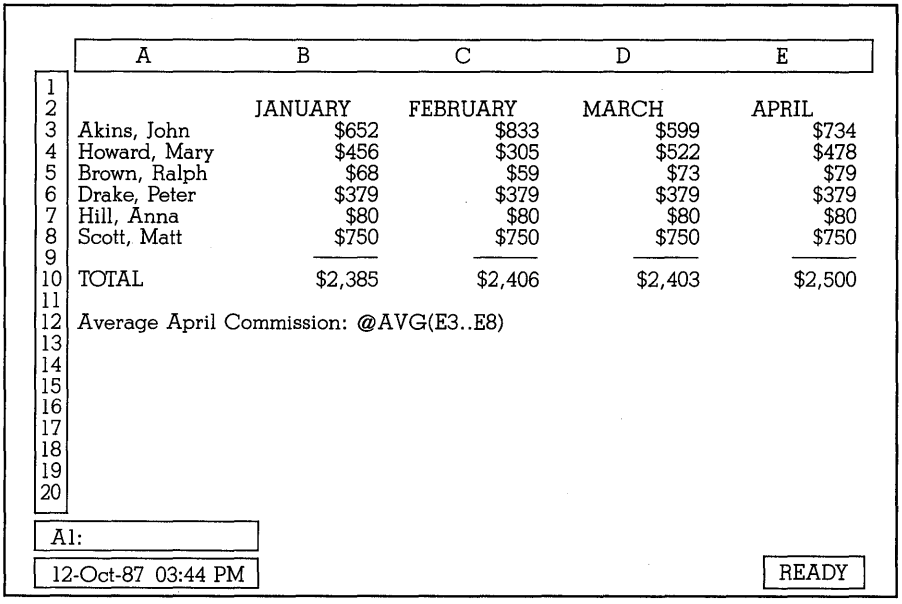
Format: @AVG(*List*)

List = one or more numeric or block values

@AVG calculates the average of all values in *List*. If more than one block is listed, they must be separated by commas. If any of the cells referenced contains ERR, the resulting value is ERR.

Examples

The following spreadsheet uses the @AVG function to calculate the average commission in April. The examples below refer to other cells in the same spreadsheet.



	A	B	C	D	E
1					
2		JANUARY	FEBRUARY	MARCH	APRIL
3	Akins, John	$652	$833	$599	$734
4	Howard, Mary	$456	$305	$522	$478
5	Brown, Ralph	$68	$59	$73	$79
6	Drake, Peter	$379	$379	$379	$379
7	Hill, Anna	$80	$80	$80	$80
8	Scott, Matt	$750	$750	$750	$750
9					
10	TOTAL	$2,385	$2,406	$2,403	$2,500
11					
12	Average April Commission: @AVG(E3..E8)				
13					
14					
15					
16					
17					
18					
19					
20					

Examples

@AVG(B3..E3) = $704.50
@AVG(B10..E10) = $2,423.50
@AVG(B3..B8,D3..D8) =$400.50

@CELL

Format: @CELL(*Attribute,Block*)

Attribute = any one of the attributes listed below

Block = a block value

@CELL returns the requested attribute of the upper-left cell in *Block*.

You can enter attributes in either upper or lower case, but you must surround them with double quotes. you can also reference a cell containing an attribute.

The following attributes are allowed:

"address" The address of the upper-left cell in *Block*.

"row" The row number of the upper-left cell in *Block* (from 1 to 8192).

"col" The column number of the upper-left cell in *Block (from 1 to 256 corresponding to columns A through IV)*

"contents" The actual contents of the upper-left cell in *Block*

"type" The type of data in the upper-left cell in *Block*:

b if cell is blank
v if the cell contains a number or any formula
l if the cell contains a label

"prefix" The label-prefix character of the upper-left cell in *Block*:

' if label is left-aligned
^ if label is centered
" if label is right-aligned
\ if label is repeating

"protect" The protected status of the upper-left cell in *Block*:

0 if cell is not protected
1 if cell is protected

"width" The width of the column containing the upper-left cell in *Block* (between 1 and 240)

"format" The current display format of the upper-left cell in *Block*:

Fn is Fixed (n = 0-15)
En is Exponential (n = 0-15)
Cn is Currency (n = 0-15)
+ is +/- (bar graph format)
G is General
Pn is Percent (n = 0-15)
D1-D5 is Date
D1 = DD-MMM-YY
D2 = DD-MMM
D3 = MMM-YY
D4 = MM/DD/YY, DD/MM/YY, DD.MM.YY, YY-MM-DD
D5 = MM/DD, DD/MM, DD.MM, MM-DD

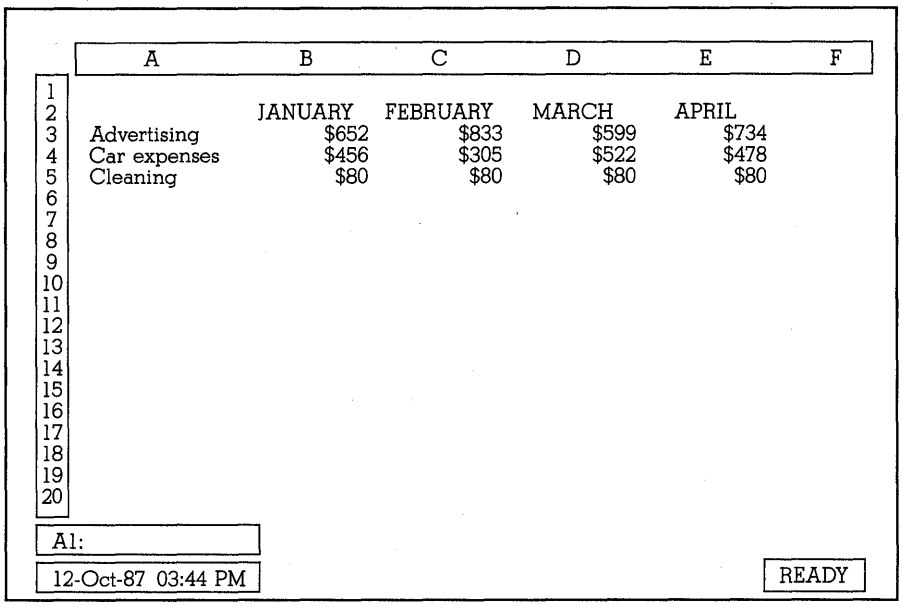
D6-D9 is Time
D6 = HH:MM:SS AM/PM
D7 = HH:MM AM/PM
D8 = HH:MM:SS-24hr, HH.MM.SS-24hr, HH,MM,SS-24hr, HHhMMmSSs
D9 = HH:MM-24hr, HH.MM-24hr, HH,MM, HHhMMm

T is Show Formulas (Text)

H is Hidden
, is Commas used to separate thousands

Examples

The following examples refer to cells in the spreadsheet below.



	A	B	C	D	E	F
1						
2		JANUARY	FEBRUARY	MARCH	APRIL	
3	Advertising	$652	$833	$599	$734	
4	Car expenses	$456	$305	$522	$478	
5	Cleaning	$80	$80	$80	$80	
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						

A1:
12-Oct-87 03:44 PM
READY

@CELL("prefix",A3) = '
@CELL("format",B5) = C0
@CELL("type",D4) = v
@CELL("address",A3) = A3
@CELL("row",B4) = 4

@CELLINDEX

Format: @CELLINDEX(*Block,Col,Row*)

Same as Cell Pointer, but returns the requested attribute of the cell in the specified column and specified row of Block.

@CELLPOINTER

Format: @CELLPOINTER(*Attribute*)

Attribute = one or more of the possible attributes

@CELLPOINTER is similar to @CELL in that it returns the requested attribute of a cell. The only difference is that it reads the cell containing the cell selector. You cannot specify another cell. However, if you move the cell selector to a different cell, then press *F9*, CALC, the results of the @CELLPOINTER formula are updated.

The attributes available are the same as those used with @CELL (see page 136). You can enter attribute names in either upper or lower case, but each must be surrounded by double quotes.

This function is useful in macros and @IF functions for quickly determining certain characteristics about the current cell, such as whether there is a label or a value currently in it. For example, the function statement:

@IF(@CELLPOINTER("type")=v,"value","label")

tells Quattro to write "protected" in the cell if the current cell is write-protected; otherwise, it writes "unprotected."

Examples

The following examples refer to cell A1, which contains the date value 11/19/87.

@CELLPOINTER("address") = A1
@CELLPOINTER("col") = 1
@CELLPOINTER("contents") = 32100
@CELLPOINTER("format") = D4
@CELLPOINTER("type") = v

@CHAR

Format: @CHAR(*Code*)

Code = a numeric value between 1 and 255

@CHAR returns the ASCII character corresponding to the given code. This is useful in generating symbols not found on the keyboard, such as ASCII graphics characters.

Refer to any standard ASCII table for the codes corresponding to each character.

@CHAR is especially useful in setting up an ASCII table with your spreadsheet. Fill a column of cells with values from 1 to 240, using the **B**lock **F**ill command (*/BF*). In the cell to the right of the first number, use the

@CHAR function to find the ASCII character for 1, for example @CHAR(A1). Then copy the formula down the column for the next 239 cells. The copied formulas will display the ASCII character for each number.

Examples

@CHAR(33) = !
@CHAR(34) = "
@CHAR(35) = #
@CHAR(36) = $
@CHAR(172) = 1/4
@CHAR(174) = «

@CHOOSE

Format: @CHOOSE(*Number,List*)

Number = a numeric value <= the number of items in *List*

List = a group of numeric or string values separated by commas

@CHOOSE selects and enters a value from the supplied list. The value it chooses depends on the value of *Number*. 0 chooses the first value in the list; 1 chooses the second; 2 chooses the third, and so on. If you specify a cell address for the *Number* argument, Quattro uses the number contained in the cell. If the cell is blank, the first value is chosen.

The values can be cell addresses, strings, numbers, or a mixture of the three. The total characters entered in the list, however, must not exceed 240.

@CHOOSE operates on integers only. If you supply a non-integer (such as 1.6433), the decimal values are disregarded.

Examples

@CHOOSE(0,"Howie","Sarah","Chris") = Howie
@CHOOSE(1,"Howie","Sarah","Chris") = Sarah
@CHOOSE(2,"Howie","Sarah","Chris") = Chris
@CHOOSE(A15,"Howie","Sarah","Chris") = Howie, if A15 is 0; Sarah if A15 is 1; Chris if A15 is 2.
@CHOOSE(3,"Howie","Sarah","Chris") = ERR (Number is too great).

See also @VLOOKUP and @HLOOKUP, functions that perform similar tasks.

@CLEAN

@CLEAN

Format: @CLEAN(*String*)

String = a string value @CLEAN removes all non-ASCII characters from a string. It is included here for compatibility with other products.

@CODE

Format: @CODE(*String*)

String = a string value

@CODE returns the ASCII code of the first character in *String*. This is the opposite of the @CHAR function, which returns the character corresponding to the given code.

Examples

@CODE("!") = 33
@CODE("Sam") = 83 (code for S)
@CODE("#") = 35
@CODE("$") = 36
@CODE("?") = 63
@CODE(hello) = syntax error (missing quotes)

@COLS

Format: @COLS(*Block*)

Block = a block value

@COLS returns the number of columns within the given block.

Examples

@COLS(A1..IV1) = 256
@COLS(A1..A1) = 1
@COLS(NAME) = 30 (if the block named NAME contains 30 columns

@COS

Format: @COS(X)

X = a numeric value

@COS returns the cosine of the angle *X*. *X* must be given in radians, not degrees. To convert degrees to radians, use the @RADIANS function. (See page 186.)

Examples

@COS(@RADIANS(60)) = 0.5
@COS(@RADIANS(75)) = 0.258819045
@COS(@RADIANS(45)) = 0.707106781

@COUNT

Format: @COUNT(*List*)

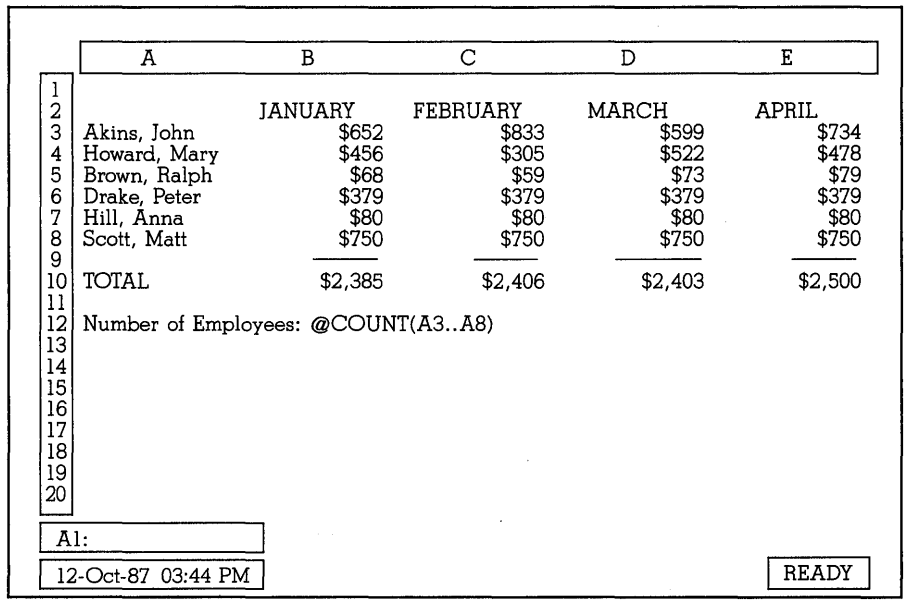
List = one or more numeric or block values, separated by commas

@COUNT returns the number of non-blank cells in *List*. If more than one block is listed, they must be separated by commas.

Any single cells in *List* are counted as 1, even if they are blank. This can be worked around by always using blocks [for example, (A3..A4) instead of just (A3)].

Examples

The following spreadsheet uses @COUNT to total the number of employees (6). The examples below refer to other cells in the same spreadsheet.



	A	B	C	D	E
1					
2		JANUARY	FEBRUARY	MARCH	APRIL
3	Akins, John	$652	$833	$599	$734
4	Howard, Mary	$456	$305	$522	$478
5	Brown, Ralph	$68	$59	$73	$79
6	Drake, Peter	$379	$379	$379	$379
7	Hill, Anna	$80	$80	$80	$80
8	Scott, Matt	$750	$750	$750	$750
9					
10	TOTAL	$2,385	$2,406	$2,403	$2,500
11					
12	Number of Employees: @COUNT(A3..A8)				
13					
14					
15					
16					
17					
18					
19					
20					

A1:

12-Oct-87 03:44 PM

READY

@COUNT(B3..B8) = 6
@COUNT(C1..C12) = 10
@COUNT(A11) = 1
@COUNT(A11..B11) = 0
@COUNT(D3..D9,E1..E4) = 11
@COUNT(A1..A5,B12,D3..D12) = 12

@CTERM

Format: @CTERM(*Interest,FutValue,PresValue*)

Interest = a numeric value representing the fixed interest rate per compounding period

FutValue = a numeric value representing the value the investment will reach at some point

PresValue = a numeric value representing the present value of the investment

@CTERM calculates the number of time periods required for an investment of *PresValue* to reach a value of *FutValue,* while earning interest of *Interest* per compounding period. It used the following formula:

natural log (*FutValue/PresValue*)
natural log (1 + *Interest*)

Examples

@CTERM(0.07,5000,3000) = 7.550042
@CTERM(0.10,5000,3000) = 5.359612
@CTERM(0.12,5000,3000) = 4.50747
@CTERM(0.12,10000,7000) = 3.147261

@CURVALUE

Format: @CURVALUE(*GeneralAction,SpecificAction*)

GeneralAction = a general menu category
SpecificAction = a menu item that requires setting

@CURVALUE returns the current value of a menu command setting. It is used in macros, usually to base the next action on a particular menu setting. Both *ActionCategory* and *Action* must be surrounded by double quotes. They must together create one of the menu-equivalent commands listed in Chapter 4.

Examples

@CURVALUE("print","block") = the block currently specified with the **B**lock Print command
@CURVALUE("file","save") = the name of the last file retrieved

@DATE

Format: @DATE(*Yr,Mo,Day*)

Yr = a numeric value between 0 and 199
Mo = a numeric value between 1 and 12
Day = a numeric value between 1 and 31

@DATE returns the "serial number" of the date specified with year, month, and day arguments. This serial number can range from 0 to 73,050, and represents the number of days from December 31, 1899 up to the date referenced in the formula. The highest date available is December 31, 2099. Date serial numbers are used in spreadsheet calculations. (The fractional portion of a date serial number is used for the time functions.)

To display a date serial number in a date format, use the **B**lock **D**isplay Format **D**ate command (see page 41), or change the default display format

to Date (*\DFDD*). This suppresses the numeric display, showing instead the date in its more common form (for example, Jan-1-87 instead of 31778).

Any illegal dates [for example, @DATE(87,2,29)] return ERR as their value. (This date corresponds to February 29, 1987, which is impossible; 1987 is not a leap year.)

Examples

@DATE(87,1,1) = 31778
@DATE(87,9,13) = 32033
@DATE(00,1,1) = 1

@DATEVALUE

Format: @DATEVALUE(*DateString*)

> *DateString* = a string value in any valid date format, surrounded by quotes

@DATEVALUE returns a serial date value that corresponds to the value in *DateString*. If the value in *DateString* is not in the correct format, or is not enclosed in quotes, an ERR value is returned.

You can display resulting date string values in standard date formats using the **B**lock **D**isplay Format **D**ate command, or by changing the default display format to **D**ate (see page 41).

There are five valid formats for *DateString*:

- DD-MMM-YY ("04-Jul-87")
- DD-MMM ("04-Jul") (assumes the current year)
- MMM-YY ("Jul-87") (assumes the first of the month)
- The Long International date format specified as the system default, one of which is MM/DD/YY ("07/04/87")
- The Short International date format specified as the system default, one of which is MM/DD ("07/18"). This format assumes the current year.
- **NOTE:** The easiest way to enter a date value in Quattro is with the date prefix (*Ctrl-D*). The @DATEVALUE function is included, however, for compatibility with other products.

Examples

@DATEVALUE("07/04/87") = 31962
@DATEVALUE("JUL-86") = 31594
@DATEVALUE("07/18") = 31976

@DATEVALUE("04-may") = 31901
@DATEVALUE("May 1 1987") = syntax error (an unacceptable format)
@DATEVALUE(07/04/87) = ERR (no quotes)

@DAVG

Format: @DAVG(*Block,Column,Criteria*)

Block = the cell block containing the database, including field names

Column = the number of the column containing the field you want to average. The first column is 0, second is 1, and so on.

Criteria = a logical expression or a cell block containing search criteria

@DAVG averages selected field entries in a database. It includes only those entries in Column number *Column* whose records meet the criteria specified in *Criteria*.

Criteria can be a search formula within quotes, such as "F4>20000" or "E4='Tom'", or it can be the coordinate of a block containing a *criteria table*, specifying search information.

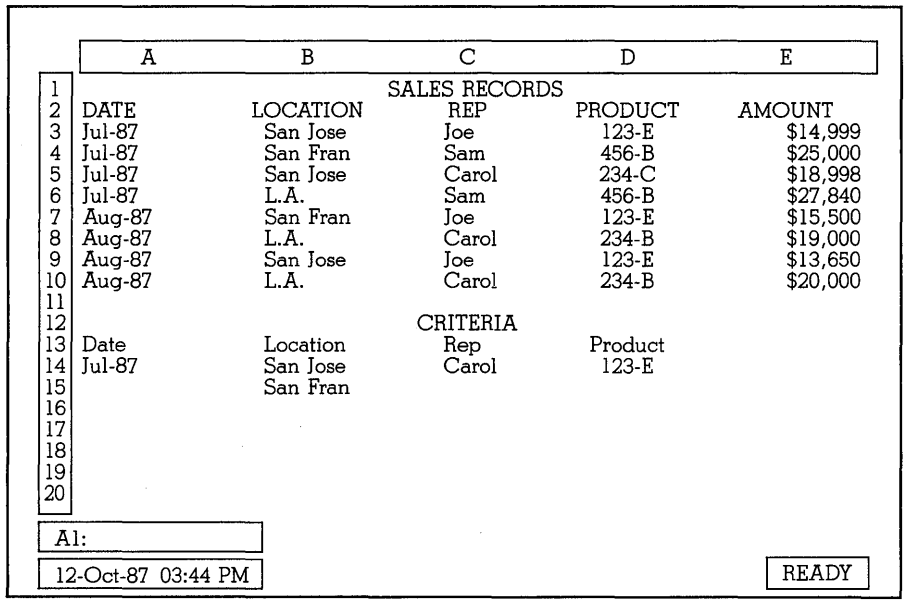
If you use a search formula for *Criteria*, it must be surrounded by quotes. You can use single quotes to surround text in the formula that would normally require double quotes. And, the formula must reference the first cell in the column to be searched. For more details on search formulas and criteria tables, see "Searching Through a Database" in Chapter 9 of the *Quattro User's Guide*.

The field specified in your criteria and the field being averaged need not be the same. The field averaged is that contained within the column you specify as *Column*.

You can specify all or part of your database as *Block*, but field names *must* be included for each field you include in the block.

Examples

The following examples refer to the database and criteria tables shown below.



	A	B	C	D	E
1			SALES RECORDS		
2	DATE	LOCATION	REP	PRODUCT	AMOUNT
3	Jul-87	San Jose	Joe	123-E	$14,999
4	Jul-87	San Fran	Sam	456-B	$25,000
5	Jul-87	San Jose	Carol	234-C	$18,998
6	Jul-87	L.A.	Sam	456-B	$27,840
7	Aug-87	San Fran	Joe	123-E	$15,500
8	Aug-87	L.A.	Carol	234-B	$19,000
9	Aug-87	San Jose	Joe	123-E	$13,650
10	Aug-87	L.A.	Carol	234-B	$20,000
11					
12			CRITERIA		
13	Date	Location	Rep	Product	
14	Jul-87	San Jose	Carol	123-E	
15		San Fran			
16					
17					
18					
19					
20					

A1:

12-Oct-87 03:44 PM

READY

@DAVG(A2..E10,4,A13..A14) = $21,709 (average of July sales)
@DAVG(A2..E10,4,B13..B15) = $17,629 (average of Bay Area sales)
@DAVG(A2..E10,4,"C3='Carol'") = $19,329 (average of Carol's sales)
@DAVG(A2..E10,4,"D3='123-E'") = $14,716 (average of 123-E sales)
@DAVG(A2..E10,5,A13..A14) = ERR (*Column* figure too high)
@DAVG(A3..E10,4,A13..A14) = 19998.28571 (incorrect—field names not included)
@DAVG(A2..E10,2,A13..A14) = ERR (cannot average labels)

@DAY

Format: @DAY(*DateTimeNumber*)

DateTimeNumber = a number under 73050.9999999 (Dec. 31, 2099)

Converts the date/time serial number you supply as *DateTimeNumber* into the number associated with that day (1-31). Decimal (time) portions of the number are ignored.

Examples

@DAY(31779) = 2 (1/2/87)
@DAY(32134) = 23 (9/23/87)
@DAY(@DATE(87,9,10)) = 10

@DCOUNT

Format: @DCOUNT(*Block,Column,Criteria*)

Block = the cell block containing the database, including field names

Column = the number of the column containing the field you want to count. The first column is 0, second is 1, and so on.

Criteria = logical expression of a cell block containing search criteria

@DCOUNT counts selected field entries in a database. It includes only those entries in Column number *Column* whose records meet the criteria specified in block *Criteria*.

Criteria can be a search formula within quotes, such as "F4>20000" or "E4='Tom'", or it can be the coordinate of a block containing a *criteria table*, specifying search information.

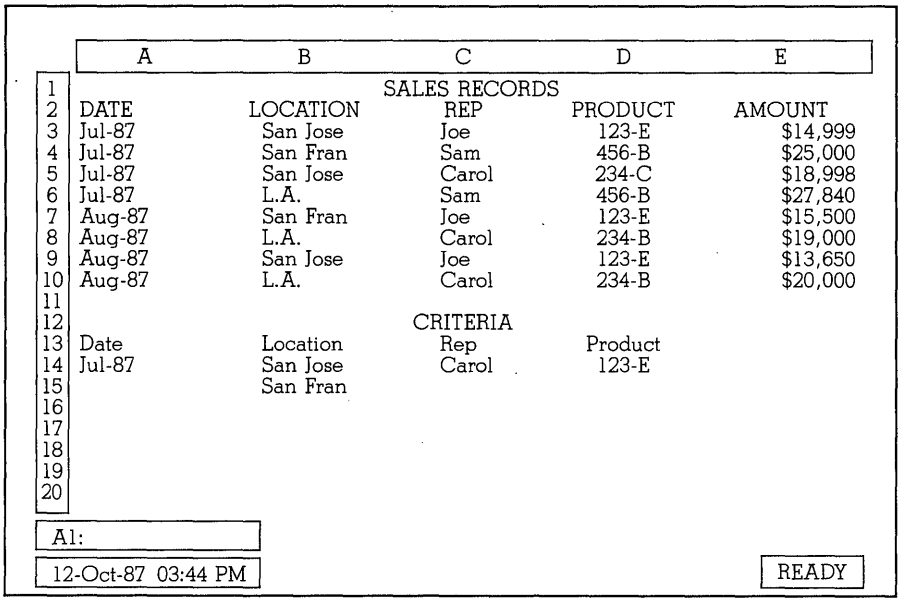
If you use a search formula for *Criteria*, it must be surrounded by quotes. You can use single quotes to surround text in the formula that would normally require double quotes. And, the formula must reference the first cell in the column to be searched. For more details on search formulas and criteria tables, see "Searching Through a Database" in Chapter 9 of the *Quattro User's Guide*.

The field specified in your criteria and the field being counted need not be the same. The field counted is that contained within the column you specify as *Column*.

You can specify all or part of your database as *Block*, but field names *must* be included for each field you include in the block.

Examples

The following examples refer to the database and criteria tables shown below.



	A	B	C	D	E
1			SALES RECORDS		
2	DATE	LOCATION	REP	PRODUCT	AMOUNT
3	Jul-87	San Jose	Joe	123-E	$14,999
4	Jul-87	San Fran	Sam	456-B	$25,000
5	Jul-87	San Jose	Carol	234-C	$18,998
6	Jul-87	L.A.	Sam	456-B	$27,840
7	Aug-87	San Fran	Joe	123-E	$15,500
8	Aug-87	L.A.	Carol	234-B	$19,000
9	Aug-87	San Jose	Joe	123-E	$13,650
10	Aug-87	L.A.	Carol	234-B	$20,000
11					
12			CRITERIA		
13	Date	Location	Rep	Product	
14	Jul-87	San Jose	Carol	123-E	
15		San Fran			
16					
17					
18					
19					
20					

A1:

12-Oct-87 03:44 PM

READY

@DCOUNT(A2..E10,4,A13..A14) = 4 (number of July sales)
@DCOUNT(A2..E10,4,"B3='San Jose'#OR# B3='San Fran'") = 5
(number of Bay Area sales)
@DCOUNT(A2..E10,4,C13..C14) = 3 (number of Carol's sales)
@DCOUNT(A2..E10,4,"D3='123-E'") = 3 (number of 123-E sales)
@DCOUNT(A2..E10,5,A13..A14) = ERR (*Column* figure too high)

@DCOUNT(A3..E10,4,A13..A14) = 7 (incorrect—field names not included)

@DDB

Format: @DDB(*Cost,Salvage,Life,Period*)

Cost = A numeric value representing the amount paid for an asset

Salvage = A numeric value representing the worth of an asset at the end of its useful life

Life = A numeric value representing the expected useful life of an asset

Period = A numeric value representing the time period for which you want to determine the depreciation expense

@DDB determines accelerated depreciation values for an asset, given the initial cost, life expectancy, end value, and depreciation period. It calculates depreciation using the double-declining balance method:

$$\frac{\text{Book Value for } Period * 2}{Life}$$

Examples

The following examples show depreciation values for the first five years of an investment.

@DDB(15000,3000,10,1) = $3,000
@DDB(15000,3000,10,2) = $2,400
@DDB(15000,3000,10,3) = $1,920
@DDB(15000,3000,10,4) = $1,536
@DDB(15000,3000,10,5) = $1,229

@DEGREES

Format: @DEGREES(X)

X = a numeric value representing radians

@RADIANS converts the given number of radians to degrees, using the following formula:

Converts radians to degrees using the following formula:

$$1 \text{ rad} = \frac{180}{\pi} \text{ degree}$$

Examples
@DEGREES(0.017) = 0.974028
@DEGREES(0.5) = 28.69789

@DMAX

Format: @DMAX(*Block,Column,Criteria*)

Block = the cell block containing the database, including field names

Column = the number of the column containing the field for which you want to find the maximum value. The first column is 0, second is 1, and so on.

Criteria = logical expression of a cell block containing search criteria

@DMAX finds the maximum value of selected field entries in a database. It includes only those entries in Column number *Column* whose records meet the criteria specified in block *Criteria.*

Criteria can be a search formula within quotes, such as "F4>20000" or "E4='Tom'", or it can be the coordinate of a block containing a *criteria table,* specifying search information.

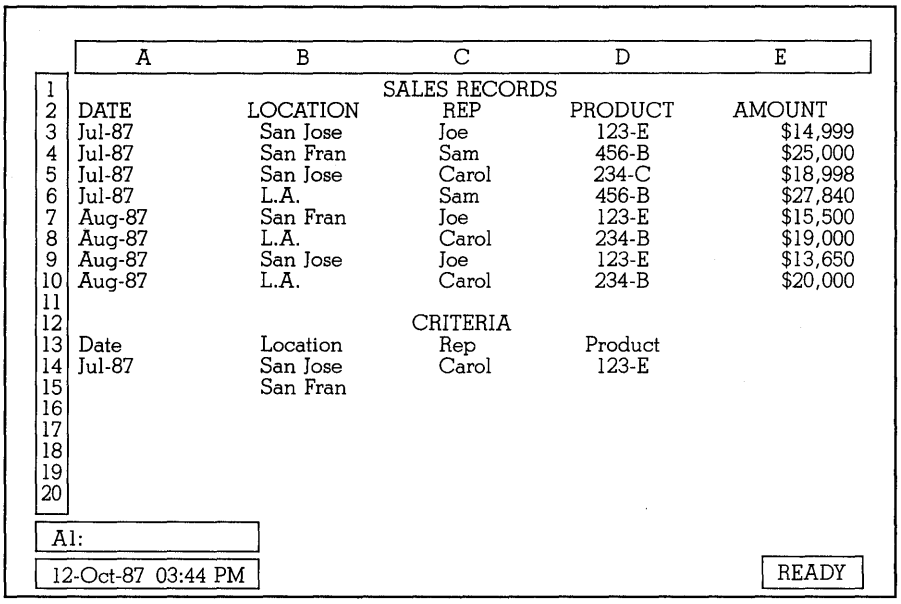
If you use a search formula for *Criteria,* it must be surrounded by quotes. You can use single quotes to surround text in the formula that would normally require double quotes. And, the formula must reference the first cell in the column to be searched. For more details on search formulas and criteria tables, see "Searching Through a Database" in Chapter 9 of the *Quattro User's Guide.*

The field specified in your criteria and the field you are finding the maximum value for need not be the same. The field you are finding the maximum value for is that contained within the column you specify as *Column.*

You can specify all or part of your database as *Block,* but field names *must* be included for each field you include in the block.

Examples

The following examples refer to the database and criteria tables shown below.



	A	B	C	D	E
1			SALES RECORDS		
2	DATE	LOCATION	REP	PRODUCT	AMOUNT
3	Jul-87	San Jose	Joe	123-E	$14,999
4	Jul-87	San Fran	Sam	456-B	$25,000
5	Jul-87	San Jose	Carol	234-C	$18,998
6	Jul-87	L.A.	Sam	456-B	$27,840
7	Aug-87	San Fran	Joe	123-E	$15,500
8	Aug-87	L.A.	Carol	234-B	$19,000
9	Aug-87	San Jose	Joe	123-E	$13,650
10	Aug-87	L.A.	Carol	234-B	$20,000
11					
12			CRITERIA		
13	Date	Location	Rep	Product	
14	Jul-87	San Jose	Carol	123-E	
15		San Fran			
16					
17					
18					
19					
20					

@DMAX(A2..E10,4,"A3=@DATEVALUE('Jul-87') #OR# 'Jul-87'"
= $27,840 (highest July sale)
@DMAX(A2..E10,4,B13..B15) = $25,000 (highest Bay Area sale)
@DMAX(A2..E10,4,C13..C14) = $20,000 (highest of Carol's sales)
@DMAX(A2..E10,4,D13..D14) = $15,500 (highest of 123-E sales)
@DMAX(A2..E10,5,A13..A14) = ERR (*Column* figure too high)

@DMAX(A3..E10,4,A13..A14) = 27840 (incorrect—field names not included)
@DMAX(A2..E10,2,Rep=Joe) = Syntax error (no quotes around search formula)

@DMIN

Format: @DMIN(*Block,Column,Criteria*)

Block = the cell block containing the database, including field names

Column = the number of the column containing the field you want to find the minimum value for. The first column is 0, second is 1, and so on.

Criteria = logical expression of a cell block containing search criteria

@DMIN finds the minimum value of selected field entries in a database. It includes only those entries in Column number *Column* whose records meet the criteria specified in block *Criteria.*

Criteria can be a search formula within quotes, such as "F4>20000" or "E4='Tom", or it can be the coordinate of a block containing a *criteria table,* specifying search information.

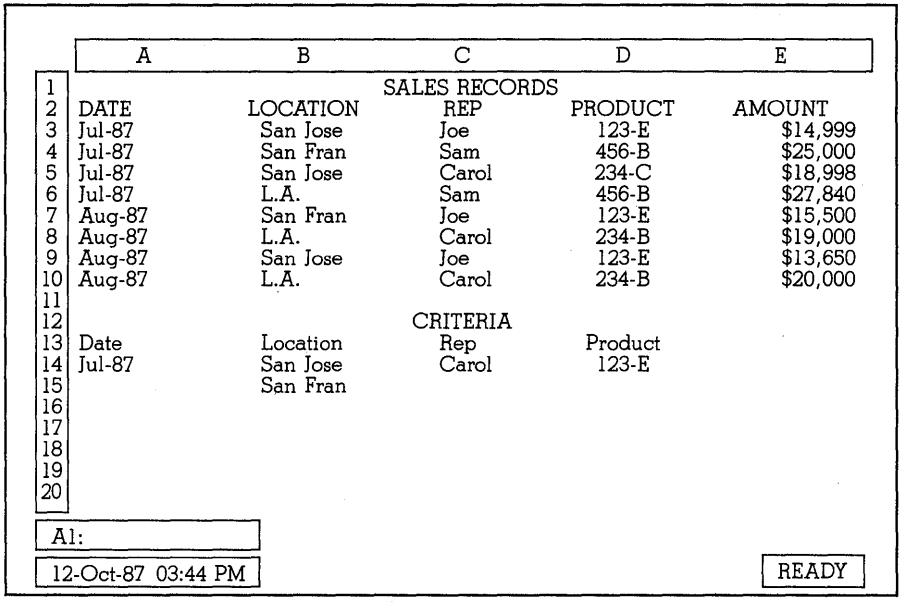
If you use a search formula for *Criteria,* it must be surrounded by quotes. You can use single quotes to surround text in the formula that would normally require double quotes. And, the formula must reference the first cell in the column to be searched. For more details on search formulas and criteria tables, see "Searching Through a Database" in Chapter 9 of the *Quattro User's Guide.*

The field specified in your criteria and the field for which you are finding the minimum need not be the same. The field for which you are finding the minimum is that contained within the column you specify as *Column.*

You can specify all or part of your database as *Block,* but field names *must* be included for each field you include in the block.

Examples

The following examples refer to the database and criteria tables shown below.



	A	B	C	D	E
1			SALES RECORDS		
2	DATE	LOCATION	REP	PRODUCT	AMOUNT
3	Jul-87	San Jose	Joe	123-E	$14,999
4	Jul-87	San Fran	Sam	456-B	$25,000
5	Jul-87	San Jose	Carol	234-C	$18,998
6	Jul-87	L.A.	Sam	456-B	$27,840
7	Aug-87	San Fran	Joe	123-E	$15,500
8	Aug-87	L.A.	Carol	234-B	$19,000
9	Aug-87	San Jose	Joe	123-E	$13,650
10	Aug-87	L.A.	Carol	234-B	$20,000
11					
12			CRITERIA		
13	Date	Location	Rep	Product	
14	Jul-87	San Jose	Carol	123-E	
15		San Fran			
16					
17					
18					
19					
20					

@DMIN(A2..E10,4,A13..A14) = $14,999 (smallest July sale)
@DMIN(A2..E10,4,B13..B15) = $13,650 (smallest Bay Area sale)
@DMIN(A2..E10,4,"C4='Carol'") = $18,998 (smallest of Carol's sales)
@DMIN(A2..E10,4,D13..D14) = $13,650 (smallest 123-E sale)
@DMIN(A2..E10,5,A13..A14) = ERR (*Column* figure too high)
@DMIN(A3..E10,4,A13..A14) = 27840 (incorrect—field names not included)
@DMIN(A2..E10,2,A13..A14) = ERR (can't use @DMIN on labels)

@DSTD

Format: @DSTD(*Block,Column,Criteria*)

Block = the cell block containing the database, including field names

Column = the number of the column containing the field you want to find the standard deviation for. The first column is 0, second is 1, and so on.

Criteria = logical expression of a cell block containing search criteria

@DSTD finds the standard deviation for selected field entries in a database. It includes only those entries in Column number *Column* whose records meet the criteria specified in block *Criteria*.

Criteria can be a search formula within quotes, such as "F4>20000" or "E4='Tom'", or it can be the coordinate of a block containing a *criteria table*, specifying search information.

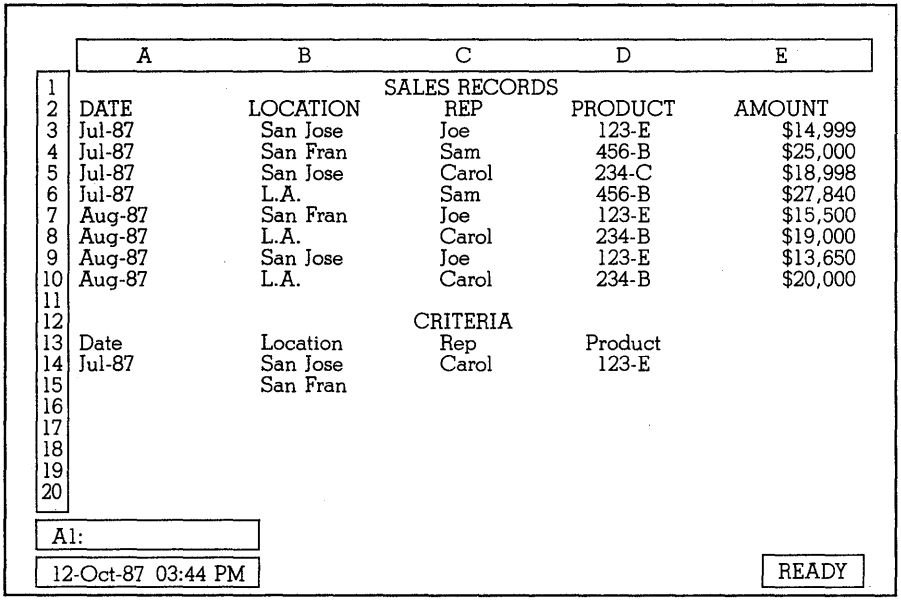
If you use a search formula for *Criteria*, it must be surrounded by quotes. You can use single quotes to surround text in the formula that would normally require double quotes. And, the formula must reference the first cell in the column to be searched. For more details on search formulas and criteria tables, see "Searching Through a Database" in Chapter 9 of the *Quattro User's Guide*.

The field specified in your criteria and the field you are finding the standard deviation for need not be the same. The field you are finding the standard deviation for is that contained within the column you specify as *Column*.

You can specify all or part of your database as *Block*, but field names *must* be included for each field you include in the block.

Examples

The following examples refer to the database and criteria tables shown below.



	A	B	C	D	E
1			SALES RECORDS		
2	DATE	LOCATION	REP	PRODUCT	AMOUNT
3	Jul-87	San Jose	Joe	123-E	$14,999
4	Jul-87	San Fran	Sam	456-B	$25,000
5	Jul-87	San Jose	Carol	234-C	$18,998
6	Jul-87	L.A.	Sam	456-B	$27,840
7	Aug-87	San Fran	Joe	123-E	$15,500
8	Aug-87	L.A.	Carol	234-B	$19,000
9	Aug-87	San Jose	Joe	123-E	$13,650
10	Aug-87	L.A.	Carol	234-B	$20,000
11					
12			CRITERIA		
13	Date	Location	Rep	Product	
14	Jul-87	San Jose	Carol	123-E	
15		San Fran			
16					
17					
18					
19					
20					

@DSTD(A2..E10,4,A13..A14) = $5,019.80 (std. dev. of July sales)
@DSTD(A2..E10,4,B13..B15) = $4,086.26 (std. dev. of BayArea sales)
@DSTD(A2..E10,4,C13..C14) = $471.88 (std. dev. of Carol's sales)
@DSTD(A2..E10,4,D13..D14) = $781.26 (std. dev. of 123-E sales)
@DSTD(A2..E10,5,A13..A14) = ERR (*Column* figure too high)
@DSTD(A3..E10,4,A13..A14) = 4614 (incorrect—field names not included)
@DSTD(A2..E10,2,A13..A14) = ERR (can't use @DSTD on labels)

@DSUM

Format: @DSUM(*Block,Column,Criteria*)

Block = the cell block containing the database, including field names

Column = the number of the column containing the field you want to total. The first column is 0, second is 1, and so on.

Criteria = logical expression of a cell block containing search criteria

@DSUM totals selected field entries in a database. It includes only those entries in Column number *Column* whose records meet the criteria specified in block *Criteria.*

Criteria can be a search formula within quotes, such as "F4>20000" or "E4='Tom'", or it can be the coordinate of a block containing a *criteria table,* specifying search information.

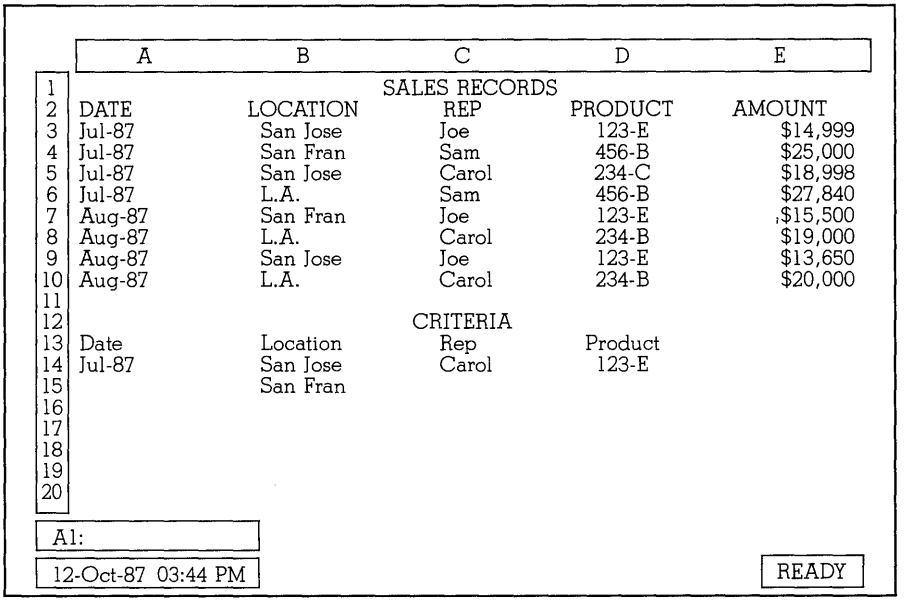
If you use a search formula for *Criteria,* it must be surrounded by quotes. You can use single quotes to surround text in the formula that would normally require double quotes. And, the formula must reference the first cell in the column to be searched. For more details on search formulas and criteria tables, see "Searching Through a Database" in Chapter 9 of the *Quattro User's Guide.*

The field specified in your criteria and the field you are finding the sum of need not be the same. The field you are finding the sum of is that contained within the column you specify as *Column.*

You can specify all or part of your database as *Block,* but field names *must* be included for each field you include in the block.

Examples

The following examples refer to the database and criteria tables shown below.



	A	B	C	D	E
1			SALES RECORDS		
2	DATE	LOCATION	REP	PRODUCT	AMOUNT
3	Jul-87	San Jose	Joe	123-E	$14,999
4	Jul-87	San Fran	Sam	456-B	$25,000
5	Jul-87	San Jose	Carol	234-C	$18,998
6	Jul-87	L.A.	Sam	456-B	$27,840
7	Aug-87	San Fran	Joe	123-E	$15,500
8	Aug-87	L.A.	Carol	234-B	$19,000
9	Aug-87	San Jose	Joe	123-E	$13,650
10	Aug-87	L.A.	Carol	234-B	$20,000
11					
12			CRITERIA		
13	Date	Location	Rep	Product	
14	Jul-87	San Jose	Carol	123-E	
15		San Fran			
16					
17					
18					
19					
20					

A1:

12-Oct-87 03:44 PM READY

@DSUM(A2..E10,4,A13..A14) = $86,837 (total of July sales)
@DSUM(A2..E10,4,B13..B15) = $88,147 (total of Bay Area sales)
@DSUM(A2..E10,4,C13..C14) = $57,998 (total of Carol's sales)
@DSUM(A2..E10,4,"D3='123-E'") = $44,149 (total of 123-E sales)
@DSUM(A2..E10,5,A13..A14) = ERR (*Column* figure too high)
@DSUM(A3..E10,4,A13..A14) = 139988 (incorrect—field names not included)
@DSUM(A2..E10,2,A13..A14) = ERR (cannot total labels)

@DVAR

Format: @DVAR(*Block,Column,Criteria*)

Block = the cell block containing the database, including field names

Column = the number of the column containing the field for which you want to compute variance. The first column is 0, second is 1, and so on.

Criteria = logical expression of a cell block containing search criteria

@DVAR calculates variance for selected field entries in a database. It includes only those entries in Column number *Column* whose records meet the criteria specified in block *Criteria.*

Criteria can be a search formula within quotes, such as "F4>20000" or "E4='Tom'", or it can be the coordinate of a block containing a *criteria table,* specifying search information.

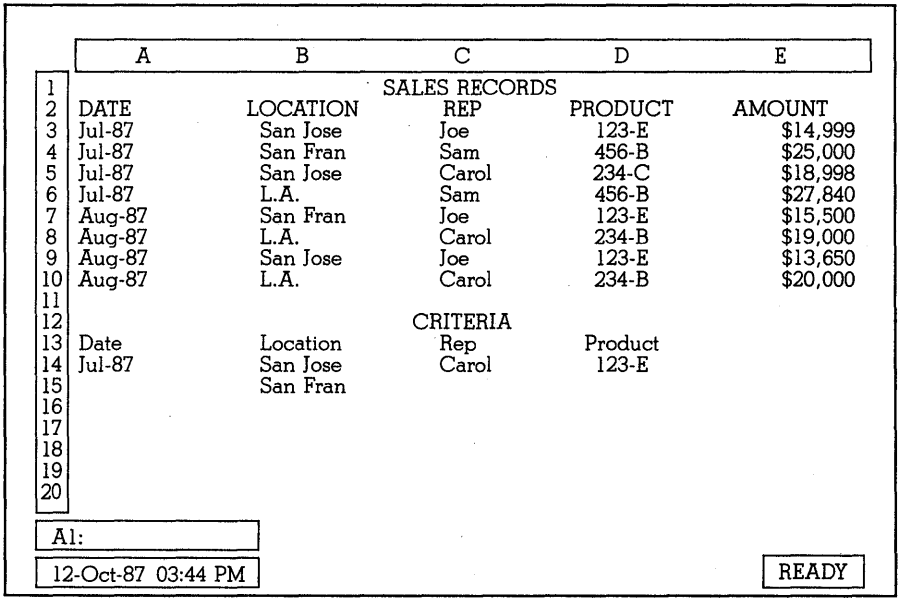
If you use a search formula for *Criteria,* it must be surrounded by quotes. You can use single quotes to surround text in the formula that would normally require double quotes. And, the formula must reference the first cell in the column to be searched. For more details on search formulas and criteria tables, see "Searching Through a Database" in Chapter 9 of the *Quattro User's Guide.*

The field specified in your criteria and the field that you are calculating the variance for need not be the same. The field counted is that contained within the column you specify as *Column.*

You can specify all or part of your database as *Block,* but field names *must* be included for each field you include in the block.

Examples

The following examples refer to the database and criteria tables shown below.



	A	B	C	D	E
1			SALES RECORDS		
2	DATE	LOCATION	REP	PRODUCT	AMOUNT
3	Jul-87	San Jose	Joe	123-E	$14,999
4	Jul-87	San Fran	Sam	456-B	$25,000
5	Jul-87	San Jose	Carol	234-C	$18,998
6	Jul-87	L.A.	Sam	456-B	$27,840
7	Aug-87	San Fran	Joe	123-E	$15,500
8	Aug-87	L.A.	Carol	234-B	$19,000
9	Aug-87	San Jose	Joe	123-E	$13,650
10	Aug-87	L.A.	Carol	234-B	$20,000
11					
12			CRITERIA		
13	Date	Location	Rep	Product	
14	Jul-87	San Jose	Carol	123-E	
15		San Fran			
16					
17					
18					
19					
20					

@DVAR(A2..E10,4,A13..A14) = 25198366 (variance of July sales)
@DVAR(A2..E10,4,B13..B15) = 16697557 (variance of Bay Area sales)
@DVAR(A2..E10,4,"C3='Carol'") = 222667.6 (variance of Carol's sales)
@DVAR(A2..E10,4,D13..D14) = 610366.9 (variance of 123-E sales)
@DVAR(A2..E10,5,A13..A14) = ERR (*Column* figure too high)
@DVAR(A3..E10,4,A13..A14) = 21291726 (incorrect—field names not included)
@DVAR(A2..E10,2,A13..A14) = ERR (can't use @DVAR on labels)

@ERR

Format: @ERR

@ERR returns the value ERR in the current cell and in any other cells that reference it, either directly or indirectly. (Exceptions to this are @COUNT, @ISERR, @ISNA, @ISNUMBER, @ISSTRING, and @CELL formulas; these will not result in ERR if they reference an ERR cell.)

The ERR value resulting from this function is the same as the ERR value produced by Quattro when it encounters an error. It is used, most often with the @IF function, to bring attention to error conditions.

Examples

@ERR = ERR
@IF(B6>B7,0,@ERR) = 0 (if B6>B7) or ERR (if B6<B7)

@EXACT

Format: @EXACT(*String1*,*String2*)

String1 = a valid string value
String2 = a valid string value

@EXACT compares the values of *String1* and *String2*. If the values are *exactly* identical, including capitalization and diacritical marks (such as ~), it returns a 1 value. If there are any differences, it returns a 0 value.

If you're comparing strings, they must be surrounded by double quotes. You can compare the contents of label cells only. If you attempt to compare one or more numbers or empty cells, the result is ERR. When comparing labels, label prefixes are ignored.

To compare strings or cell contents without regard to capitalization or diacritical marks, use the @IF function. For example, *@IF(C3=B3,1,0)* will

return a 1 value if the contents of the cells are the same but capitalized differently.

Examples

@EXACT("client","Client") = 0
@EXACT("client","client") = 1
@EXACT(client,client) = syntax error (no quotes)
@EXACT("client","client"."client") = syntax error (more than two strings)
@EXACT(29,"29") = ERR (the first string is a value)
@EXACT(A1,"yes") = 1 (if A1 contains the label *yes*)

@EXP

Format: @EXP(X)

X = a numeric value <= 709

@EXP returns the mathematical constant e, raised to the *X*th power. This function is the inverse of a natural logarithm, @LN (see page 174).

Examples

@EXP(3.4) = 29.9641004
@EXP(1) = 2.71828183 (the actual value of "e")
@SQRT(@EXP(2)) = 2.71828183
@LN(@EXP(2.5)) = 2.5

@FALSE

Format: @FALSE

@FALSE returns the logical value 0 and is usually used in @IF formulas. The zero it returns is the same as any other zero, but the @FALSE function makes the formula easier to read.

See also @TRUE on page 199.

Examples

@FALSE = 0
@IF(C3=100,@FALSE,10) = 0 (if C3 = 100) or 10 (if C3 <> 100)

@IF(C3=100,@FALSE,@TRUE) = 0 (if C3 = 100) or 1 (if C3 <> 100)

@FILEEXISTS

Format: @FILEEXISTS(*FileName*)

FileName = any file name

@FILEEXISTS returns a 1 if *FileName* exists in the default data directory and returns a 0 if it doesn't. *FileName* must be surrounded by quotes, and must include any file name extension attached to the file name. To search for a file in a directory other than the default data directory, include the directory path in *FileName*.

@FIND

Format: @FIND(*SubString,String,StartNumber*)

SubString = a valid string value, representing the value to search for

String = a valid string value, representing the value to search through

StartNumber = a numeric value >= 0, representing the character position to begin searching with

@FIND searches through the given *String* for the value given as *SubString*. If it finds *Substring*, it returns the character position at which the first occurrence was found. *StartNumber* begins the search at that number of characters into the string. 0 = the first character in the string, 1 = the second, and so on. The value of *StartNumber* must not be more than the number of characters in *String* minus 1.

@FIND is case-sensitive, and is also sensitive to diacritical marks used in non-English languages. You can overcome the case-sensitivity of this function by using @UPPER to force one or more of the strings into all caps, for example:

@FIND(@UPPER(C3),@UPPER(C4),0)

forces both the substring in cell C3 and the string in cell C4 to uppercase, then searches for the substring.

@FIND is most often used in conjunction with two other string functions: @REPLACE (to perform "search and replace" operations on strings) and @MID (to access substrings).

If @FIND fails to find any occurrences of *SubString*, or if the *StartNumber* given is invalid, the result is ERR.

Examples

@FIND("i","find",0) = 1
@FIND("nd","find",2) = 2
@FIND("F","find",0) = ERR
@FIND("f","find",3) = ERR
@FIND("d","find",4) = ERR
@FIND(n,find,0) = syntax error
@FIND("hi",C4,0) = 1 (if C4 contains *ship*)

@FV

Format: @FV(*Payment,Interest,Life*)

Payment = a numeric value representing the amount of equal payments to be made

Interest = a numeric value >= 0, representing periodic interest rate

Life = a numeric value representing number of periods of the investment

@FV returns the future value of an investment where *Payment* is invested for *Life* periods at the rate of *Interest* per period. The payment is calculated using the following formula:

$$FV = Payment * \frac{(1 + Interest)^{(Life)} - 1}{Interest}$$

Examples

@FV(200,.12,5) = $1,270.56
@FV(500,0.9,4) = $6,684.50
@FV(800,0.9,3) = $5,208.00

@FV(800,0.9,A3) = $40,929.67 (if A3 = 6)

@HEXTONUM

Format: @HEXTONUM(*String*)

String = a hexadecimal number surrounded by quotes

@HEXTONUM converts the hexadecimal number in the string to the corresponding decimal value.

Examples

@HEXTONUM("a") = 10
@HEXTONUM("10") = 16

@HLOOKUP

Format: @HLOOKUP(*X,Block,Row*)

X = a numeric or string value
Block = a block value
Row = a numeric value => 0

@HLOOKUP searches (horizontally) through the first row of the given *Block* for the given value *X*. When found, it returns the value itself (if *Row* = 0), or the value displayed the specified number of rows beneath it (as indicated by *Row*).

@HLOOKUP provides an efficient way to access information stored in a table. For example, you might have a table listing days of the week by number:

0	1	2	3	4	5	6
Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday

You could use the @HLOOKUP function to enter the day of the week for a given date. For example:

@HLOOKUP(@MOD(@DATE(87,7,11),7),A1..G2,1) = Saturday

The value of *X* can be either a character string or a number, the address or block name of any cell containing a label or value, or any expression that results in a number or string. If *X* is a string, Quattro looks for an exact match of either upper or lowercase. If *X* is a number and Quattro can't find an equal number, it locates the highest number in the row not more than *X*.

The second argument, *Block*, specifies the coordinates of the table to be used for the lookup. This table must have its *index* values in the first row. The index values are the comparison figures used to determine which of the columns of the table will be used. Because of the nature of a lookup table, these values (if numbers) must be in ascending numerical order. Also, there must be no blank cells in the index row, nor any blank rows in the table.

Quattro searches through the index row of the table from left to right looking for a match to *X*. If it finds an exact match, Quattro stops at that column. If an exact match isn't found, Quattro stops at the value closest to but not greater than *X*. If *X* is a label, it stops at the nearest match. If *X* is a number and the index row contains only labels, Quattro stops at the rightmost column.

The final argument *Row* tells Quattro how many rows down in the column the return value is. This argument can be any value from 0 up to the number of rows in the table, or any expression or cell address resulting in such a value. A value of 0 tells Quattro to return the actual value found in the index row. A value of 1 instructs Quattro to return the value directly below the one found in the index row; 2 tells Quattro to return the value two rows below, and so on.

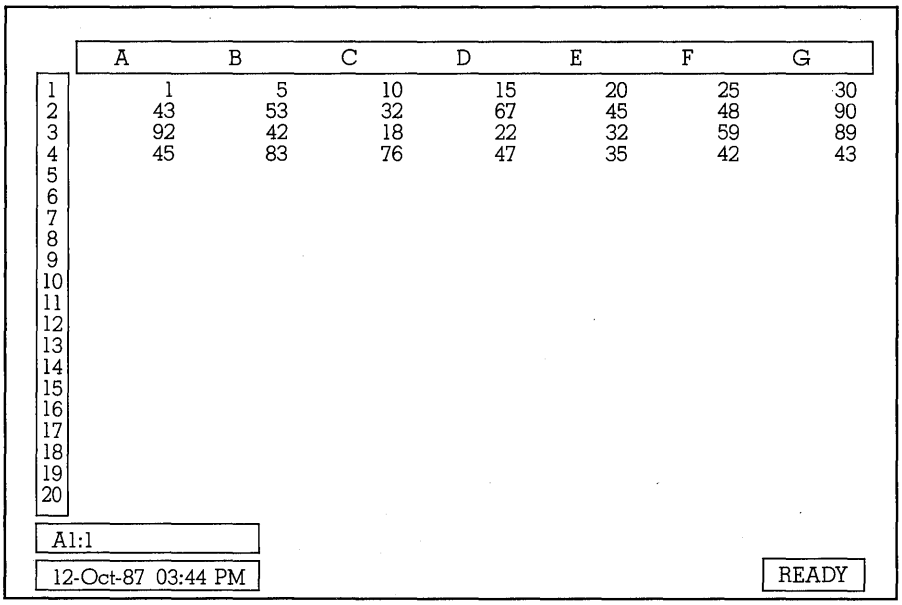
Note: If *X* is a string value and *Row* = 0, Quattro will return the offset number of the column *X* is found in, not the value of *X*.

Any of the following instances will result in ERR:

- *Row* is less than 0 or greater than the number of rows in *Block*.
- *X* is less than the smallest value in the topmost row of *Block*.
- *X* is a string or label and Quattro fails to find a match in the top row of *Block*.

Examples

The following examples refer to data in the lookup table below. In the first example, Quattro searches across the first row of the specified block (row 1), looking for the largest number equal to or less than 17. It stops at cell D1, then moves down the specified number of rows (3). It stops at cell D4 and returns the value 47.



@HLOOKUP(17,A1..G4,3) = 47
@HLOOKUP(10,A1..G4,0) = 10
@HLOOKUP(6,A1..G4,2) = 42
@HLOOKUP(50,A1..G4,3) = 43
@HLOOKUP("string",A1..G4,3) = ERR (exact string not found)
@HLOOKUP("18",A1..G4,2) = ERR (18 is shown as string)
@HLOOKUP(18,A1..G4,4) = ERR (row value > # rows)
@HLOOKUP(18,A1..G2,3) = ERR (row value > # rows in given block)

See also @VLOOKUP for searching vertically through a table (page 201).

@HOUR

Format: @HOUR(*DateTimeNumber*)

DateTimeNumber = a numeric value between 0 and 73050.9999999, representing a date/time serial number

@HOUR returns the hour portion of *DateTimeNumber*. *DateTimeNumber* must be a valid date/time serial number. Because only the decimal portion of a serial number pertains to time, the integer portion of the number will be disregarded. The result is between 0 (12:00AM) and 23 (11:00PM).

To extract the hour portion of a string that is in time format (instead of serial format), use @TIMEVALUE within the @HOUR function to translate the time into a serial number. To return standard hours (1-12) instead of military hours (1-24), use the @MOD function (see page 178).

(See also @TIME on page 198 and @TIMEVALUE on page 198.)

Examples:

@HOUR(.25) = 6
@HOUR(.5) = 12
@HOUR(.75) = 18
@HOUR(@TIMEVALUE("10:08am")) = 10
@MOD(@HOUR(@TIMEVALUE("9:31:52 AM")),12) = 9

@IF

Format: @IF(*Cond,TrueExpr,FalseExpr*)

Cond = a logical expression representing the condition to be tested

TrueExpr = any numeric or string value representing the value to use if *Cond* is true

FalseExpr = a numeric or string value representing the value to use if *Cond* if false

@IF evaluates the logical condition given as *Cond*. If the condition is found to be true, it returns the value given as *TrueExp*. If the condition is false, it returns the value given as *FalseExpr*.

The formula entered as *Cond* can be any logical expression that can be evaluated as true or false, for example, *B6<0* or *C3*D2=53*. If the expression is found to be true, the *TrueExpr* value is entered into the current cell. Otherwise, *FalseExpr* is entered.

You can use compound conditions by connecting expressions with #AND# or #OR#. If you use #AND# , both conditions given must be met to evaluate true. If you use #OR#, the expression is true if either of the conditions is met. For example, *A3<10#AND#A3>5* means that the value in A3 must be between 6 and 9 to evaluate true. You can also use the #NOT# operator to negate a condition. For example, *#NOT#(B3>10)* evaluates true if B3 is *not* greater than 10.

The expressions given as *TrueExpr* and *FalseExpr* can be either numeric values or text. Numeric values can be numbers or formulas resulting in

numbers. If text is used, the string must be surrounded by double quotes, for example, *@IF(D6=5,"John","Harry")*. You can also use cell references to use the contents of other cells in the spreadsheet. For example, *@IF(B10<18,D5,C4)* enters the contents of D5 if the condition is true, and enters the contents of C4 if the condition is false.

If the condition you specify with *Cond* searches a cell for a number and the cell contains a label, ERR is returned. Likewise, if you search for a label and find a numeric value, ERR is returned.

Although logical expressions typically reference other cells, this is not required. Any expression resulting in a numeric result is accepted, for example, *A1=1* or *A1="Fred"*. If the result of *Cond* is greater than 0, *TrueExpr* is the result; otherwise *FalseExpr* is the result.

@IF functions can be *nested*, or used within one another. In other words, *TrueExpr* can contain yet another test to further validate *Cond*. For example, *@IF(B5>C6,@IF(B5>C7,1,2),3)* tells Quattro to see if the contents of B5 are greater than C6. If they are, it then checks to see if B5 is greater than C7; if so, it enters a 1 in the cell. If not, it enters a 2. If B5 is *not* greater than C6, it enters a 3. There's no limit on the number of levels. @IF expressions can be nested, as long as the entire expression doesn't exceed 240 characters.

Examples

@IF(8=7,4,5) = 5
@IF(B4<100,"Yes","No") = Yes if B4 < 100, otherwise, No
@IF(C10=BLOCK,45,50) = 45 if C10 = the cell named BLOCK, otherwise 50
@IF(50-35,10,20) = 10
@IF(50-60,10,20) = 20

@INDEX

Format: @INDEX(*Block,Column,Row*)

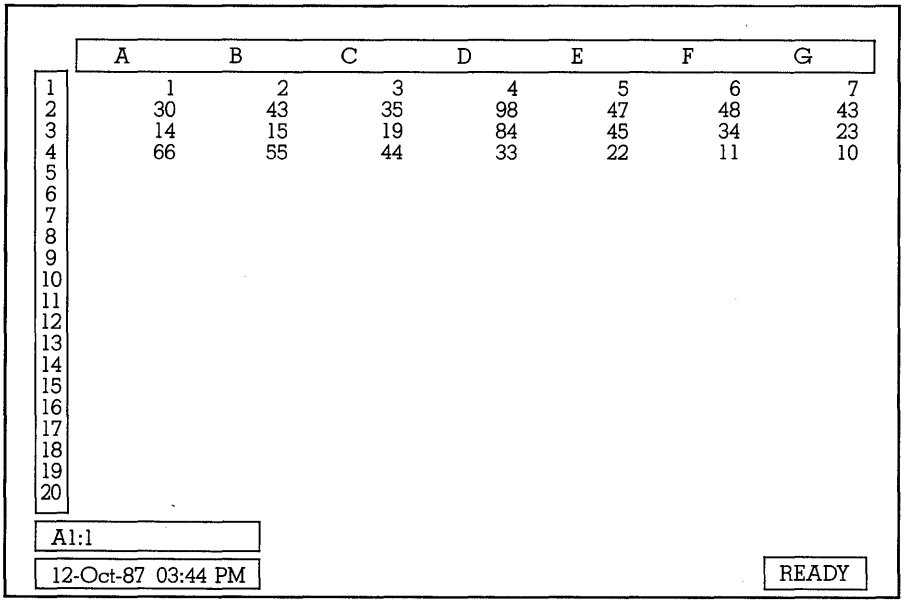
Block	=	a block value
Column	=	a numeric value >= 0
Row	=	a numeric value >= 0

@INDEX, like the @LOOKUP functions, is used with data tables. It searches through the table given as *Block* and returns the value specified with the Column and Row values. The Column and Row values are not the actual coordinates of the resulting cell, but *offset* values. In other words, @INDEX begins in the top left cell of the given block, moves right the number of columns specified by *Column* and down the number of rows specified by *Row*. It then returns the value in the current cell.

The *Column* and *Row* values must be numbers greater than or equal to zero and less than the number of rows or columns in the block. If a fractional number is used (for example, 2.35), the fractional part is dropped (*not* rounded).

Example

The following examples reference cells in the data table below.



@INDEX(A1..G10,4,3) = 22
@INDEX(A1..G10,2,3) = 44
@INDEX(C2..E4,1,2) = 33
@INDEX(C2..E4,3,3) = ERR (too many rows)
@INDEX(B1..D3,-3,2) = ERR (negative column number)

@INT

Format: @INT(X)

X = a numeric value

@INT drops the fractional portion of *X*, returning its integer value. @INT does not round off *X*. For that function, see @ROUND (page 189).

Examples

@INT(499.99) = 499
@INT(0.1245) = 0
@INT(C4) = 5 if C4 contains a value between 5 and 6

@IRR

Format: @IRR(*Guess,Block*)

Guess = a numeric value that estimates the internal rate of return on an investment

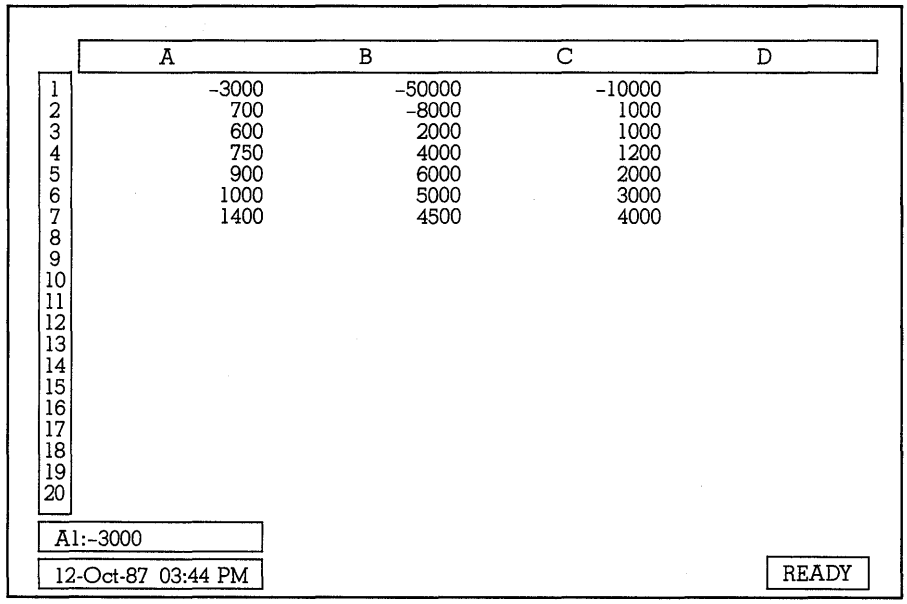
Block = a cell block that contains cash flow amounts for the investment

@IRR determines the internal rate of return on an investment. It references a block in your spreadsheet that contains cash flow information and uses the supplied internal rate of return estimate to calculate the results.

Before you use this function, you must set up a cash flow table, showing expected cash flow amounts over a period of time. Quattro assumes that the amounts are received at regular intervals. Negative amounts are interpreted as cash outflows, and positive amounts as inflows. The first amount must be a negative number, to reflect the initial investment. Following amounts can be in even or irregular amounts.

Examples

The following examples reference the cash flow tables below.



@IRR(12%,A1..A7) = 16.85%
@IRR(12%,B1..B7) = –20.60%
@IRR(12%,C1..C7) = 4.70%

@ISERR

Format: @ISERR(X)

X = a cell address or expression

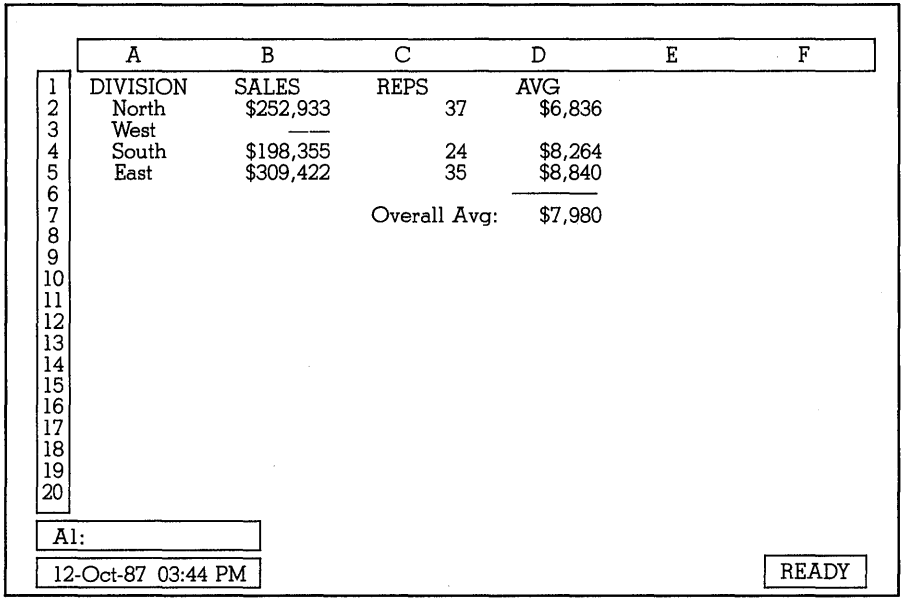
@ISERR is normally used to check the contents of a cell for errors. If the cell contains ERR, a value of 1 is returned; otherwise, 0 is returned. You can also use formulas or numeric values with the @ISERR function.

When an error occurs in a cell, any formula referencing that cell results in ERR, and any cell referencing that cell becomes ERR. This sometimes causes the value of ERR to ripple through your spreadsheet. The @ISERR function, used with the @IF function, can arrest this ripple effect, as demonstrated below.

In the spreadsheet below, sales for each division are divided by the number of sales reps to determine average sales per rep for each division. The West Division is late with its figures, which would normally cause an ERR value to be displayed both for that division's average and the overall average

(you cannot divide by zero). In this case, however, the @ISERR function is used within an @IF formula to enter "- -" for the division's average if the division's sales figure is blank.

D3: @IF(@ISERR(B3/C3),"- -",B3/C3)



Examples

@ISERR(C2) = 1 is C2 contains ERR, otherwise, it equals 0
@ISERR(10/0) = 1
@ISERR(C2/B3) = 1 if B3 is 0, ERR, or NA, or if C2 is NA or ERR; otherwise 0
@ISERR(45+C3) = 1 if C3 is NA or ERR; otherwise 0
@IF(@ISERR(A2),0,A5) = 0 is A2 is ERR, otherwise it returns the value in A5

@ISNA

Format: @ISNA(X)

X = a cell address or expression

@ISNA tests for the special value NA in a cell. If the cell contains an NA value, it returns a 1; otherwise, it returns a 0. NA is considered a special value because the only way it can appear on the spreadsheet is through the

use of the @NA function (see 180). If a cell contains the label "NA" (not produced by @NA), it is not recognized by the @ISNA function.

Examples

@ISNA("NA") = 0
@ISNA(@NA) = 1
@ISNA(A18) = 1 if A18 contains NA produced by @NA

@ISNUMBER

Format: @ISNUMBER(*X*)

X = a cell address or expression

@ISNUMBER examines *X* and determines if it contains a numeric value. If *X* is blank or contains a numeric value, ERR, or NA, it returns a 1. If *X* is a label or text, it returns a 0. @ISNUMBER is usually used with the @IF function to determine whether or not an entry is a value.

Examples

@ISNUMBER(88) = 1
@ISNUMBER("88") = 0 (quotes signify a text string)
@ISNUMBER(9/15/87) = 1
@ISNUMBER(@ERR) = 1 (ERR and NA are considered numeric values)

@ISSTRING

Format: @ISSTRING(*X*)

X = a cell address or expression

@ISSTRING examines *X* and determines if it contains a label or text string. If it does (even if it's an empty text string), it returns a 1. If it is blank or contains a numeric or date value, it returns a 0.

Usually, @ISSTRING is used to test the contents of a cell. You can test any expression, however. In order for an expression to be considered a text string, it must be surrounded by double quotes.

Examples

@ISSTRING(55) = 0
@ISSTRING(2/5/88) = 0
@ISSTRING("Hello, world.") = 1
@ISSTRING("55") = 1

@ISSTRING(A15) = 1 if A15 contains a label, otherwise = 0
@ISSTRING("") = 1 ("" is an empty string)
@ISSTRING(@NA) = 0 (NA and ERR are considered values)

@LEFT

Format: @LEFT(*String,Num*)

String = a string value
Num = a numeric value >= 0

@LEFT returns the leftmost *Num* characters of *String*. It allows you to extract a specified number of characters from the left side of a string or label.

If *String* is a numeric or date value or a blank cell, @LEFT returns ERR. If *Num* is longer than the length of *String*, all of *String* is returned.

Examples

@LEFT("Jennifer",5) = Jenni
@LEFT("Jennifer",15) = Jennifer
@LEFT("155",1) = 1
@LEFT(" Jennifer",6) = J (including 5 leading spaces)
@LEFT(123,1) = ERR (123 is a value)

@LENGTH

Format: @LENGTH(*String*)

String = a string value

@LENGTH returns the number of characters in *String*, including spaces. You can combine strings or cell addresses with an ampersand (&). When *String* is a text string, it must be surrounded by double quotes.

Examples

@LENGTH("Hello, world.") = 13
@LENGTH(" Jennifer") = 9 (including preceding space)
@LENGTH("Greetings "&"earthling") = 19 (including space after Greetings)
@LENGTH(29584949) = ERR (29584949 is a value, not a string)

@LENGTH(A6&B10) = the total number of characters in A6 and B10

@LN

Format: @LN(*X*)

X = a numeric value > 0

@LN returns the natural logarithm of *X*. A natural logarithm uses the mathematical constant e as a base. @LN produces the inverse of @EXP.

Examples

@LN(3) = 1.098612288
@LN(@EXP(10)) = 10
@LN(16)/@LN(2) = 4
@LN(-4) = ERR (-4 is less than 0)

@LOG

Format: @LOG(*X*)

X = a numeric value > 0

@LOG returns the base 10 logarithm of *X*.

Examples

@LOG(1000) = 3
@LOG(10^23.8) = 23.8
@LOG(16)/@LOG(2) = 4

@LOWER

Format: @LOWER(*String*)

String = a string value

@LOWER returns *String* in lowercase characters. Numbers and symbols within a string are unaffected. Numeric and date values return ERR.

Examples

@LOWER("UPPER") = upper
@LOWER("Hello, world.") = hello, world.
@LOWER("145 Bancroft Lane") = 145 bancroft lane

@LOWER(4839) = ERR
@LOWER(@LEFT("johnson",1)) = j

@MAX

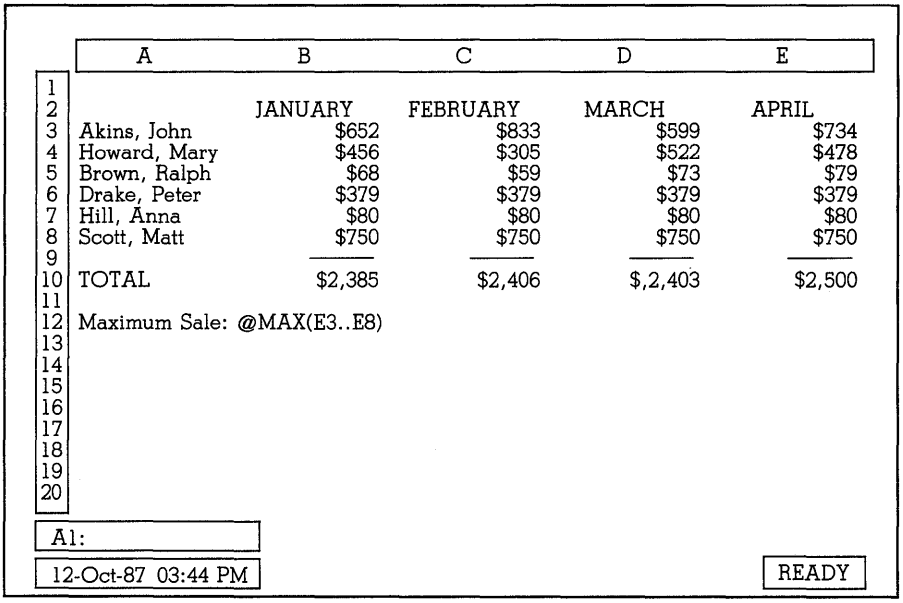
Format: @MAX(*List*)

List = one or more numeric or block values

@MAX returns the largest numeric or date value in *List*. If more than one block is listed, commas must separate the blocks. If any of the cells referenced contains ERR, the resulting value is ERR.

Examples

The following spreadsheet uses @MAX to find the highest sales per rep in April. The examples below refer to other cells in the same spreadsheet.



	A	B	C	D	E
1					
2		JANUARY	FEBRUARY	MARCH	APRIL
3	Akins, John	$652	$833	$599	$734
4	Howard, Mary	$456	$305	$522	$478
5	Brown, Ralph	$68	$59	$73	$79
6	Drake, Peter	$379	$379	$379	$379
7	Hill, Anna	$80	$80	$80	$80
8	Scott, Matt	$750	$750	$750	$750
9					
10	TOTAL	$2,385	$2,406	$,2,403	$2,500
11					
12	Maximum Sale: @MAX(E3..E8)				
13					
14					
15					
16					
17					
18					
19					
20					

A1:

12-Oct-87 03:44 PM READY

@MAX(B3..B8) = $750
@MAX(C3..C8,E3..E8) = $833
@MAX(A1..E10) = $2,500

@MAX(B3..C8,E3) = $833

@MEMAVAIL

Format: @MEMAVAIL

@MEMAVAIL returns the number of bytes of conventional memory currently available.

Example

@MEMAVAIL = 47819 (if 47,819 bytes of memory are available)

@MEMEMSAVAIL

Format: @MEMEMSAVAIL

@MEMEMSAVAIL returns the number of bytes of expanded memory (EMS) currently available. If your system doesn't contain expanded memory, it returns NA.

Example

@MEMEMSAVAIL = 12000 (if your systems contains 12,000 bytes of free EMS)

@MID

Format: @MID(*String,StartNumber,Num*)

String	=	a string value
StartNumber	=	a numeric value >= 0
Num	=	a numeric value >= 0

@MID extracts the first *Num* characters of *String* starting at character number *StartNumber*. It is similar to the @LEFT function, which extracts *Num* characters of *String* beginning with the first character. The difference is that you can specify the character number to start with.

String can be any text string (surrounded by quotes) or reference to a cell containing a label. If *StartNumber* is greater than the length of *String* or if *Num* is 0, the result is "", or an empty string.

Examples

@MID("Abraham Lincoln",8,7) = Lincoln
@MID("George Washington",7,4) = Wash
@MID("Theodore Roosevelt",19,5) = ""
@MID(A23,@FIND("Roosevelt",A23,0),@LENGTH("Roosevelt")) =
Roosevelt (if A23 = Franklin Roosevelt)

@MIN

Format: @MIN(*List*)

List = one or more numeric or block values

@MIN returns the smallest numeric value in *List*. If *List* contains more than one value, commas must separate the values. Labels are treated in all statistical functions as 0, so if there are any labels in *List*, @MIN will return 0.

Examples

The following spreadsheet uses @MIN to find the lowest commission in April. The examples below refer to other cells in the same spreadsheet.

	A	B	C	D	E
1					
2		JANUARY	FEBRUARY	MARCH	APRIL
3	Akins, John	$652	$833	$599	$734
4	Howard, Mary	$456	$305	$522	$478
5	Brown, Ralph	$68	$59	$73	$79
6	Drake, Peter	$379	$379	$379	$379
7	Hill, Anna	$80	$80	$80	$80
8	Scott, Matt	$750	$750	$750	$750
9					
10	TOTAL	$2,385	$2,406	$ 2,403	$2,500
11					
12	Lowest April Commission: @MIN(E3..E8)				
13					
14					
15					
16					
17					
18					
19					
20					

A1:

12-Oct-87 03:44 PM READY

@MIN(B3..B8) = $68

@MIN(B5..E5,B7..E7) = $59
@MIN(B3..E3) = $599

@MINUTE

Format: @MINUTE(*DateTimeNumber*)

DateTimeNumber = a numeric value between 1 and 73050.9999999, representing a date/time serial number

@MINUTE returns the minute portion of *DateTimeNumber*. *DateTimeNumber* must be a valid date/time serial number. Because only the decimal portion of a serial number pertains to time, the integer portion of the number is disregarded. The result is between 0 and 59.

To extract the minute portion of a string that is in time format (instead of serial format), use @TIMEVALUE within the @MINUTE function to translate the time into a serial number (see page 198). You can also use @TIME to enter a time value instead of a serial number (see page 198).

Examples:

@MINUTE(.36554) = 46
@MINUTE(.2525) = 3
@MINUTE(35) = 0
@MINUTE(@TIME(3,15,22)) = 15
@MINUTE(@TIMEVALUE("10:08 am")) = 8

@MOD

Format: @MOD(*X*,*Y*)

X = a numeric value
Y = a numeric value <> 0

@MOD divides the *X* value by *Y* and returns the remainder value. Because you cannot divide a number by zero, ERR results if the value of *Y* is zero.

Examples

@MOD(3,1) = 0 (3 divided by 1 leaves no remainder)
@MOD(5,2) = 1 (5 divided by 2 leaves a remainder of 1)
@MOD(3,1.1) = 0.8
@MOD(4,0) = ERR
@IF(@MOD(YEAR,4)=0#AND#@MOD(YEAR,100)<>0#OR#@MOD(YEAR,400)=

0,"Leap Year","Not Leap Year")
(checks to see if the year entered in the block named YEAR is a leap year or not)

@MONTH

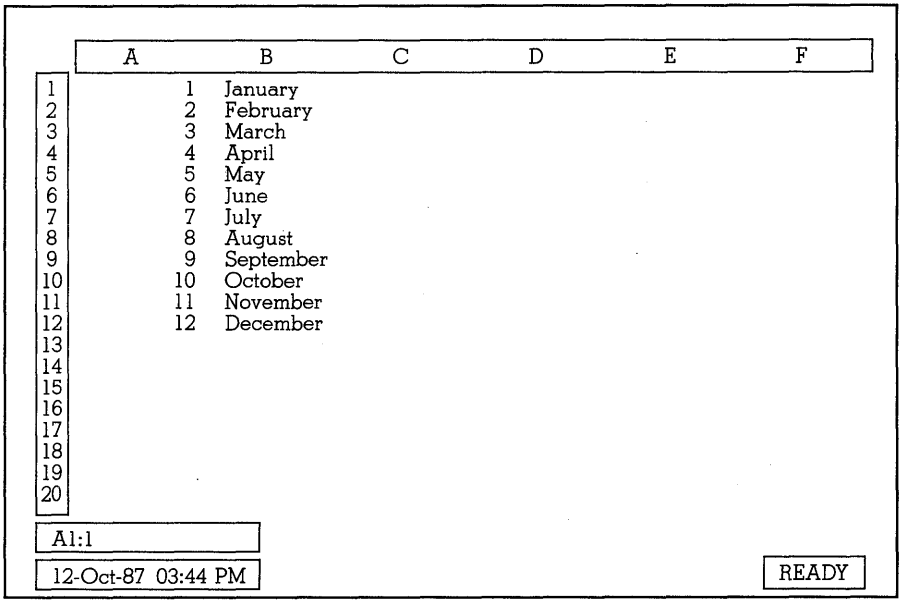
Format: @MONTH(*DateTimeNumber*)

DateTimeNumber = a numeric value between 0 and 73050.9999999, representing a date serial number

@MONTH returns the month portion of *DateTimeNumber*. *DateTimeNumber* must be a valid date/time serial number. Only the integer portion is used. The result is between 1 (January) and 12 (December).

To extract the month portion of a string that is in Date format (instead of serial format), use @DATEVALUE within the @MONTH function to translate the date into a serial number (see page 145). You can also use @DATE to enter a date value instead of a serial number (see page 144).

To display the *name* of the resulting month instead of the number, use a lookup table with the numbers corresponding to their names as shown in the example below:



You could then use the @VLOOKUP function to display the corresponding month name:

@VLOOKUP(@MONTH(*DateTimeNumber*),A1..B12,1)

Examples:

@MONTH(69858) = 4
@MONTH(58494) = 2
@MONTH(.37373) = 1
@MONTH(@DATEVALUE("3/5/88")) = 3
@MONTH(@DATE(88,3,5)) = 3
@MOD(@MONTH(@DATEVALUE("3/5/88")),12) = 3

@N

Format: @N(*Block*)

Block = a block value

@N inspects *Block* and returns the numeric value of the upper-left cell. If that cell contains a label or is blank, it returns a 0.

This function is used by other spreadsheet programs to avoid unnecessary ERR values resulting from labels included in calculations. This is unnecessary with Quattro, however, because labels are already considered zero values in calculations. The @N function is included in Quattro only for compatibility with other products.

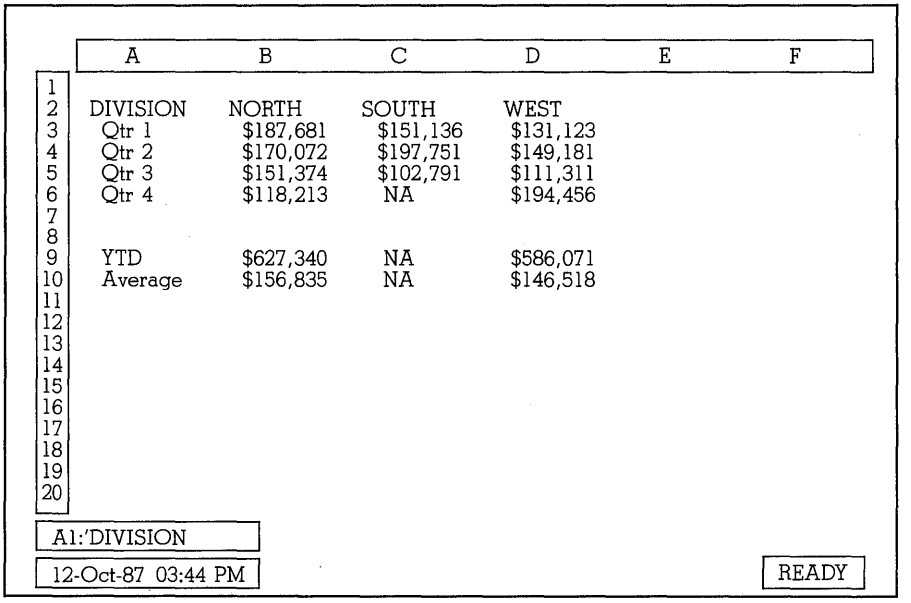
@NA

Format: @NA

@NA returns the special value NA (Not Available). NA is displayed in the cell and in any cells containing formulas that reference that cell.

@NA is used to indicate values not yet available. It ensures that formulas relying on information that is not provided do not display inaccurate data.

In the example below, @NA has been entered for the South's Qtr 4 results. As you can see, the NA cascades through to the totals with no special steps taken. When the @NA is replaced with a valid value, the totals will immediately reflect the correct figures.



Examples

@NA = NA
@IF(B6=0,@NA,B6) = NA if B6 = 0, otherwise the value of B6 is displayed

@NOW

Format: @NOW

@NOW returns the serial number corresponding to the current date and time. To display the number as a date, use the **B**lock **D**isplay Format **D**ate command (see page 41). To display the number as time, use the **B**lock **D**isplay **F**ormat **D**ate **T**ime command (see page 64). You can also change the default display format to **D**ate or **T**ime (*\DFDD* or *\DFDT*).

The value generated by @NOW is updated to the current date and time each time you press the CALC key (*F9*), or perform any function that reevaluates the spreadsheet.

The integer part of a date/time serial number pertains to the date; the decimal portion pertains to time. To extract just the date portion, use @INT(@NOW). To extract just the time portion, use @MOD(@NOW,1).

Examples

@NOW = 31905.572338 (5/8/87, 1:45 PM)
@INT(@NOW) = 31905 (5/8/87)
@MOD(@NOW,1) = 0.572338 (1:45 PM)
@INT(@MOD(@NOW,7)) = 6 (Friday, 5/8/87)

@NPV

Format: @NPV(*Interest,Block*)

Interest = a numeric value representing a fixed periodic interest rate

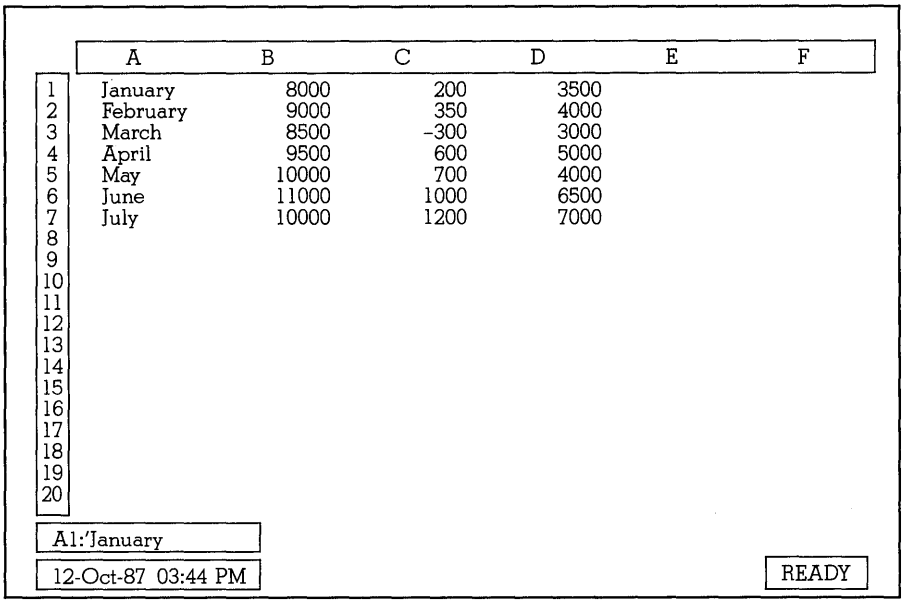
Block = a cell block containing expected cash flow information

@NPV calculates the current value of a set of estimated cash flow values (*Block*), discounted at the given interest rate (*Interest*). It is helpful in determining how much an investment is currently worth, based on expected earnings, although its accuracy is entirely dependent upon the accuracy of the cash flow table.

The cash flow table you reference should show expected income and debits over a period of time. Quattro assumes that the amounts are received *at the end of* regular intervals and that the length of this interval is the same as the period on which interest is compounded. In other words, if monthly cash flow is estimated, *Interest* needs to show monthly interest. To convert annual interest to monthly interest, simply divide by 12.

Examples

The following examples reference the cash flow tables below.



	A	B	C	D	E	F
1	January	8000	200	3500		
2	February	9000	350	4000		
3	March	8500	-300	3000		
4	April	9500	600	5000		
5	May	10000	700	4000		
6	June	11000	1000	6500		
7	July	10000	1200	7000		
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						

@NPV(1.25%,B1..B7) = $62,683.77
@NPV(15%/12,C1..C7) = $3,507
@NPV(15%/12,D1..D7) = $31,216
–2000+@NPV(15%/12,D1..D7) = $29,215.77 (assumes an initial cash outflow of $2,000

@NUMTOHEX

Format: @NUMTOHEX(*X*)

X = a numeric value

@NUMTOHEX converts the number X to its corresponding hexadecimal string value.

Examples

@NUMTOHEX (10) = 'A
@NUMTOHEX (16) = '10

@PI

Format: @PI

@PI returns the value of Π (3.141592653589794...), the classic ratio of a circle's circumference to its diameter.

To figure the circumference of a circle given the diameter, use the following formula:

@PI * Diameter

To figure the area of a circle, given the radius, use the formula:

@PI * Radius^2

or

@PI * (Diameter/2)^2

Examples

@PI*13 = 40.84 (circumference)
@PI*(7.5)^2 = 176.7142 (area)
@PI*B3 = the circumference of a circle whose diameter is in B3

@PROPER

Format: @PROPER(*String*)

String = a string value

@PROPER converts the first letter of every word in *String* to upper case, and the rest of the characters to lower case. A word is defined as an unbroken string of alphabetic characters. Any blank spaces, punctuation symbols, or numbers mark the ending of a word.

Examples

@PROPER("GEORGE washINGTON") = George Washington
@PROPER("FIRST QUARTER") = First Quarter
@PROPER("JOHN J. SMITH") = John J. Smith
@PROPER("1979's results") = 1979'S Results

@PMT

Format: @PMT(*Principal,Interest,Life*)

Principal = a numeric value representing amount borrowed

Interest = a numeric value > –1, representing interest rate

Life = a numeric value <> 0, representing the number of periods of the loan

@PMT calculates the fully amortized payment of borrowing *Principal* dollars at *Interest* percent per period over *Life* number of periods. It assumes that interest is paid at the end of each period. The payment is calculated using the following formula:

$$PMT = \frac{Principal * Interest}{1 - (1 + Interest)^{(-Life)}}$$

You can enter the value for *Interest* as a decimal or a fraction, for example, 9.5% or .095. The amount you specify for *Interest* must correlate with the unit used for *Life*. In other words, if the interest is compounded yearly, the amount entered for *Life* must represent years. If the interest is compounded monthly, *Life* must represent the number of months the loan is for. To calculate monthly payments using an annual interest rate, divide the interest rate by 12. For example,

@PMT(10000,15%/12,36)

calculates a monthly payment for a $10,000 loan at an annual 15% interest rate.

Examples

@PMT(1000,0.12,5) = $277.40
@PMT(500,0.16,12) = $96.20
@PMT(5000,16%/12,12) = $453.65
@PMT(12000,0.11,15) = $1,668.78

@PV

Format: @PV(*Payment,Interest,Life*)

Payment = a numeric value representing the amount of the periodic payment

Interest = a numeric value >= –1, representing periodic interest rate

Life = a numeric value representing the number of payments to be made

@PV calculates the present value of an investment where *Payment* is received for *Life* periods and is discounted at the rate of *Interest* per period. Present value is calculated using the following formula:

$$PV = Payment * \frac{1 - (1 + Interest)^{(-Life)}}{Interest}$$

Examples

@PV(277,0.12,5) = $998.52
@PV(600,0.17,10) = $2,795.16
@PV(100,0.11,12) = $649.23

@RADIANS

Format: @RADIANS(*X*)

X = a numeric value representing degrees

@RADIANS converts the given number of degrees to radians, using the following formula:

$$1° = \frac{\pi}{180} = 0.017 \text{ radians}$$

Examples
@RADIANS (1) = 0.017453
@RADIANS (57) = 0.994838
@RADIANS (@DEGREES (3.5)) = 3.5

@RAND

Format: @RAND

@RAND returns a fractional random number between 0 and 1. This offers a sampling of figures, useful for generating sample data for simulated situations.

To generate random numbers in another range, multiply @RAND by the difference between the new high and low ends, then add the new low end number. For example, to indicate a range of 10 to 100, enter *@RAND*90+10*. This extends the upper limit to 100 and increases the lower limit to 10.

Examples

@RAND = a random number between 0 and 1
@RAND*9+1 = a random number between 1 and 10
–@INT(@RAND*90+10) = a random integer between –10 and –100
@RAND*1000 = a random number between 0 and 1000

@RAND+5 = a random number between 5 and 6

@RATE

Format: @RATE(*FutValue,PresValue,Term*)

FutValue = a numeric value representing the future value of an investment

PresValue = a numeric value representing the current value of an investment

Term = a numeric value representing the length of the investment in terms of the number of compounding periods

@RATE calculates the interest rate required in order for an investment of *PresValue* to be worth *FutValue* within *Term* compounding periods. If *Term* represents years, an annual interest rate results; if *Term* represents months, a monthly interest rate results, and so on.

@RATE uses the following formula to calculate interest rate:

[(*FutValue*/*PresValue*)^(1/*Term*)] – 1

Examples

@RATE(10000,7000,6*12) = .50% (monthly)
@RATE(1200,1000,3) = 6.27% (yearly)
@RATE(500,100,25) = 6.65% (yearly)

@REPEAT

Format: @REPEAT(*String,Num*)

String = a string value
Num = a numeric value >= 0

@REPEAT returns *Num* copies of *String* as one continuous label. This function is similar to the repeating label prefix (\) in that it repeats one or more characters. The difference is that you can specify exactly how many times you want the string to be repeated. The \ label prefix adjusts the display to fill the column, even when the width is changed. The @REPEAT function displays a fixed number of characters and does not change.

When you specify a text string with @REPEAT, it must be surrounded by double quotes.

Examples

@REPEAT("-",20) = ————————————————————
@REPEAT("good day!",3) = good day!good day!good day!
@REPEAT(A5,5) = the contents of A5 repeated 5 times
@REPEAT("-",@CELL("width",A1..A1)) = ————————————
(if column A is 12 characters wide. If you change the column width, pressing CALC (*F9*) adjusts the repeat string to fill the cell).

@REPLACE

Format: @REPLACE(*String,StartNum,Num,NewString*)

String	=	a string value, representing the text to operate on
StartNum	=	a numeric value >= 0, representing the character position to begin with
Num	=	a numeric value >= 0, representing the number of characters to delete
NewString	=	a string value, representing the characters to insert at position *Num*

@REPLACE lets you replace characters in a label with a new string. It searches through the given *String* until it reaches the given character position *Startnum*. Then it removes *Num* number of characters from the string, replacing them with *Newstring*.

Both *String* and *Newstring* can be either cell references or text strings. If text strings, they must be surrounded by double quotes.

To replace one string with another, specify 0 as *StartNum* and for *Num* enter a number equal to or greater than the number of characters in *String*.

To insert one string into another string, specify 0 as *Num*.

To add one string to the end of another, specify as *StartNum* a number one greater than the number of characters in *String*.

To delete part or all of a string, specify "" as *NewString*.

The utility of @REPLACE is greatly increased when it is used in conjunction with other String functions. For instance, to replace one word with another within a sentence, you can use @FIND and @LENGTH to simplify the search-and-replace operation. For example:

@REPLACE(A7,@FIND("man",A7,0),@LENGTH("man"),"person")

searches through A7 for *man*, then replaces *man* with *person*.

Examples

@REPLACE("Sales Salaries",6,0,"Reps' ") = Sales Reps' Salaries
@REPLACE("355 Howard",11,0," St.") = 355 Howard St.
@REPLACE("McDonald Corp.",2,6,"Douglas") = McDouglas Corp.
@REPLACE("Leslie J. Cooper",7,3,"") = Leslie Cooper

@RIGHT

Format: @RIGHT(*String,Num*)

String = a string value
Num = a numeric value >= 0

@RIGHT returns the last *Num* characters of *String*. It allows you to extract a specified number of characters from the right side of a string or label.

If *String* is a not a valid string, @RIGHT returns ERR. If *Num* is 0, the result is "", or an empty string. If *Num* is greater than the number of characters in *String*, the entire string is returned.

Examples

@RIGHT("Jennifer Meyer",5) = Meyer
@RIGHT("Jennifer Meyer",25) = Jennifer Meyer
@RIGHT("155",1) = 5
@RIGHT("Jennifer ",6) = fer (including 3 subsequent spaces)
@RIGHT(123,1) = ERR (123 is a value)
@RIGHT(A10,5) = the last five characters of A10
@RIGHT(A16,@LENGTH(A16)-@FIND("Roosevelt",A16,0)) =
Roosevelt (if A16 = Theodore Roosevelt)

@ROUND

Format: @ROUND(*X,Num*)

X = a numeric value
Num = a numeric value between –15 and 15

@ROUND adjusts the precision of *X* to *Num* decimal points. *Num* specifies the power of 10 to which *X* is rounded. If *Num* is positive, *X* is rounded *Num* digits to the *right* of the decimal point. If *Num* is negative, *X* is rounded *Num* digits to the *left* of the decimal point. For example, if *Num* is –3, *X* is rounded to the nearest thousand.

If *Num* is 0, *X* is rounded to an integer. If *Num* is not an integer, it is rounded off to the nearest integer.

Examples

@ROUND(12345.54321,0) = 12346
@ROUND(12345.54321,2) = 12345.54
@ROUND(12345.54321,-2) = 12300

@ROWS

Format: @ROWS(*Block*)

Block = a block value

@ROWS returns the number of rows within the given block.

Examples

@ROWS(A1..A1) = 1
@ROWS(A1..C15) = 15
@ROWS(B100..B8192) = 8093
@ROWS(NAME) = 30 (if the block named NAME contains 30 rows)

@S

Format: @S(*Block*)

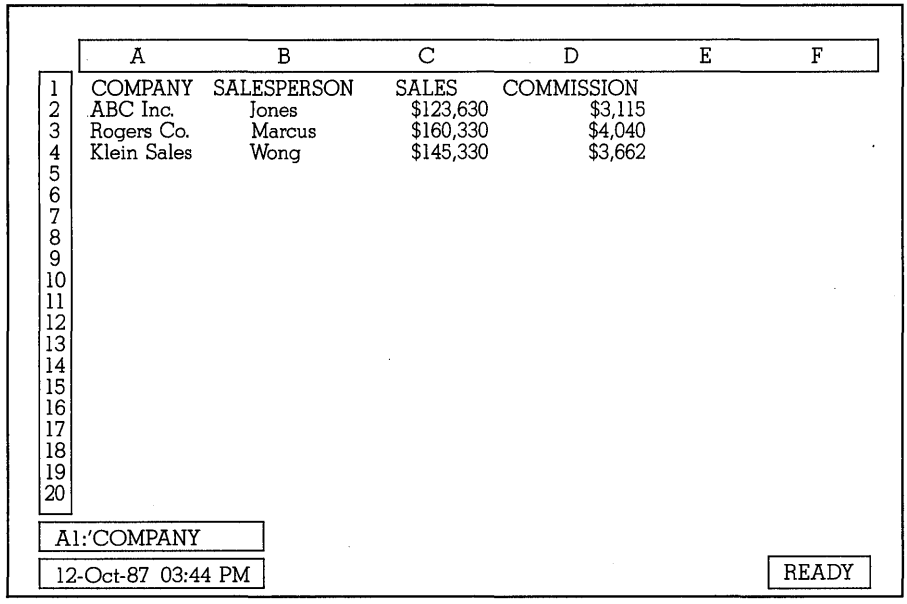
Block = a block value

@S inspects *Block* and returns the string value of the upper-left cell. If that cell contains a numeric or date value or is blank, it returns "" (an empty string).

@S is often used to avoid unnecessary ERR values in the spreadsheet. For example, suppose you want to combine labels in cells A1 and A2, you enter +A1+A2 in cell A5. If cell A1 contains a value and cell A2 contains a string, the result in cell A5 will be ERR. However, if you use @S, as in *+@S(A1)+@S(A2)*, this will guarantee that ERR will not appear in A5.

Examples

The following examples refer to cells in the spreadsheet below.



@S(A1..A6) = Company
@S(A2..A2) = ABC Inc.
@S(C2..C4) = (blank)
@S(B1..B1)&": "&@S(B2..B2) = Salesperson: Jones
@S(B3)&@S(C3) = Marcus

@SECOND

Format: @SECOND(*DateTimeValue*)

DateTimeNumber = a numeric value between 0 and 73050.9999999 representing a date/time serial number

@SECOND returns the second portion of *DateTimeNumber*. *DateTimeNumber* must be a valid date/serial number. Because only the decimal portion of a serial number pertains to time, the integer portion of the number is disregarded. The result is between 0 and 59.

To extract the second portion of a string that is in time format (instead of serial format), use @TIMEVALUE within the @SECOND function to translate the time into a serial number (see page 198). You can also use @TIME to enter a time value instead of a serial number (see page 198).

Examples:

@SECOND(.3655445) = 23
@SECOND(.2543222) = 13
@SECOND(35) = 0
@SECOND(@TIME(3,15,22)) = 22
@SECOND(@TIMEVALUE("10:08:45 am")) = 45
@SECOND(@TIMEVALUE("10:08 am")) = 0

@SIN

Format: @SIN(*X*)

X = a numeric value

@SIN returns the sine of the angle *X*. *X* must be given in radians, not degrees. To convert angles to radians, use the @RADIANS function (see page 186).

Examples

@SIN(@RADIANS(30)) = 0.5
@SIN(0.5) = 0.47942554
@SIN(1) = 0.84147098

@SLN

Format: @SLN(*Cost,Salvage,Life*)

Cost = a numeric value representing the amount paid for an asset

Salvage = a numeric value representing the value of an asset at the end of its useful life

Life = a numeric value representing the number of years of useful life for the asset

@SLN calculates the straight-line depreciation for an asset over one period of its life, using the following formula:

$$\frac{Cost - Salvage}{Life}$$

To compute accelerated depreciation (allowing higher depreciation values in the first years of the asset's life), use the @SYD function (see page 196). To calculate depreciation using the double-declining balance method, use the @DDB function (see page 149).

Examples

@SLN(15000,3000,10) = $1,200
@SLN(5000,500,5) = $900
@SLN(1800,0,3) = $600

@SQRT

Format: @SQRT(*X*)

X = a numeric value >= 0

@SQRT returns the square root of *X*. If *X* is a negative value, the result of @SQRT is ERR. If *X* is a string or reference to a cell containing a label, the function returns 0.

Examples

@SQRT(9) = 3
@SQRT(2) = 1.414213562
@SQRT(144) = 12
@SQRT(@SQRT(16)) = 2
@SQRT(-4) = ERR

@STD

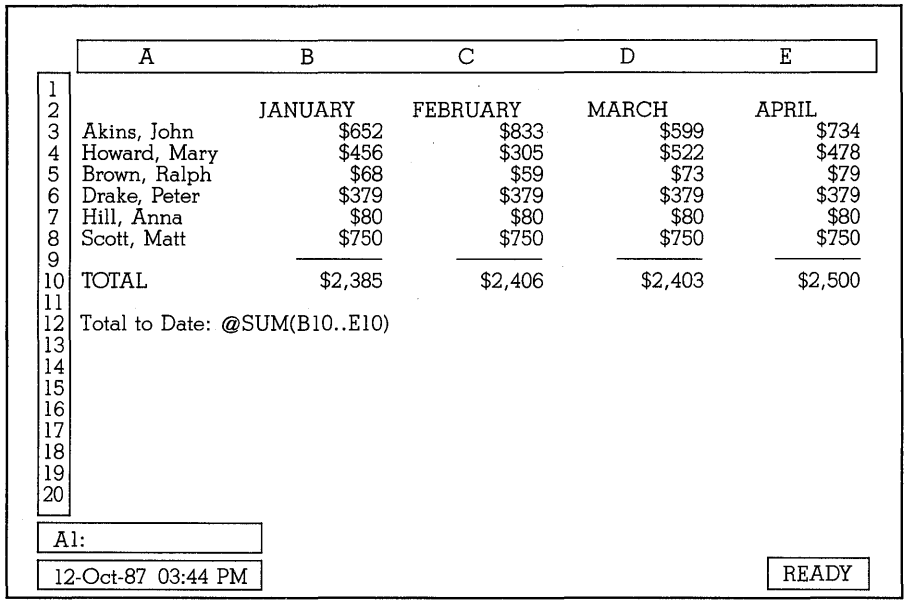
Format: @STD(*List*)

List=one or more numeric or block values

@STD returns the standard deviation (the square root of the variance) of all values in *List*. *List* can be any combination of single cell references, cell blocks, and numeric values. When more than one component is used, they must be separated by commas. Any labels or blank cells within a cell block are ignored by @STD.

@STD determines how much individual values in *List* differ from the average (mean) of all values in *List*. It can be used to verify the reliability of the average; the lower the value returned by @STD, the less individual values vary from the average.

The examples below use @STD to calculate the standard deviation of various sales revenue figures shown in the spreadsheet.



	A	B	C	D	E
1					
2		JANUARY	FEBRUARY	MARCH	APRIL
3	Akins, John	$652	$833	$599	$734
4	Howard, Mary	$456	$305	$522	$478
5	Brown, Ralph	$68	$59	$73	$79
6	Drake, Peter	$379	$379	$379	$379
7	Hill, Anna	$80	$80	$80	$80
8	Scott, Matt	$750	$750	$750	$750
9					
10	TOTAL	$2,385	$2,406	$2,403	$2,500
11					
12	Total to Date: @SUM(B10..E10)				
13					
14					
15					
16					
17					
18					
19					
20					

A1:

12-Oct-87 03:44 PM READY

Examples

@STD(B5..E5) = $7.33
@STD(A3..E3,260) = $587.78
@STD(B3..E8) = $271.78

@STRING

Format: @STRING(*X,DecPlaces*)

X = a numeric value
DecPlaces = a numeric value between 0 and 15

@STRING converts *X* to a string, rounding *X* to the decimal precision indicated by *DecPlaces*.

Once a number or date has been converted to a label using @STRING, no display formatting can be done with it. To format strings derived from numbers as anything other than General format, you must build a macro that uses the {CONTENTS} keyword (see 214).

Examples

@STRING(3.59,0) = 4
@STRING(98.6,2) = 98.60

@STRING(0.3902,0) = 0
@STRING("Harry",0) = 0

@SUM

Format: @SUM(*List*)

List = one or more numeric or block values

@SUM returns the total of all numeric values in *List*. *List* can be any combination of single cell references, cell blocks, and numeric values. When more than one component is used, they must be separated by commas. Any labels or blank cells within a cell block are ignored by @SUM.

CAUTION: Any dates in the cell block will be converted to serial numbers and included in the calculation. Since this will most likely throw off your final sum, you should avoid including dates in the @SUM argument.

Examples

The following spreadsheet uses @SUM to calculate the total sales for January through April. The examples below refer to other cells in the same spreadsheet.

	A	B	C	D	E
1					
2		JANUARY	FEBRUARY	MARCH	APRIL
3	Akins, John	$652	$833	$599	$734
4	Howard, Mary	$456	$305	$522	$478
5	Brown, Ralph	$68	$59	$73	$79
6	Drake, Peter	$379	$379	$379	$379
7	Hill, Anna	$80	$80	$80	$80
8	Scott, Matt	$750	$750	$750	$750
9					
10	TOTAL	$2,385	$2,406	$2,403	$2,500
11					
12	Total to Date: @SUM(B10..E10)				
13					
14					
15					
16					
17					
18					
19					
20					

A1:

12-Oct-87 03:44 PM READY

@SUM(B5..E5) = $279
@SUM(A3..E3,260) = $3,078

@SUM(A5,534) = 534
@SUM(B4..E4,B8..E8) = $4,761
@SUM(B3..E8) = $9,694

@SYD

Format: @SYD(*Cost,Salvage,Life,Period*)

Cost = a numeric value representing the initial cost of an asset

Salvage = a numeric value representing the value of the asset at the end of its life expectancy

Life = a numeric value representing the length of the asset's life expectancy

Period = a numeric value representing the period for which you want to calculate depreciation

@SYD calculates depreciation amounts for an asset using an accelerated depreciation method. This allows higher depreciation in the earlier years of the asset's life. @SYD uses the following formula to compute depreciation:

$$\frac{(Cost - Salvage) * (Life - Period + 1)}{(Life * (Life + 1) / 2)}$$

Examples

The following examples show depreciation values for the first five years of an asset's life. These can be compared to those calculated with the @DDB function, which distributes more of the depreciation in the first year of life.

@SYD(12000,1000,5,1) = $3,667
@SYD(12000,1000,5,2) = $2,933
@SYD(12000,1000,5,3) = $2,200
@SYD(12000,1000,5,4) = $1,467
@SYD(12000,1000,5,5) = $733

@DDB(12000,1000,5,1) = $4,800
@DDB(12000,1000,5,2) = $2,880
@DDB(12000,1000,5,3) = $1,728
@DDB(12000,1000,5,4) = $1,037

@DDB(12000,1000,5,5) = $555

@TAN

Format: @TAN(*X*)

X = a numeric value

@TAN returns the tangent of the angle *X*. *X* must be given in radians, not degrees. To convert angles to radians, use the @RADIANS function (see page 186).

Examples

@TAN(4) = 1.157821
@TAN(@PI/4) = 1
@TAN(@RADIANS(45)) = 1

@TERM

Format: @TERM(*Payment,Interest,FutValue*)

Payment = a numeric value representing a period payment amount toward an investment

Interest = a numeric value representing a fixed, periodic interest rate accrued by the investment

FutValue = a numeric value representing the future value of the investment

@TERM computes the number of payment periods required in order to accumulate an investment of *FutValue*, making regular payments of *Payment* and accruing interest at the rate of *Interest*.

Examples

@TERM(300,6%,5000) = 11.9
@TERM(500,7%,1000) = 1.94
@TERM(1000,10%,50000) = 18.8

@TERM(100,5%,1000) = 8.3

@TIME

Format: @TIME(*Hr,Min,Sec*)

Hr = a number between 0 and 23, representing Hour
Min = a number between 0 and 59, representing Minute
Sec = a number between 0 and 59, representing Second

@TIME returns the date/time serial number represented by Hr:Min:Sec. Each of these arguments must be within the valid ranges above. Any fractional portions are truncated. You can display resulting time string values in standard time formats using the **B**lock **D**isplay Format **D**ate **T**ime command (see page 64), or you can change the default display format to **T**ime (*\DFDT*).

Examples

@TIME(3,0,0) = 0.125 (3:00 am)
@TIME(3,30,15) = 0.14600694444 (3:30:50 am)
@TIME(18,15,59) = 0.76109953704 (5:15:59 pm)
@TIME(B15,23,45) = 0.099826388889 (when the value in B15 is 2)
@TIME(@HOUR(C3),A4,B10) = 0.575139 (when C3 = 0.5577876543, A4 = 48, and B10 =12)

@TIMEVALUE

Format: @TIMEVALUE(*TimeString*)

TimeString = a string value in any valid time format, surrounded by quotes

@TIMEVALUE returns a serial time value that corresponds to the value in *TimeString*. If the value in *TimeString* is not in the correct format, or is not enclosed in quotes, an ERR value is returned.

You can display resulting time string values in standard time formats using the **B**lock **D**isplay Format **D**ate **T**ime command, or you can change the default display format to **T**ime (see page 64).

There are four valid formats for *TimeString*:

- HH:MM:SS AM/PM (03:45:30 PM)
- HH:MM AM/PM (03:45 PM)

- The Long International time format chosen as a system default, one of which is HH:MM:SS (15:45:30)
- The Short International time format chosen as a system default, one of which is HH:MM (15:45)

Examples

@TIMEVALUE("03:30:15 AM") = 0.1460069444
@TIMEVALUE("03:00") = 0.125
@TIMEVALUE("18:15:59") = 0.76109953704
@TIMEVALUE("3.45") = ERR

@TODAY

Format: @TODAY = @INT(@NOW)

@TODAY enters the numeric value of the system's date.

@TRIM

Format: @TRIM(*String*)

String = a string value

@TRIM removes any extra spaces from *String*: spaces following the last non-space character or preceding the first non-space character and duplicate spaces between words. Normal strings are not affected. If *String* is empty, or contains a numeric value, it returns ERR.

Examples

@TRIM(" too many spaces ") = too many spaces
@TRIM("no extra spaces") = no extra spaces
@TRIM(125) = ERR

@TRUE

Format: @TRUE

@TRUE returns the logical value 1 and is usually used in @IF formulas. The 1 it returns is the same as the regular numeral 1, but the @TRUE function makes the formula easier to read.

See also @FALSE on page 160.

Examples

@TRUE = 1
@IF(C3=100,@TRUE,10) = 1 (if C3 = 100) or 10 (if C3 <> 100)
@IF(C3=100,@TRUE,@FALSE) = 1 (if C3 = 100) or 0 (if C3 <> 100)

@UPPER

Format: @UPPER(*String*)

String = a string value

@UPPER returns *String* in uppercase characters. Numbers and symbols within a string are unaffected. If *String* is blank, or contains a numeric or date value, the result is ERR.

Examples

@UPPER("upper") = UPPER
@UPPER("Hello, world.") = HELLO, WORLD.
@UPPER("145 Bancroft Lane") = 145 BANCROFT LANE
@UPPER(4839) = ERR
@UPPER(@LEFT("johnson",1)) = J

@VALUE

Format: @VALUE(*String*)

String = a string value

@VALUE converts *String* into a numeric value. *String* can contain any of the arithmetic operators, but must not contain dollar signs, commas, or embedded spaces. One decimal place is permitted, and leading and trailing spaces are ignored.

This function is useful for converting data that was imported with the File Import command, but was not automatically converted into values.

Examples

@VALUE(" 3.59") = 3.59
@VALUE(" 98.6 ") = 98.6
@VALUE(12/4) = 3
@VALUE(" 88.039") = 88.039
@VALUE("$56.43") = 56.43

@VALUE("34,200") = 34,200

@VAR

Format: @VAR(List)

List = a list of values

@VAR calculates the variance of all non-blank, numeric cells in *List*, using the "N" method:

$$\frac{(n * (\text{Sum } (X*X))) - ((\text{Sum } X) * (\text{Sum } X))}{n * n}$$

where n is the count of non-null cells and x is each non-null cell in the list.

Caution: If *List* contains text or a reference to a single cell containing a label (for example, @VAR(B1) where B1 = Adam), @VAR will return ERR. However, label cells within references to multiple cell blocks (such as @VAR(B1..B5)) will be ignored.

Examples @VAR(23,24,25) = .666666667 @VAR("Adam",53) = ERR @VAR(B1..B4) = 54.6875 (if B1=10, B2=15, B3="Susan", B4=20)

@VLOOKUP

Format: @VLOOKUP(*X,Block,Column*)

X = a string or numeric value
Block = a block value
Column = a numeric value => 0

@VLOOKUP works along the same basic principles as @HLOOKUP, except the rows and columns are reversed.

@VLOOKUP searches (vertically) down the first column of the given *Block* for the given value X. When found, it returns the value itself (if *Column* = 0), or the value displayed the specified number of columns to the right (as indicated by *Column*).

The value of *X* can be either a character string or a number, the address or block name of any cell containing a label or value, or any expression that results in a number or string. If *X* is a string, Quattro looks for a match in either upper or lowercase. If *X* is a number and Quattro can't find an equal number, it locates the highest number in the column not greater than *X*.

The second argument (*Block*) specifies the coordinates of the table to be used for the lookup. This table must have its *index* values in the first column. These values (if numbers) must be in ascending numerical order. Also, there must be no blank cells in the index column, nor any blank columns in the table.

Quattro looks through the index column of the table from top to bottom looking for a match to *X*. If it finds an exact match, Quattro stops at that column. If Quattro can't find a match for *X*, it stops at the value closest to but not greater than *X*. If *X* is a number and the index column contains only labels, Quattro returns ERR.

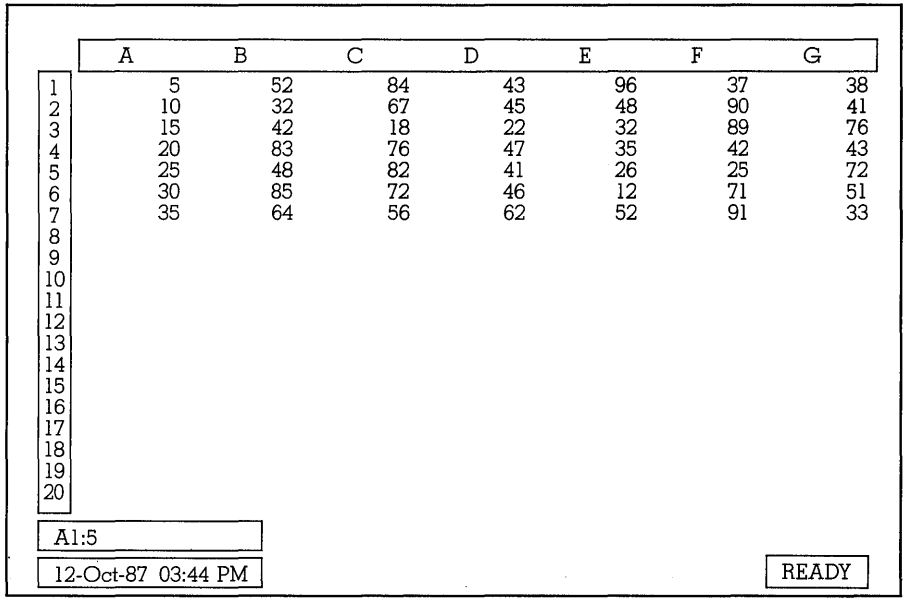
The final argument (*Column*) tells Quattro how many columns to the right the return value is. This argument can be any value from 0 up to the number of columns in the table, or any expression or cell address resulting in such a value. A value of 0 tells Quattro to return the actual value found in the index column. A value of 1 instructs Quattro to return the value directly to the right of the one found in the index column; a value of 2 tells Quattro to return the value two columns to the right, and so on.

The following instances will result in ERR:

- *Column* is less than 0 or greater than the number of columns in *Block*.
- *X* is less than the smallest value in the first column of *Block*.

Examples

The following examples refer to data in the lookup table below. In the first example, Quattro searches down the first column of the specified block (column A), looking for the largest number equal to or less than 17. It stops at cell A3, then moves across the specified number of columns (3). It stops at cell D3 and returns the value 22.



	A	B	C	D	E	F	G
1	5	52	84	43	96	37	38
2	10	32	67	45	48	90	41
3	15	42	18	22	32	89	76
4	20	83	76	47	35	42	43
5	25	48	82	41	26	25	72
6	30	85	72	46	12	71	51
7	35	64	56	62	52	91	33
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							

@VLOOKUP(17,A1..G7,3) = 22
@VLOOKUP(10,A1..G7,0) = 10
@VLOOKUP(6,A1..G7,2) = 84
@VLOOKUP(50,A1..G7,3) = 62
@VLOOKUP("string",A1..G7,3) = ERR (no labels in block)
@VLOOKUP("18",A1..G7,2) = ERR (no labels in block)
@VLOOKUP(18,A1..G7,8) = ERR (col value > # cols)
@VLOOKUP(18,A1..C7,4) = ERR (col value > # cols in given block)

See also @HLOOKUP for searching horizontally through a table (page 163).

@YEAR

Format: @YEAR(*DateTimeNumber*)

DateTimeNumber = a numeric value between 0 and 73050.9999999 representing a date/time serial number

@YEAR returns the year portion of *DateTimeNumber*. The result will be between 0 (1900) and 199 (2099). To display the actual year, just add 1900 to the result of @YEAR. If you want to extract the year portion of a string that is in Date format, use @DATEVALUE within the @YEAR function to coerce the string into a serial number.

Examples

@YEAR(22222) = 60 (1960)
@YEAR(A6)+1900 = 19nn, where n is the year value in A6
@YEAR(@DATEVALUE("12-Oct-54")) = 54

C H A P T E R

3

Macro Commands

Macro Commands by Type

Macro commands fall into five categories:

- **System** commands affect the screen display and computer bell.
- **Interaction** commands let you create interactive macros that pause for the user to enter data from the keyboard.
- **Program Flow** commands are programming commands that let you include branching and looping in your macro.
- **Cell** commands affect the data stored in specified cells.
- **File** commands work with data within files other than your current spreadsheet file.

Table 3.1 lists the commands in each category, along with the syntax and a brief description. The next section, "Macro Command Descriptions," describes each command in detail.

Table 3.1: Macro Commands Listed by Category

Command	Description
System Commands	
{BEEP *Number*}	Sounds the computer's bell
{INDICATE *String*}	Sets the mode indicator to display *String*
{PANELOFF}	Prevents menus and prompts from being displayed
{PANELON}	Restores menus and prompts disabled by {PANELOFF}
{WINDOWSOFF}	Prevents the screen from being updated
{WINDOWSON}	Restores screen updating disabled by {WINDOWSOFF}
Interaction Commands	
{ ? }	Pauses the macro and accepts input from the keyboard
{BREAKOFF}	Disables the *Ctrl-Break* key used to abort a macro
{BREAKON}	Restores the *Ctrl-Break* key after it was disabled by {BREAKOFF}
{GET *Location*}	Pauses the macro, accepts one keystroke, and stores it in *Location*
{GETLABEL *PromptString,Location*}	Pauses the macro, displays *PromptString*, and stores the subsequent keystrokes as a label in *Location*
{GETNUMBER *PromptString, Location*}	Pauses the macro, displays *PromptString*, and stores the subsequent keystrokes as a numeric value in *Location*
{LOOK *Location*}	Places the first keystroke typed in *Location*
{MENUBRANCH *Location*}	Pauses the macro, and lets you build a custom menu at *Location*
{MENUCALL *Location*}	Pauses the macro, and executes a customized menu as a subroutine
{WAIT *TimeNumber*}	Pauses the macro for the amount of time specified in *TimeNumber*

Table 3.1: Macro Commands Listed by Category

Command	Description
Program Flow Commands	
{BRANCH *Location*}	Passes execution control to another macro at *Location*
{DEFINE *Location1: Type1,...*}	Defines the type of arguments passed to a subroutine and tells where to store them
{DISPATCH *Location*}	Branches indirectly or directly to an alternate macro branch
{FOR *CounterLoc,Start#, Stop#,Step#,StartLoc*}	Executes a subroutine the specified number of times
{FORBREAK}	Terminates execution of a {FOR} command
{IF *Condition*}	Checks to see if a condition is TRUE or FALSE before continuing execution
{ONERROR *BranchLocation, MessageLocation*}	Continues execution at a specified location after an error is detected
{QUIT}	Terminates macro execution and returns keyboard control
{RESTART}	Terminates macro execution at the end of the current subroutine
{RETURN}	Terminates the current subroutine and returns control to main routine
{STEPOFF}	Exits Single-Step mode, executing the macro at normal pace
{STEPON}	Enters Single-Step mode, in which the macro is executed step by step, advancing when you press the space bar
{SubRoutine, *ArgumentList*}	Jumps to the specified subroutine, and passes any given arguments
Cell Contents Commands	
{BLANK *Location*}	Erases a cell or block of cells
{CONTENTS *Dest,Source, (Value:Type)*}	Copies the contents of one cell to another location
{LET *Location,Value:Type*}	Places the given value as a number or string in *Location*
{PUT *Location,Column#, Row#, (Number | String)*}	Places a number or string in a given *Location* offset by the specified number of columns and rows
{RECALC *Location, Condition,Iteration#*}	Recalculates rows of the spreadsheet a specified number of times
{RECALCCOL *Location, Condition,Iteration#*}	Recalculates columns of the spreadsheet a specified number of times

Table 3.1: Macro Commands Listed by Category

Command	Description
File Commands	
{CLOSE}	Closes an open file
{FILESIZE *Location*}	Calculates the number of bytes in the current file, and stores the value in *Location*
{GETPOS *Location*}	Places the position of the file pointer in *Location*
{OPEN *Filename, AccessMode*}	*Opens a file for reading, writing, or modifying*
{READ *#Bytes,Location*}	Reads the specified number of bytes and stores them in *Location*
{READLN *Location*}	Reads a line of characters and stores it in *Location*
{SETPOS *FilePosition*}	Sets the file pointer to the value of *FilePosition*
{WRITE *String*}	Writes a string of characters to the current file
{WRITELN *String*}	Writes a string of characters to the current file and ends it with a carriage return/line-feed
Miscellaneous Commands	
{ ; }	Lets you enter explanatory remarks in a macro
{ }	Lets you enter a blank line in a macro

The /x Commands

Quattro also accepts the /x commands used by other spreadsheet programs. These commands all have macro command equivalents and work the same as those equivalents (with the exception of X1 and Xn, as noted below. The only difference is that they can be entered from the keyboard during recording instead of using the MACRO LIST key.

Table 3.2 lists each of the /x commands and their equivalents.

Table 3.2: /x Commands and the Corresponding Macro Commands

/x Command	Macro Command Equivalent
/xc	{SubRoutine}
/xl	{GETLABEL}
/xn	{GETNUMBER}
/xg	{BRANCH}
/xi	{IF}
/xm	{MENUBRANCH}
/xq	{QUIT}
/xr	{RETURN} and {RET}

The arguments for /x commands are entered directly after the command. For example,

```
/xlName? {~}C10{~}
```

displays a prompt that reads, "Name?", then stores the response in cell C10. The tilde {~} key-equivalent commands insert carriage returns after the prompt and the cell address. Tildes are required after each argument. (The {CR} command cannot be used for carriage returns in /x commands.) Unlike their macro command equivalents, /xl and /xn do not require a location parameter. If no location is included, the current cell is assumed.

Macro Command Descriptions

The rest of this chapter describes each macro command. It shows the format to use and defines any arguments required.

{ }

Format: { }

{ } inserts a blank line in the macro without stopping the macro. Quattro continues executing the macro on the next line.

See { ; } for an example.

{;}

Format: {;*String*}

String = any combination of numbers or alphabetic characters, up to 237 characters in length

{;}

{ ; } allows you to enter explanatory remarks into your macro. When Quattro encounters this macro command, it skips over it.

The { ; } command is especially convenient as a way to temporarily hide other macro statements, such as a branch to a macro subroutine not yet completed.

Example

The following example runs the Int-update subroutine that calculates interest to date, then branches to the Print_invoice subroutine to print the invoice.

```
{; calculate interest to date}
{Int-update}
{}
{; Print invoice}
{BRANCH Print_invoice}
```

{ ? }

Format: { ? }

{ ? } pauses macro execution and returns control to the keyboard. The user can then enter any necessary information. When the user presses *Enter*, control passes back to Quattro and the macro continues. The keystrokes entered by the user (except the *Enter*) are either stored in the current cell or are used to select a menu item.

{ ? } is similar to the {GETLABEL} and {GETNUMBER} commands. Like these macro commands, it collects information from the user. There are several major differences, however. { ? } does not display a prompt or let you specify a storage location other than the current cell. It also does not distinguish between label and value entries. On the other hand, it allows the user more freedom and flexibility in response to the command. The user can move anywhere in the spreadsheet, making any changes desired.

You can precede { ? } with a {GOTO} command to move the selector to the cell you want to store the information in. You can also have the macro enter a "prompt" in the current cell to be replaced by the information entered by the user. Because the carriage return entered by the user returns control to Quattro, be sure to include a carriage return (tilde) after { ? }. This enters the user's input in the given cell.

Caution: The user has total keyboard control between { ? } and pressing *Enter* and can use function keys, access menus, and issue commands. Therefore, you should use this command with caution. Don't include it in macros that may be used by inexperienced users.

Example

The following macro moves the selector to cell E15, enters the label "Check #?" in the cell to act as a prompt, then passes control to the keyboard so the user can enter a check number. If the transaction was cash, the user can enter the label "Cash". The information entered by the user replaces the "Check #?" prompt.

```
{GOTO}E15~
Check #?~
{?}~
```

{BEEP}

Format: {BEEP *Number*}

Number = 1 to 4 (optional)

{BEEP} sounds the computer's built-in speaker. *Number* dictates the exact tone of the beep. If *Number* is omitted, {BEEP 1} is sounded. You can choose from the following tones:

{BEEP 1}	Low
{BEEP 2}	Standard
{BEEP 3}	Medium
{BEEP 4}	High

{BEEP} is used to catch the user's attention. You can use it in interactive macros to introduce a prompt for information, or to indicate that a macro has finished.

Example

The following macro checks a block named errorCheck for an error condition detected by the formula @ISERR(*X*). If there's no error, it branches to a macro called "continue," which carries on the previous procedure. If there is an error, it gives a low, then medium beep and moves the selector to the block called "messagearea," where an error message is stored.

```
{IF errorCheck=0}{BRANCH continue}
{BEEP 4}{BEEP 2}{GOTO}messagearea~
```

{BLANK}

Format: {BLANK *Location*}

Location = cell or block you want erased

{BLANK} erases the cell or block referred to as *Location*. This is the same as the Block Erase command, although because it doesn't directly access Quattro menus, you can use it while another menu is on the screen. For instance, {BLANK} can erase a cell from within the Graph menu without having to exit to the main menu.

Example

The following macro erases the block named PARTLIST, then uses the Combine command to place the contents of the spreadsheet file PARTS.WKQ into that area of the spreadsheet.

```
\f          {BLANK partlist}
            {GOTO}partlist~
            /{File,CopyFile}{clear}c:\quattro\parts{~}
```

{BRANCH}

Format: {BRANCH *Location*}

Location = location or name of another macro

{BRANCH} exits the current macro and begins executing the macro specified by *Location*. If *Location* references a block of cells, Quattro branches to the top left cell.

{BRANCH} is usually used to determine the flow of execution based on a condition test. For example, it can be used to execute a different macro depending on the contents of a certain cell.

{BRANCH} is similar to {Subroutine} in that it passes control to another macro. However, unlike {Subroutine}, it doesn't hold your place in the original macro, waiting for control to return. Therefore, you should use {BRANCH} when you don't intend to return to the original macro. To temporarily exit the current macro, use the {SubRoutine} command (see page 243). You can also use the \x equivalent \xg to branch to another macro (see page 208).

Example

The following macro branches to a macro called "HIGH" if the value in cell D10 is higher than 1000; otherwise, it continues execution of the same macro.

{IF D10>1000}{BRANCH high}

{BREAKOFF}

Format: {BREAKOFF}

{BREAKOFF} disables the *Ctrl-Break* key, which can be used to immediately terminate a macro. After {BREAKOFF}, the user won't be able to exit a macro either inadvertently or deliberately until the end of the macro or until a {BREAKON} command is called.

Caution: The {BREAKOFF} command should be used only when necessary. Without access to the *Ctrl-Break* key, the only way to stop a "runaway" macro is to reboot.

Example

The following excerpt from a macro disables the *Ctrl-Break* key while the user inputs a name.

```
Enter your name here:{CR}
{BREAKOFF}
{?}~
{BREAKON}
```

{BREAKON}

Format: {BREAKON}

{BREAKON} enables the *Ctrl-Break* key after a previous {BREAKOFF} command has disabled the *Ctrl-Break* key.

{BREAKON} should be used as soon as possible after {BREAKOFF}, because with *Ctrl-Break* disabled, the only way to halt a "runaway" macro is to reboot.

See {BREAKOFF} for an example of using both {BREAKOFF} and {BREAKON}.

{CLOSE}

Format: {CLOSE}

{CLOSE} ends access to an open file. This allows another file to be opened (only one can be open at a time). {CLOSE} completes the process of writing information to a file, including update of the disk directory. This step is crucial to the integrity of any file. Should your computer be turned off before a file is closed, that file's contents may become corrupted or lost.

The only situation in which {CLOSE} would fail is in the event of a disk error, such as when a disk is removed from the disk drive before the file is closed. In this case, {ONERROR} is useful in intercepting the error.

Example

The following macro opens a new file in Drive A called "AFILE", writes the ASCII text line "Hello, world." to the file, and closes the file.

```
\f          {OPEN "A:AFILE",W}
            {WRITELN "Hello, world."}
            {CLOSE}
```

{CONTENTS}

Format: {CONTENTS *Destination,Source,Width,Format*}

Destination	=	Cell you want data written to
Source	=	Cell you want data copied from
Width	=	optional column width (1 to 72)
Format	=	optional format code (see table below)

The {CONTENTS} command copies the contents of *Source* into *Destination*, but, unlike {LET} or other copy commands, if *Source* contains a value entry, it translates the copied value into a left-aligned label. It also lets you specify a different display format and column width using the *Width* and *Format* arguments.

Width can be any number from 1 to 72. Quattro will not alter the width of the destination column, but will simply treat the resulting string as though it came from a column with the specified width. For example, if a value is displayed as ***** in the source column because the column is not wide enough, specifying a wider column width will allow the value to be copied as it would be displayed within that width, not as *****. *Width* is optional, but must be provided if the *Format* argument is to be used. If you don't specify *Width*, the width of the source column is assumed.

Format can be any number from 0 to 127. Each number in this range corresponds to a specific display format and decimal precision (see Table 3.3). *Format* affects the Destination entry only, not the Source value.

Table 3.3 lists the special codes used to indicate display formats with the *Format* argument.

Table 3.3: Display Format Codes

Code	Description
0 – 15	Fixed (0 – 15 decimals)
16 – 31	Scientific (0 – 15 decimals)
32 – 47	Currency (0 – 15 decimals)
48 – 63	% [Percent] (0 – 15 decimals)
64 – 79	, [Comma] (0 – 15 decimals)
112	+/- [Bar graph]
113	General
114	Date [1] (DD-MMM-YY)
115	Date [2] (DD-MMM)
116	Date [3] (MMM-YY)
117	Text
118	Hidden
119	Time [1] (HH:MM:SS AM/PM)
120	Time [2] (HH:MM AM/PM)
121	Date [4] (Long International)
122	Date [5] (Short International)
123	Time [3] (Long International)
124	Time [4] (Short International)
127	Default (set with the Default Format Display command)

Examples

The following examples assume that cell C18 contains the value 48,988 in Comma format with a column width of 12.

{CONTENTS A18,C18} places the 12-character label 48,988 in cell A18.

{CONTENTS E10,C18,3} places the 3-character label *** in cell E10. (Only asterisks are copied because the value doesn't fit within 3 spaces.)

{CONTENTS A5,C18,15,34) places the 15-character label
" $48,988.00" in cell A5 (five spaces are inserted at the beginning).

{DEFINE}

Format: {DEFINE *Location1:Type1,Location2:Type2,...*}

Location = the cell in which you want to store the argument being passed

Type = the data type of the argument passed (string or value)

When you pass control to a subroutine with the {SubRoutine} command, you can also pass arguments for use by that subroutine. If you do so, you must include a {DEFINE} command in the first line of the subroutine. This command lets you define the data type of each argument passed and indicate which cells to store them in. If no {DEFINE} command is included, the arguments are ignored.

The {DEFINE} command can define numerous arguments passed to the subroutine. It defines them sequentially. In other words, the first location and type given are assigned to the first argument passed; the second location and type are assigned to the second argument, and so on.

For each argument, you must specify a location. This tells Quattro where to copy each argument. If there are more arguments than locations, the extra arguments are ignored. If there are more locations given than there are arguments passed, the macro is immediately terminated, and an error message is displayed.

Type is optional. It tells Quattro whether the argument is a value or string. If no data type is given, the argument is assumed to be a literal string (even if it's a valid block name, cell address, or value).

Example

In the following example, the \f macro passes three arguments (principal, interest, and term) to a subroutine called CalcLoan. CalcLoan stores the arguments in named blocks and defines them as values. It then uses those arguments to calculate the monthly payment on a loan and stores the result in a cell named "amount." It then creates a label in a cell called "payment" displaying that amount as currency, and returns control back to the main macro. The main macro then displays the result in the current cell preceded by the string "The monthly payment will be."

```
\f          {CalcLoan 79500,12%,30}

CalcLoan    {DEFINE prin:value,int:value,term:value}
            {LET amount,@PMT(prin,int/12,term*12)}
            {CONTENTS payment,amount,9,34}
```

	{RETURN}
prin	79500
int	0.12
term	30
amount	817.747
payment	$817.75
	+"The monthly payment will be "&@TRIM(PAYMENT)

{DISPATCH}

Format: {DISPATCH *Location*}

Location = a single cell containing the address or block name of another macro

{DISPATCH} continues macro execution at the cell referenced by *Location*. This is similar to {BRANCH}, except that the reference is indirect. Instead of specifying the cell that contains the macro to branch to, {DISPATCH} specifies a cell that contains the address of the macro to branch to.

{DISPATCH} is useful when you want to branch to one of several alternative macros, depending upon circumstances. You can also set up a macro that modifies the contents of *Location* itself, depending on user input or test conditions.

Example

The following macro utilizes {DISPATCH jumpcell} to create a form letter that changes, depending on the result of a credit check.

CREDIT	OK
\f	{GOTO}txtArea~ {IF credit="OK"} {LET jumpcell,"OK_Note"} {BRANCH continue} {LET jumpcell,"BUM_Note"} {BRANCH continue}
continue	Because of your current standing with P.K. Finance,{down} we have decided to {DISPATCH jumpcell}.{down}{down}
jumpcell	OK_Note

OK_Note	extend your credit limit.{down 2} {BRANCH finish}
BUM_Note	CANCEL your account.{down 2} {BRANCH finish}
finish	Please call Mr. Kahn at our office for details.{down 2} Sincerely,{down} James Madison (Account officer){down}
txtArea	Because of your current standing with P.K. Finance, we have decided to extend your credit limit. Please call Mr. Kahn at our office for details. Sincerely, James Madison (Account officer)

{FILESIZE}

Format: {FILESIZE *Location*}

Location = any cell address or block name

{FILESIZE} calculates the number of bytes in the currently open file and places the result in *Location* as a value. If *Location* is a block, the result is placed in the upper-left cell of the block. If no file is open, Quattro ignores this command and continues executing the current macro.

Examples

The following example opens the file "MYFILE.TXT" and places its size (in bytes) in the cell named numBytes.

\f	{OPEN "MYFILE.TXT",R} {FILESIZE numbytes}
numbytes	4632

The example below shows use of {FILESIZE} and other file macro commands. This macro creates a file, writing text to it, and then reads it back character by character. {FILESIZE} is used to calculate how many times the {FOR} loop should get another character. Before the file is read,

the file pointer is set to the beginning with {SETPOS}. As the characters are read in, they are concatenated to reconstruct the original sentence.

```
\m          {OPEN "MYFILE.TXT",W}
            {WRITE "The quick brown fox jumped"}
            {WRITE "over the lazy dog."}
            {FILESIZE bytes}
            {SETPOS 0}
            {LET string,""}
            {FOR counter,1,bytes,1,readChar}
            {CLOSE}

readChar    {READ 1,Char}
            {LET string,+String&char}
            {Calc}

counter     46
bytes       45
string      The quick brown fox jumped over the lazy dog.
char
```

{FOR}

Format: {FOR *Counter,Start#,Stop#,Step#,StartLocation*}

Counter	=	a cell used to count the number of macro iterations
Start#	=	the initial value to be placed in *Counter*
Step#	=	the amount added to *Counter* after each iteration
Stop#	=	the maximum value for *Counter*
StartLocation	=	the cell containing the subroutine to be executed

The {FOR} command repeatedly executes a macro subroutine beginning in *StartLocation*, creating a macro *loop*. Quattro keeps track of how many times the macro is executed by using the cell specified with *Counter* as an incremental counter. The initial value of the counter is *Start#*. Each time the macro loop is executed, the number stored in *Counter* is increased by *Step#*. When the value in *Counter* reaches or exceeds *Stop#*, execution stops.

{FOR} is similar to the FOR..NEXT control structure found in many programming languages. It offers the capability of executing a particular subroutine a predetermined number of times. Although this can be

effectively implemented with other macro commands, {FOR} is a more readable version, and is easier to debug and modify.

The {FOR} command is most flexible if you store the values for *Start#*, *Stop#*, and *Step#* in spreadsheet cells, then reference those cells with the macro. Then you'll be able to alter the command without editing the macro.

Examples

{FOR D15,1,5,1,E30} executes the subroutine beginning in cell E30 five times.

{FOR D15,1,10,2,E30} executes the subroutine ten times.

{FOR D15,1,5,0,E30} executes the subroutine continously until you press *Ctrl-Break*.

{FOR D15,1,0,1} does not execute the subroutine.

The following macro automatically creates Qtr labels for reports.

```
\q          {; position cursor where Qtr labels should
            appear}
            {FOR counter, 1987,1990,1,list}
            {; return to original position}
            {LEFT}{END}{LEFT}

counter

list        Qtr1{right}
            Qtr2{right}
            Qtr3{right}
            Qtr4{right}
```

{FORBREAK}

Format: {FORBREAK}

{FORBREAK}, when used within a subroutine called by {FOR}, cancels the execution of the subroutine, and effectively terminates the processing of {FOR}. Macro execution continues normally on the line below the {FOR} command.

If {FORBREAK} appears any place other than in a subroutine called by {FOR}, macro execution is halted, and Quattro displays an error message.

{FORBREAK} is usually used in conjunction with an {IF} command to exit the macro if a specific condition is reached, for example, if an error condition is found.

Example

The following example shows how a {FORBREAK} command terminates a loop used to enter names into a list. Enter "STOP" to stop the loop.

```
\f          {GOTO}list_top~
            {FOR counter,1,10000,1,get_name}

counter

get_name    {GETLABEL "Enter next name: ",name_cell}
name_cell   stop
            {DOWN}
            {IF name_cell="STOP"}{FORBREAK}

list_top
```

{GET}

Format: {GET *Location*}

Location = the cell in which to store the keystroke entered by the user

{GET} pauses macro execution, accepts one keystroke from the user, then stores a representation of that keystroke into *Location* as a left-aligned label. It then continues macro execution without waiting for a carriage return. Almost any Quattro keystroke can be recorded with {GET}. If the key pressed results in a non-printable character, such as *Esc* or *Enter*, its key-equivalent command is recorded. The only exceptions are *F1*, which is not available from within a macro, and *Ctrl-Break*, which will interrupt the macro (as long as {BREAKOFF} is not in effect). Unlike { ? }, the keystroke is not passed on to Quattro.

{GET} is of limited use in most applications because there is no prompt displayed and you can only type one character. An appropriate application for {GET} would be to prompt the user about whether to continue or stop. You can use the {LOOK} command with {GET} to check the typeahead buffer for user response. See also {MENUBRANCH}, {MENUCALL}, { ? }, {LOOK}, {GETNUMBER}, and {GETLABEL}.

Example

The following asks the user if he or she wants to continue.

```
\f          {RIGHT 10}
            {; display msg_area in top left of screen}
```

	{GOTO}msg_area~
again	Press Y to continue...~
	{GET keystroke}
	{IF keystroke<>"Y"}{BEEP 2}{HOME}{QUIT}
	{BEEP 4}{BRANCH again}
keystroke	
msg-area	Press Y to continue...

{GETLABEL}

Format: {GETLABEL *PromptString,Location*}

PromptString = a string displayed to the user as a prompt

Location = a cell in which to store the user's response

{GETLABEL} pauses macro execution, displays *PromptString* at the top of the screen, and accepts keystrokes input by the user until *Enter* is pressed. At that time, Quattro stores the characters typed as a left-aligned label in *Location*. Unlike {GET}, Quattro's special keys are not accepted.

PromptString must be a literal string (not a cell reference), and quotation marks are optional. The prompt can be up to 70 characters long, and is truncated if longer. The response can be as long as 240 characters.

If {PANELOFF} is in effect, {GETLABEL} ignores it, allowing the control panel area to be viewed and updated as the input is typed. Once input is completed (indicated by *Enter*), the {PANELOFF} state is restored.

See also {GET}, {GETNUMBER}, and {PANELOFF}. The /x command equivalent to {GETLABEL} is /xL (see page 208)

Example

The following example stores the user's last name in the cell named last_cell.

\f	{GETLABEL"enter your last name:",last_cell}
last_cell	

{GETNUMBER}

Format: {GETNUMBER *PromptString,Location*}

PromptString = a string displayed to the user as a prompt

Location = a cell in which to store the user's response

{GETNUMBER} is similar to {GETLABEL}, except that it accepts only a numeric value, a formula resulting in a numeric value, or the cell address or block name of a cell returning a value. If a text value is input, ERR is entered in the cell. The prompt can be up to 70 characters long, and the response can be as long as 240 characters. The /x command equivalent to {GETNUMBER} is /xn (see page 208).

Example

The following example stores the user's age in the cell named age_cell.

```
\f          {GETNUMBER "Enter your age:",age_cell}
age_cell
```

{GETPOS}

Format: {GETPOS *Location*}

Location = the cell in which to store the retrieved value

{GETPOS} places the position of the file pointer into Location as a value. If no file is open, the command is ignored.

The file pointer is a number corresponding to the position at which the next character written to the file will be placed. If the file pointer is 0, the next character written to the file with {WRITE} or {WRITELN} will be placed at the beginning of the file. If the file already contains information, it will be overwritten, beginning at the first position in the file. After the file is written to, the file pointer is always positioned immediately after the last character written. If a file is newly created, then written to for the first time, the file pointer will also be the size of the file.

Example

In the following macro example, the text file TEST.TXT is opened for reading, a line is read in, and the length of the line is calculated by subtracting the starting position from the ending position. Note that the calculated length is two characters longer than the apparent length of the string. This is because the CR and LF characters, found at the end of each line in a typical text file, are stripped away by Quattro when the text is read into cells. The text file read in here was created by the macro example provided in the {WRITELN} command description.

\f	{OPEN "A:test.TXT",R} {GETPOS start} {READLN input} {GETPOS end} {CLOSE} {Let num_char,+end-start}
input	This is a short line.
start	0
end	23
num_char	23

{IF}

Format: {IF *Condition*}

Condition = a logical expression (or address of a cell that contains a label, value, or an expression)

{IF} operates in much the same manner as the @IF function. {IF} evaluates *Condition*; if the condition is numeric and non-zero, it is considered to be TRUE and macro execution continues in the same cell; if the condition is anything else, it is considered FALSE and macro execution continues in the cell directly below the {IF} command. However, unlike @IF, {IF} commands cannot be nested within each other.

{IF} is similar to the IF..THEN..ELSE control structure found in many programming languages. For instance, the following macro implements the IF..THEN..ELSE routine below it.

```
\f      {IF gpa>.59}{BRANCH passStudent}
        {BRANCH failStudent}
```

In a programming language, the above sequence would read:

```
IF gpa>.59
THEN GOTO passStudent
ELSE GOTO failStudent
```

When using {IF}, be sure to include a {BRANCH} or {QUIT} in the same cell as the {IF}, as shown above. Otherwise, you could find both your THEN and ELSE clauses being executed, instead of one or the other.

{IF} statements can be stacked in order to branch according to the case. The /x command equivalent to {IF} is /xI (see page 208).

Example

The following example uses a string of {IF} commands to award a grade based upon a test result:

```
\g      {IF result>=.90}{BRANCH give_A}
        {IF result>=.80}{BRANCH give_B}
        {IF result>=.70}{BRANCH give_C}
        {IF result>=.56}{BRANCH give_D}
        {BRANCH give_F}
```

In a programming language, the above sequence would read:

```
IF result >= .90
   THEN give_A
      ELSE IF result >= .80
         THEN give_B
            ELSE IF result >= .70
               THEN give_C
                  ELSE IF result >= .56
                     THEN give_D
                        ELSE give_F
```

{INDICATE}

Format: {INDICATE *String*}

String = any character string (not a cell reference)

{INDICATE} sets the mode indicator in the lower-right corner of the screen to read whatever is given as *String*. *String* is optional and need not be surrounded by quotes. If *String* is longer than five characters, only the first five characters are used. If spaces are to appear in the mode indicator (as in the second example below), use a quoted string. To restore the mode indicator to its normal settings, use {INDICATE} with no arguments. To remove the mode indicator display, use {INDICATE""}.

Examples

\f	{INDICATE Save!}
\g	{INDICATE " Go! "} – note use of spaces
\h	{INDICATE E14} – prompt reads "E14", because cell references are ignored...
\i	{INDICATE} – restores normal mode indicator.

{LET}

Format: {LET *Location,Value:Type*}

Location = cell in which to store the specified value

Value = numeric or string value to be stored in *Location*

:Type = *:string* or *:value*; *:string* stores the value or formula as a label, and *:value* stores the actual value or value resulting from a formula (optional)

With the {LET} command, you can enter a value into a cell while a macro is running, without having to actually move the cell selector to *Location*. {LET} enters the value or string you specify with *Value* in the cell specified with *Location*.

You can use the optional *Type* argument to specify whether to store *Value* as a actual number or as a string. If you specify a formula as a string, the formula is written into *Location* as a string, not the resulting value. For example, {LET A1,B3*23:string} stores the formula B3*23 as a label in cell A1. If you omit the *Type* argument, Quattro tries to store the value as a numeric value; if unsuccessful, it stores the value as a string.

Examples

The examples below assume that cell A1 contains the label "Dear", cell A2 contains the label "Sir", and cell A3 contains the value 25. To the right of each {LET} is shown the result:

\m	{LET F1,25}	25
	{LET F2,A3}	25
	{LET F3,+A1&" "&A2}	Dear Sir
	{LET F4,+A1&" "&A2:value}	Dear Sir
	{LET F5,+A1&" "&A2:string}	+A1&" "&A2
	{LET F6,+A1&A3}	ERR

{LOOK}

Format: {LOOK *Location*}

Location = a cell in which to store a typed character

When Quattro is executing a macro, it doesn't respond to keystrokes entered by the user (except *Ctrl-Break*). If the user presses keys during a

macro, those keystrokes are stored in the computer's "typeahead buffer," and are responded to when the macro pauses or is terminated.

The {LOOK} command checks this typeahead buffer for any stored keystrokes. If any are found, it places the first keystroke typed in *Location*.

{LOOK} can be used while processing long macros, such as creating and printing a report, to check for a keystroke which might signal that the user wishes to quit (see the first example below).

{LOOK} does not remove the keystroke from the buffer. If a macro does nothing other than {LOOK}, the key will still be passed on to Quattro when the macro ends. If you wish to remove any pending keystrokes from the buffer, you can use the {GET} command, as in the second example below. See also {GET} on page 221.

Examples

In the following example, Quattro gives you 15 seconds to select the next menu choice. If you do not, you must reenter the password.

```
start_time
keystroke
pass

\m              {GOTO}msg_area~
                A. Add name{down}
                B. Edit name{down}
                C. Delete name{down}
                Enter choice: {right}
                {}
                {LET start_time,@NOW}
check_it        {LOOK keystroke}
                {IF keystroke=""}
                {IF @NOW>start-time+@TIME(0,0,15)}
                {BRANCH password}
                {IF keystroke<>""}{BRANCH take_action}
                {beep 4}
                {BRANCH check_it}

password        {; get password from user}
                {GETLABEL "Enter password : ",pass}

take_action     {}
                {beep 3}

msg_area
```

{MENUBRANCH}

Format: {MENUBRANCH *Location*}

Location = a cell block containing a custom Quattro menu

{MENUBRANCH} pauses macro execution and displays the custom menu stored at *Location* (see below for instructions on building a custom Quattro menu structure). After the user selects an item from the menu, macro execution continues with the cell below the description of the menu choice. Often this is a {BRANCH}. The only exception is if the user presses *Esc*; in this case, the macro continues in the {MENUBRANCH} cell. {MENUCALL} is different from {MENUBRANCH} in that it calls the menu as a subroutine.

With the sole exception of the *Esc* key, a custom menu acts exactly the same as one of Quattro's own menus. The user can use the arrow keys to look at each item, and can select a menu item by either pressing *Enter* while highlighting it, or pressing the first letter of the menu item's name.

To build a custom menu:

1. Locate an area of the spreadsheet with enough empty columns for each of the items in your menu, plus an extra blank one on the right.
2. Along the top row, place each menu item's title in a separate column in the same row), leaving no blank cells between items. These titles must be labels; values are ignored.
3. In the row immediately below the titles row, place a more detailed description of the menu item. The user will see this description on the Description/Prompt Line when this item is highlighted on the menu.
4. Place the desired macro commands for each of the menu items in the rows immediately below the descriptions.
5. If you want to reference the menu by name instead of address, give the upper-left cell of the menu (the first menu item's title) a descriptive block name. It is not necessary to name the entire contents of the custom menu, because Quattro continues to add items to the menu until it reaches a blank column on the right.

Some suggestions when building custom menus:

- Never begin two or more items with the same letter. This would prevent the user from choosing menu items by typing the first letter.
- Use the second line of the menus to fully describe the effect of choosing that item.

- Wherever appropriate, include a "Quit" or "Return to Quattro" menu item. Forcing the user to press *Break* to return to Ready mode is not particularly "user friendly".
- The width of the columns containing the menu items has no bearing on the menu's appearance when it is displayed from a {MENUBRANCH}. However, choosing a remote area of the spreadsheet and widening the columns which contain the titles, descriptions, and macros improves their readability.

The /x command equivalent to {MENUBRANCH} is /xm (see page 208).

Example

The following macro displays a custom menu that offers the user three choices. If the user presses *Esc*, the menu is redisplayed with the {BRANCH \f} command.

```
\f          {MENUBRANCH QuitMenu}
            {BRANCH \f}

QuitMenu    Continue                Return                  Quit
            Continue this macro     Return to Ready         Leave Quattro
            {BRANCH \f}             {BRANCH continue}       {MENUBRANCH
                                                            Sure}
            {BEEP 4}

            Sure                    No                      Yes
                                    Stay in Quattro         Return to DOS
                                    {BRANCH \f}             /qy

            continue
```

{MENUCALL}

Format: {MENUCALL *Location*}

Location = a cell block containing a custom Quattro menu

{MENUCALL} pauses macro execution and displays the custom menu stored at *Location*. {MENUCALL} treats the called menu as a subroutine. After the user selects an item from the menu, macro execution continues in the cell that contained the {MENUCALL} command.

For a complete description of custom menus and how to build them, see {MENUBRANCH}.

Example

The following macro uses two consecutive custom menus to let the user change search criteria for a database. It then copies records that meet the criteria to the Output Block and branches to another macro to print the Output Block as labels.

```
\m          {;Choose members for label print}
            {MENUCALL m_status}
            {MENUCALL m_paid}
            {\ Query,Extract}
            {BRANCH print_labels}

m_status    Active                  Retired
            Active Members          Retired Members
            {LET status, "A"}       {LET status, "R"}

m_paid      Paid                    Unpaid
            Dues are paid           Dues are unpaid
            {LET paid, "T"}         {LET paid, "F"}
```

Database Criteria Block:

```
            STATUS      PAID
            A           F
```

{ONERROR}

Format: {ONERROR *BranchLocation,MessageLocation*}

BranchLocation = the first cell of the macro to be executed in case of an error

MessageLocation = the cell in which to store any error message (optional)

Normally, if an error occurs during macro execution, the macro is terminated and Quattro displays an error message on the screen's message line. The {ONERROR} command tells Quattro to branch to a different location (*BranchLocation*) if an error is encountered and store the error message in *MessageLocation* for future reference. The *MessageLocation* argument is optional. If omitted, however, no error message will be displayed or stored.

Only one {ONERROR} command can be in effect at a time, so each time it is called, the most recent {ONERROR} replaces the previous one. If an error occurs, the {ONERROR} state is "used up" and must be re-declared. It is

best to declare {ONERROR} at the very beginning of your macro, or at least before any procedure is likely to result in an error.

In general, {ONERROR} captures errors that arise when commands are entered from the keyboard. It will not capture macro programming errors, such as an incorrect sequence of commands, or an attempt to call a non-existing subroutine. Nor will it detect syntax errors within a macro itself, such as an error resulting from incorrect usage of a macro keyword.

Typical errors that {ONERROR} will detect are:

- disk errors, such as a disk drive door left open, a full disk, or failure to find a file of a given name
- an attempt to copy to a protected cell

For details on exactly which error are detected by {ONERROR}, see Appendix C, "Error Messages," in the *Quattro User's Guide.*

Examples

The following macro also uses {ONERROR} to trap possible errors in extracting the block "numbers" from the spreadsheet, saving it to a file on Drive A.

```
\g          {ONERROR fileErr,fErrMess}
            {/ File,ExtractValues}{clear}

fileErr     {BEEP 3}{GOTO}MessArea~
            Oops! There seems to have been an error:
            {Down}
            Fix the disk drive, and I'll try again...{Down}
            Press Enter to continue...{Down}{?}
            {BRANCH \g}

numbers     $125.35
            $95.00
            $30.35

fErrMess
MessArea
```

{OPEN}

Format: {OPEN *Filename,AccessMode*}

Filename = the name of a file

AccessMode = R, M, or W

{OPEN} establishes a connection to *Filename*, allowing other file-access macro commands ({READ}, {WRITE}, etc...) to be used with this file. There are three different access modes: Read-Only, Modify, and Write (described below). *Filename*'s full name and access path must be given, and the entire name must be in quotes. For instance, to open and modify the file DATA.TXT in a subdirectry called FILES on Drive C, use the command `{OPEN "C:\FILES\DATA.TXT",M}`. If any part of the access path or filename is left out, the file may not be found. (See Appendix B in the *Quattro User's Guide* for more on DOS directories.)

Once {OPEN} is executed successfully, Quattro skips to the next row and continues executing instructions there. {OPEN R} and {OPEN M} will fail if the file is not found. {OPEN W} will only fail if either the supplied access path or filename is somehow invalid. If, for any reason, {OPEN} fails, Quattro continues executing commands in the current cell. {ONERROR} cannot be used to "trap" errors that occur in the {OPEN} command. For an example of how to effectively trap errors, see the third example below.

Warning: Although {OPEN} can provide access to any type of file (including.WKQ spreadsheet files), Quattro's file-access macro commands are designed to work only with ASCII (plain text) files. Using these commands with any other file type is not recommended, and can result in corruption of that file.

Access modes:

R (read-only)	Allows only reading from this file, insuring that no changes are made to it. Only {READ} and {READLN} can be used with a file opened as read-only.
M (modify)	Opens an existing file for modification. All reading and writing commands can be used with a file opened for modification.
W (write)	Opens a new file with the name given in the {OPEN} command. If a file already exists with that name, the existing file is erased. All reading and writing commands can be used with a file opened for writing.

Examples

The following example opens the file named "MYDATA.TXT" for reading. If the file doesn't exist in Quattro's default data directory, the command fails.

```
{OPEN "MYDATA.TXT",R}
```

The next example creates a file name "MYDATA.TXT" on Drive A for writing. If there is already a file on Drive A with that name, it is erased. This command will fail only if there is no disk in Drive A.

```
{OPEN "A:MYDATA.TXT",W}
```

The last example demonstrates how to do error trapping in an {OPEN} macro. When you press *Alt-H*, Quattro attempts to open the file C:\MYDIR\DATA.TXT. If that succeeds, the macro stops, since there are no more commands in the row below. If {OPEN} fails (meaning the file does not exist) the adjacent {BRANCH} is executed. "tryAgain" then attempts to create the file. If that succeeds, the macro restarts, since the {OPEN M} should succeed now that the file has been created. If "tryAgain" fails, that can mean only one thing: the directory path given was somehow invalid. Therefore, the adjacent {BRANCH} goes to "badDir", which displays a relevant error message with pertinent instructions.

```
\H          {OPEN "C:\MYDIR\DATA.TXT",M}
            {BRANCH tryAgain}

tryAgain    {OPEN "C:\MYDIR\DATA.TXT",W}
            {BRANCH badDir} {BRANCH \H}

badDir      {Goto}errLoc~{BEEP 4}
            The directory C:\MYDIR\ does not exist.
            {Down}
            Use the /System command to create it, then try again.
            {Down}

errLoc
```

{PANELOFF}

Format: {PANELOFF}

{PANELOFF} disables normal display of menus and prompts during execution of a macro. It can significantly speed up execution time for macros that use keystrokes to walk through menus, as it spares Quattro the time normally needed to draw its menus on the screen. Its effect is automatically cancelled by Quattro once the macro stops running, so you needn't worry about having the user locked out of the menus. To cancel its effect during macro execution, use the {PANELON} command.

Note: {PANELOFF} does not disable menu created for {MENUCALL} and {MENUBRANCH} commands.

This command can be used with {WINDOWSON} to completely disable normal screen updating.

See also {PANELON}, {WINDOWSON}.

Example

The following macro turns off the display of menus, specifies a new default data directory, then turns menu display back on.

```
\P          {PANELOFF}
            {/FD c:\receipts
            {PANELON}
```

{PANELON}

Format: {PANELON}

{PANELON} enables display of Quattro menus and prompts which have been disabled with {PANELOFF}. {PANELON} has no effect if used without an accompanying {PANELOFF}. Therefore, it can be used repeatedly with no ill effects.

This command can be used with {WINDOWSON} to completely restore normal screen updating.

See also {PANELOFF} and {WINDOWSON}. See {PANELOFF} an example.

{PUT}

Format: {PUT *Location,Column#,Row#,Value:Type*}

Location = a block within which *Value* will be stored, either as a value or label, as specified by *Type*

Column# = the number of columns into the specified block to store *Value*

Row# = the number of rows into the specified block to store *Value*

Value = any string or numeric value

Type = *string* or *value*; *string* stores the value or formula as a label, and *value* stores the actual value or value resulting from a formula (optional)

{PUT}, like the {LET} command, copies a value to a particular cell in the spreadsheet. However, instead of placing the value directly in the specified cell, {PUT} copies *Value* into the cell that is offset *Column#* columns and *Row#* rows into *Location*.

{PUT} processes *Value* the same way as {LET} does, including the use of ":string" and ":value". If neither of these two optional arguments is supplied, {PUT} tries to store the value as a numeric value; if unsuccessful, it stores the value as a string, or label.

The values for *Column#* and *Row#* can be any number between 0 and one less than the number of columns or rows within *Location*, respectively. A value of 0 implies the 1st column or row, 1 implies the 2nd, and so on. If *Column#* or *Row#* are larger than the number of columns or rows in the block, the macro stops. (This error cannot be trapped by {ONERROR}.)

See also {LET}.

Examples

Each of the following examples assumes that cell A41 contains the value 25, the block NUMBERS has been defined as A44..B50, and DATA is a cell containing the value 295.

{PUT NUMBERS,1,4,A41:value}	copies the value 25 into the cell at the intersection of the second column and the fifth row of the block NUMBERS (cell B47).
{PUT NUMBERS,1,5,A41:string}	copies the string "A41" into the cell at the 2nd column and the 6th row of the block NUMBERS (cell B48).
{PUT NUMBERS,1,6,DATA}	copies the contents of the block DATA to the 2nd column and 7th row of NUMBERS (B49). If there is no block named DATA, this example instead places a label ("DATA") into cell B49.

{QUIT}

Format: {QUIT}

{QUIT} terminates all macro execution, and returns control of Quattro to the user.

Use {QUIT} in conjunction with {IF}, {LOOK}, or one of the menu macro commands, to terminate a macro under user control.

The /x command equivalent to {QUIT} is /xq (see page 208).

Examples

The following macro displays a menu that has a "Quit" option, which returns the user to Quattro's Ready mode.

```
\g          {MENUBRANCH quitMenu}

quitMenu    Continue          Quit
            Keep going        Quit to Ready mode
            {BRANCH \g}       {QUIT}
```

{READ}

Format: {READ *#Bytes,Location*}

#Bytes = the number of bytes of characters to read from a file

Location = the cell in which to store the characters read

{READ} reads *#Bytes* bytes of characters from the currently open file (starting at the current position of the file pointer), and stores them as a label in *Location*. The currently open file is left unchanged, and the file pointer is moved to the position following the last character read.

{READ} is similar to {READLN}, except for two differences. While {READLN} reads one line of characters (terminated by a carriage-return/line-feed pair), {READ} reads the precise number of characters specified. This allows you to read, for example, fields within a record rather than an entire record. The second major difference is that while {READLN} strips out the carriage-return/line-feed pair at the end of a line, {READ} manipulates these as if they were no different from other characters. Should you use {READ} to read a file created by {WRITELN}, you will see two graphics characters at the end of each string read. These will be the carriage-return and line-feed characters. {READ} is best used only in conjunction with {WRITE}, or when you know the text file structure in detail.

{ONERROR} can be used to trap errors such as the disk drive not being ready.

Example

The following example shows the opening of a text file containing a phone directory database, and the grabbing of a name and phone number from the third record. The macro that created this text file is shown in the description of the {WRITE} command:

```
\l          {OPEN "A:PHONEDIR.PRN",R}
            {SETPOS rec_length*(rec_number-1)}
            {READ name_length,name}
            {READ phon_length,phone}
            {CLOSE}

rec_number  3
rec_length  25
name_length 13          name    Hall, Sue Ann
phon_length 12          phone   617-234-5678
```

{READLN}

Format: {READLN *Location*}

Location = the cell in which to store the characters read

{READLN} is similar to {READ}. Instead of using a number of bytes to determine the amount of text to be read, {READLN} automatically reads forward from the current file pointer location up to and including the carriage-return/line-feed at the end of the line. Unlike {READ} however, it does not read the carriage-return/line-feed into the cell.

Use {READ} to read lines from a record-structured file, where the lines are of uniform length. {READLN} can read the contents of a file one row at a time, making formatting on the spreadsheet much easier than {READ}.

Like the other file-access macros, if {READLN} fails, the macro continues execution in the current cell. If it is successful, the macro skips to the row below, and execution continues there. {ONERROR} can be used to trap disk and file errors, such as a disk drive door being left open.

Example

In the following macro, the text file TEXT.TXT is opened for reading, a line is read in, and the length of the line is calculated by subtracting the starting position from the ending position. Note that the calculated length is two characters longer than the apparent length of the string. This is because the carriage-return and line-feed characters found at the end of each line in a

typical text file are stripped away by Quattro when the text is read into cells.

\m	{OPEN "A:test.TXT",R} {GETPOS start} {READLN input} {GETPOS end} {CLOSE} {LET num_char,+end-start}	
input	This is a short line.	21
start	0	
end	23	
num_char	23	

{RECALC}

Format: {RECALC *Location,Condition,Iteration#*}

Location = the cell block to recalculate

Condition = the condition to be met before recalculation is halted (optional)

Iteration# = the maximum number of times to recalculate *Location* trying to meet *Condition* (optional)

{RECALC} causes Quattro to recalculate a selected portion of the spreadsheet in a row-by-row order. This is different from normal recalculation, where Quattro recalculates the entire spreadsheet in natural order, which means that before a formula is calculated, each cell it references is recalculated first.

With the optional *Condition* argument, you can tell Quattro to recalculate formulas in a block repeatedly until the specified *Condition* is met. You can also supply an *Iteration#* argument to specify the maximum number of times to recalculate formulas trying to satisfy *Condition*. In order to use *Iteration#*, the *Condition* argument must also be supplied.

{RECALC} is useful in that it allows very rapid recalculation of selected parts of a spreadsheet. This is very useful where the entire spreadsheet is so large that recalculations would excessively slow your work.

{RECALC} overrides the recalculation method currently specified for the spreadsheet, enforcing a row-by-row recalculation. If all the formulas

reference only cells above, or to the left, in the same row, the spreadsheet will be correctly calculated. Should there be references to cells to the right or above in the same column, you must use {RECALCCOL}. Should there be references to cells below and to the right of your formula, you will have to use {CALC} to recalculate the entire spreadsheet.

{RECALC} immediately displays the results of your recalculation. You do not need to use a command like a {LET} or {PGDN}{PGUP} to force redisplay.

Warning: If there are formulas within the block being recalculated that depend on other formulas outside of the block, they may not evaluate correctly. Make sure that the block specified in *Location* encompasses all of the cells referenced by formulas within it.

Example

The following example shows a block of formula entries all linked to the value in the top-left cell (C5). Below are shown the results of calculating the block with the CALC key (in Natural order), with {RECALC}, and with {RECALCCOL}. Each shows different results.

	(C5)n +C5+1 +C6+1 +C7+1	+C8+1 +D5+1 +D6+1 +D7+1
After CALC key Natural order (with 1 in C5)	1 2 3 4	5 6 7 8
After {RECALC} (with 100 in C5)	100 101 102 103	5 6 7 8
After {RECALCCOL} (with 100 in C5)	100 101 102 103	104 105 106 107

{RECALCCOL}

Format: {RECALCCOL *Location,Condition,Iteration#*}

Location = the cell block to recalculate

Condition = the condition to be met before recalculation is halted (optional)

Iteration# = the maximum number of times to recalculate *Location* trying to meet *Condition* (optional)

{RECALCCOL} is similar to {RECALC}, except {RECALCCOL} performs a columnwise recalculation instead of a rowwise calculation.

See {RECALC} for an example using {RECALCCOL}.

{RESTART}

Format: {RESTART}

{RESTART} terminates macro execution at the end of the subroutine in which it is encountered. In effect, {RESTART} can be thought of as replacing the next {RETURN} statement (or the end of the subroutine) with {QUIT}.

{RESTART} is typically used for error handling. If an application is already nested at near the maximum number of levels and a severe error occurs that requires the application to terminate, {RESTART} ensures that additional subroutine calls can be made.

Example

The following example, a deeply nested macro (nested_macro), calls a subroutine (getnum) that asks the user for a number, which is stored in A1. The subroutine then checks A1 for an ERR value (which would occur if the user entered something other than a number). If it finds an error, {RESTART} clears the path to calling macros, effectively saying, "This is now the main macro." (This ensures that another subroutine can be called.) It then calls the showerror subroutine, which beeps and displays an error message. When control returns to the getnum routine, the macro ends with the {RETURN} command, treating it like a {QUIT} command. If the user correctly enters a number, control returns directly to the nested_macro, the cell selector moves right five cells, and the macro continues.

```
nested_macro    {getnum}
                {right 5}
                  .
                  .
                  .

getnum          {GETNUMBER "Number:",A1}
```

```
                {IF @ISERR(A1)}{RESTART}
                {IF @ISERR(A1)}{showerror}
                {RETURN}
showerror       {BEEP}You didn't enter a number; try again.
```

{RETURN}

Format: {RETURN}

{RETURN} terminates the currently executing subroutine, and returns control to the macro that called it.

A {RETURN} command at the end of a subroutine is optional, since Quattro automatically returns from a subroutine when it reaches a blank cell or a cell containing a value. {RETURN} is usually used in conjunction with the {IF} command, to return to the main macro if a certain condition is met.

Subroutines are used to incorporate command sequences that are executed repeatedly in an application, or that are used by several macros. By pulling out the sequence as a separate routine, you can avoid reentering it each time it's used.

See {SubRoutine} for an example of using {RETURN}.

{SETPOS}

Format: {SETPOS *FilePosition*}

FilePosition = the number of bytes into a file to set the file pointer to

{SETPOS} moves the file pointer of the currently open file to the value *FilePosition*. (See {GETPOS} for a discussion of the file pointer.) *FilePosition* refers to the offset, in number of bytes, where you want to position the file pointer. Therefore, the first position in the file is numbered 0, not 1.

If *FilePosition* is greater than the number of bytes in the currently open file, the file pointer is set to the position following the last character in the file.

If no file is currently open when {SETPOS} is encountered, macro execution begins with the next command in the same cell as the {SETPOS} command. If the {SETPOS} is successful, the rest of that cell's commands are ignored, and execution continues on the row below.

{ONERROR} can be used to trap errors such as the disk drive not being ready.

See also {ONERROR} and {GETPOS}. For an example using {SETPOS}, see {READ}.

{STEPOFF}

Format: {STEPOFF}

{STEPOFF} tells Quattro to exit Single-Step mode, in which the macro is executed in slow-motion for debugging. The macro is then executed at the normal pace.

See {STEPON} for more information.

{STEPON}

Format: {STEPON}

{STEPON} causes Quattro to enter Single-Step mode, in which the macro is executed one step at a time for degbugging purposes. When Quattro encounters a {STEPON} command, it pauses and waits for the user to press the space bar. When the space bar is pressed, the commands in the next cell of the macro are executed and the macro pauses again. Single-Step mode continues until a {STEPOFF} command is encountered or until the macro ends.

{STEPON} is similar to using the DEBUG key (Shift-F8) to enter Single-Step mode, except that it can be embedded within a macro. The DEBUG key must be used before a macro begins executing. See "Debugging a Macro" in Chapter 12 of the *Quattro User's Guide* for more information.

Example

The following example uses the {STEPON} and {STEPOFF} commands to debug the last routine of a macro. Notice that the empty pairs of brackets, placed at the beginning and end of the routine, make it easy to add {STEPON} and {STEPOFF} commands when there is a problem.

```
Cleanup     {; Save spreadsheet to disk and say goodbye}
            { }{STEPON}
            {File,Save}~~
            {GOTO}quit_msg~
quit_msg    { }{STEPOFF}
            {QUIT}
```

{SubRoutine}

Format: {*SubRoutine ArgumentList*}

SubRoutine = the name of the subroutine being called (cannot be a cell address)

ArgumentList = a list of one or more arguments to be passed to the specified subroutine (optional)

{*SubRoutine*} calls the subroutine of the specified name, passing along any arguments given in *ArgumentList*. (Those arguments are then defined within the subroutine with the {DEFINE} command—see {DEFINE} for more information.)

Any macro can be called as a subroutine. The macro statements in the subroutine are executed until either {RETURN} or a blank cell is encountered. Then Quattro returns control to the calling routine, continuing execution with the macro statement in the cell below where it left off. If the macro subroutine ends with {QUIT}, both it and the calling macro(s) will be terminated.

Note: If a subroutine is given the same name as a macro command, issuing the command will execute the subroutine, not the macro command.

Subroutines can be "nested" (they can call more subroutines) up to 15 levels.

Macros can call themselves as subroutines (recursion), but this is not recommended, as serious complications can arise from an improperly designed recursive scheme.

The /x command equivalent to {Subroutine} is /xc (see page 208).

Example

The following example invokes a subroutine that forces the user to verify printing twice before it begins.

```
print       {GETLABEL "Do you want to print this (Y/N)?",
            ans_cell}
            {IF @UPPER(ans_cell)="Y"}{CHK_TWICE}
              .
              .
              .
Chk_twice   {; Double-check the print request}
            {GETLABEL "Are you certain (Y/N)?",ans_cell}
            {IF @UPPER(ans_cell)="N"} {RETURN}
            {GETLABEL "Ready to print now (Y/N)?",ans_cell}
            {IF @UPPER(ans_cell)="Y"} \PG
            {RETURN}
ans_cell    Y (or N)
```

{WAIT}

Format: {WAIT *DateTimeNumber*}

> *DateTimeNumber* = the date and time at which macro execution can resume

{WAIT} forces Quattro to pause all macro execution until the time corresponding to *DateTimeNumber* arrives. *DateTimeNumber* is a standard date/time serial number (both date and time portions must be included. If the current date or time is already later than *DateTimeNumber*, the macro terminates immediately.

While execution is suspended, the macro is inactive, a WAIT indicator flashes, and normal spreadsheet operation can only be restored by pressing *Ctrl-Break*.

This command is not the most efficient means for checking that a period of time has elapsed, since no processing of any kind can take place while {WAIT} is executing.

Examples

The following macro beeps, displays a "Go home" message, then suspends all execution until 8:00 AM tomorrow.

```
{BEEP 3}
Time to go home!{DOWN}
{WAIT @TODAY+1+@TIMEVALUE("8:00 AM")}
```

The next macro waits until this time tomorrow.

```
{WAIT @NOW+1}
```

This macro uses {WAIT} to alert the user by sounding 5 short beeps, spaced two seconds apart.

```
\h          {FOR counter,1,5,1,loopHere}

loopHere    {BEEP 3}
            {WAIT @NOW+@TIME(0,0,2)}

counter     6
```

{WINDOWSOFF}

Format: {WINDOWSOFF}

{WINDOWSOFF} disables normal screen updating during execution of a macro. It can significantly speed up execution time for most macros, since it spares Quattro the time normally needed to redraw the screen each time a cell is changed. Its effect is automatically cancelled by Quattro once the macro stops running, so you needn't worry about having the user "locked out" of the screen. To cancel its effect within the same macro, use the {WINDOWSON} command.

Use {WINDOWSOFF} along with {PANELOFF} to completely disable normal screen updating.

Example

The following macro uses {WINDOWSOFF} and {WINDOWSON} to turn off screen updating while a list of vendors is sorted, thereby speeding up the sort operation.

```
\w          {GOTO}sort_message~
            {WINDOWSOFF}
            {SORT vendor_name}
            {WINDOWSON}

sort        {define vendor_nm}
            {/ Sort,Block}

vendor_nm
            ~
            {/ Block,Copy}vendor_nm~key_nm~
```

```
                {/ Sort,Key1}
key_nm
                ~a
                {/ Sort,Go}
                sort_message    SORT IS IN PROGRESS
                Database:
                General Cement Co.
                Alveoli Mfg., Inc.
                Sandab Development
                Consolidated Dust
```

{WINDOWSON}

Format: {WINDOWSON}

{WINDOWSON} re-enables normal screen updating during execution of a macro, canceling the effects of a previous {WINDOWSOFF}. However, the screen will not be updated until a {Calc} key-equivalent command is encountered, or until the macro ends. If {WINDOWSON} is called when screen updating is already in effect (the normal mode), the command is ignored.

See {WINDOWSOFF} for an example using {WINDOWSON}.

{WRITE}

Format: {WRITE *String*}

String = the string of characters to be written into the open file

{WRITE} copies the characters of *String* to the currently open file, starting at the location of the file pointer. The file pointer is advanced to the position following the last character written, and the file is expanded if necessary.

{WRITE} does not automatically place the carriage-return and line-feed characters at the end of lines. To include these characters, use the {WRITELN} command.

String can be any quoted string, block name or cell address referring to a label, or an expression resulting in a string.

If there is no file open, or if there is insufficient room on the disk to expand the file as required, the {WRITE} command will fail. Macro execution will then continue with the first command after {WRITE} (in the same cell). If the {WRITE} is successful, the rest of that cell's commands are ignored, and execution continues on the next row below.

{ONERROR} can be used to trap errors such as the disk drive not being ready.

After you've finished accessing a particular file with {WRITE}, it is imperative that the file be closed with {CLOSE}. Otherwise, there is a good chance the the file will become corrupted if the computer is turned off or otherwise interrupted.

See also {WRITELN}.

Example

The following example creates a text file with fixed-length fields that will serve as a phone list database. After the macro is executed, the file will look like this:

```
Golden, David 415-921-7774; Hack, Edmund 201-133-3511; Hall, Sue Ann 617-234-5678
```

See the description of the {READ} command to see how to read a file like this.

```
\k      {OPEN "A:PHONEDIR.PRN",W}
        {WRITE "Golden, David "}
        {WRITE "415-921-7774;"}
        {WRITE "Hack, Edmund "}
        {WRITE "201-133-3511;"}
        {WRITE "Hall, Sue Ann "}
        {WRITE "617-234-5678"}
        {CLOSE}
```

{WRITELN}

Format: {WRITELN *String*}

String = the string of characters to be written into the open file as a single line

{WRITELN} copies the characters of *String* to the currently open file, starting at the location of the file pointer, and ends the string with the

carriage-return and line-feed characters. The file pointer is advanced to the position following the last character written, and the file is expanded if necessary.

To enter a blank line in the file, use the command {WRITELN ""}. To enter characters without carriage-return/line-feed characters, use the {WRITE} command.

String can be any quoted string, block name or cell address referring to a label, or an expression resulting in a string.

If there is no file open, or if there is insufficient room on the disk to expand the file as required, the {WRITELN} command will fail. Macro execution will then continue with the first command after {WRITELN} (in the same cell). If the {WRITELN} is successful, the rest of that cell's commands are ignored, and execution continues on the next row below.

{ONERROR} can be used to trap errors such as the disk drive not being ready.

After you've finished accessing a particular file with {WRITELN}, it is imperative that the file be closed with {CLOSE}. Otherwise, there is a good chance the the file will become corrupted if the computer is turned off or otherwise interrupted.

See also {WRITE}.

Example

The following example shows how {WRITELN} is used to write a line of text to the file TEXT.TXT. This line is terminated by the CR (carriage-return) and LF (line-feed) characters.

```
\z          {OPEN "A:TEST.TST",W}
            {WRITELN "This is a short line."}
            {CLOSE}
```

C H A P T E R

4

Menu-Equivalent Commands

When you record a macro in Quattro, the keys you use to select a command from a menu are translated to what are called *menu-equivalent commands*. These commands represent the menu commands selected and are displayed within the recorded macro instead of the actual keystrokes pressed.

Menu-equivalent commands have two purposes. They make a macro that uses menu commands much easier to read, and therefore edit. They also make it possible to use the macro with different Quattro menu structures. For the second reason especially, it is best to use menu-equivalent commands even when typing in a macro as a label. The extra keystrokes they require will be well worth it if you end up altering your menus with the Quattro menu builder, or loading a different menu structure.

Menu-equivalent commands are also used by the Transcript program add-in. Transcript records all your actions in Quattro, like one giant macro, in a file.

Menu-equivalent commands break down all of Quattro's menu commands into two groups. This first, the general action, describes the basic category of the command, such as Block or Format. The second group, the specific action, pinpoints exactly what the command does. For example, {/ MenuColors Frame} affects the color of the menu frame and correlates to the **D**efault **C**olors **S**preadsheet **F**rame command (*/DCSF*).

The rest of this chapter is devoted to three tables that list the menu-equivalent commands in different arrangements.

- **Table 4.1** shows the Quattro menu tree and lists the menu-equivalent commands for each item. This is most helpful when you're writing

macros and need to look up the menu-equivalent command for the menu item you want to use.

- **Table 4.2** shows the Lotus 1-2-3-compatible menu tree and the menu-equivalent commands for each item. Use this table if you have reconfigured Quattro with the Lotus 1-2-3-compatible menu tree.
- **Table 4.3** lists each of the menu-equivalent commands alphabetically. This is useful when you see the command (in a macro or transcript), and want to know what it means. It lists the Lotus 1-2-3-compatible commands and examples showing syntax.

Table 4.1: Menu Equivalents for Quattro Commands

Quattro Command	Menu Equivalent
BLOCK	
Copy	{/ Block,Copy}
Move	{/ Block,Move}
Erase	{/ Block,Erase}
Display Format	{/ Block,Format}
Label Align	{/ Block,Align}
Fill	{/ Math,Fill}
Search/Replace	
Block	{/ Audit,ReplaceRange}
Search	{/ Audit,SearchString}
Replace	{/ Audit,ReplaceString}
Go	{/ Audit,Replace}
Quit	
Reformat	{/ Block,Justify}
Advanced	
Values	{/ Block,Values}
Transpose	{/ Block,Transpose}
Protect	{/ Block,Protect}
Unprotect	{/ Block,Unprotect}
Block Names	
Create	{/ Name,Create}
Delete	{/ Name,Delete}
Labels	
Right	{/ Name,RightCreate}
Down	{/ Name,UnderCreate}
Left	{/ Name,LeftCreate}
Up	{/ Name,AboveCreate}
Reset	{/ Name,Reset}
Make Table	{/ Name,Table}
COLUMNS	
Insert	{/ Column,Insert}
Delete	{/ Column,Delete}
Width	{/ Column,Width}
Reset	{/ Column,Reset}
Hide	{/ Column,Hide}
Expose	{/ Column,Display}
Set Global	{/ Defaults,ColWidth}
Quit	
ROW	
Insert	{/ Row,Insert}
Delete	{/ Row,Delete}

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
ERASE	{/ Basics,Erase}
FILE	
Retrieve	{/ File,Retrieve}
Save	{/ File,Save}
Combine	
Copy	
File	{/ File,CopyFile}
Block	{/ File,CopyRange}
Quit	
Add	
File	{/ File,AddFile}
Block	{/ File,AddRange}
Quit	
Subtract	
File	{/ File,SubtractFile}
Block	{/ File,SubtractRange}
Quit	
Quit	
Import	
ASCII Text File	{/ File,ImportText}
Comma & "" Delimited file	{/ File,ImportNumbers}
Xtract	
Formulas	{/ File,ExtractFormulas}
Values	{/ File,ExtractValues}
Erase	{/ File,Erase}
Directory	{/ File,Directory}
Parse	
Create	{/ Parse,CreateLine}
Edit	{/ Parse,EditLine}
Input	{/ Parse,Input}
Output	{/ Parse,Output}
Go	{/ Parse,Go}
Reset	{/ Parse,Reset}
Quit	
OS	{/ Basics,OS}
GRAPH	
Graph Type	{/ Graph,Type}
X Axis Values	{/ XAxis,Labels}
Series Values	
1st Series	{/ 1Series,Block}
2nd Series	{/ 2Series,Block}
3rd Series	{/ 3Series,Block}

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
4th Series	{/ 4Series,Block}
5th Series	{/ 5Series,Block}
6th Series	{/ 6Series,Block}
View	{/ Graph,View}
Quit	
Titles	
1st Line	{/ Graph,MainTitle}
2nd Line	{/ Graph,XYSize}
X	{/ XAxis,Title}
Y	{/ YAxis,Title}
Size	
1st Line	{/ Graph,MainFontSize}
2nd Line	{/ Graph,Line2Size}
X and Y	{/ Graph,XYSize}
View	{/ Graph,View}
Quit	
Font	{/ GraphPrint,Fonts}
View	{/ Graph,View}
Quit	
Customize	
Series	
Markers	
1st Series	{/ 1Series,Markers}
2nd Series	{/ 2Series,Markers}
3rd Series	{/ 3Series,Markers}
4th Series	{/ 4Series,Markers}
5th Series	{/ 5Series,Markers}
6th Series	{/ 6Series,Markers}
Update	{/ Graph,UpdateMarkers}
View	{/ Graph,View}
Quit	
Patterns	
1st Series	{/ 1Series,Pattern}
2nd Series	{/ 2Series,Pattern}
3rd Series	{/ 3Series,Pattern}
4th Series	{/ 4Series,Pattern}
5th Series	{/ 5Series,Pattern}
6th Series	{/ 6Series,Pattern}
Update	{/ Graph,UpdatePatterns}
View	{/ Graph,View}
Quit	

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
Legends	
On/Off	{/ Graph,Legend}
1st Series	{/ 1Series,Legend}
2nd Series	{/ 2Series,Legend}
3rd Series	{/ 3Series,Legend}
4th Series	{/ 4Series,Legend}
5th Series	{/ 5Series,Legend}
6th Series	{/ 6Series,Legend}
View	{/ Graph,View}
Quit	
Interior Labels	
1st Series	{/ CompGraph,ALabels}
2nd Series	{/ CompGraph,BLabels}
3rd Series	{/ CompGraph,CLabels}
4th Series	{/ CompGraph,DLabels}
5th Series	{/ CompGraph,ELabels}
6th Series	{/ CompGraph,FLabels}
View	{/ Graph,View}
Quit	
Override Type	
1st Series	{/ 1Series,Type}
2nd Series	{/ 2Series,Type}
3rd Series	{/ 3Series,Type}
4th Series	{/ 4Series,Type}
5th Series	{/ 5Series,Type}
6th Series	{/ 6Series,Type}
View	{/ Graph,View}
Quit	
1st Series	
Block	{/ 1Series,Block}
Legend	{/ 1Series,Legend}
Interior Labels	{/ 1Series,Labels}
Placement of Labels	{/ 1Series,LabelsLoc}
Color	{/ 1Series,Color}
Override Type	{/ 1Series,Type}
Markers	{/ 1Series,Markers}
Fill Pattern	{/ 1Series,Pattern}
Reset	{/ Graph,Reset1}
View	{/ Graph,View}
Quit	
2nd Series	
Block	{/ 2Series,Block}
Legend	{/ 2Series,Legend}
Interior Labels	{/ 2Series,Labels}
Placement of Labels	{/ 2Series,LabelsLoc}

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
Color	{/ 2Series,Color}
Override Type	{/ 2Series,Type}
Markers	{/ 2Series,Markers}
Fill Pattern	{/ 2Series,Pattern}
Reset	{/ Graph,Reset2}
View	{/ Graph,View}
Quit	
3rd Series	
Block	{/ 3Series,Block}
Legend	{/ 3Series,Legend}
Interior Labels	{/ 3Series,Labels}
Placement of Labels	{/ 3Series,LabelsLoc}
Color	{/ 3Series,Color}
Override Type	{/ 3Series,Type}
Markers	{/ 3Series,Markers}
Fill Pattern	{/ 3Series,Pattern}
Reset	{/ Graph,Reset3}
View	{/ Graph,View}
Quit	
4th Series	
Block	{/ 4Series,Block}
Legend	{/ 4Series,Legend}
Interior Labels	{/ 4Series,Labels}
Placement of Labels	{/ 4Series,LabelsLoc}
Color	{/ 4Series,Color}
Override Type	{/ 4Series,Type}
Markers	{/ 4Series,Markers}
Fill Pattern	{/ 4Series,Pattern}
Reset	{/ Graph,Reset4}
View	{/ Graph,View}
Quit	
5th Series	
Block	{/ 5Series,Block}
Legend	{/ 5Series,Legend}
Interior Labels	{/ 5Series,Labels}
Placement of Labels	{/ 5Series,LabelsLoc}
Color	{/ 5Series,Color}
Override Type	{/ 5Series,Type}
Markers	{/ 5Series,Markers}
Fill Pattern	{/ 5Series,Pattern}
Reset	{/ Graph,Reset5}
View	{/ Graph,View}
Quit	

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
6th Series	
Block	{/ 6Series,Block}
Legend	{/ 6Series,Legend}
Interior Labels	{/ 6Series,Labels}
Placement of Labels	{/ 6Series,LabelsLoc}
Color	{/ 6Series,Color}
Override Type	{/ 6Series,Type}
Markers	{/ 6Series,Markers}
Fill Pattern	{/ 6Series,Pattern}
Reset	{/ Graph,Reset6}
View	{/ Graph,View}
Quit	
View	{/ Graph,View}
Quit	
X-Axis	
Title	{/ XAxis,Title}
X Values	{/ XAxis,Labels}
Scale	{/ XAxis,ScaleMode}
Display Scaling	{/ XAxis,ShowScale}
Lower	{/ XAxis,Min}
Upper	{/ XAxis,Max}
Increment	{/ XAxis,Step}
Format of Ticks	{/ XAxis,Format}
No. of Minor Ticks	{/ XAxis,Skip}
Alternate Ticks	{/ XAxis,Alternate}
View	{/ Graph,View}
Reset	{/ XAxis,Reset}
Update	{/ XAxis,Update}
Quit	
Y-Axis	
Title	{/ YAxis,Title}
Scale	{/ YAxis,ScaleMode}
Lower	{/ YAxis,Min}
Upper	{/ YAxis,Max}
Increment	{/ YAxis,Step}
Format of Ticks	{/ YAxis,Format}
No of Minor Ticks	{/ YAxis,Skip}
Display scaling	{/ YAxis,ShowScale}
View	{/ Graph,View}
Reset	{/ YAxis,Reset}
Update	{/ YAxis,Update}
Quit	
Colors	
Series	
1st Series	{/ 1Series,Color}

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
2nd Series	{/ 2Series,Color}
3rd Series	{/ 3Series,Color}
4th Series	{/ 4Series,Color}
5th Series	{/ 5Series,Color}
6th Series	{/ 6Series,Color}
View	{/ Graph,View}
Quit	
Titles	
1st Line	{/ Graph,MainColor}
2nd Line	{/ Graph,SubColor}
X Title	{/ XAxis,Color}
Y Title	{/ YAxis,TitleColor}
View	{/ Graph,View}
Quit	
Frame	{/ Graph,FrameColor}
Grid	{/ Graph,GridColor}
Background	{/ Graph,BackColor}
Update	{/ Graph,UpdateColors}
View	{/ Graph,View}
Quit	
Pies	
Label Format	{/ Pie,ValueFormat}
Exploded	
1st Slice	{/ PieExploded,1}
2nd Slice	{/ PieExploded,2}
3rd Slice	{/ PieExploded,3}
4th Slice	{/ PieExploded,4}
5th Slice	{/ PieExploded,5}
6th Slice	{/ PieExploded,6}
7th Slice	{/ PieExploded,7}
8th Slice	{/ PieExploded,8}
9th Slice	{/ PieExploded,9}
View	{/ Graph,View}
Quit	
Pattern	
1st Slice	{/ PiePattern,1}
2nd Slice	{/ PiePattern,2}
3rd Slice	{/ PiePattern,3}
4th Slice	{/ PiePattern,4}
5th Slice	{/ PiePattern,5}
6th Slice	{/ PiePattern,6}
7th Slice	{/ PiePattern,7}
8th Slice	{/ PiePattern,8}
9th Slice	{/ PiePattern,9}
View	{/ Graph,View}
Quit	

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
Colors	
1st slice	{/ PieColor,1}
2nd slice	{/ PieColor,2}
3rd slice	{/ PieColor,3}
4th slice	{/ PieColor,4}
5th slice	{/ PieColor,5}
6th slice	{/ PieColor,6}
7th slice	{/ PieColor,7}
8th slice	{/ PieColor,8}
9th slice	{/ PieColor,9}
View	{/ Graph,View}
Quit	
View	{/ Graph,View}
Quit	
Resolution	{/ Graph,ScreenMode}
Grids	
Color	{/ Graph,GridColor}
Line Type	{/ Graph,GridType}
Frame Graph	{/ Graph,FrameGraph}
View	{/ Graph,View}
Quit	
View	{/Graph,View}
Quit	
Reset	
Graph	{/ Graph,ResetAll}
1st Series	{/ Graph,Reset1}
2nd Series	{/ Graph,Reset2}
3rd Series	{/ Graph,Reset3}
4th Series	{/ Graph,Reset4}
5th Series	{/ Graph,Reset5}
6th Series	{/ Graph,Reset6}
X-Axis	{/ XAxis,Reset}
Name	
Display	{/ Graph,NameUse}
Store	{/ Graph,NameCreate}
Erase	{/ Graph,NameDelete}
Reset	{/ Graph,NameReset}
Print	
Printers	
1st Printer	
Type of printer	{/ GPrinter1,Type}
Make	{/ GPrinter1,ShowMake}
Model	{/ GPrinter1,ShowModel}
Mode	{/ GPrinter1,ShowMode}

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
Device	{/ GPrinter1,Device}
Baud rate	{/ GPrinter1,Baud}
Parity	{/ GPrinter1,Parity}
Stop bits	{/ GPrinter1,Stop}
Quit	
2nd Printer	
Type of printer	{/ GPrinter2,Type}
Make	{/ GPrinter2,ShowMake}
Model	{/ GPrinter2,ShowModel}
Mode	{/ GPrinter2,ShowMode}
Device	{/ GPrinter2,Device}
Baud rate	{/ GPrinter2,Baud}
Parity	{/ GPrinter2,Parity}
Stop bits	{/ GPrinter2,Stop}
Quit	
Plotter Speed	{/ GraphPrint,PlotSpeed}
Quit	
Destination	{/ GraphPrint,ShowDest}
Layout	
Measurement	{/ GraphPrint,Dimensions}
Left Edge	{/ GraphPrint,Left}
Top Edge	{/ GraphPrint,Top}
Height	{/ GraphPrint,Height}
Width	{/ GraphPrint,Width}
Orientation	{/ GraphPrint,Rotated}
Break Page	{/ GraphPrint,FFMode}
Quit	
Colors	
Copy the Screen's	{/ GPrintColors,UseGraphs}
Pie	
1st Slice	{/ GPrintPieColors,1}
2nd Slice	{/ GPrintPieColors,2}
3rd Slice	{/ GPrintPieColors,3}
4th Slice	{/ GPrintPieColors,4}
5th Slice	{/ GPrintPieColors,5}
6th Slice	{/ GPrintPieColors,6}
7th Slice	{/ GPrintPieColors,7}
8th Slice	{/ GPrintPieColors,8}
9th Slice	{/ GPrintPieColors,9}
Quit	

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
Series	
1st Series	{/ GPrintColors,1Series}
2nd Series	{/ GPrintColors,2Series}
3rd Series	{/ GPrintColors,3Series}
4th Series	{/ GPrintColors,4Series}
5th Series	{/ GPrintColors,5Series}
6th Series	{/ GPrintColors,6Series}
Quit	
Titles	
1st Line	{/ GPrintColors,Title}
2nd Line	{/ GPrintColors,SubTitle}
X Title	{/ GPrintColors,XTitle}
Y Title	{/ GPrintColors,YTitle}
Frame	{/ GPrintColors,Frame}
Grid	{/ GPrintColors,Grid}
Background	{/ GPrintColors,Background}
Quit	
Go	{/ GraphPrint,Print}
Write EPS/PIC	
EPS File	{/ GraphFile,PostScript}
PIC File	{/ GraphFile,PIC}
Quit	
Reset	{/ GraphPrint,Reset}
Update	{/ GraphPrint,Update}
Quit	
View	{/ Graph,View}
Quit	
MACRO	
Macro Actions	
Name	{/ Name,Create}
Delete	{/ Name,Delete}
Execute	{/ Name,Execute}
Clear Breakpoints	{/ Dbg,Reset}
QuattroAdd-Ins	
Load	{/ Addins,Load}
Run	{/ Addins,Run}
Unload	{/ Addins,Unload}
PRINT	
Block	{/ Print,Block}
Left Heading	{/ Print,LeftBorder}
Top Heading	{/ Print,TopBorder}
Destination	
Printer	{/ Print,OutputPrinter}
File	{/ Print,OutputFile}

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
Page Layout	
Header	{/ Print,Header}
Footer	{/ Print,Footer}
Break Pages	{/ Print,Breaks}
Margins & Length	
Page Length	{/ Print,PageLength}
Left	{/ Print,LeftMargin}
Top	{/ Print,TopMargin}
Right	{/ Print,RightMargin}
Bottom	{/ Print,BottomMargin}
Quit	
Create Break	{/ Print,CreatePageBreak}
Setup String	{/ Print,Setup}
Update	{/ Print,Update}
Quit	
Format	{/ Print,Format}
Adjust Printer	
Skip Line	{/ Print,SkipLine}
Form Feed	{/ Print,FormFeed}
Align	{/ Print,Align}
Reset	
All	{/ Print,ResetAll}
Print Block	{/ Print,ResetBlock}
Borders	{/ Print,ResetBorders}
Layout	{/ Print,ResetDefaults}
Go	{/ Print,Go}
Quit	
LAYOUT	
Titles	
Horizontal	{/ Titles,Horizontal}
Vertical	{/ Titles,Vertical}
Both	{/ Titles,Both}
Clear	{/ Titles,Clear}
Windows	
Horizontal	{/ Windows,Horizontal}
Vertical	{/ Windows,Vertical}
Sync	{/ Windows,Synch}
Unsync	{/ Windows,Unsynch}
Clear	{/ Windows,Clear}
Descriptor Line	{/ Startup,MoveStatus}
Quit	

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
DEFAULT	
Hardware	
Screen	
Text Screen Options	
Color	{/ ScreenHardware,Color}
IBM Monochrome	{/ ScreenHardware,Mono}
Screen Snows	{/ ScreenHardware,Retrace}
Use Special Driver	{/ ScreenHardware,UseSpecialText}
Driver Name	{/ ScreenHardware,TextDriver}
Graph Screen Options	
Graphic Screen	{/ ScreenHardware,GraphScreenType}
Resolution	{/ Graph,ScreenMode}
Aspect Ratio	{/ ScreenHardware,AspectRatio}
Quit	
Text Printer	
Device	{/ Hardware,Device}
Baud	{/ Hardware,Baud}
Stop Bits	{/ Hardware,StopBits}
Parity	{/ Hardware,Parity}
Auto LF	{/ Hardware,AutoLf}
1 Sheet	{/ Hardware,SingleSheet}
Quit	
Normal Memory	{/ Basics,ShowMem}
EMS	{/ Basics,ShowEMS}
Coprocessor	{/ Basics,ShowCoProc}
Quit	
Colors	
Menu	
Frame	{/ MenuColors,Frame}
Banner	{/ MenuColors,Banner}
Text	{/ MenuColors,Text}
Key Letter	{/ MenuColors,First Letter}
Highlight	{/ MenuColors,Menu Bar}
Settings	{/ MenuColors,Settings}
Explanation	{/ MenuColors,Explanation}
Drop Shadow	{/ Startup,Shadow}
Quit	
SpreadSheet	
Borders	{/ Color,Frame}
Cells	{/ Color,Cells}
Titles	{/ Color,Titles}
Highlight	{/ Color,Cursor}
Descriptor Line	{/ Color,Status}

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
Status Line	{/ Color,Indicators}
Input Line	{/ Color,Edit}
Unprotected	{/ Color,Unprotect}
Labels	{/ ValueColors,Labels}
Quit	
Conditional	
On/Off	{/ ValueColors,Enable}
ERR	{/ ValueColors,Err}
Smallest Normal Value	{/ ValueColors,Min}
Greatest Normal Value	{/ ValueColors,Max}
Below Normal Color	{/ ValueColors,Low}
Normal Cell Color	{/ ValueColors,Normal}
Above Normal Color	{/ ValueColors,High}
Quit	
Palettes	
Color	{/ Color,ColorPalette}
Monochrome	{/ Color,BWPalette}
Quit	
Help	
Banner	{/ HelpColors,Banner}
Text	{/ HelpColors,Text}
Keywords	{/ HelpColors,Keyword}
Highlight	{/ HelpColors,Highlight}
Quit	
Quit	
International	
Currency	{/ Intl,Currency}
Punctuation	{/ Intnl,Punctuation}
Date	{/ FormatChanges,IntlDate}
Time	{/ FormatChanges,IntnlTime}
Quit	
Formats	
Display	{/ Defaults,Format}
Align	{/ Defaults,Alignment}
Hide Zeroes	{/ Defaults,Zero}
Width of Col	{/ Defaults,ColWidth}
Clock	{/ Defaults,ClockFormat}
Quit	
Directories	
Resource	{/ Defaults,ResourceDirectory}
Data	{/ Defaults,Directory}
Quit	

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
Recalculation	
Mode	{/ Defaults,RecalcMode}
Order	{/ Defaults,RecalcOrder}
Iteration	{/ Defaults,RecalcIteration}
Circular Cell	{/ Audit,ShowCirc}
Quit	
Protection	
Enable	{/ Protection,Enable}
Disable	{/ Protection,Disable}
Startup	
Autoload File	{/ Startup,File}
Startup Macro	{/ Startup,Macro}
Extension	{/ Startup,Extension}
Beep	{/ Startup,Beep}
Help Access Method	{/ Defaults,HelpAccess}
Default Add-Ins	
1	{/ DefaultAddins,1}
2	{/ DefaultAddins,2}
3	{/ DefaultAddins,3}
4	{/ DefaultAddins,4}
5	{/ DefaultAddins,5}
6	{/ DefaultAddins,6}
7	{/ DefaultAddins,7}
8	{/ DefaultAddins,8}
Reset	
1	{/ DefaultAddins,Reset1}
2	{/ DefaultAddins,Reset2}
3	{/ DefaultAddins,Reset3}
4	{/ DefaultAddins,Reset4}
5	{/ DefaultAddins,Reset5}
6	{/ DefaultAddins,Reset6}
7	{/ DefaultAddins,Reset7}
8	{/ DefaultAddins,Reset8}
All	{/ DefaultAddins,All}
Quit	
Quit	
Menu Tree	
Main Menus	{/ Startup,Menus}
Alternate Menus	{/ Startup,AltMenus}
Reset Alternate	{/ Startup,ResetAlt}
Switch Menus	
Main Menu	{/ Startup,UseMain}
Alternate Menu	{/ Startup,UseAlt}
Quit	

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
Compatibility Options	
Menus Options	
Keep Wide	{/ Startup,WideMenus}
Remember	{/ Startup,Remember}
Confirm Options	
Borland Style	{/ Startup,BorlandConfirm}
During Macros	{/ Startup,ConfirminMacro}
Macro Recording	{/ Startup,Record}
Quit	
Quit	
Update	{/ Defaults,Update}
Quit	
ADVANCED	
Database	
Sort	
Block	{/ Sort,Block}
1st Key	{/ Sort,Key1}
2nd Key	{/ Sort,Key2}
3rd Key	{/ Sort,Key3}
4th Key	{/ Sort,Key4}
5th Key	{/ Sort,Key5}
Go	{/ Sort,Go}
Reset	{/ Sort,Reset}
Sort Order	
Numbers before Labels	{/ Startup,CellOrder}
Label Order	{/ Startup,LabelOrder}
Quit	
Quit	
Query	
Block	{/ Query,Block}
Assign Names	{/ Query,AssignNames}
Formula Criterion	{/ Query,FormulaCriteria}
Criteria Table	{/ Query,CriteriaBlock}
Locate	{/ Query,Locate}
Extract	{/ Query,Extract}
Unique	{/ Query,Unique}
Output Block	{/ Query,Output}
Delete	{/ Query,Delete}
Reset	{/ Query,Reset}
Quit	
Form Input	{/ Block,Input}
Quit	

Table 4.1: Menu Equivalents for Quattro Commands, Continued

Quattro Command	Menu Equivalent
What-If	
1 Variable	{/ Math,1 Cell What-If}
2 Variables	{/ Math,2 Cell What-If}
Reset	{/ Math,Reset What-If}
Quit	
Regression	
Independent	{/ Regression,Independent}
Dependent	{/ Regression,Dependent}
Output	{/ Regression,Output}
Y Intercept	{/ Regression,Intercept}
Go	{/ Regression,Go}
Reset	{/ Regression,Reset}
Quit	
Frequency	{/ Math,Distribution}
Matrix	
Invert	{/ Math,Invert Matrix}
Multiply	{/ Math,Multiply Matrix}
Quit	
Quit	
QUIT	{/ Basics,Quit}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands

Lotus Command	Menu Equivalent
WORKSHEET	
Global	
Format	{/ Defaults,Format}
Label-Prefix	{/ Defaults,Alignment}
Column-Width	{/ Defaults,ColWidth}
Recalculation	
Natural	{/ CompCalc,Natural}
Columnwise	{/ CompCalc,ColWise}
Rowwise	{/ CompCalc,RowWise}
Automatic	{/ CompCalc,Automatic}
Manual	{/ CompCalc,Manual}
Iteration	{/ Defaults,RecalcIteration}
Protection	
Enable	{/ Protection,Enable}
Disable	{/ Protection,Disable}
Default	
Printer	
Interface	{/ Hardware,LotusDevice}
Auto-LF	{/ Hardware,AutoLf}
Left	{/ Print,LeftMargin}
Right	{/ Print,RightMargin}
Top	{/ Print,TopMargin}
Bottom	{/ Print,BottomMargin}
Page-Length	{/ Print,PageLength}
Wait	{/ Hardware,SingleSheet}
Setup	{/ Print,Setup}
Name	
Quit	
Directory	{/ Defaults,Directory}
Status	
Normal Memory	{/ Basics,ShowMem}
Expanded memory	{/ Basics,ShowEMS}
Co-Processor	{/ Basics,ShowCoProc}
ShowCirc	{/ Audit,ShowCirc}
Quit	
Update	{/ Defaults,Update}
Other	
International	
Punctuation	{/ Intnl,Punctuation}
Currency	{/ Intnl,Currency}
Date	{/ FormatChanges,IntnlDate}
Time	{/ FormatChanges,IntnlTime}
Quit	

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Help	{/ Defaults,HelpAccess}
Clock	{/ Defaults,ClockFormat}
Resource Directory	{/ Defaults,ResourceDirectory}
Quit	
Zero	{/ Defaults,Zero}
Insert	
Column	{/ Column,Insert}
Row	{/ Row,Insert}
Delete	
Column	{/ Column,Delete}
Row	{/ Row,Delete}
Column	
Set-Width	{/ Column,Width}
Reset-Width	{/ Column,Reset}
Hide	{/ Column,Hide}
Display	{/ Column,Display}
Erase	{/ Basics,Erase}
Titles	
Both	{/ Titles,Both}
Horizontal	{/ Titles,Horizontal}
Vertical	{/ Titles,Vertical}
Clear	{/ Titles,Clear}
Window	
Horizontal	{/ Windows,Horizontal}
Vertical	{/ Windows,Vertical}
Sync	{/ Windows,Synch}
Unsync	{/ Windows,Unsynch}
Clear	{/ Windows,Clear}
Status	
Normal Memory	{/ Basics,ShowMem}
Expanded Memory	{/ Basics,ShowEMS}
Co-Processor	{/ Basics,ShowCoProc}
Quit	
Page	{/ Print,CreatePageBreak}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
RANGE	
Format	{/ Block,Format}
Label	{/ Block,Align}
Erase	{/ Block,Erase}
Name	
Create	{/ Name,Create}
Delete	{/ Name,Delete}
Labels	
Right	{/ Name,RightCreate}
Down	{/ Name,UnderCreate}
Left	{/ Name,LeftCreate}
Up	{/ Name,AboveCreate}
Reset	{/ Name,Reset}
Table	{/ Name,Table}
Justify	{/ Block,Justify}
Protect	{/ Block,Protect}
Unprotect	{/ Block,Unprotect}
Input	{/ Block,Input}
Value	{/ Block,Values}
Transpose	{/ Block,Transpose}
Search/Replace	
Range	{/ Audit,ReplaceRange}
Search String	{/ Audit,SearchString}
Replace String	{/ Audit,ReplaceString}
Go	{/ Audit,Replace}
Quit	
COPY	{/ Block,Copy}
MOVE	{/ Block,Move}
FILE	
Retrieve	{/ File,Retrieve}
Save	{/ File,Save}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Combine	
Copy	
Entire-File	{/ File,CopyFile}
Named-Range	{/ File,CopyRange}
Add	
Entire-File	{/ File,AddFile}
Named-Range	{/ File,AddRange}
Subtract	
Entire-File	{/ File,SubtractFile}
Named-Range	{/ File,SubtractRange}
Xtract	
Formulas	{/ File,ExtractFormulas}
Values	{/ File,ExtractValues}
Erase	{/ File,Erase}
List	{/ File,List}
Import	
Text	{/ File,ImportText}
Numbers	{/ File,ImportNumbers}
Directory	{/ File,Directory}
PRINT	
Printer	{/ Print,OutputPrinter}
File	{/ Print,OutputFile}
GRAPH	
Type	{/ Graph,Type}
X	{/ XAxis,Labels}
A	{/ 1Series,Block}
B	{/ 2Series,Block}
C	{/ 3Series,Block}
D	{/ 4Series,Block}
E	{/ 5Series,Block}
F	{/ 6Series,Block}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Reset	
Graph	{/ Graph,ResetAll}
X	{/ XAxis,Reset}
A	{/ Graph,Reset1}
B	{/ Graph,Reset2}
C	{/ Graph,Reset3}
D	{/ Graph,Reset4}
E	{/ Graph,Reset5}
F	{/ Graph,Reset6}
Quit	
View	{/ Graph,View}
Save	{/ GraphFile,PIC}
Options	
Legend	
A	{/ 1Series,Legend}
B	{/ 2Series,Legend}
C	{/ 3Series,Legend}
D	{/ 4Series,Legend}
E	{/ 5Series,Legend}
F	{/ 6Series,Legend}
Format	
Graph	{/ CompGraph,GraphFormat}
A	{/ CompGraph,AFormat}
B	{/ CompGraph,Bformat}
C	{/ CompGraph,Cformat}
D	{/ CompGraph,DFormat}
E	{/ CompGraph,Eformat}
F	{/ CompGraph,Fformat}
Quit	
Titles	
First	{/ Graph,MainTitle}
Second	{/ Graph,SubTitle}
X-Axis	{/ XAxis,Title}
Y-Axis	{/ YAxis,Title}
Main Font	{/ GraphPrint,Fonts}
Character Size	
First Title	{/ Graph,MainFontSize}
Second Title	{/ Graph,Line2Size}
X and Y	{/ Graph,XYSize}
Grid	{/ CompGraph,Grid}
Scale	
Y Scale	
Automatic	{/ CompGraph,YAuto}
Manual	{/ CompGraph,YManual}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Lower	{/ YAxis,Min}
Upper	{/ YAxis,Max}
Format	{/ YAxis,Format}
Indicator	{/ YAxis,ShowScale}
Quit	
X Scale	
Automatic	{/ CompGraph,XAuto}
Manual	{/ CompGraph,XManual}
Lower	{/ XAxis,Min}
Upper	{/ XAxis,Max}
Format	{/ XAxis,Format}
Indicator	{/ XAxis,ShowScale}
Quit	
Skip	{/ XAxis,Skip}
Color	{/ Graph,MainColor}
B&W	
Data-Labels	
A	{/ CompGraph,ALabels}
B	{/ CompGraph,BLabels}
C	{/ CompGraph,CLabels}
D	{/ CompGraph,DLabels}
E	{/ CompGraph,ELabels}
F	{/ CompGraph,FLabels}
Quit	
Patterns	
A	{/ 1Series,Pattern}
B	{/ 2Series,Pattern}
C	{/ 3Series,Pattern}
D	{/ 4Series,Pattern}
E	{/ 5Series,Pattern}
F	{/ 6Series,Pattern}
Update	{/ Graph,UpdatePatterns}
View	{/ Graph,View}
Quit	
Markers	
A	{/ 1Series,Markers}
B	{/ 2Series,Markers}
C	{/ 3Series,Markers}
D	{/ 4Series,Markers}
E	{/ 5Series,Markers}
F	{/ 6Series,Markers}
Update	{/ Graph,UpdateMarkers}
View	{/ Graph,View}
Quit	
Quit	

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Name	
Use	{/ Graph,NameUse}
Create	{/ Graph,NameCreate}
Delete	{/ Graph,NameDelete}
Reset	{/ Graph,NameReset}
Graph Customize	
A	
Range	{/ 1Series,Block}
Legend	{/ 1Series,Legend}
Interior Labels	{/ 1Series,Labels}
Placement of Labels	{/ 1Series,LabelsLoc}
Color	{/ 1Series,Color}
Override Type	{/ 1Series,Type}
Markers	{/ 1Series,Markers}
Fill Pattern	{/ 1Series,Pattern}
Reset	{/ Graph,Reset1}
View	{/ Graph,View}
Quit	
B	
Range	{/ 2Series,Block}
Legend	{/ 2Series,Legend}
Interior Labels	{/ 2Series,Labels}
Placement of Labels	{/ 2Series,LabelsLoc}
Color	{/ 2Series,Color}
Override Type	{/ 2Series,Type}
Markers	{/ 2Series,Markers}
Fill Pattern	{/ 2Series,Pattern}
Reset	{/ Graph,Reset2}
View	{/ Graph,View}
Quit	
C	
Range	{/ 3Series,Block}
Legend	{/ 3Series,Legend}
Interior Labels	{/ 3Series,Labels}
Placement of Labels	{/ 3Series,LabelsLoc}
Color	{/ 3Series,Color}
Override Type	{/ 3Series,Type}
Markers	{/ 3Series,Markers}
Fill Pattern	{/ 3Series,Pattern}
Reset	{/ Graph,Reset3}
View	{/ Graph,View}
Quit	
D	
Range	{/ 4Series,Block}
Legend	{/ 4Series,Legend}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Interior Labels	{/ 4Series,Labels}
Placement of Labels	{/ 4Series,LabelsLoc}
Color	{/ 4Series,Color}
Override Type	{/ 4Series,Type}
Markers	{/ 4Series,Markers}
Fill Pattern	{/ 4Series,Pattern}
Reset	{/ Graph,Reset4}
View	{/ Graph,View}
Quit	
E	
Range	{/ 5Series,Block}
Legend	
Interior Labels	{/ 5Series,Labels}
Placement of Labels	{/ 5Series,LabelsLoc}
Color	{/ 5Series,Color}
Override Type	{/ 5Series,Type}
Markers	{/ 5Series,Markers}
Fill Pattern	{/ 5Series,Pattern}
Reset	{/ Graph,Reset5}
View	{/ Graph,View}
Quit	
F	
Range	{/ 6Series,Block}
Legend	{/ 6Series,Legend}
Interior Labels	{/ 6Series,Labels}
Placement of Labels	{/ 6Series,LabelsLoc}
Color	{/ 6Series,Color}
Override Type	{/ 6Series,Type}
Markers	{/ 6Series,Markers}
Fill Pattern	{/ 6Series,Pattern}
Reset	{/ Graph,Reset6}
View	{/ Graph,View}
Quit	
X-Axis	
Title	{/ XAxis,Title}
X Values	{/ XAxis,Labels}
Scale	{/ XAxis,ScaleMode}
Lower	{/ XAxis,Min}
Upper	{/ XAxis,Max}
Increment	{/ XAxis,Step}
Format of Tics	{/ XAxis,Format}
No. of Minor Tics	{/ XAxis,Step}
Alternate Tics	{/ XAxis,Alternate}
View	{/ Graph,View}
Reset	{/ XAxis,Reset}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Update	{/ XAxis,Update}
Quit	
Y-Axis	
Title	{/ YAxis,Title}
Scale	{/ YAxis,ScaleMode}
Lower	{/ YAxis,Min}
Upper	{/ YAxis,Max}
Increment	{/ YAxis,Step}
Format of Tics	{/ YAxis,Format}
No. of Minor Tics	{/ YAxis,Step}
Alternate Tics	{/ YAxis,Update}
View	{/ Graph,View}
Reset	{/ YAxis,Reset}
Update	{/ YAxis,Update}
Quit	
Colors	
Series	
A	{/ 1Series,Color}
B	{/ 2Series,Color}
C	{/ 3Series,Color}
D	{/ 4Series,Color}
E	{/ 5Series,Color}
F	{/ 6Series,Color}
View	{/ Graph,View}
Quit	
Titles	
First	{/ Graph,MainColor}
Second	{/ Graph,SubColor}
X	{/ XAxis,TitleColor}
Y	{/ YAxis,TitleColor}
View	{/ Graph,View}
Quit	
Frame	{/ Graph,FrameColor}
Grid	{/ Graph,GridColor}
Background	{/ Graph,BackColor}
Update	{/ Graph,UpdateColors}
View	{/ Graph,View}
Quit	
Pie	
Label Format	{/ Pie,ValueFormat}
Explode	
1st Slice	{/ PieExploded,1}
2nd Slice	{/ PieExploded,2}
3rd Slice	{/ PieExploded,3}
4th Slice	{/ PieExploded,4}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
5th Slice	{/ PieExploded,5}
6th Slice	{/ PieExploded,6}
7th Slice	{/ PieExploded,7}
8th Slice	{/ PieExploded,8}
9th Slice	{/ PieExploded,9}
View	{/ Graph,View}
Quit	
Patterns	
1st Slice	{/ PiePattern,1}
2nd Slice	{/ PiePattern,2}
3rd Slice	{/ PiePattern,3}
4th Slice	{/ PiePattern,4}
5th Slice	{/ PiePattern,5}
6th Slice	{/ PiePattern,6}
7th Slice	{/ PiePattern,7}
8th Slice	{/ PiePattern,8}
9th Slice	{/ PiePattern,9}
View	{/ Graph,View}
Quit	
Colors	
1st Slice	{/ PieColor,1}
2nd Slice	{/ PieColor,2}
3rd Slice	{/ PieColor,3}
4th Slice	{/ PieColor,4}
5th Slice	{/ PieColor,5}
6th Slice	{/ PieColor,6}
7th Slice	{/ PieColor,7}
8th Slice	{/ PieColor,8}
9th Slice	{/ PieColor,9}
View	{/ Graph,View}
Quit	
View	{/ Graph,View}
Quit	
Resolution	{/ Graph,ScreenMode}
Grid	
Color	{/ Graph,GridColor}
Line Type	{/ Graph,GridType}
Frame Graph	{/ Graph,FrameGraph}
View	{/ Graph,View}
Quit	
View	{/ Graph,View}
Quit	

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Print	
Printers	
1st Printer	
Type of Printer	{/ GPrinter1,Type}
Make	{/ GPrinter1,ShowMake}
Model	{/ GPrinter1,ShowModel}
Mode	{/ GPrinter1,ShowMode}
Device	{/ GPrinter1,Device}
Baud Rate	{/ GPrinter1,Baud}
Parity	{/ GPrinter1,Parity}
Stop Bits	{/ GPrinter1,Stop}
Quit	
2nd Printer	
Type of Printer	{/ GPrinter2,Type}
Make	{/ GPrinter2,ShowMake}
Model	{/ GPrinter2,ShowModel}
Mode	{/ GPrinter2,ShowMode}
Device	{/ GPrinter2,Device}
Baud Rate	{/ GPrinter2,Baud}
Parity	{/ GPrinter2,Parity}
Stop Bits	{/ GPrinter2,Stop}
Quit	
Plotter Speed	{/ GraphPrint,PlotSpeed}
Quit	
Destination	{/ GraphPrint,ShowDest}
Layout	
Measurement	{/ GraphPrint,Dimensions}
Left Edge	{/ GraphPrint,Left}
Top Edge	{/ GraphPrint,Top}
Height	{/ GraphPrint,Height}
Width	{/ GraphPrint,Width}
Rotated	{/ GraphPrint,Rotated}
Break Page	{/ GraphPrint,FFMode}
Quit	
Colors	
Copy the Screen's	{/ GPrintColors,UseGraphs}
Pie Color	
1st Slice	{/ GPrintPieColors,1}
2nd Slice	{/ GPrintPieColors,2}
3rd Slice	{/ GPrintPieColors,3}
4th Slice	{/ GPrintPieColors,4}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
5th Slice	{/ GPrintPieColors,5}
6th Slice	{/ GPrintPieColors,6}
7th Slice	{/ GPrintPieColors,7}
8th Slice	{/ GPrintPieColors,8}
9th Slice	{/ GPrintPieColors,9}
Quit	
Series	
A	{/ GPrintColors,1Series}
B	{/ GPrintColors,2Series}
C	{/ GPrintColors,3Series}
D	{/ GPrintColors,4Series}
E	{/ GPrintColors,5Series}
F	{/ GPrintColors,6Series}
Quit	
Titles	
First	{/ GPrintColors,Title}
Second	{/ GPrintColors,SubTitle}
X	{/ GPrintColors,XTitle}
Y	{/ GPrintColors,YTitle}
Quit	
Frame	{/ GPrintColors,Frame}
Grid	{/ GPrintColors,Grid}
Background	{/ GPrintColors,Background}
Quit	
Go	{/ GraphPrint,Print}
Write EPS File	{/ GraphFile,Postscript}
Reset	{/ GraphPrint,Reset}
Update	{/ GraphPrint,Update}
Quit	
Quit	
DATA	
Fill	{/ Math,Fill}
Table	
1	{/ Math,1 Cell What-If}
2	{/ Math,2 Cell What-If}
Reset	{/ Math,Reset What-If}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Sort	
Data-Range	{/ Sort,Block}
Primary-Key	{/ CompSort,Key1}
Secondary-Key	{/ CompSort,Key2}
Reset	{/ Sort,Reset}
Go	{/ Sort,Go}
3	{/ CompSort,Key3}
4	{/ CompSort,Key4}
5	{/ CompSort,Key5}
Quit	
Query	
Input	{/ Query,Block}
Criterion	{/ Query,CriteriaBlock}
Output	{/ Query,Output}
Find	{/ Query,Locate}
Extract	{/ Query,Extract}
Unique	{/ Query,Unique}
Delete	{/ Query,Delete}
Reset	{/ Query,Reset}
Assign Names	{/ Query,AssignNames}
Logical Criteria	{/ Query,FormulaCriterion}
Quit	
Distribution	{/ Math,Distribution}
Matrix	
Invert	{/ Math,InvertMatrix}
Multiply	{/ Math,MultiplyMatrix}
Regression	
X-Range	{/ Regression,Independent}
Y-Range	{/ Regression,Dependent}
Output-Range	{/ Regression,Output}
Intercept	{/ Regression,Intercept}
Reset	{/ Regression,Reset}
Go	{/ Regression,Go}
Quit	
Parse	
Format-Line	
Create	{/ Parse,CreateLine}
Edit	{/ Parse,EditLine}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Input-Column	{/ Parse,Input}
Output-Range	{/ Parse,Output}
Reset	{/ Parse,Reset}
Go	{/ Parse,Go}
Quit	
SYSTEM	{/ Basics,OS}
Add-In/Macro	
Macro Actions	
Name	{/ Name,Create}
Delete	{/ Name,Delete}
Execute	{/ Name,Execute}
Clear Breakpoints	{/ Name,BkptReset}
Quattro Add-Ins	
Load	{/ Addins,Load}
Run	{/ Addins,Run}
Unload	{/ Addins,Unload}
Install	
Hardware	
Text Screen Options	
Color	{/ ScreenHardware,Color}
IBM Monochrome	{/ ScreenHardware,Mono}
Screen Snows	{/ ScreenHardware,Retrace}
Use Special Driver	{/ ScreenHardware,UseSpecialText}
Driver Name	{/ ScreenHardware,TextDriver}
Graph Screen Options	
Graphic Screen	{/ ScreenHardware,GraphScreenType}
Resolution	{/ Graph,ScreenMode}
Aspect Ratio	{/ ScreenHardware,AspectRatio}
Quit	
Colors	
Menu	
Frame	{/ MenuColors,Frame}
Banner	{/ MenuColors,Banner}
Text	{/ MenuColors,Text}
Key Letter	{/ MenuColors,First Letter}
Highlight	{/ MenuColors,Menu Bar}
Settings	{/ MenuColors,Settings}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
Explanation	{/ MenuColors,Explanation}
Drop Shadow	{/ Startup,Shadow}
Quit	
Spreadsheet	
Borders	{/ Color,Frame}
Cells	{/ Color,Cells}
Titles	{/ Color,Titles}
Highlight	{/ Color,Cursor}
Descriptor Line	{/ Color,Status}
Status Line	{/ Color,Indicators}
Input Line	{/ Color,Edit}
Unprotected	{/ Color,Unprotect}
Labels	{/ ValueColors,Labels}
Quit	
Conditional	
On/Off	{/ ValueColors,Enable}
ERR	{/ ValueColors,Err}
Smallest Normal Value	{/ ValueColors,Min}
Greatest Normal Value	{/ ValueColors,Max}
Below Normal Color	{/ ValueColors,Low}
Normal Cell Color	{/ ValueColors,Normal}
Above Normal Color	{/ ValueColors,High}
Quit	
Palettes	
Color	{/ Color,ColorPalette}
Monochrome	{/ Color,BWPalette}
Quit	
Help	
Banner	{/ HelpColors,Banner}
Text	{/ HelpColors,Text}
Keyword	{/ HelpColors,Keyword}
Highlight	{/ HelpColors,Highlight}
Quit	
Quit	
Startup	
Autoload File	{/ Startup,File}
Startup Macro	{/ Startup,Macro}
Extension	{/ Startup,Extension}
Beep	{/ Startup,Beep}
Default Add-Ins	
1	{/ DefaultAddins,1}
2	{/ DefaultAddins,2}
3	{/ DefaultAddins,3}

Table 4.2: Menu Equivalents for the Lotus 1-2-3 Commands, Continued

Lotus Command	Menu Equivalent
4	{/ DefaultAddins,4}
5	{/ DefaultAddins,5}
6	{/ DefaultAddins,6}
7	{/ DefaultAddins,7}
8	{/ DefaultAddins,8}
Reset	
1	{/ DefaultAddins,Reset1}
2	{/ DefaultAddins,Reset2}
3	{/ DefaultAddins,Reset3}
4	{/ DefaultAddins,Reset4}
5	{/ DefaultAddins,Reset5}
6	{/ DefaultAddins,Reset6}
7	{/ DefaultAddins,Reset7}
8	{/ DefaultAddins,Reset8}
All	{/ DefaultAddins,ResetAll}
Quit	
Quit	
Menu Tree	
Main Menus	{/ Startup,Menus}
Alternate Menus	{/ Startup,AltMenus}
Reset Alternate	{/ Startup,ResetAlt}
Switch Menus	
Main Menu	{/ Startup,UseMain}
Alternate Menu	{/ Startup,UseAlt}
Quit	
Quit	
Toggle Descriptor Line	{/ Startup,MoveStatus}
Compatibility Options	
Menu Options	
Keep Wide	{/ Startup,WideMenus}
Remember	{/ Startup,Remember}
Confirm Options	
Borland Style	{/ Startup,BorlandConfirm}
During Macros	{/ Startup,ConfirminMacro}
Macro Recording	{/ Startup,Record}
Quit	
Quit	
Update	{/ Defaults,Update}
QUIT	{/ Basics,Quit}

Table 4.3: Menu Equivalents and Associated Keystrokes

Menu Equivalent	Quattro Menu	Lotus Menu	Example
1Series[1]			
Block	/GS1	/GA	{/ 1Series,Block}F67..H67~
Pattern	/GCS1F	/GOPA[2]	{/ 1Series,Pattern}j
Color	/GCS1C	/GGCSA[2]	{/ 1Series,Color}
Type	/GCS1T	/GGAO	{/ 1Series,Type}c
Legend	/GCS1L	/GOLA	{/ 1Series,Legend}sales~
Labels	/GCS1I	/GODA	{/ 1Series,Labels}F67..H67~
LabelsLoc	/GCS1I	/GGAP	{/ 1Series,LabelsLoc}a
Markers	/GCS1M	/GOMA[2]	{/ 1Series,Markers}a
2Series[1]			
Block	/GS2	/GB	{/ 2Series,Block}F67..H67~
Pattern	/GCS2F	/GOPB[2]	{/ 2Series,Pattern}a
Color	/GCS2C	/GGCSB[2]	{/ 2Series,Color}
Type	/GCS2T	/GGBO	{/ 2Series,Type}n
Legend	/GCS2L	/GOLB	{/ 2Series,Legend}sales~
Labels	/GCS2N	/GODB	{/ 2Series,Labels}F67..H67~
LabelsLoc	/GCS2P	/GGBP	{/ 2Series,LabelsLoc}a
Markers	/GCS2M	/GOMB[2]	{/ 2Series,Markers}h
3Series[1]			
Block	/GS3	/GC	{/ 3Series,Block}F67..H67~
Pattern	/GCS3F	/GOPC[2]	{/ 3Series,Pattern}a
Color	/GCS3C	/GGCSC[2]	{/ 3Series,Color}
Type	/GCS3T	/GGCO	{/ 3Series,Type}m
Legend	/GCS3L	/GOLC	{/ 3Series,Legend}sales~
Labels	/GCS3N	/GODC	{/ 3Series,Labels}F67..H67~
LabelsLoc	/GCS3P	/GGCP	{/ 3Series,LabelsLoc}a
Markers	/GCS3M	/GOMC[2]	{/ 3Series,Markers}m
4Series[1]			
Block	/GS4	/GD	{/ 4Series,Block}F67..H67~
Pattern	/GCS4F	/GOPD[2]	{/ 4Series,Pattern}e
Color	/GCS4C	/GGCSD[2]	{/ 4Series,Color}
Type	/GCS4T	/GGDO	{/ 4Series,Type}b
Legend	/GCS4L	/GOLD	{/ 4Series,Legend}sales~
Labels	/GCS4N	/GODD	{/ 4Series,Labels}F67..H67~
LabelsLoc	/GCS4P	/GGDP	{/ 4Series,LabelsLoc}a
Markers	/GCS4M	/GOMD[2]	{/ 4Series,Markers}a

1. See also Graph, XAxis, YAxis, 1Series, 2Series, 3Series, 4Series, 5Series, 6Series, PieColor, PieExploded, PiePattern, GPrintColors, GPrintPieColors, CompGraph, GPrinter1, GPrinter2, and GraphFile.
2. Not in Lotus 1-2-3.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
5Series[1]			
Block	/GS5	/GE	{/ 5Series,Block}F67..H67~
Pattern	/GCS5F	/GOPE[2]	{/ 5Series,Pattern}h
Color	/GCS5C	/GGCSE[2]	{/ 5Series,Color}
Type	/GCS5T	/GGEO	{/ 5Series,Type}c
Legend	/GCS5L	/GOLE	{/ 5Series,Legend}sales~
Labels	/GCS5N	/GODE	{/ 5Series,Labels}F67..H67~
LabelsLoc	/GCS5P	/GGEP	{/ 5Series,LabelsLoc}a
Markers	/GCS5M	/GOME[2]	{/ 5Series,Markers}f
6Series[1]			
Block	/GS6	/GF	{/ 6Series,Block}F67..H67~
Pattern	/GCS6F	/GOPF[2]	{/ 6Series,Pattern}d
Color	/GCS6C	/GGCSF[2]	{/ 6Series,Color}
Type	/GCS6T	/GGFO	{/ 6Series,Type}n
Legend	/GCS6L	/GOLF	{/ 6Series,Legend}sales~
Labels	/GCS6N	/GODF	{/ 6Series,Labels}F67..H67~
LabelsLoc	/GCS6P	/GGFP	{/ 6Series,LabelsLoc}a
Markers	/GCS6M	/GOMF[2]	{/ 6Series,Markers}l
AddIns			
Load	/ML	/EL[2]	{/ AddIns,Load}~
Run	/MR	/ER[2]	{/ AddIns,Run}1
Unload	/MU	/EU[2]	{/ AddIns,Unload}1
Audit			
Replace	/BSG	/RSG	{/ Audit,Replace}
SearchString	/BSS	/RSS	{/ Audit,SearchString}3~
ReplaceString	/BSR	/RSN	{/ Audit,ReplaceString}5~
ReplaceRange	/BSB	/RSR	{/ Audit,ReplaceRange}A1..B3~
ShowCirc	/DR	/WS	{/ Audit,ShowCirc}
Basics			
Erase	/E	/WE	{/ Basics,Erase}
OS	/FO	/S	{/ Basics,OS}
Quit	/Q	/Q	{/ Basics,Quit}
ShowMem	/DH	/WS	{/ Basics,ShowMem}
ShowEMS	/DH	/WS	{/ Basics,ShowEMS}
ShowCoProc	/DH	/WS	{/ Basics,ShowCoProc}

1. See also Graph, XAxis, YAxis, 1Series, 2Series, 3Series, 4Series, 5Series, 6Series, PieColor, PieExploded, PiePattern, GPrintColors, GPrintPieColors, CompGraph, GPrinter1, GPrinter2, and GraphFile.
2. Not in Lotus 1-2-3.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
Block			
Copy	/BC	/C	{/ Block,Copy}A1..B3~C4..D6~
Move	/BM	/M	{/ Block,Move}A1..B3~C4..D6~
Erase	/BE	/RE	{/ Block,Erase}A1..B3~
Reformat	/BD	/RF	{/ Block,Format}s3~A1..B3~
Align	/BL	/RL	{/ Block,Align}lA1..B3~
Transpose	/BAT	/RT	{/ Block,Transpose}A1..B3~C4..D6~
Values	/BAV	/RV	{/ Block,Values}A1..B3~C4..D6~
Input	/ADF	/RI	{/ Block,Input}A1..B3~
Protect	/BAP	/RP	{/ Block,Protect}A1..B3~
Unprotect	/BAU	/RU	{/ Block,Unprotect}A1..B3~
Justify	/BR	/RJ	{/ Block,Justify}A1..B3~
Color[3]			
Frame	/DCSB	/ICSB	{/ Color,Frame}
Edit	/DCSI	/ICSI	{/ Color,Edit}
Cells	/DCSC	/ICSC	{/ Color,Cells}
Cursor	/DCSH	/ICSH	{/ Color,Cursor}
Unprotect	/DCSU	/ICSU	{/ Color,Unprotect}
Indicators	/DCSS	/ICSS	{/ Color,Indicators}
Titles	/DCST	/ICST	{/ Color,Titles}
Status	/DCSD	/ICSD	{/ Color,Status}
Column			
Hide	/CH	/WCH	{/ Column,Hide}aa1~
Display	/CE	/WCD	{/ Column,Display}aa1~
Reset	/CR	/WCR	{/ Column,Reset}
Width	/CW	/WCS	{/ Column,Width}12~
Delete	/CD	/WDC	{/ Column,Delete}i1~
Insert	/CI	/WIC	{/ Column,Insert}i1~
CompCalc			
Natural	N/A	/WGRN	{/ CompCalc,Natural}
ColWise	N/A	/WGRC	{/ CompCalc,ColWise}
RowWise	N/A	/WGRR	{/ CompCalc,RowWise}
Automatic	N/A	/WGRA	{/ CompCalc,Automatic}
Manual	N/A	/WGRM	{/ CompCalc,Manual}
CompGraph			
XManual	N/A	/GOSXM	{/ CompGraph,XManual}
XAuto	N/A	/GOSXA	{/ CompGraph,XAuto}
YManual	N/A	/GOSYM	{/ CompGraph,YManual}
YAuto	N/A	/GOSYA	{/ CompGraph,YAuto}
Grid	N/A	/GOG	{/ CompGraph,Grid}h~m

3 . See also ValueColors, MenuColors, and HelpColors.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
GraphFormat	N/A	/GOFG	{/ CompGraph,GraphFormat}1
AFormat	N/A	/GOFA	{/ CompGraph,AFormat}1
BFormat	N/A	/GOFB	{/ CompGraph,BFormat}1
CFormat	N/A	/GOFC	{/ CompGraph,CFormat}1
DFormat	N/A	/GOFD	{/ CompGraph,DFormat}1
EFormat	N/A	/GOFE	{/ CompGraph,EFormat}1
FFormat	N/A	/GOFF	{/ CompGraph,FFormat}1
ALabels	/GCSI1	/GOLA	{/ CompGraph,ALabels}A1..B3~a
BLabels	/GCSI2	/GOLB	{/ CompGraph,BLabels}A1..B3~a
CLabels	/GCSI3	/GOLC	{/ CompGraph,CLabels}A1..B3~a
DLabels	/GCSI4	/GOLD	{/ CompGraph,DLabels}A1..B3~a
ELabels	/GCSI5	/GOLE	{/ CompGraph,ELabels}A1..B3~a
FLabels	/GCSI6	/GOLF	{/ CompGraph,FLabels}A1..B3~a
CompPrint			
AsDisplayed	N/A	/PPOOA	{/ CompPrint,AsDisplayed}
Formulas	N/A	/PPOOC	{/ CompPrint,Formulas}
Breaks	N/A	/PPOOF	{/ CompPrint,Breaks}
NoBreaks	N/A	/PPOOU	{/ CompPrint,NoBreaks}
CompSort			
Key1	N/A	/DSP	{/ CompSort,Key1}A1~D~
Key2	N/A	/DSS	{/ CompSort,Key2}B1~D~
Key3	N/A	/DS3[2]	{/ CompSort,Key3}C1~D~
Key4	N/A	/DS4[2]	{/ CompSort,Key4}D1~D~
Key5	N/A	/DS5[2]	{/ CompSort,Key5}E1~D~
Default[4]			
HelpAccess	/DSH	/WGDOH	{/ Defaults,HelpAccess}r
Directory	/DDD	/WGDD	{/ Defaults,Directory}C:\QUATTRO~
ClockFormat	/DFC	/WGDOC	{/ Defaults,ClockFormat}n
Format	/DFD	/WGF	{/ Defaults,Format}c0~
Alignment	/DFA	/WGA	{/ Defaults,Alignment}c
ColWidth	/DFW	/WGC	{/ Defaults,ColWidth}12~
Zero	/DFH	/WGZ	{/ Defaults,Zero}n
Update	/DU	/WGDU	{/ Defaults,Update}
RecalcOrder	/DRO	N/A	{/ Defaults,RecalcOrder}c
RecalcMode	/DRM	N/A	{/ Defaults,RecalcMode}a
RecalcIteration	/DRI	/WGRI	{/ Defaults,RecalcIteration}1~
ResourceDirectory	/DDR	/WGDR[2]	{/ Defaults,ResourceDirectory} C:\QUATTRO~

2. Not in Lotus 1-2-3.

4. See also Protection, Column, Row, Hardware, and Intnl.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
Default AddIns			
1	/DSD1	/ISD1	{/ DefaultAddIns,1}~
2	/DSD2	/ISD2	{/ DefaultAddIns,2}~
3	/DSD3	/ISD3	{/ DefaultAddIns,3}~
4	/DSD4	/ISD4	{/ DefaultAddIns,4}~
5	/DSD5	/ISD5	{/ DefaultAddIns,5}~
6	/DSD6	/ISD6	{/ DefaultAddIns,6}~
7	/DSD7	/ISD7	{/ DefaultAddIns,7}~
8	/DSD8	/ISD8	{/ DefaultAddIns,8}~
Reset1	/DSDR1	/ISDR1	{/ DefaultAddIns,Reset1}
Reset2	/DSDR2	/ISDR2	{/ DefaultAddIns,Reset2}
Reset3	/DSDR3	/ISDR3	{/ DefaultAddIns,Reset3}
Reset4	/DSDR4	/ISDR4	{/ DefaultAddIns,Reset4}
Reset5	/DSDR5	/ISDR5	{/ DefaultAddIns,Reset5}
Reset6	/DSDR6	/ISDR6	{/ DefaultAddIns,Reset6}
Reset7	/DSDR7	/ISDR7	{/ DefaultAddIns,Reset7}
Reset8	/DSDR8	/ISDR8	{/ DefaultAddIns,Reset8}
ResetAll	/DSDRA	/ISDRA	{/ DefaultAddIns,ResetAll}
File[5]			
Retrieve	/FR	/FR	{/ File,Retrieve}TEST.WK1~
Save	/FS	/FS	{/ File,Save}TEST.WK1~
ExtractFormulas	/FXF	/FXF	{/ File,ExtractFormulas}NEW~A1..B3~r
ExtractValues	/FXV	/FXV	{/ File,ExtractValues}NEW~A1..B3~r
Update	/DU	/WGDU	{/ File,Update}
Directory	/FD	/FD	{/ File,Directory}C:\QUATTRO~
ImportNumbers	/FIC	/FIN	{/ File,ImportNumbers}COMMA.PRN~
ImportText	/FIA	/FIT	{/ File,ImportText}ASCII.PRN~
AddFile	/FCAF	/FCAF	{/ File,AddFile}SALES.WKQ~
SubtractFile	/FCSF	/FCSF	{/ File,SubtractFile}SALES.WKQ~
CopyFile	/FCCF	/FCCF	{/ File,CopyFile}SALES.WKQ~
AddRange	/FCAB	/FCAB	{/ File,AddRange}A1..H8~SALES.WKQ~
SubtractRange	/FCSB	/FCSB	{/ File,SubtractRange}A1..H8~SALES.WKQ~
CopyRange	/FCCB	/FCCB	{/ File,CopyRange}A1..H8~SALES.WKQ~
Erase	/FE	/FE	{/ File,Erase}w SALES.WKQ~
List	N/A	/FL	{/ File,List}w
FormatChanges			
IntlDate	/DID	/WGDOID	{/ FormatChanges,IntlDate}d
IntnlTime	/DIT	/WGDOIT	{/ FormatChanges,IntlTime}d

5. See also Defaults Directory.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
GPrintColors[1]			
UseGraphs	/GPCC	/GPCC[2]	{/ GPrintColors,UseGraphs}
Title	/GPCT1	/GPCT1[2]	{/ GPrintColors,Title}b
SubTitle	/GPCT2	/GPCT2[2]	{/ GPrintColors,SubTitle}b
Background	/GPCB	/GPCB[2]	{/ GPrintColors,Background}b
Frame	/GPCF	/GPCF[2]	{/ GPrintColors,Frame}b
Grid	/GPCG	/GPCG[2]	{/ GPrintColors,Grid}b
XTitle	/GPCTX	/GPCTX[2]	{/ GPrintColors,XTitle}b
YTitle	/GPCTY	/GPCTY[2]	{/ GPrintColors,YTitle}b
1Series	/GPSC1	/GPCSA[2]	{/ GPrintColors,1Series}b
2Series	/GPSC2	/GPCSB[2]	{/ GPrintColors,2Series}b
3Series	/GPSC3	/GPCSC[2]	{/ GPrintColors,3Series}b
4Series	/GPSC4	/GPCSD[2]	{/ GPrintColors,4Series}b
5Series	/GPSC5	/GPCSE[2]	{/ GPrintColors,5Series}b
6Series	/GPSC6	/GPCSF[2]	{/ GPrintColors,6Series}b
GPrinter1[1]			
Device	/GPP1D	/GPP1D[2]	{/ GPrinter1,Device}2
Baud	/GPP1B	/GPP1B[2]	{/ GPrinter1,Baud}9
Stop	/GPP1S	/GPP1S[2]	{/ GPrinter1,Stop}1
Parity	/GPP1P	/GPP1P[2]	{/ GPrinter1,Parity}1
Type	/GPP1T	/GPP1T[2]	{/ GPrinter1,Type}ip~
ShowMake	/GPP1	/GPP1[2]	{/ GPrinter1,ShowMake}
ShowModel	/GPP1	/GPP1[2]	{/ GPrinter1,ShowModel}
ShowMode	/GPP1	/GPP1[2]	{/ GPrinter1,ShowMode}
GPrinter2[1]			
Device	/GPP2D	/GPP2D[2]	{/ GPrinter2,Device)2
Baud	/GPP2B	/GPP2B[2]	{/ GPrinter2,Baud}1
Stop	/GPP2S	/GPP2S[2]	{/ GPrinter2,Stop}1
Parity	/GPP2P	/GPP2P[2]	{/ GPrinter2,Parity}1
Type	/GPP2T	/GPP2T[2]	{/ GPrinter2,Type}ip~
ShowMake	/GPP2	/GPP2[2]	{/ GPrinter2,ShowMake}
ShowModel	/GPP2	/GPP2[2]	{/ GPrinter2,ShowModel}
ShowMode	/GPP2	/GPP2[2]	{/ GPrinter2,ShowMode}

1. See also Graph, XAxis, YAxis, 1Series, 2Series, 3Series, 4Series, 5Series, 6Series, PieColor, PieExploded, PiePattern, GPrintColors, GPrintPieColors, CompGraph, GPrinter1, GPrinter2, and GraphFile.
2. Not in Lotus 1-2-3.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
GPrintPieColors[1]			
1	/GPCP1	/GPCP1[2]	{/ GPrintPieColors,1}b
2	/GPCP2	/GPCP2[2]	{/ GPrintPieColors,2}b
3	/GPCP3	/GPCP3[2]	{/ GPrintPieColors,3}b
4	/GPCP4	/GPCP4[2]	{/ GPrintPieColors,4}b
5	/GPCP5	/GPCP5[2]	{/ GPrintPieColors,5}b
6	/GPCP6	/GPCP6[2]	{/ GPrintPieColors,6}b
7	/GPCP7	/GPCP7[2]	{/ GPrintPieColors,7}b
8	/GPCP8	/GPCP8[2]	{/ GPrintPieColors,8}b
9	/GPCP9	/GPCP9[2]	{/ GPrintPieColors,9}b
Graph[1]			
Type	/GG	/GT	{/ Graph,Type}1
MainTitle	/GT1	/GOTF	{/ Graph,MainTitle}Main Test~
SubTitle	/GT2	/GOTS	{/ Graph,SubTitle}Tester~
MainColor	/GCCT	/GGCT1[2]	{/ Graph,MainColor}b
SubColor	/GCCS	/GGCT2[2]	{/ Graph,SubColor}b
BackColor	/GCCB	/GGCB[2]	{/ Graph,BackColor}b
MainFontSize	/GTC1	/GOTCM[2]	{/ Graph,MainFontSize}a~
GridType	/GCGL	/GOGL[2]	{/ Graph,GridType}1
GridColor	/GCGC	/GGCG[2]	{/ Graph,GridColor}b
FrameColor	/GCCF	/GGCF[2]	{/ Graph,FrameColor}b
ResetAll	/GRG	/GRG	{/ Graph,ResetAll}
View	/GV	/GV	{/ Graph,View}
NameUse	/GND	/GNU	{/ Graph,NameUse}GRAPH~
NameCreate	/GNS	/GNC	{/ Graph,NameCreate}GRAPH~
NameDelete	/GNE	/GND	{/ Graph,NameDelete}GRAPH~
NameReset	/GNR	/GNR	{/ Graph,NameReset}
XYSize	/GTC2	/GOTCS[2]	{/ Graph,XYSize}a~
Legend	/GCSLE	N/A	{/ Graph,Legend}
ScreenMode	/GCR	/GGR	{/ Graph,ScreenMode}
Reset1	/GR1	/GRA	{/ Graph,Reset1}
Reset2	/GR2	/GRB	{/ Graph,Reset2}
Reset3	/GR3	/GRC	{/ Graph,Reset3}
Reset4	/GR4	/GRD	{/ Graph,Reset4}
Reset5	/GR5	/GRE	{/ Graph,Reset5}
Reset6	/GR6	/GRF	{/ Graph,Reset6}
UpdateColors	/GCCU	/GGCU	{/ Graph,UpdateColors}
UpdateMarkers	/GCSMU	/GOMU	{/ Graph,UpdateMarkers}
UpdatePatterns	/GCSPU	/GOPU	{/ Graph,UpdatePatterns}
FrameGraph	/GCGF	/GGGF	{/ Graph,FrameGraph}y
Line2Size	/GTS2	/GOTCS	{/ Graph,Line2Size}a~

1. See also Graph, XAxis, YAxis, 1Series, 2Series, 3Series, 4Series, 5Series, 6Series, PieColor, PieExploded, PiePattern, GPrintColors, GPrintPieColors, CompGraph, GPrinter1, GPrinter2, and GraphFile.
2. Not in Lotus 1-2-3.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
GraphFile[1]			
File	/GPDN	/GPDN[2]	{/ GraphFile,File}filename~
Type	/GPDT	/GPDT[2]	{/ GraphFile,Type}~~~
PIC	/GPWP	/GS	{/ GraphFile,PIC}junk~
Postscript	/GPWE	/GPW	{/GraphFile,Postscript}Q.EPS~
GraphPrint[1]			
Print	/GPG	/GPG[2]	{/ GraphPrint,Print}
Fonts	/GTF	/GOTT[2]	{/ GraphPrint,Fonts}a
Width	/GPLW	/GPLW[2]	{/ GraphPrint,Width}15~
Height	/GPLH	/GPLH[2]	{/ GraphPrint,Height}30~
Top	/GPLT	/GPLT[2]	{/ GraphPrint,Top}4~
Left	/GPLL	/GPLL[2]	{/ GraphPrint,Left}3~
Rotated	/GPLR	/GPLR[2]	{/ GraphPrint,Rotated}P~
Dimensions	/GPLM	/GPLM[2]	{/ GraphPrint,Dimensions}i
Update	/GPU	/GPU[2]	{/ GraphPrint,Update}
Reset	/GPR	/GPR[2]	{/ GraphPrint,Reset}
FFMode	/GPLB	/GPLB[2]	{/ GraphPrint,FFMode}y~
PlotSpeed	/GPPP	/GPPP[2]	{/ GraphPrint,PlotSpeed}1~
DestIsFile	/GPDF	/GPDF[2]	{/ GraphPrint,DestIsFile}
DestIsPtr1	/GPD1	/GPD1[2]	{/ GraphPrint,DestIsPtr1}
DestIsPtr2	/GPD2	/GPD2[2]	{/ GraphPrint,DestIsPtr2}
ShowDest	/GPD	/GPD[2]	{/ GraphPrint,ShowDest}
Hardware[6]			
Device	/DHTD	/WGDPI	{/ Hardware,Device}2
Baud	/DHTB	N/A	{/ Hardware,Baud}8
StopBits	/DHTS	N/A	{/ Hardware,StopBits}2
Parity	/DHTP	N/A	{/ Hardware,Parity}n
AutoLf	/DHTA	/WGDPA	{/ Hardware,AutoLf}y
SingleSheet	/DHT1	/WGDPW	{/ Hardware,SingleSheet}y
HelpColors			
Highlight	/DCHH	/ICHH	{/ HelpColors,Highlight}
Keyword	/DCHK	/ICHK	{/ HelpColors,Keyword}
Text	/DCHT	/ICHT	{/ HelpColors,Text}
Banner	/DCHB	/ICHB	{/ HelpColors,Banner}

1. See also Graph, XAxis, YAxis, 1Series, 2Series, 3Series, 4Series, 5Series, 6Series, PieColor, PieExploded, PiePattern, GPrintColors, GPrintPieColors, CompGraph, GPrinter1, GPrinter2, and GraphFile.

2. Not in Lotus 1-2-3.

6. See also Default, ScreenHardware, Startup, GPrinter1, and GPrinter2.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
Intnl			
Punctuation	/DIP	/WGDOIP	{/ Intnl,Punctuation}a~
Currency	/DIS	/WGDOIC	{/ Intnl,Currency}Ł~p~
CurrencyLocation	N/A	N/A	{/ Intnl,CurrencyLocation}p~
CurrencySymbol	N/A	N/A	{/ Intnl,CurrencySymbol}Ł~
Math[7]			
1 Cell What-If	/AW1	/DT1	{/ Math,1 Cell What-If}F67..H70~E67~
2 Cell What-If	/AW2	/DT2	{/ Math,2 Cell What-If}F67..H10~E67~F66~
Reset What-If	/AWR	/DTR	{/ Math,Reset What-If}
Invert Matrix	/AMI	/DMI	{/ Math,InvertMatrix}A1..C3~D1~
Multiply Matrix	/AMM	/DMM	{/ Math,MultiplyMatrix}A1..C3~D1..F3~G1~
Distribution	/AF	/DD	{/ Math,Distribution}B6..B17~D6..D9~
Fill	/BF	/DF	{/ Math,Fill}A1..B3~1~2~~
MenuColors			
Frame	/DCMF	/ICMF[2]	{/ MenuColors,Frame}
Banner	/DCMB	/ICMB[2]	{/ MenuColors,Banner}
First Letter	/DCMK	/ICMK[2]	{/ MenuColors,First Letter}
Text	/DCMT	/ICMT[2]	{/ MenuColors,Text}
Settings	/DCMS	/ICMS[2]	{/ MenuColors,Settings}
Menu Bar	/DCMH	/ICMH[2]	{/ MenuColors,Menu Bar}
Explanation	/DCME	/ICME[2]	{/ MenuColors,Explanation}
Name			
Create	/BAC	/RNC	{/ Name,Create}junk~A1..B3~
Delete	/BAD	/RND	{/ Name,Delete}junk~
Reset	/BAR	/RNR	{/ Name,Reset}
RightCreate	/BALR	/RNLR	{/ Name,RightCreate}A1..A3~
UnderCreate	/BALD	/RNLD	{/ Name,UnderCreate}A1..B1~
LeftCreate	/BALL	/RNLL	{/ Name,LeftCreate}B1..B3~
AboveCreate	/BALA	/RNLU	{/ Name,AboveCreate}C1..C3~
Execute	/ME	/AE	{/ Name,Execute}junk~
Table	/BAM	/RNT	{/ Name,Table}A1..B3~
BkptReset	/MC	/AC[2]	{/ Name,BkptReset}
Parse[7]			
EditLine	/FPE	/DPFE	{/ Parse,EditLine}{right}{right}L~
CreateLine	/FPC	/DPFC	{/ Parse,CreateLine}
Input	/FPI	/DPI	{/ Parse,Input}A1..C10~
Output	/FPO	/DPO	{/ Parse,Output}D1~
Go	/FPG	/DPG	{/ Parse,Go}
Reset	/FPR	/DPR	{/ Parse,Reset}

2. Not in Lotus 1-2-3.

7. See also Math, Sort, Regression, CompSort, Parse, Query, Startup CellOrder, and Startup LabelOrder.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
Pie[1]			
ValueFormat	/GCPL	/GGLP	{/ Pie,ValueFormat}v
PieColor[1]			
1	/GCPC1	/GGPC1[2]	{/ PieColor,1}b
2	/GCPC2	/GGPC2[2]	{/ PieColor,2}b
3	/GCPC3	/GGPC3[2]	{/ PieColor,3}b
4	/GCPC4	/GGPC4[2]	{/ PieColor,4}b
5	/GCPC5	/GGPC5[2]	{/ PieColor,5}b
6	/GCPC6	/GGPC6[2]	{/ PieColor,6}b
7	/GCPC7	/GGPC7[2]	{/ PieColor,7}b
8	/GCPC8	/GGPC8[2]	{/ PieColor,8}b
9	/GCPC9	/GGPC9[2]	{/ PieColor,9}b
PieExploded[1]			
1	/GCPE1	/GGPE1[2]	{/ PieExploded,1}d
2	/GCPE2	/GGPE2[2]	{/ PieExploded,2}e
3	/GCPE3	/GGPE3[2]	{/ PieExploded,3}e
4	/GCPE4	/GGPE4[2]	{/ PieExploded,4}d
5	/GCPE5	/GGPE5[2]	{/ PieExploded,5}e
6	/GCPE6	/GGPE6[2]	{/ PieExploded,6}d
7	/GCPE7	/GGPE7[2]	{/ PieExploded,7}e
8	/GCPE8	/GGPE8[2]	{/ PieExploded,8}d
9	/GCPE9	/GGPE9[2]	{/ PieExploded,9}e
PiePattern[1]			
1	/GCPP1	/GGPP1[2]	{/ PiePattern,1}j
2	/GCPP2	/GGPP2[2]	{/ PiePattern,2}a
3	/GCPP3	/GGPP3[2]	{/ PiePattern,3}h
4	/GCPP4	/GGPP4[2]	{/ PiePattern,4}j
5	/GCPP5	/GGPP5[2]	{/ PiePattern,5}a
6	/GCPP6	/GGPP6[2]	{/ PiePattern,6}g
7	/GCPP7	/GGPP7[2]	{/ PiePattern,7}f
8	/GCPP8	/GGPP8[2]	{/ PiePattern,8}i
9	/GCPP9	/GGPP9[2]	{/ PiePattern,9}j

1. See also Graph, XAxis, YAxis, 1Series, 2Series, 3Series, 4Series, 5Series, 6Series, PieColor, PieExploded, PiePattern, GPrintColors, GPrintPieColors, CompGraph, GPrinter1, GPrinter2, and GraphFile.
2. Not in Lotus 1-2-3.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
Print			
Block	/PB	/PPR	{/ Print,Block}A1..B3~
Header	/PPH	/PPOH	{/ Print,Header}head~
Footer	/PPF	/PPOF	{/ Print,Footer}foot~
LeftBorder	/PL	/PPOBC	{/ Print,LeftBorder}A1..A3~
TopBorder	/PT	/PPOBR	{/ Print,TopBorder}A1..C1~
LeftMargin	/PPML	/PPOML	{/ Print,LeftMargin}20~
RightMargin	/PPMR	/PPOMR	{/ Print,RightMargin}56~
TopMargin	/PPMT	/PPOMT	{/ Print,TopMargin}4~
BottomMargin	/PPMB	/PPOMB	{/ Print,BottomMargin}3~
PageLength	/PPMP	/PPOP	{/ Print,PageLength}20~
Setup	/PPS	/PPOS	{/ Print,Setup}\015~
Align	/PAA	/PPA	{/ Print,Align}
SkipLine	/PAS	/PPL	{/ Print,SkipLine}
FormFeed	/PAF	/PPP	{/ Print,FormFeed}
Go	/PG	/PPG	{/ Print,Go}
ResetAll	/PRA	/PPCA	{/ Print,ResetAll}
ResetBlock	/PRP	/PPCR	{/ Print,ResetBlock}
ResetBorders	/PRB	/PPCB	{/ Print,ResetBorders}
ResetDefaults	/PRD	/PPCF	{/ Print,ResetDefaults}
Format	/PF	N/A	{/ Print,Format}a
Breaks	/PPB	N/A	{/ Print,Breaks}n
Update	/PPU	/WGDU	{/ Print,Update}
OutputFile	/PDF	/PF	{/ Print,OutputFile}file name
OutputPrinter	/PDP	/PP	{/ Print,OutputPrinter}
CreatePageBreak	/PFC	/WP	{/ Print,CreatePageBreak}
Protection			
Disable	/DPD	/WGPD	{/ Protection,Disable}
Enable	/DPE	/WGPE	{/ Protection,Enable}
Query[7]			
Block	/ADQB	/DQI	{/ Query,Block}A1..B3~
CriteriaTable	/ADQC	/DQC	{/ Query,CriteriaTable}A6..B7~
Output	/ADQO	/DQO	{/ Query,Output}A10..C10~
FormulaCriterion	/ADQF	/DQL[2]	{/ Query,FormulaCriterion}salary>100~
Extract	/ADQE	/DQE	{/ Query,Extract}
Unique	/ADQU	/DGU	{/ Query,Unique}
Delete	/ADQD	/DQD	{/ Query,Delete}
Locate	/ADQL	/DLF	{/ Query,Locate}
Reset	/ADQR	/DQR	{/ Query,Reset}
AssignNames	/ADQA	/DQA[2]	{/ Query,AssignNames}

2. Not in Lotus 1-2-3.

7. See also Math, Sort, Regression, CompSort, Parse, Query, Startup CellOrder, and Startup LabelOrder.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
Regression[7]			
Independent	/ARI	/DRX	{/ Regression,Independent}C3..C8~
Dependent	/ARD	/DRY	{/ Regression,Dependent}D3..D8~
Output	/ARO	/DRO	{/ Regression,Output}E3~
Intercept	/ARY	/DRI	{/ Regression,Intercept}z
Go	/ARG	/DRG	{/ Regression,Go}
Reset	/ARR	/DRR	{/ Regression,Reset}
Row			
Delete	/RD	/WDR	{/ Row,Delete}b8000~
Insert	/RI	/WIR	{/ Row,Insert}b8000~
ScreenHardware			
Retrace	/DHSS	/IHS[2]	{/ ScreenHardware,Retrace}y
Mono	/DHSI	/IHI[2]	{/ ScreenHardware,Mono}y
Color	/DHSC	/IHC[2]	{/ ScreenHardware,Color}y
UseSpecialText	/DHSU	/IHU[2]	{/ ScreenHardware,UseSpecialText}y
TextDriver	/DHSD	/IHD[2]	{/ ScreenHardware,TextDriver}driver~
GraphScreenType	/DHSG	/IHG[2]	{/ ScreenHardware,GraphScreenType}a
AspectRatio	/DHSGA	/IHA[2]	{/ ScreenHardware,AspectRatio}
Sort[7]			
Block	/ADSB	/DSD	{/ Sort,Block}A1..E10~
Key1	/ADS1	N/A	{/ Sort,Key1}A1~D
Key2	/ADS2	N/A	{/ Sort,Key2}B1~D
Key3	/ADS3	N/A	{/ Sort,Key3}C1~D
Key4	/ADS4	N/A	{/ Sort,Key4}D1~D
Key5	/ADS5	N/A	{/ Sort,Key5}E1~D
Go	/ADSG	/DSG	{/ Sort,Go}
Reset	/ADSR	/DSR	{/ Sort,Reset}
Startup			
Extension	/DSE	/ISE[2]	{/ Startup,Extension}wk1~
Macro	/DSS	/ISS[2]	{/ Startup,Macro}\O
File	/DSA	/ISA[2]	{/ Startup,File}AUTO123.WK1~
Menus	/DSMM	/ISMM[2]	{/ Startup,Menus}QUATTRO.RSC~
AltMenus	/DSMA	/ISMA[2]	{/ Startup,AltMenus}123.ALT
WideMenus	/DSCK	/ISCK[2]	{/ Startup,WideMenus}y
Remember	/DSCR	/ISCR	{/ Startup,Remember}n
Beep	/DSB	/ISB	{/ Startup,Beep}y

2. Not in Lotus 1-2-3.

7. See also Math, Sort, Regression, CompSort, Parse, Query, Startup CellOrder, and Startup LabelOrder.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
Record	/DSCM	/ISCM	{/ Startup,Record}k
BorlandLoss	/DSCB	/ISCB	{/ Startup,BorlandLoss}y
ConfirminMacro	/DSCD	/ISCD	{/ Startup,ConfirminMacro}n
MoveStatus	/LD	/IST	{/ Startup,MoveStatus}
ResetAlt	/DSMR	/ISMR	{/ Startup,ResetAlt}
UseMain	/DSMSM	/ISMSM	{/ Startup,UseMain
UseAlt	/DSMSA	/ISMSA	{/ Startup,UseAlt}
Shadow	/DCMD	/ICMD	{/ Startup,Shadow}
CellOrder	/ADSSN	/DSCN	{/ Startup,CellOrder}N
LabelOrder	/ADSSL	/DSCL	{/ Startup,LabelOrder}A
Titles			
Both	/LTB	/WTB	{/ Titles,Both}
Horizontal	/LTH	/WTH	{/ Titles,Horizontal}
Vertical	/LTV	/WTV	{/ Titles,Vertical}
Clear	/LTC	/WTC	{/ Titles,Clear}
ValueColors			
Enable	/DCCO	/ICCO[2]	{/ ValueColors,Enable}e~
Err	/DCCE	/ICCE[2]	{/ ValueColors,Err}
Min	/DCCS	/ICCS[2]	{/ ValueColors,Min}-30~
Max	/DCCE	/ICCE[2]	{/ ValueColors,Max}30~
Low	/DCCB	/ICCB[2]	{/ ValueColors,Low}
Normal	/DCCN	/ICCN[2]	{/ ValueColors,Normal}
High	/DCCA	/ICCA[2]	{/ ValueColor,High}
Labels	/DCSL	/ICSL[2]	{/ ValueColors,Labels}
Windows			
Horizontal	/LWH	/WWH	{/ Windows,Horizontal}
Vertical	/LWV	/WWV	{/ Windows,Vertical}
Clear	/LWC	/WWC	{/ Windows,Clear}
Synch	/LWS	/WWS	{/ Windows,Synch}
Unsynch	/LWU	/WWU	{/ Windows,Unsynch}

2. Not in Lotus 1-2-3.

Table 4.3: Menu Equivalents and Associated Keystrokes, Continued

Menu Equivalent	Quattro Menu	Lotus Menu	Example
XAxis			
ScaleMode	/GCXS	/GGXS	{/ XAxis,ScaleMode}m
Min	/GCXL	/GGXL	{/ XAxis,Min}20~
Max	/GCXU	/GGXU	{/ XAxis,Max}40~
Step	/GCXI	/GGXI[2]	{/ XAxis,Step}4~
Format	/GCXF	/GGXF	{/ XAxis,Format}f~
Title	/GCXT	/GGXT	{/ XAxis,Title}Big X~
TitleColor	/GCCTX	/GGCTX[2]	{/ XAxis,TitleColor}
Labels	/GX		{/ XAxis,Labels}F67..H67~
Alternate	/GCXA	/GGXA[2]	{/ XAxis,Alternate}y
Skip	/GCXN	/GGXN[2]	{/ XAxis,MinorTicks}{esc}
Update	/GCXU	/GGXU	{/ XAxis,Update}
Reset	/GCXR	/GGXR[2]	{/ XAxis,Reset}
ShowScale	/GCXD	/GGXD	{/ XAxis,ShowScale}y
YAxis			
ScaleMode	/GCYS	/GGXS	{/ YAxis,ScaleMode}m
Min	/GCYL	/GGYL	{/ YAxis,Min}20~
Max	/GCYU	/GGYU	{/ YAxis,Max}35~
Step	/GCYI	/GGYI[2]	{/ YAxis,Step}5~
Format	/GCYF	/GGYF	{/ YAxis,Format}f~
Title	/GCYT	/GGYT	{/ YAxis,Title}Super Y~
TitleColor	/GCCTY	/GGCTY[2]	{/ YAxis,TitleColor}
Skip	/GCYN	/GGYN[2]	{/ YAxis,MinorTicks}2~
Update	/GCYU	/GGYU[2]	{/ YAxis,Update}
Reset	/GCYR	/GGYR[2]	{/ YAxis,Reset}
ShowScale	/GCYD	/GGYD	{/ YAxis,ShowScale}n

2. Not in Lotus 1-2-3.

A P P E N D I X

A

Quattro Keys and Indicators

This appendix includes five tables that define spreadsheet indicators (both mode indicators and status indicators) that appear on the bottom line of your Spreadsheet Screen and the function of special keys in Quattro.

Table A.1: Spreadsheet Modes

Mode	Description
EDIT	The spreadsheet is in Edit mode, allowing changes to be made to the contents of the current cell. To enter Edit mode, press *F2*.
ERROR	An error has occurred. Press *Esc* to go back to what you were doing before the error, or correct the error.
FIND	The system is searching for entries that match the conditions you specified with the Query command.
FRMT	You are editing a format line during a Parse operation.
HELP	A help screen is displayed. To display help, press *F1*. To exit, press *Esc*.
LABEL	The entry you are making is text (a label).
MENU	A menu is displayed. To display the main menu, press /. To remove a menu, press *Esc*.
POINT	The spreadsheet is in Point mode, allowing you to specify a cell or block with the cell selector.
READY	Quattro is ready for the next entry or command.
REP	The value you enter (in a menu command) will replace the existing value.
VALUE	The entry you are making is a number or formula.
WAIT	An action is in progress. You must wait before proceeding.

Table A.2: Spreadsheet Status Conditions

Status	Description
CALC	At least one formula in the spreadsheet needs to be recalculated. Press *F9* to recalculate the formula(s).
CAPS	The *Caps Lock* key is on.
CIRC	A formula in the spreadsheet contains a circular reference (it refers to itself or to another formula that refers back to it).
DEBUG	The macro Debug Window is displayed and/or a macro is being debugged.
END	The *End* key is on.
MACRO	A macro is being executed.
NUM	The *Num Lock* key is on.
OVR	The *Insert* key is on.

Table A.3: The Direction Keys in a Spreadsheet

Key	Description
Left arrow	Moves left one cell.
Right arrow	Moves right one cell.
Up arrow	Moves up one cell.
Down arrow	Moves down one cell.
Ctrl-Left arrow or *Shift-Tab*	Moves left one screen.
Ctrl-Right arrow or *Tab*	Moves right one screen.
Pg Up	Moves up one screen.
Pg Dn	Moves down one screen.
End	Must be used with another direction key.
End-Home	Moves to lower-right corner of the filled-in part of the spreadsheet.
End-Up arrow	If the current cell contains an entry, it moves upward to the next filled-in cell beneath an empty one. If the current cell is blank, it moves upward to the next filled-in cell encountered.
End-Down arrow	If the current cell contains an entry, it moves downward to the next filled-in cell above an empty one. If the current cell is blank, it moves downward to the next filled-in cell encountered.
End-Right arrow	If the current cell contains an entry, it moves right to the next filled-in cell followed by an empty one. If the current cell is blank, it moves to the next filled-in cell to the right.
End-Left arrow	If the current cell contains an entry, it moves left to the next filled-in cell preceded by an empty one. If the current cell is blank, it moves to the next filled-in cell to the left.
GOTO (*F5*)	Moves to any cell you specify.

Table A.4: The Quattro Function Keys

Key	Name	Description
F1	HELP	Displays information about the current area, e.g., a highlighted menu item or active Quattro function. If pressed within a help screen, it displays information about the help function. Press *ESC* to exit help.
Alt-F1	PREVIOUS HELP	Redisplays the last help message that was viewed. If pressed within a help screen, it returns you to the initial help index.
F2	EDIT	Puts Quattro into Edit mode so you can make changes to a cell entry.
F3	NAMES	Displays a list of block names for the spreadsheet. Press + on the numeric keypad to show coordinates. Selecting a block name enters it on the input line.
Alt-F3	FUNCTIONS	Displays a list of functions for the spreadsheet. Press + on the numeric keypad to show syntax. Selecting a function enters it on the input line.
Shift-F3	MACROS	Displays a menu of macro commands for the spreadsheet. Selecting a command enters it on the input line.
F4	ABS	Makes the cell address to the left of the cursor absolute. Press repeatedly to cycle through the possible absolute combinations, e.g., B4, B$4, $B4, B4.
F5	GOTO	Moves the cell selector to a specifed cell address.
F6	WINDOW	Moves the cell selector to the inactive window when the screen is split into two windows.
F7	QUERY	Repeats the previous What-If command.
Alt-F7	TABLE	Repeats the last Query command, and *Alt-F7* repeats the last What-If command.
Shift-F7	ADD-IN	Repeats the last Add-In command.

Table A.4: The Quattro Function Keys, Continued

Key	Name	Description
F8	MACRO	Displays a list of existing macros. Select one to execute it.
Shift-F8	DEBUG	Enters Single-Step mode for debugging macros.
Alt-F8	RECORD	Enters macro Record mode, in which all keys you press and commands you enter are recorded in a macro. If you're in Record mode, pressing *Alt-F8* again exits Record mode.
F9	CALC	In Ready mode, calculates any formulas that have been entered or changed since you turned off automatic recalculation or last pressed CALC. In Value or Edit mode, it converts the formula on the input line to the end result.
F10	GRAPH	Displays the current graph. Press *Esc* to return to the spreadsheet.

Other Special Keys

A few other keys on your keyboard perform special functions. They are described in Table A.5.

Table A.5: Special Keys in Quattro

Key	Description
+ (EXPAND)	If you are in a menu, it displays the command settings for the menus. (Pressed only on the numeric pad.)
– (CONTRACT)	If you are in a menu, it shrinks the menus, so that settings are no longer displayed. (Pressed only on the numeric pad.)
Alt	Pressed with certain function keys to invoke commands.
Backspace	Used in typing or editing an entry to erase the character to the left of the cursor. If pressed within a help screen, it returns you to the previous level of help. If pressed while pointing out a cell block, it returns you to the cell that was current before you entered Point mode.
Caps Lock	Enters Caps mode, in which all letters you enter are displayed in capital letters. Press again to exit Caps mode.
Ctrl-Backspace	Erases any existing entry in a prompt line.
Ctrl-Break	Exits from a menu immediately to the spreadsheet's Ready mode. If pressed while a macro is executing, it terminates the macro.
Ctrl-Enter	Lets you assign the highlighted menu command to a special key as a "shortcut."
Del	Erases the contents of the current cell.
Enter	In the spreadsheet, writes the entry on the input line into the current cell and returns to Ready mode. In a menu or choice list, selects the highlighted item. Also used with *Ctrl* to store a menu command in a special key as a "shortcut."
Esc (Escape)	Backs out of whatever you are doing, e.g., backs you out of a menu, erases any changes you made to an entry on

	the input line, or removes a system prompt from the screen without invoking a command. If pressed within the Help function, it returns you directly to the spreadsheet.
Num Lock	Lets you use the numeric keypad on the right of the keyboard to enter numbers. Press again to reuse these keys as direction keys.
Print Screen	Prints the contents of the current screen exactly as displayed.
Scroll Lock	Locks the screen so that pressing *Up arrow* or *Down arrow* scrolls the screen instead of moving the cursor.
Shift	Pressed with alphanumeric keys, enters an uppercase character.

A P P E N D I X

B

Hardware Notes

This appendix covers the hardware equipment that Quattro can work with. It lists only the hardware that has been thoroughly tested with Quattro. For information on other compatible or equivalent systems, consult your computer dealer.

Personal Computers

Quattro runs on the IBM PC, XT, AT, Portable, 3270, and computers that are truly IBM-PC compatible. If you want to be able to view graphs on the screen, the computer must also have graphics capability.

Graphics Cards

Quattro can run on several types of graphics cards, including the following:

- IBM Color/Graphics Adapter
- Hercules Monochrome Graphics Adapter
- IBM 3270/PC and 3270/AT with APA
- AT&T 6300 640x400
- MCGA (IBM Model 30)
- IBM VGA
- IBM Enhanced Graphics Adapter
- Sigma 400
- IBM 8514 Graphics Adapter

Text Printers

Quattro uses any standard ASCII printer for printing spreadsheets and other text. Some popular printers are

- IBM Proprinter
- Epson (FX, MX, and RX series)
- C. Itoh 8510
- Okidata Microline series (ML 92 and ML 93)
- Okidata Serial Microline series (ML 92S and ML 93S)
- Epson-compatibles

Graphics Printers

Quattro requires a graphics plotter or printer device in order to print graphs. Printers and plotters supported include the following:

- Apple LaserWriter
- C. Itoh Printers
  - Model 85105
  - C-310 XP
  - C-315 XP
  - C-715 (Epson LQ 1500 Identity)
  - C-815 (Proprinter Mode)
  - C-815 (Toshiba 351 Mode)
  - C-815 (Native Mode)
- Calcomp Plot Master
- Canon Laser Printer
- Epson Printers
  - MX with GrafTrax Plus
  - FX, RX, LX Series
  - FX 86, FX 286
  - JX 80
  - EX 800
  - EX 800 with Color Option
  - LQ 800

- LQ 2500
- LQ 2500 with Color Option
- GQ 1500 with .5 mb RAM
- GQ 1500 with 1.5 mb Option

■ Fujitsu Printers

- DPL24I
- DPL24C
- DL2400 (DPL24C Mode)
- DL2400 (JX-80 Mode)

■ Hewlett Packard Printers

- HP PaintJet
- HP LaserJet
- HP LaserJet Plus
- HP LaserJet II

■ HP Compatible Plotters

- HP 7220
- HP 7470
- HP 7475
- HP 7550
- ColorPro
- Sweet P

■ IBM Printers

- Graphics Printer
- Color Jetprinter
- Proprinter
- Quietwriter
- Personal Pageprinter (PostScript)

■ Lotus .PIC Output

■ NEC Printers

- P560/P565 Black
- P560XL/P565XL Color
- LC 860+
- LC 860+ in LaserJet mode

- LC 890 in PostScript mode
- LC 890 in LaserJet mode

■ Okidata Printers

- Okidata 84 Step 2
- Okidata 92
- Okidata 192
- Okidata 193 Plus

■ Panasonic Printers

- KX-P1092

■ Quadram Printers

- QuadLaser

■ Texas Instruments Printers

- Model 850
- Model 855
- Model 860 XL
- Model 865

■ Toshiba Printers

- Model 321
- Model 351
- Page Laser 12

Note: Many other printers will work using emulations of the above printers.

Expanded Memory Cards

You can add expanded memory cards to your computer to increase the amount of data it can "memorize" and the speed with which it processes data. Among the cards supported by Quattro are:

■ AST RAMpage
■ Intel AboveBoard
■ Quatram Liberty
■ STB Memory Champion

A P P E N D I X

C

Printer Setup Strings

When you print a spreadsheet or graph, you can send your printer specific print instructions sending it a *setup string* with one of the setup string commands. In a setup string, you can specify line spacing, print mode, typeface, and so on.

A setup string consists of one or more ASCII *character codes*, which are translated from the printer control commands listed in your printer's manual. Each printer recognizes its own set of control commands. The following tables list the ASCII codes for Epson, IBM, and Hewlett-Packard printer commands.

To enter a command for a printer other than an Epson, IBM, or Hewlett-Packard LaserJet, refer to your printer's manual for the control or escape commands. To translate your printer's escape (*Esc*) command, type *\027*, then enter the letter or number following *Esc* in the print command.

To enter a control command (a key combination that uses the *Ctrl* key), look up the letter after *Ctrl* on an ASCII table (see Appendix C). Type *\0* and the corresponding Hex code shown on the table.

You can enter more than one code in a setup string, as long as the printer supports it. For example, the Epson FX 85 can print near-letter-quality, bold, elite-pitch characters, but it cannot print italicized characters in near-letter-quality mode. When entering multiple codes, don't insert spaces between them.

Table C.1: Epson Printer Control Codes

Code	Instruction
Print Mode:	
\027X1	Select near-letter-quality mode
\027X0	Select draft mode
\015	Select compressed mode
\018	Cancel compressed mode
\014	Select expanded mode (one line)
\020	Cancel expanded mode (one line)
\027W1	Select expanded mode
\027W0	Cancel expanded mode
\027M	Select elite pitch
\027P	Select pica pitch
\027E	Select emphasized mode
\027F	Cancel emphasized mode
\027G	Select double-strike mode
\027H	Cancel double-strike mode
\027S0	Select superscript
\027S1	Select subscript
\027T	Cancel superscript/subscript
\027-1	Select/cancel underlining
\0274	Select italic mode
\0275	Cancel italic mode
\027R(n)	Select international character set
\027p	Select/cancel proportional mode
\027(space)(n)	Select character space
\027a(n)	Near-letter-quality justification
Page Formatting:	
\027Q(n)	Set right margin
\027I(n)	Set left margin
\027N(n)	Select skip-over perforation
\027O	Cancel skip-over perforation
\0272	Single-space text
\027J24	Double-space text
\027C(n)	Set page length in number of lines
\027C0(n)	Set page length in number of inches

* The justification values for (n) for are as follows:
0 = left justification
1 = centering
2 = right justification
3 = full justification

Table C.2: IBM Printer Control Codes

Code	Instruction
Print Mode:	
\015	Select compressed mode
\018	Cancel compressed mode
\014	Select expanded mode (one line)
\020	Cancel expanded mode (one line)
\027W1	Select expanded mode
\027W0	Cancel expanded mode
\027E	Select emphasized mode
\027F	Cancel emphasized mode
\027G	Select double-strike mode
\027H	Cancel double-strike mode
\027S0	Select superscript
\027S1	Select subscript
\027T	Cancel superscript/subscript
\027-1	Select/cancel underlining
\0276	Select international character set
\0277	Select standard character set
Page Formatting:	
\027X(n1)(n2)	Set margins (1=left; 2=right)
\027N(n)	Select skip-over perforation
\027O	Cancel skip-over perforation
\0272	Single-space text
\027A24	Double-space text
\027C(n)	Set page length in lines
\027C0(n)	Set page length in inches

Table C.3: Hewlett-Packard LaserJet Printer Control Codes

Code	Instruction
Print Mode:	
\027E\027&l0O	Portrait
\027E\027&l0O\027&k2S	Portrait compressed print
\027E\027&l1O	Landscape
\027E\027&l1O\027&k2S	Landscape compressed print

For more detailed information about any of these printer commands, see the owner's manual for your printer.

A P P E N D I X

D

ASCII Codes

The American Standard Code for Information Exchange (ASCII) is ASCII code that translates alphabetic and numeric characters and symbols and control instructions into 7-bit binary code. Table D.1 shows both printable characters and control characters.

IBM Extended ASCII Character Set

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
0	0		32	20		64	40	@	96	60	`
1	1	☺	33	21	!	65	41	A	97	61	a
2	2	☻	34	22	"	66	42	B	98	62	b
3	3	♥	35	23	#	67	43	C	99	63	c
4	4	♦	36	24	$	68	44	D	100	64	d
5	5	♣	37	25	%	69	45	E	101	65	e
6	6	♠	38	26	&	70	46	F	102	66	f
7	7	•	39	27	'	71	47	G	103	67	g
8	8	◘	40	28	(	72	48	H	104	68	h
9	9	○	41	29	)	73	49	I	105	69	i
10	A	◙	42	2A	*	74	4A	J	106	6A	j
11	B	♂	43	2B	+	75	4B	K	107	6B	k
12	C	♀	44	2C	,	76	4C	L	108	6C	l
13	D	♪	45	2D	-	77	4D	M	109	6D	m
14	E	♫	46	2E	.	78	4E	N	110	6E	n
15	F	☼	47	2F	/	79	4F	O	111	6F	o
16	10	►	48	30	0	80	50	P	112	70	p
17	11	◄	49	31	1	81	51	Q	113	71	q
18	12	↕	50	32	2	82	52	R	114	72	r
19	13	‼	51	33	3	83	53	S	115	73	s
20	14	¶	52	34	4	84	54	T	116	74	t
21	15	§	53	35	5	85	55	U	117	75	u
22	16	▬	54	36	6	86	56	V	118	76	v
23	17	↨	55	37	7	87	57	W	119	77	w
24	18	↑	56	38	8	88	58	X	120	78	x
25	19	↓	57	39	9	89	59	Y	121	79	y
26	1A	→	58	3A	:	90	5A	Z	122	7A	z
27	1B	←	59	3B	;	91	5B	[	123	7B	{
28	1C	∟	60	3C	<	92	5C	\	124	7C	|
29	1D	↔	61	3D	=	93	5D	]	125	7D	}
30	1E	▲	62	3E	>	94	5E	^	126	7E	~
31	1F	▼	63	3F	?	95	5F	_	127	7F	⌂

IBM Extended ASCII Character Set, continued

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	β
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	∅
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	∈
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	▒	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▓	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	▉	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	n
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	2
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

A P P E N D I X

E

Error Messages

This appendix lists, alphabetically, each error message that could possibly display on your screen. It briefly describes the error that caused the message and offers resolutions. To remove an error message from the screen, press *Esc.*

ADD-IN INIT ERROR—An error was found in the program add-in you tried to run.
Resolution: Recopy the add-in from your Quattro distribution diskettes into the QUATTRO directory.

CANNOT FIND SCREEN DRIVER—In loading Quattro, a screen driver was not found in the current resource directory.
Resolution: Make sure that you have specified the proper directory for resource files, and that the resource directory has all screen driver files. If in doubt, recopy the files from the Quattro master diskettes to the resource directory.

CANNOT HIDE ALL COLS—You attempted to hide all columns of the spreadsheet, which is impossible.
Resolution: Respecify the columns you want to hide.

CANNOT LOAD DRIVER—Unable to find driver.
Resolution: Use the Quattro system disks to reload all drivers into the Quattro directory.

CANNOT OPEN FILE—Quattro can't find a file by the name you specified.
Resolution: Check your file name. Make sure you're accessing the correct directory.

CLOSE FILE ERROR—The file you specified cannot be closed. Possible disk error.
Resolution: Check disk drive door on a floppy disk.

COL/ROW BOUNDARY ERR—You attempted to move or copy data where there is not enough room.
Resolution: Specify a different destination for the copy or move.

COULD NOT COMBINE ENTIRE BLOCK—The file you want to combine won't fit in the area you specified.
Resolution: Move the cursor to a new position that provides more room for data input.

CREATE FILE ERROR—Quattro can't create the file you specified.
Resolution: If you are creating the file on a floppy drive, make sure the floppy disk drive is closed. Also check to confirm that the current directory has read/write/create privileges.

DIR. DOES NOT EXIST—You specified a directory that doesn't exist.
Resolution: Make sure you typed the exact path name of the directory.

DISK IS FULL—The disk you're writing on is full.
Resolution: If you're using a floppy-based system, insert a new, formatted disk and try again. If you need to format the disk, use the **O**S command to access DOS. If you're using a hard-disk system, write the file to a disk in Drive A, then clean out some of the files on your hard disk.

DISK WRITE PROTECTED—The disk specified for a save is write protected
Resolution: Remove the write-protect tab from your floppy disk.

DRIVE ISN'T READY—The disk drive specified is not ready for I/O.
Resolution: Make sure the proper floppy drive has a disk, and that the drive door is closed.

ERROR LOADING ADD-IN—The program add-in you tried to load was not found or is not compatible with Quattro.
Resolution: Check to make sure your current add-in file (FILENAME.QAI) is in your current resource directory.

FILE DOESN'T EXIST—Quattro can't find the file you specified.
Resolution: Try looking in a different directory. You can change the directory temporarily with the **F**ile **D**irectory command (*/FD*), or specify a different directory with the file name.

FILE DRIVER NOT FOUND—No file driver was found for the file extension you specified.
Resolution: You cannot load a file with the specified extension. Check to make sure the extension is spelled correctly.

FILE IS CORRUPTED—The file you tried to read is corrupted.
Resolution: The file you specified cannot be read by Quattro; its data has been corrupted.

FILE NOT COMPATIBLE—The format of the file you tried to load is not compatible with any Quattro format.
Resolution: If possible, change the format of the file you are trying to load to a file type that can be loaded into Quattro.

GENERAL FAILURE—There has been a DOS general failure.
Resolution: If you tried to save to a floppy disk, make sure that the disk is formatted for DOS.

HIDDEN COLUMN—You attempted to go to a hidden column.
Resolution: Expose the hidden column (*/CE*).

ILLEGAL CRITERIA—You attempted to use an illegal function or a database function in search criteria.
Resolution: Enter your search criteria again without the illegal function or database function.

INSERT SYSTEM DISK—The system disk is need for the next step.
Resolution: Insert the Quattro System Disk in your boot drive.

INVALID ADD-IN/DRIVER—The add-in you tried to load does not have a valid format for Quattro.
Resolution: From your Quattro program disks, copy the add-in and/or driver files into the Quattro directory.

INVALID ALT MENUS—The alternate menu specified has an invalid format.
Resolution: Make sure that the file specified was created in Menu Builder, and is a current version; .RSC files renamed to .ALT files will not work, for instance.

INVALID BLOCK—The block you specified is not valid.
Resolution: Re-enter the block coordinates, separated by one or two periods. You can also use pointing to specify the block.

INVALID FILE—You tried to retrieve a file that cannot be translated by Quattro. Quattro will automatically translate the following types of files: Lotus 1-2-3, dBASE, Paradox, Symphony, and DIF.
Resolution: Translate the file into a Lotus 1-2-3 file or ASCII text file, and import the file.

INVALID FILE DRIVER—The file driver you are attempting to use is not compatible with the current version of Quattro.
Resolution: Get an updated version of the file driver.

INVALID FORMAT LINE—Parse line is invalid.
Resolution: Check to see that there is a "|" at the beginning of the parse line and that there are no invalid characters.

INVALID FORMAT POS—Quattro cannot edit a parse line on the bottom line of the screen.
Resolution: Move the cursor down one line.

INVALID GRAPH NAME—The graph does not exist.
Resolution: Check to make sure the graph name you typed is spelled correctly.

INVALID MARGIN VALUES—The combination of values you specified for the margin settings is invalid. The left margin setting cannot be greater than the right, and the top margin setting cannot be greater than the bottom. The valid ranges for each are Left and Right (0-240); Top and Bottom (0-32).
Resolution: Respecify the margin settings.

INVALID PAGE LENGTH—The page length you specified is not valid. The valid range for this setting is 1 to 100.
Resolution: Respecify page length.

INVALID PASSWORD—The password you gave is not correct.
Resolution: Try again. If you can't supply the correct password, you won't be able to retrieve the file.

INVALID PRINT BLOCK—There is no print block specified, or the block you specified is invalid.
Resolution: Specify a valid print block.

INVALID RANGE—Range specified is invalid
Resolution: Respecify the range to be valid.

INVALID SETUP—The setup string you entered is invalid
Resolution: Check Appendix C for the correct setup codes. Then reenter the setup string.

INVALID SPREADSHEET—You tried to retrieve a file that is not properly formatted as a Quattro file.
Resolution: Only attempt to retrieve Quattro files.

INVALID X BLOCK—The independent values specified for regression are invalid.
Resolution: Make sure the number of values is less than 17.

INVALID Y BLOCK—The dependent values specified for regression are invalid.
Resolution: Make sure that there are an equal number of dependent and independent values.

KEY IS IN USE—The key you specified is already used, and is not available for shortcuts.
Resolution: Choose another key for the shortcut.

KEY MUST BE INSIDE SORT BLOCK—You specified a sort key that was not part of the sort block.
Resolution: Specify a different sort key or respecify the sort block.

MACRO ERROR—The macro you were running contains an error.
Resolution: Use Debug mode to find the error, then use Edit mode to fix it.

MATRIX SIZE ERROR—You can't multiply matrices over 90 rows or columns; or, matrices that are not square cannot be inverted.
Resolution: Adjust the size of your matrix.

MATR. NOT INVERTIBLE—The matrix you attempted to invert cannot be inverted.
Resolution: Adjust your matrix so that the determinant is not zero.

MEMORY FULL—Your computer's memory is full.
Resolution: Split your spreadsheet into separate files, unload an add-in, unload an alternate menu tree, or remove a memory-resident program.

NO ALTERNATE MENU—When Quattro was loading, a specified alternate menu was not found.
Resolution: Make sure that the .ALT file specified is in your indicated resource directory.

NO ALT MENU SPACE—There is no room left for an alternate menu.
Resolution: Don't specify an alternate menu, or use options in **MEMORY FULL**.

NO BLOCK NAME—Range to delete or use does not exist.
Resolution: Make sure the block name you specified is valid.

NO COMMAND.COM FOUND—The COMMAND.COM file isn't available for DOS operations.
Resolution: Make sure that COMMAND.COM is in your boot drive.

NO HELP FILE—Quattro can't find the QUATTRO.HLP file containing help information.
Resolution: Make sure the QUATTRO.HLP file is in the directory specified with the Resource Directory. If not, change the Resource Directory or copy the file to it.

NO MATCH—Quattro couldn't find any records matching the criteria you gave.
Resolution: If you think there really are matching records, try respecifying the criteria. Remember to reference the first value in the field you want to search, not the field name.

NO MORE ADD-IN SLOTS—You already have eight add-ins loaded into Quattro. You cannot load any more.
Resolution: Unload add-ins you're not using.

NO MORE ROOM TO RECORD—Recording has gone outside the range you specified.
Resolution: Begin recording again with a larger range, or open space for further recording. In the latter case, make sure to insert any commands that may have been truncated when the overflow occurred.

NO OUTPUT BLOCK—No output range was specified for regression data.
Resolution: Specify a regression output range.

NO PARSE FORMAT LINE—You didn't include the format line with the parse block.
Resolution: Respecify the **P**arse **B**lock to include the parse line as part of the parse block.

NO PARSE INPUT BLOCK—Quattro can't find the data to parse.
Resolution: Specify a **P**arse **B**lock, then try again.

NO PARSE OUTPUT BLK.—Quattro doesn't know where to enter parsed data.
Resolution: Specify a **P**arse **O**utput Block, then try again.

NO ROOM FOR TITLE—The cursor is near the top or left edge of the screen, so it is not possible to create a title.
Resolution: Move the cursor away from the top or left edge of the screen.

NO ROOM FOR WINDOW—You attempted to open another window when two windows were already displayed, or you attempted to create a window when the cursor was on the left or top edge of the screen. (Quattro allows only two windows to be open at a time.)
Resolution: If you were near the left or top edge, then move the cursor away from the edge.

NO SEARCH STRING—No search string has been specified for search/replace.
Resolution: Enter a string for the "Search" choice on the Search/Replace menu.

NO UNPROTECTED CELL—You can't use the **F**orm Input command unless there are cells that have been explicitly unprotected.
Resolution: Set **D**efault protection to **E**nable (*/BAE*), then disable protection for the cells you want access to (*/BAU*).

NO WINDOW—You cannot change or clear window; only one window is open.
Resolution: This command does not make sense with only one window open.

NO X BLOCK—No independent range has been specified for regression.
Resolution: Enter an independent range for regression.

NO Y BLOCK—No dependent range has been specified for regression.
Resolution: Enter a dependent range for regression.

NON EMPTY CELL—The macro you're recording won't fit in the current cell and the cell below it contains data, or the macro location you specified contains data. Quattro won't store a macro in a non-empty cell.
Resolution: Re-record the macro in a different location. Be sure there are enough empty cells beneath the location you specify.

NOT EXISTING MACRO—The macro you attempted to execute doesn't exist.
Resolution: Make sure you typed the correct macro name. Try selecting it from the Names list displayed when you select **M**acro **E**xecute.

OPEN FILE ERROR—The file you specified can't be opened.
Resolution: Check your file name. Possible disk error.

OUTP. BLOCK TOO SMALL—The range you specified for **B**lock **R**eformat is too small to hold the text.
Resolution: Specify a larger range.

PRESS ANY KEY TO CONTINUE—
Resolution: Read the error message, then press the space bar to continue to work in Quattro.

PRINT PARAMETER ERROR—One or more print specifications is invalid.
Resolution: Check to make sure you don't have overlapping margins, and that your page length and top and bottom margins are compatible.

PRINTER ERROR—The printer is unable to print your spreadsheet or graph.
Resolution: Make sure the printer is turned on, securely connected to your computer, and on line.

PROTECTED CELL—The cell you attempted to enter information into or change is protected.
Resolution: Remove the protection. You can use the **B**lock **A**dvanced **U**nprotect command to unprotect the cell you want to enter data in.

READ FILE ERROR—Read file error. Invalid {read} command
Resolution: Check to make sure {open} command precedes {read} command. Also check to make sure that the read block is valid.

SPREADSHEET VERSION IS OUT OF DATE—Attempting to use an outdated spreadsheet file.
Resolution: Upgrade to the new version of Quattro.

SYNTAX ERROR—The syntax of your Formula is incorrect.
Resolution: Check the syntax and edit the formula.

TOO FEW OBSERVATIONS—There is not enough data to generate regression data.
Resolution: Add more rows of data to the independent and dependent ranges.

TOO MANY COLUMNS—You attempted to insert a column when the last column(s) had nonempty cells.
Resolution: Erase the cells in the last column(s) if appropriate.

TOO MANY ROWS—You attempted to insert a row when the last row(s) had nonempty cells.
Resolution: Erase the cells in the last row(s) if appropriate.

WRITE FILE ERROR—Cannot write the file.
Resolution: If you are creating the file on a floppy drive, make sure the floppy disk drive is closed. Also check to confirm that the current directory has read/write/create privileges.

X BLOCK REQUIRED—A range of X labels is required for an XY graph.
Resolution: Specify X labels for your graph.

Index

A

B

C

D

E

F

G

W

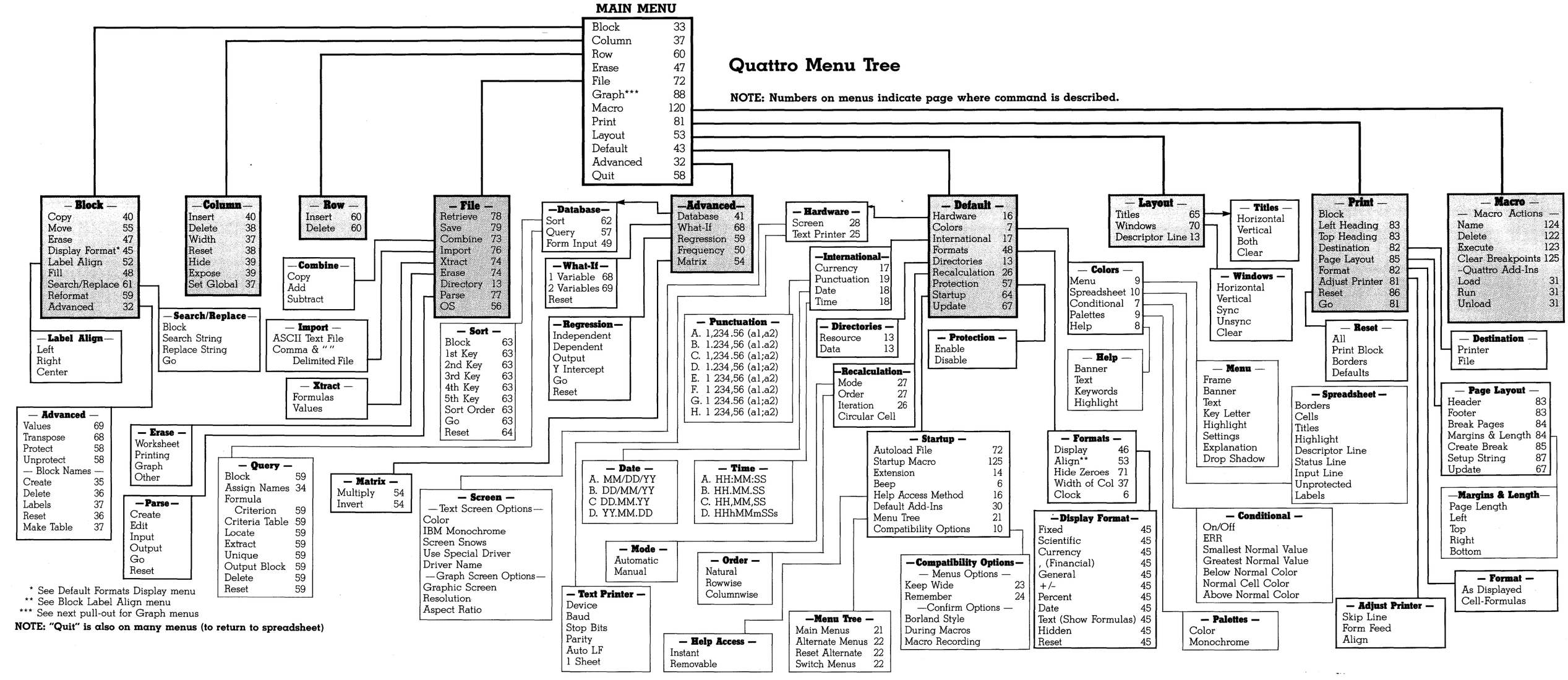
X

Y

Z

Quattro Menu Tree

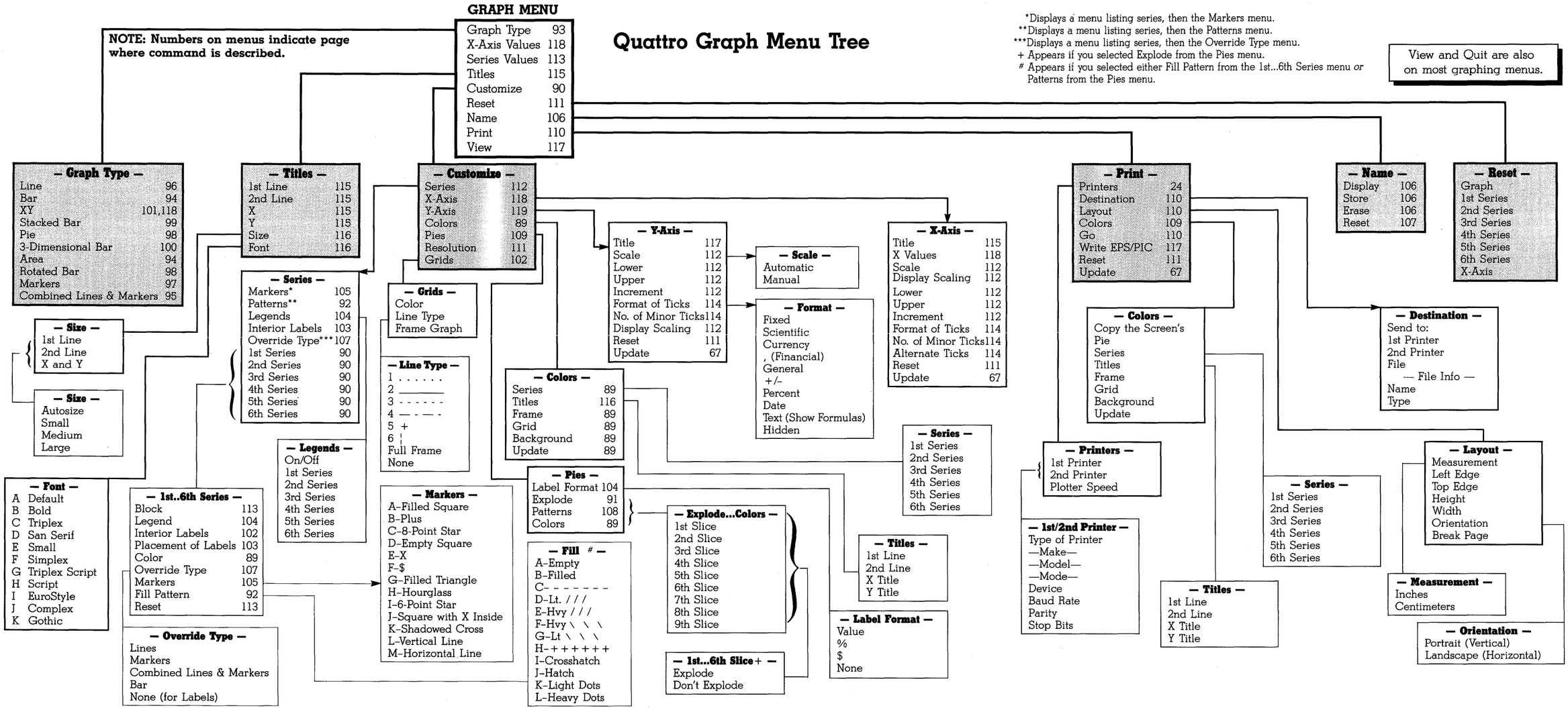
NOTE: Numbers on menus indicate page where command is described.



* See Default Formats Display menu
** See Block Label Align menu
*** See next pull-out for Graph menus

NOTE: "Quit" is also on many menus (to return to spreadsheet)

Quattro Graph Menu Tree



123.RSC MENU TREE

NOTE: Menu items with square bullet ■ indicate Quattro extensions to Lotus 1-2-3 menus.

*See next pullout for Graph menus.

MAIN MENU
Worksheet
Range
Copy
Move
File
Print
Graph*
Data
System
Add-In/Macro ■
Install ■
Quit

— Worksheet —
Global
Insert
Delete
Column
Erase
Titles
Window
Status
Page

— Global —
Format
Label-Prefix
Column-Width
Recalculation
Protection
Default
Zero

— Protection —
Enable
Disable

— Recalculation —
Natural
Columnwise
Rowwise
Automatic
Manual
Iteration

— Default —
Printer
Directory
Status
Update
Other
Resource Directory ■
Quit

— International —
Punctuation
Currency
Date
Time
Quit

— Other —
International
Help
Clock

— Status —
Normal Memory
Expanded Memory
Co-Processor
Quit

— Printer —
Interface
Auto-LF
Left
Right
Top
Bottom
Page-Length
Wait
Setup
Name
Quit

— Delete —
Column
Row

— Column —
Set-Width
Reset-Width
Hide
Display

— Status —
Normal Memory
Expanded Memory
Co-Processor
Quit

— Titles —
Both
Horizontal
Vertical
Clear

— Window —
Horizontal
Vertical
Sync
Unsync
Clear

— Insert —
Column
Row

— Range —
Format
Label
Erase
Name
Justify
Protect
Unprotect
Input
Value
Transpose
Search/Replace ■

— Name —
Create
Delete
Labels
Reset
Table

— Labels —
Right
Down
Left
Up

— Search/Replace■ —
Range
Search String
Replace String
Go
Quit

— File —
Retrieve
Save
Combine
Xtract
Erase
List
Import
Directory

— Xtract —
Formulas
Values

— Import —
Text
Numbers

— Combine —
Copy
Add
Subtract

— Subtract —
Entire-File
Named-Range

— Add —
Entire-File
Named-Range

— Copy —
Entire-File
Named-Range

— Install —
Hardware
Colors
Startup
Update

— Startup —
Autoload File
Startup Macro
Extension
Beep
Default Add-Ins
Menu Tree
Toggle Descriptor Line
Compatibility Options
Quit

— Compatibility Options —
— Menu Options —
Keep Wide
Remember
— Confirm Optins —
Borland Style
During Macros
Macro Recording
Quit

— Menu Tree —
Main Menu
Alternate Menu
Reset Alternate
Switch Menu

— Switch Menu —
Main Menu
Alternate Menu
Quit

— Hardware —
— Text Screen Options —
Color
IBM Monochrome
Screen Snows
Use Special Driver
Driver Name
— Graph Screen Options —
Graphic Screen
Resolution
Aspect Ratio
Quit

— Colors —
Menu
Spreadsheet
Conditional
Palettes
Help
Quit

—Default Add-Ins—
1
2
3
4
5
6
7
8
Reset

— Reset —
1
2
3
4
5
6
7
8
All
Quit

— Add-In/Macro —
— Macro Actions —
Name
Delete
Execute
Clear Breakpoints
— Quattro Add-Ins —
Load
Run
Unload

— Menu —
Frame
Banner
Text
Key Letter
Highlight
Settings
Explanation
Drop Shadow
Quit

— Palettes —
Color
Monochrome
Quit

— Help —
Banner
Text
Keyword
Highlight
Quit

— Sort —
Data-Range
Primary-Key
Secondary-Key
Reset
Go
3 ■
4 ■
5 ■
Quit

— Query —
Input
Criterion
Output
Find
Extract
Unique
Delete
Reset
Assign Names ■
Logical Criteria
Quit

— Conditional —
On/Off
ERR
Smallest Normal Value
Greatest Normal Color
Below Normal Color
Normal Cell Color
Above Normal Color
Quit

— Data —
Fill
Table
Sort
Query
Distribution
Matrix
Regression
Parse

—Parse —
Format-Line
Input-Column
Output-Range
Reset
Go
Quit

— Spreadsheet —
Borders
Cells
Titles
Highlight
Descriptor Line
Status Line
Input Line
Unprotected
Labels
Quit

— Print —
Printer
File

— Table —
1
2
Reset

— Matrix —
Invert
Multiply

— Regression —
X-Range
Y-Range
Output-Range
Intercept
Reset
Go
Quit

— Format-Line —
Create
Edit

123.RSC
Graph Menu Tree

GRAPH MENU

Type
X
A
B
C
D
E
F
Reset
View
Save
Options
Name
Graph Customize ■
Print ■
Quit

NOTE: Menu items with square bullet ■ indicate Quattro extensions to Lotus 1-2-3 menus.

— Reset —
Graph
X
A
B
C
D
E
F
Quit

—Legend—
A
B
C
D
E
F

— Options —
Legend
Format
Titles
Grid
Scale
Color
B&W
Data-Labels
Patterns ■
Markers ■
Quit

— Name —
Use
Create
Delete
Reset

— Format —
Graph
A
B.
C
D
E
F
Quit

— Titles —
First
Second
X-Axis
Y-Axis
Main Font
Character Size

— Data-Labels —
A
B
C
D
E
F
Quit

—Markers ■ —
A
B
C
D
E
F
Update
View
Quit

— Scale —
Y Scale
X Scale
Skip

— X Scale —
Automatic
Manual
Lower
Upper
Format
Indicator
Quit

— Y Scale —
Automatic
Manual
Lower
Upper
Format
Indicator
Quit

— Patterns ■ —
A
B
C
D
E
F
Update
View
Quit

— A..F Series —
Range
Legend
Interior Labels
Placement of Labels
Color
Override Type
Markers
Fill Pattern
Reset
View
Quit

— Graph Customize ■ —
A
B
C
D
E
F
X-Axis
Y-Axis
Colors
Pie
Resolution
Grid
View
Quit

— Pie —
Label Format
Explode
Patterns
Colors
View
Quit

— Explode —
1st Slice
2nd Slice
3rd Slice
4th Slice
5th Slice
6th Slice
7th Slice
8th Slice
9th Slice
View
Quit

— Patterns —
1st Slice
2nd Slice
3rd Slice
4th Slice
5th Slice
6th Slice
7th Slice
8th Slice
9th Slice
View
Quit

— Colors —
1st Slice
2nd Slice
3rd Slice
4th Slice
5th Slice
6th Slice
7th Slice
8th Slice
9th Slice
View
Quit

— Grid —
Color
Line Type
Frame Graph
View
Quit

— Colors —
Series
Titles
Frame
Grid
Background
Update
View
Quit

— Titles —
First
Second
X
Y
View
Quit

— Series —
A
B
C
D
E
F
View
Quit

—X-Axis —
Title
X Values
Scale
Lower
Upper
Increment
Format of Ticks
No. of Minor Ticks
Alternate Ticks
View
Reset
Update
Quit

—Y-Axis —
Title
Scale
Lower
Upper
Increment
Format of Ticks
No. of Minor Ticks
Alternate Ticks
View
Reset
Update
Quit

— Print ■ —
Printers
Destination
Layout
Colors
Go
Write EPS File
Reset
Update
Quit

— Layout —
Measurement
Left Edge
Top Edge
Height
Width
Rotated
Break Page
Quit

— Pie —
1st Slice
2nd Slice
3rd Slice
4th Slice
5th Slice
6th Slice
7th Slice
8th Slice
9th Slice
Quit

— Colors —
Copy the Screen's
Pie
Series
Titles
Frame
Grid
Background
Quit

— Series —
A
B
C
D
E
F
Quit

— Titles —
First
Second
X
Y
Quit

— Printers —
1st Printer
2nd Printer
Plotter Speed
Quit

— 1st..2nd Printer —
Type of Printer
—Make—
—Model—
—111Mode—
Device
Baud Rate
Parity
Stop Bits
Quit

QUATTRO™

4585 SCOTTS VALLEY DRIVE, SCOTTS VALLEY, CA 95066

ISBN 0-87524-171-9