

PowerMathID OS X Reference Manual

This page intentionally left blank

Table of Contents

Preface

Table of Contents	p-iii
Disclaimers	p-vii

I. Overview

Standard Features	1-1
Formatting of examples in this Manual	1-2
Formatting Interactive Keyboarding Examples	1-2
Standard Keyboard Examples	1-2
Formatting ASCII Keyboarding Examples	1-2
Cross Platform use of ASCII language	1-2
Interactive Dialog Boxes (An Overview)	1-3
Base Point Size	1-3
Pi Set	1-3
Superset	1-3
Current Element/Nested Procedures	1-3
Active Style	1-4

II. System Setup

Fonts	2-1
Install the Plugins	2-1
The “Dongle”	2-1
“Demo” Mode	2-1
Setup Is Complete (for Single Station)	2-1
The Network Server	2-2
Basic Tips for Troubleshooting	2-2
License Checkout	2-2
Give Back License	2-3
Checkout Remembered	2-3

III. Basic Concepts

Interactive Editor	3-1
ASCII Import/Export	3-1
PowerMath ASCII 4.0	3-1
Base Point Size	3-1
Level Offset Leading	3-1
Character Styles	3-2
(“Alternate” Styles)	3-2
Composition Procedures	3-2
Procedures Options	3-2
Automated Mapping to Pi Characters	3-2
The Supersets	3-2
Global Updates	3-2
The Equation Box	3-2

IV. The Main Editing Window

Equation Baseline	4-1
The Message Fields	4-1
Moving and Sizing the Editing Window	4-2
Call Editor for “Create Equation”	4-2
Set Quark to “Content Mode”	4-2
Keyboard Math Characters	4-2
Enhanced Automatic Formatting	4-2
Global Editability	4-2
Store the equation	4-2
Edit an equation	4-3

Using the Editor	4-3
Editing/deleting structures	4-3
The View/Scale Factor	4-3
V. The Equation Box	
Sizing the Equation Box	5-1
Vertically Positioning the Equation Box	5-1
Box Type (Sizing Options)	5-2
AutoCalc	5-2
When equation box “clipping” matters	5-2
“Baseline Bottom” and “Manual” sizing	5-2
Color, Shade, and “Pad” the Equation Box	5-2
Vertical Spacing Options	5-2
Set Baseline Shift to Zero	5-2
AutoLead at Ascender	5-3
Auto Space After at Descender	5-3
“Zero” value turns it off	5-3
PowerMath to PICT Converter	5-3
PICT Range Selector	5-3
Sequential Numbering Appended to PICT filenames	5-3
Root Filename and Storage Location for this batch of PICT Files	5-3
VI. Basic Specification of Type	
Base Point Size default to “Auto”	6-1
Base Point Size default to absolute value	6-1
Override the Base Point Size default	6-1
Set the Math Level Offset	6-1
Font, Face, and Scale	6-1
Text Color	6-1
VII. Accessing the Supersets and Opening Legacy Documents	
PowerMath™ SuperSets folder	7-1
Using older versions of PowerMath	7-1
Converting legacy documents to PowerMathID	7-1
Considerations when converting legacy documents	7-2
VIII. Accessing the Main Data Setups from the Superset (A General Overview)	
“Tags”: naming the elements	8-1
Accessing Procedures	8-1
Tagging	8-2
Improved procedures	8-3
Case Fraction	8-3
Built-up Divisions	8-3
Styles	8-3
Matrix	8-3
Superior/Inferior	8-3
The Stacking Procedure	8-3
Macro Procedure	8-3
Using Procedures	8-3
IX. Character Pi Interface	
Accessing the Pi Character Interface	9-1
Show Palette	9-1
Editing a Pi Character Mapping	9-2
The Pi Selector Window	9-2
Pi Characters Editing Options	9-2
The Preview Window	9-2
Auto Remap	9-3
Auto Upsize	9-3
X. Character Styles	

Accessing Character Styles	10-1
Editing Character Styles	10-1
Accessing Character Styles With the PowerMath ASCII language	

XI. Macros

Creating a Macro	11-1
Accessing a Macro Interactively	11-1
Macros in the PowerMath ASCII Filter	11-1
Using the wildcard (merge key) for macros	11-1
Using merge codes interactively	11-1
Using merge codes in ASCII	11-2

XII. Mathematics Composition Procedure

Division	12-1
Accessing the Division Procedure Interactively	12-1
Editing Division index values	12-1
Editing Division setups	12-2
Creating new Division setup	12-2
Accessing Division through the PowerMath ASCII Filter language	12-3
Summation	12-3
Editing a Summation character	12-3
Main Character Count	12-3
Accessing a sup/inf setup for the limits	12-4
Accessing Summations setups Interactively	12-4
Accessing Summations with the PowerMath ASCII Filter language	12-4
The Summation limit alignment	12-4
Setting the Summation limit alignment interactively	12-4
Setting the Summation Limit Alignment Default through the PowerMath ASCII Filter language.	12-5
Integral	12-5
Editing a Integral character	12-5
Accessing a sup/inf setup for the limits	12-5
Accessing Integrals interactively	12-6
Accessing Integrals with the PowerMath ASCII Filter language	12-6
Stack Limits	12-6
Accessing Stacked Limits through the PowerMath ASCII Filter language	12-6
Radical	12-7
Editing a Radical character	12-7
Accessing Radicals interactively	12-7
Accessing Radicals with the PowerMath ASCII filter language	12-8
Case Fraction	12-8
Accessing the Case Fraction procedure interactively	12-8
Editing Case Fraction setups	12-8
Setting a new Case Fraction setup	12-9
Accessing Case Fraction through the PowerMath ASCII filter language	12-9
Matrix	12-9
Defining a Matrix interactively	12-9
Creating a Matrix with the ASCII Filter	12-10
Math Rule	12-11
Accessing the Math Rule procedure interactively	12-11
Editing a new Math Rule setup	12-12
Cancellation Rules	12-12
Accessing Math Rule through the PowerMath ASCII language	12-12
Force Level	12-12
Using Force Level Interactively	12-12
Using Force Level though the PowerMath ASCII filter language	12-13

Stack	12-13
Choosing a Stacking procedure index interactively	12-13
Accessing Stacking with the ASCII Language	12-14
Superior and Inferiors	12-14
Accessing a Superior/Inferior setup	12-14
Editing Superior/Inferior index values	12-15
Creating a new Superior/Inferior setup	12-15
Nesting the Superior/Inferior Procedure	12-15
Stacked Super- and Subscripts	12-16
Multi-level Super- and Subscripts for oversized characters	12-16
Keying Superior/Inferior Procedure with the PowerMath ASCII filter language	12-16
Accessing Superior/ Inferior setups in the ASCII language	12-16
XIII. Customized Horizontal Placement of Superior and Inferiors	13-1
The Interface	13-1
Edit Baseline Font	13-1

Disclaimers

PowerHouse Software, Inc. ("PowerHouse") believes that the software described in this instruction manual is accurate and reliable. Great care has been taken in its preparation. However, PowerHouse makes no warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding the PowerHouse software. PowerHouse does not warrant, guarantee or make any representations regarding the use or the results of the use of PowerHouse software in terms of its correctness, accuracy, reliability, currentness or otherwise. The entire risk as to the results and performance of PowerHouse software is assumed by you. The exclusion of implied warranties is not permitted in some states. The above exclusion may not apply to you.

In no event will PowerHouse, its directors, officers, employees or agents be liable for any consequential, incidental or indirect damages (including damages for loss of business profits, business interruption, loss of business information and the like) arising out of the use or inability to use PowerHouse software even if PowerHouse has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above may not apply to you.

PowerHouse's liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort (including negligence), product liability or otherwise) will be limited to \$50.00.

The user should exercise care to assure that the use of the programs is in full compliance with laws, rules, and regulations of the jurisdictions in which they are used.

PowerMath is a proprietary product of PowerHouse Software, Inc., Andover, NH, and is not to be used, copied, or disclosed without the express written permission of PowerHouse Software, Inc. PowerMath is licensed to be used on only a single computer system per copy.

The information contained herein, the functions and specifications of PowerHouse's PowerMath are subject to change without notice. Revisions may be issued from time to time to advise of changes and /or additions.

PowerHouse Software, Inc. provides the computer software program and this User manual (the "Program"), and licenses its use on the terms stated below:

- a. You are granted a license to use the Program and Eve 3 Key under the terms stated in this agreement for personal use in your business or profession. Title and ownership of the Program and Eve 3 key remains with PowerHouse Software, Inc.
- b. The Demo version of the Program is intended as a Demo only. The Demo Program may be used by you on a computer or computers which you own or use for which the Demo Program is designed to operate.
- c. You may freely distribute this Demo Program provided that you do not require compensation for the Program and that the Demo manual be distributed with the Program.
- d. You may not sell, lease or rent this Program, Eve 3 Key or License to any other person.

Any comments or suggestions regarding this document should be forwarded to:

Mike Gorman
76 Shaw Hill Road
Andover, NH 03216
mike.gorman@gmail.com

Apple Disclaimer

Apple Computer Inc. (“Apple”) makes no warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding the Apple software.

Apple does not warrant, guarantee or make any representations regarding the use or the results of the use of Apple software in terms of its correctness, accuracy, reliability, currentness or otherwise. The entire risk as to the results and performance of Apple software is assumed by you. The exclusion of implied warranties is not permitted in some states. The above exclusion may not apply to you.

In no event will Apple, its directors, officers, employees or agents be liable for any consequential, incidental or indirect damages (including damages for loss of business profits, business interruption, loss of business information and the like) arising out of the use or inability to use the Apple software even if Apple has been advised of the possibility of such damages. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above may not apply to you.

Apple’s liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort (including negligence), product liability or otherwise will be limited to \$50.00.

I. Overview

PowerMath is a powerful and sophisticated equation typesetting and production program which is capable of producing the highest quality type for Engineering, Science, and Mathematics publications on the desktop. It is presently being used by the most demanding publishers and type suppliers worldwide, and it has become a publishing standard. This implementation of the program is a standalone application on the Macintosh, which interfaces with plugins for Adobe InDesign/InCopy. Future implementations may run as cross-platform software components, maintaining compatibility with equations produced under present and past configurations.

Standard Features

The PowerMath composition and production feature set includes:

- *Comprehensive, powerful data setups (Supersets)*
 - *Global editing capabilities, batch transformations of documents*
 - *Extensive, automated access to Pi characters and alternate fonts*
 - *Complete set of procedures automating creation of math structures*
 - *Multiple vertical alignment points*
 - *ASCII import and export for offline keyboarding and archiving*
 - *Proprietary ASCII language*
 - *“Tagging”, naming of data elements and procedures*
 - *Powerful online Macro capability, with merge codes for variable input, accessible interactively or from Ascii language*
 - *Online interface sets up customizable horizontal placement of superior/inferior characters on a per font/per baseline character basis*
 - *Automated stacking of superiors and inferiors*
 - *Automated vertical placement of superiors and inferior characters for upsized Pi characters, such as brackets and braces.*
 - *Automated control of min/max optical space above and below division and case fraction rules*
 - *Colored, shaded, and optically padded backgrounds inside equation boxes*
 - *Allows nesting of built up structures within matrices*
 - *Expanded number of procedure data setups and character styles*
 - *General purpose procedures for “Stacking”, or centering, elements over, under, and on the baseline, including such items as “lim” procedures and accented characters.*
- ... and more.

Section I

Formatting of examples in this Manual

Throughout this manual, examples will be shown illustrating how various items were keyboarded in PowerMath. The formatting conventions for these examples are as follows:

Formatting Interactive Keyboarding Examples

Interactive keying often involves the use of “invisible” keystrokes, which involve the modifier keys.

These will appear enclosed in angle brackets, for example `< >`. In these cases, the angle brackets are used for clarity only and are not actually keyed when using PowerMath. The modifiers and their notation are:

<code>< ></code>	The “Command” or “Apple” key.
<code><sh></code>	Shift
<code><opt></code>	Option
<code><ctl></code>	Control

When used in combination, the modifier keys are enclosed together in one set of angle brackets, and are separated by hyphens, as:

<code>< -sh-opt></code>	Command, shift, and option pressed simultaneously
-------------------------------	---

Sometimes a key which would normally be used to access a character is pressed simultaneously with the `< >` key, and possibly other modifiers, in order to access a control function, as:

<code>< -opt-d></code>	Command, option, and ‘d’ pressed simultaneously
------------------------------	---

Standard Keyboard Examples

Other single keys or functions enclosed in angle brackets are:

<code><ret></code>	The Return key
<code><tab></code>	The tab key
<code><ok></code>	Click on the OK button in a dialog box
<code><cancel></code>	Click on the Cancel button in a dialog box
<code><ctl-rArr></code>	Control and “right arrow” pressed simultaneously
<code><ctl-lArr></code>	Control and “left arrow” pressed simultaneously
<code><ctl-uArr></code>	Control and “up arrow” pressed simultaneously
<code><ctl-dArr></code>	Control and “down arrow” pressed simultaneously

Complete keyboarding sequences are formatted in helvetica bold, as:

`x=< -d>a+b<ret>c+d<ret><ok>`

Formatting Ascii Keyboarding Examples

The Ascii coding language involves frequent use of curly braces, brackets and other symbols. These are always “literal”, and never represent control combinations like those which are used in formatting interactive keyboarding. In the following Ascii coding sequence, *all* characters are actually keyed:

`x=*frac*{a+b}{c+d}`

Note: The the Ascii language is *case sensitive*. It is very important to keep this in mind, as you will see.

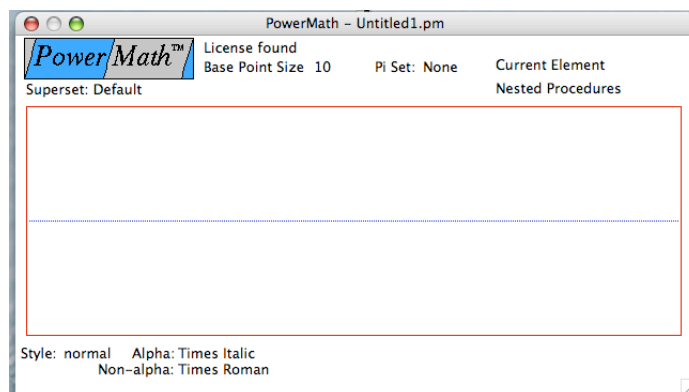
Cross Platform use of ASCII language

Note: The ASCII language can be keyed from any source. When keying on a PC

Section I

machine, conversion to a Macintosh is mostly straight across. However, some font characters are placed differently in the Windows ASCII table from the Macintosh Font table. Smart quotes, for example, do not have the same ASCII number cross platform. You should be prepared to test some keyboard characters before converting PC keyed ASCII into PowerMath.

*Interactive
Dialog Boxes
(An Overview)*



A brief description of a PowerMath editing window will simplify explanations later on in the manual.

When you open a PowerMath editing window (from the standalone app, or from the InDesign plugin in a document) you should see this dialog box.

The version of PowerMath

you are using will appear at the top of the window.

You can see if the dongle is present on the system or if you are running in demo mode through the Eve3™ Key number. If no licensing key is present, the program runs in demo mode.

Base Point Size

The Base Point size of the active equation is shown directly to the right of the PowerMath logo. If the preference for Base Point size is set to auto, that number will reflect the paragraph style in effect when PowerMath was launched. For example, if you are setting your display math at 10/12, but examples are set at 9/11, PowerMath will make that change based on the document Style Sheet point size.

Eve3™ KEY Number 18B PRESENT
Base Point Size 12.0 Pi Set: None

Pi Set

Directly to the right of the Base Point size is shown the current *Pi Set* mapping. *Pi set precedence codes* are the keyboard way of accessing sets of remapped keyboard characters. PowerMath enables four levels (level-1, -3, -5, -7) in addition to normal for character mapping. Default is None. This means that most characters are literal Apple keyboard characters. You key an “a” you get an “a”. If for example you key < -1> for level 1, you activate the keyboard to the mapping of that level. In the default superset, key an “a” in level one and you get a right arrow “→”. Pi character mapping is explained later in the manual.

Note: When the Pi Set Precedence is set on 1, 3, 5, or 7, that will affect only the next keyboard stroke. Subsequently, the Pi Set will reset to “None”.

Note: Some keystrokes have been “Auto Mapped”, which means these will automatically remap to Pi Set 1 characters without any need for a precedence code. “Auto Mapping” is customizable by the user, and will be explained later in the manual.

Superset

Beneath the PowerMath logo is the listing of the active Superset. Only one Superset (typographic preferences) can be active in a document at any given time. Attempts to mix Supersets in a document will be defeated by subsequent updates to equations.

*Current
Element/Nested
Procedures*

At the top right hand corner of the PowerMath equation box, you will be able to see what procedures have been begun and nested. Starting from left to right, the first procedure is listed followed by next. For example, if you have a radical in the numerator of a fraction which is in the top limit of a summation, the window will appear as shown to the left.

DEMI
Pi Set: 1
Radical Character
 $\Sigma \pm \sqrt{}$

Style: Normal Alpha: CoreTTI (Plain)
Non-alpha: TimesTen Roman (Plain)

Section I

When you satisfy (end) the procedure, it will disappear from the Nested Procedure window.

Active Style

At the bottom of the PowerMath window, the active style is listed. The default setup is “Normal”. The window will show the font used for Alpha and Non-Alpha characters. When a style change has been made, it will remain until changed again.

II. System Setup

This section describes what you need to do initially to set up your system in order to use PowerMath.

The program now consists of two main components:

1. InDesign plugins: “PowerMathCS3.InDesignPlugin” for Indesign CS3, and “PowerMathCS4.InDesignPlugin” for CS4. These should be manually copied to the "Plug-Ins" folders for InDesign CS3 and CS4.
2. A standalone application: “PowerMathApp”. First, mount “PowerMathID.dmg”, then double click on "PowerMath.mpkg". Follow the instructions in order to install "PowerMathApp" and associated packages.

Download the “PowerMath Support Files” (PowerMath manual, fonts, font metrics, supersets, superset manual) from:

http://www.phsoftware.com/pages/powerMath_ordering.html

Using the standalone application alone

Many program functions (including editing and copying preference "supersets", and creating, editing, and storing PowerMath equations in various data formats) can be done from the standalone application, without starting up InDesign. Most of the program functions described in the pages that follow will be the same whether the PowerMath application is called directly from the finder outside of any host program, or from a plugin within a host program such as Adobe InDesign.

Calling the application directly from the finder can be very useful for quick and easy access to data structures such as supersets and equations stored as libraries in various formats, or simply for learning PowerMath, without the overhead, learning curve, or expense of host programs during training and initial keyboarding.

The standalone application interfaces to InDesign documents through the PowerMath plugin. New equations, or equations previously stored in the document or directly on disk by the standalone, can be accessed by calling the standalone application through the PowerMath palette or menu from InDesign.

Fonts

PowerMath, as shipped, uses a number of custom fonts (packaged with the product,). However, the user is in no way constrained to use all or any of these fonts, as will be pointed out in the sections which discuss data setups. Obviously, whichever fonts are chosen by the user must be made available from the his font archives on site.

The “Dongle”

The full production version of PowerMath is protected by a protection key, also known as an “Eve-3 Key” or simply as a “Dongle”. To install the Eve-3 Key, plug the Key into any USB port, on a computer on which the PowerMath installer has been run.

“Demo” Mode

Demo mode allows the creation of “Demo Equations”, which appear on screen in the host document with a gray background and which print lightly shaded. Please note that Demo equations cannot be converted to production equations.

Ascii input in Demo mode

The PowerMath plugin must be installed at least in Demo mode (freeware) in order to view and print equations created in an InDesign document by PowerMath / OS X.

PowerMath also features an Ascii-coded text conversion capability which can be seen using the program in demo mode, although equations created this way without the dongle will also be Demo Equations.

Section II

*Setup Is Complete
(for Single Station)*

If you have purchased a single-station Key, you can now start up Indesign if desired, and run PowerMath in full production mode on the machine which has the Key.

[In development, availability to be announced]

The Network Server

Section II

III- Basic Concepts

The PowerMath program uses a fundamentally new approach to the creation and editing of mathematical structures in an InDesign document. While other programs have tried either to use the text capabilities supplied by InDesign to plugin developers, or to create and import external files such as EPS files, PowerMath employs its own text editing and typographic capabilities and then places its product directly into the InDesign document. In this way, the developers were free to create equation composition functionality which goes far beyond that supplied to developers by general document producing programs, while eliminating the need for the creation and importation of separate “art” files for each equation.

The concepts which are briefly mentioned here are essential to an understanding of the nature and purpose of PowerMath, and they will be explained in detail later in this manual.

Interactive Editor

New formulas can be created and existing ones edited in the fully interactive main editing window. This window shares some of the text parameters which are available within the InDesign document and it is the starting point for accessing most of the capabilities of the PowerMath program.

ASCII Import/Export

Equations can also be keyed offline as encoded Ascii strings, as part of an externally keyed Ascii document. Once the document has been imported into InDesign, all or any number of these equations can be automatically transformed into interactive PowerMath equations by selecting ‘PowerMath™ ASCII Import’ from the InDesign PowerMath palette.

PowerMath ASCII 4.0

The Ascii import/export language was completely revised for PowerMath 4.0. It is much more readable and familiar to mathematics compositors (somewhat resembling TeX). It features a “tagging”, or naming, capability which allows users and third party developers to customize access to data setups for composition procedures and for automated “mapping” to Pi Characters.

MathMonarch™

WestWords Inc. (of Logan Utah) has undertaken a development effort which has produced a capability to batch migrate MathType equations from MSWord into PowerMath in a host document, and back again as a fully automated process. The program which does this, owned and available for sale through WestWords, is called MathMonarch.

Base Point Size

All elements of PowerMath equations are scaled relative to one constant value, namely the Base Point Size. Character sizes and leading, pi character parameters such as size, spacing, baseline deflection and overbar weights, and vertical movement for procedural elements such as those for buildups and limits are all done relative to the Base Point Size. In this way, all or any part of math set with PowerMath can be rescaled for any reason (such as a change in the client’s overall spec or for copying and pasting to footnotes) simply by adjusting this one constant value.

Level Offset Leading

Levels are the basic units for vertical placement of the elements of built up equations in PowerMath. In an equation containing a 3-line buildup, for example, the numerator of the buildup is placed on level +1, the denominator is on level -1, and the main baseline of the equation is said to be on the zero level. Similarly, for a 3-line summation, the upper limit is automatically set as a superior to level +1 while the lower limit is automatically set as an inferior to level -1. This vertical placement value is specified as a percentage of the Base Point Size and the default is 70 percent (or 7 points for 10 point math) per level.

Section III

Character Styles ("Alternate" Styles)

Styles can be created and assigned to equation text elements. These allow for scaling, font assignment and face assignment with both alpha and non-alpha setups within each style. Once created, these styles can be accessed through keyboard shortcuts (or by their assigned "tags") and they are globally editable.

Composition Procedures

Mathematical structures (such as buildups, matrices, radicals, case fractions, and summations or integrals with limits,) are typeset in PowerMath using Procedures, all of which work basically the same way. The structure is initiated by menu selection or keyboard shortcut, and as each content element (such as the numerator or denominator of a division) is keyed, the return key is used to step to the next content element or eventually to step out of the structure.

Procedures Options

Values describing such things as sizing, vertical movement, or rule weights can be assigned by the user for the various PowerMath Procedures. "Data sets" of such values are available to most procedures ("Procedures Options") so that the user need only select indexes to these sets (or their assigned "tags") in order to select whichever customized setup for a given procedure is most appropriate for a specific purpose. These are also globally editable.

Automated Mapping to Pi Characters

Pi characters are accessed by mapping the keyable input characters to format-like pi character setups which include font access, character scaling, space left and right, placement of overbars for radicals, and limit placement for summations, integrals, and upsized brackets and braces. Eight Pi precedence codes are used so that the number of these remapped setups available at any one time is eight times the 256 keyable input characters, for a total of 2048. Access to these globally editable mappings is available from the keyboard or through their assigned tags.

The Supersets

The three kinds of data setups mentioned above (character styles, procedures options, and mapping to Pi characters) taken as a whole can be managed as Supersets, and any number of these supersets can be created and stored as files by the PowerMath user. One superset is assigned to, and becomes a part of, each document. Any superset can be reused in any number of documents and the superset which is current to a document can be replaced by any other superset at any time. A complete explanation of this most important concept can be found later in this manual.

Global Updates

All, or any of variously definable subsets, of the PowerMath equations in a document can be updated automatically to reflect the latest revisions to elements of the Supersets. This is a key feature of the program, and it provides an important production efficiency.

The Equation Box

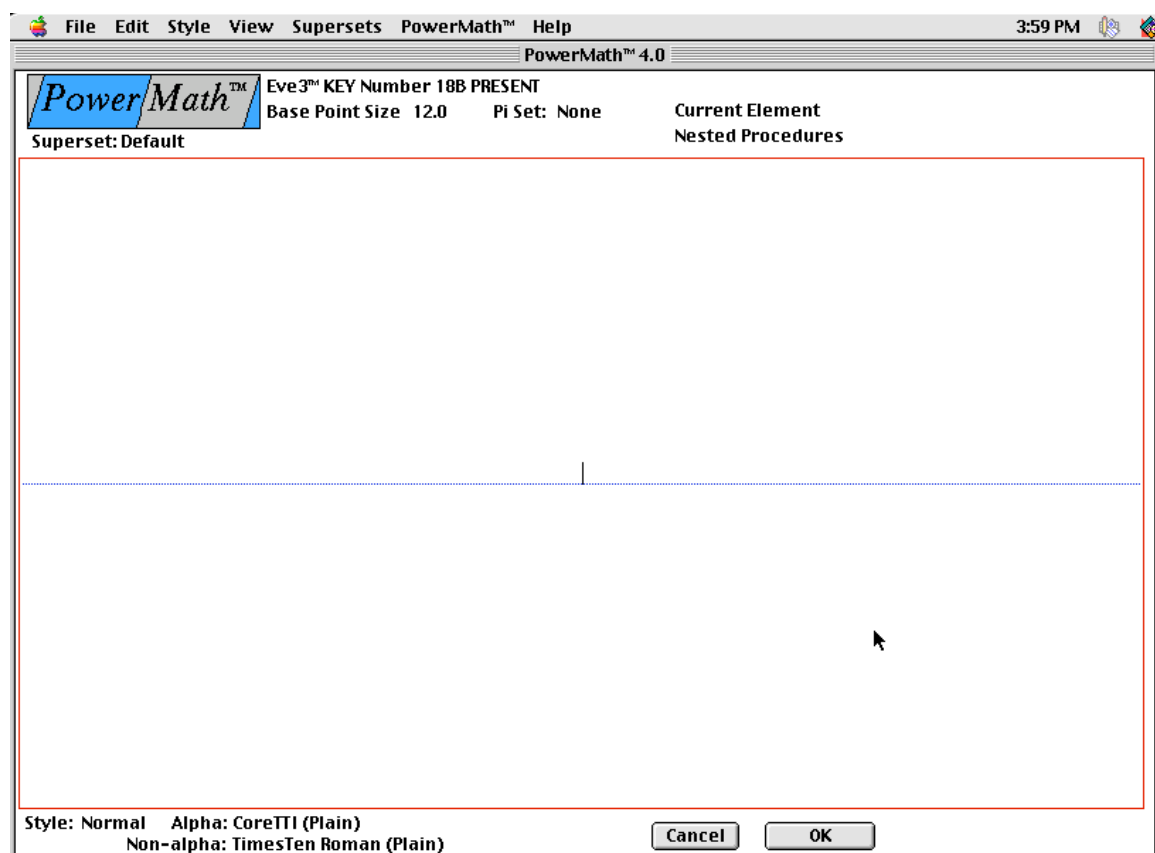
Whenever PowerMath puts an equation into the InDesign document, it does so by placing it within an equation box, which behaves like an anchored text or picture box within the document. Basically, the plug-in automatically sizes this box to precisely contain the new or edited equation, with some further sizing options available to the user. Vertical spacing above and below the box will also be automatically adjusted. The equation box also features user-specifiable parameters such as background color and opacity, and internal "pad space" around the equation.

IV. The Main Editing Window

This anchored box is automatically inserted into the InDesign text box wherever the cursor was at the time that PowerMath was called, with the main equation baseline, the top equation baseline, or the bottom equation baseline matching the current text baseline.

The PowerMath Main Editing Window is used to create equations interactively, and also to interactively edit existing equations. This is the main access point to the PowerMath typesetting and editing software.

You can access the editing window for creation of new equations by choosing “New



Equation” from the InDesign PowerMath palette, or for editing existing equations by selecting the equation either with the selection tool or the text tool, or by placing the cursor immediately to the right of the equation box in the InDesign document, and choosing “Edit Equation” from the palette.

Equation Baseline

The line across the center of the editing window shows the baseline of the equation. This sits on the same baseline the cursor was on in the InDesign document when PowerMath was called.

The Message Fields

Superset: Shows the name of the master data setup file, the “Superset”, which is currently loaded for this document.

Eve3 Key: Shows the presence of the “Dongle”, locally or on the network.

NOTE: You can click on the “Eve3 Key” field to see when your entitlement to pre-paid software upgrades expires. Purchasers of Software Subscriptions (see Ordering Information) will see this field set to one year from date of purchase.

Base Point Size: Shows the overall scaling constant in effect for this equation.

Section III

Pi Set: When one of the 8 Pi Precedence codes has been keyed and is pending for the next character to be keyed, this will show 1, 3, 5, 7, opt+1, opt+2, opt+3, Or opt+7.

Nested Procedures: When a mathematical structure (such as a buildup or a radical) is being typeset using a PowerMath Procedure, the procedure(s) currently in effect are noted here by symbols.

Current Element: When a PowerMath Procedure is being used, the part of the structure which is currently being set (such as the numerator of a buildup) is noted here.

Style: Shows the name of the character style assigned to the next character to be keyed. Note that font assignments within styles can be specified as automatically split for Alpha vs. Non-alpha characters.

*Moving and Sizing
the Editing Window*

You can click on the title bar and drag the editing window to a new position in the usual way. Sizing is also somewhat standard, just click in the box in the lower left corner and drag to resize the editing window. The window cannot, however, be made small enough to obscure the essential controls. The size and position of this window are saved in the PowerMath Preferences file, so that they will be remembered the next time the window is opened.

*Call Editor for
"Create Equation"*

The editor is called from the InDesign document for the creation of equations by choosing "New Equation" from the InDesign palette.

Important: You must have InDesign in "text" mode, with the text tool selected. in order to create an equation.

*Set InDesign to
"Content" mode*

Important: You must have InDesign in "Select" mode, with the selector arrow tool in effect and an existing equation selected, in order to edit an equation, or place the text tool just to the right of the equation.

*Keyboard Math
Characters*

Once the editing window has opened, the user can simply begin keyboarding characters. Default parameters and type characteristics which are appropriate for mathematics typesetting will be automatically accessed. Let's take a look (in the left margin) at our first example.

Example 4.1:

$$2x = y$$

Keyboarding Sequence: **2x=y<ok>**

*Enhanced Automatic
Formatting*

Notice that although only *content* was keyboarded, *formatting* was automatic. The numeric character is roman, the alpha "variables" are italic, the equal sign came from a Pi font, and space was inserted to the left and right of the equal sign.

While this automatic formatting is usual for equation editors, the degree of control given to the PowerMath user over customization of automated formatting is most *unusual*. As we shall see, PowerMath provides the ability to choose any character from any font to be automatically accessed in any situation, with fully automated control over scaling, baseline deflection, space left and right, and association of characters with procedures (such as a "Sigma" character with a summation procedure).

Global Editability

Furthermore, however many hundreds or thousands of times these characters and parameters may be accessed in a math document, the entire document (or variously selectable parts of it) can be transformed in a single "batch" global edit. A full discussion of these capabilities will be provided in the chapters which follow.

Store the equation

The last action taken in any PowerMath keyboarding sequence is **<ok>**, clicking on the "OK" button in order to store the equation back into the InDesign text box. This results in the creation of a "user box" which contains the equation. A user box is a

Section IV

special entity provided by InDesign to plug-in developers. It functions in the document as an anchored box, a lot like an anchored text or picture box. This equation box will be the subject of the next section.

Edit an equation

You can edit a PowerMath equation simply by selecting it and calling the edit function from the palette. This will cause the main editing window to reopen, with the equation still fully constructed and ready to be interactively edited. Upon completion of edits, click on **<ok>** to store the equation back into InDesign.

Using the Editor

Straightforward editing of simple character strings in the PowerMath editor is done in the usual way, using the “Delete” key, and “Cut”, “Copy”, and “Paste” from the “Edit” menu. Note, however, that Powermath data can only be copied and pasted while the editing window remains open. Parts of equations cannot be copied from one equation and subsequently pasted into another when the editing window is reopened.

Editing/deleting structures

Editing of mathematical structures which have been created by PowerMath Procedures is more complex. This is because the program inserts markers into the text stream which delineate parts of structures, such as numerator, rule, and denominator of a built-up fraction. These markers must be protected from normal deletion so that the equation is never left in the meaningless state of having some, but not a complete set, of these markers present for any such structure. For this reason, the program only allows procedures to be selected for cut, copy, or paste in their entirety. This is done by clicking anywhere within a procedure and keying **<sh-uArr>**, which results in selection of the entire procedure (markers and text contents) for editing. When structures are nested (as for a division over a division), you can click inside a sub-structure, key **<sh-uArr>** to select just that part, and optionally key **<sh-uArr>** again to select the whole nested structure.

The View/Scale Factor

When editing an equation in the main editing window, it is possible to enlarge or reduce the viewing “Scale” factor by choosing “View/Scale” from the PowerMath menu bar. The number chosen is a percentage of the actual print size. Specifying this factor here sets it for the current editing session only, and it will return to the default scale factor the next time the editing window is opened. This item is also available under the “General Preferences” dialog, shown in the next section, and it is here that the default scale factor is “permanently” set.

NOTE: Changing the viewing scale factor has no effect on the actual print size of any equation.

We will return to the editing window later in this manual, and take a detailed look at specifying type parameters generally, as well as at accessing and modifying the main procedures and data structures of the program. First, however, we’ll try to gain a more complete understanding of how the equation box itself relates to the InDesign document, and what controls over the equation box are available to the user.

V. The Equation Box

Let's just review what we know so far about the equation box which contains a PowerMath equation in an InDesign text box. The box itself is a special kind of entity called an equation box, created by our InDesign plug-in, which functions in a document much like an anchored text or picture box.

The left side of the box is positioned in the InDesign text box at the current insertion point (the position of the cursor when PowerMath is called), and it is at this position that the main baseline of the equation within the user box is almost always (see "Set Baseline Shift to Zero") placed. But what determines the size of the box, and what mechanism allows for vertical space above and below the equation in the document? What related options are available to the user?

The basic strategy employed by PowerMath to size equation boxes is to utilize custom-integrated automated derivation of font metrics data in order to precisely contain the equation. This means that the box is drawn to precisely clear the highest ascender on the highest baseline level, the lowest descender, and the leftmost and rightmost characters (including removal of any negative sidebearings so that nothing gets "clipped").

The basic strategy here is to increase the current line leading by the number of levels up for built up structures, multiplied by the "level offset leading" (See "III. Basic Concepts" for an explanation of level offset leading). Similarly, the current "space after" is increased by the number of levels down, again multiplied by the level offset leading.

Beyond these strategies, a number of important options are offered to the user in the "Equation Box" dialog, which is accessed from the PowerMath "Style" menu.

Sizing the Equation Box

It is important to realize that it matters if the equation appears "clipped" on the screen. When printing, the equations will print clipped exactly as they might be on the screen if the box is resized.

Color, Shade, and "Pad" the Equation Box

The design of many textbooks requires that some standalone equations appear with a colored background, and with that background optically spaced evenly around the equation. As you can see from the options dialog above, PowerMath provides an easy way to accomplish this.

Example 5.1:


$$2x = y$$

Example 5.1 was created using this new option. Magenta was selected from the "Box Color" menu, with a shade value of 50% and an optical pad space value of 20 points. Color management in this situation is handled directly by InDesign.

Note: By making use of our font metrics interface, PowerMath is able to keep the pad space constant no matter what characters or structures are added to or taken away from the equation.

Vertical Spacing Options

Some additional vertical spacing options are available to control how the equation baseline sits relative to the current insertion point in the text box, and to allow some extra vertical spacing above and below oversized characters.

Section V

Auto Baseline Shift options

Example 5.2:

$$\text{insertion B} \begin{array}{r} 1 \\ - 2 \end{array} \begin{array}{r} - 3 \\ 6 \end{array} \text{R}$$

Example 5.3:

$$\text{insertion B} \begin{array}{r} 1 \\ - 2 \end{array} \begin{array}{r} - 3 \\ 6 \end{array} \text{R}$$

AutoLead at Ascender

Auto Space After at Descender

“Zero” value turns it off

PowerMath to PICT Converter

PICT Range Selector

Multilevel Base Align to Surrounding Text

- ☒ Mainline Base Align
- ☐ Bottom Level Base Align
- ☐ Top Level Base Align

☐ Base Offset Brackets on Auto Ups

☐ Autoleading off

AutoLead at Ascender

Auto SpaceAfter at Descender

Default Pad Space

Radio buttons which control baseline shift for equation boxes are now available in the equation box dialog for the current equation, and also in the general preferences dialog as a “permanent” option.

Normally, the equation box will be baseline shifted so that a built-up structure will keep its main baseline at the same vertical position as the baseline at the current insertion point in the document, as in example 5.2. When Bottom Level Base Align is selected, no baseline shift takes place, so that the equation is set as in example 5.3

Two additional fields in the Supersets/General Preferences dialog are used to allow for vertical spacing of equation boxes in the document, this time allowing space for the ascenders and descenders of oversize characters in an equation. These fields can be thought of as a “tolerance” amount for ascenders and descenders which, if exceeded, will result in extra lead.

The Autolead at Ascender field should be set to a reasonable value for a character ascender in 10 point type, for example 8 points. In this case, extra space will be allocated for whichever (if any) ascender rises most above 8 points from the baseline on the highest level of any structure in the equation. For example, a bracket which rises 10 points from the baseline of the top row in example 5.2 would cause an extra 2 points of lead for the equation line.

Similarly, the AutoSpaceAfter at Descender field should be set to a reasonable value for a character descender in 10 point type, for example 4 points. In this case, extra space will be allocated for whichever (if any) descender drops most below 4 points from the baseline on the lowest level of any structure in the equation. For example, a bracket which drops 6 points from the baseline of the bottom row in example 5.2 would cause an extra 2 points of lead for the line following the equation.

NOTE: These tolerances are only enabled when the paragraph format for the equation line specifies an actual lead amount, *not* “Autolead”!

A value of zero in these tolerance fields is taken to be a special case which signifies “do nothing”. In other words, a “zero” entry turns this feature off.

Whenever a user wants to convert the unique PowerMath equation data formats which are drawn in “user boxes” to a more standard format, he can access “PowerMath to PICT Converter” [IN PROGRESS, AVAILABILITY TO BE ANNOUNCED]. He may, for example, want to do this prior to converting a InDesign document to HTML for web publishing.

This is a one-way process. Once this conversion has been done, equations cannot be converted back to editable or globally updatable PowerMath equations.

The process allows for selection of some or all of the equations in the document for batch processing. When it is first called, the following selector dialog is presented:

After clicking on the appropriate option for range selection, the filenames options are

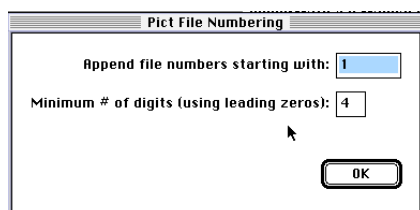
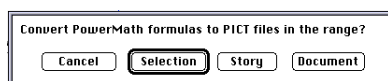
Section V

*Sequential
Numbering
Appended to PICT
filenames*

*Root Filename and
Storage Location for
this batch of PICT
Files*

presented. The program will have to create a “PICT” file for each equation, which will be named and stored in an appropriate location by the user.

Filenames are generated automatically. Prior to selecting the root filename (see below), the program asks for numbers to be appended to it sequentially as equations in the current range are batch processed into PICT files. Once these selections have been made, the root filename and storage location (usually separate chapter folders) can be selected.

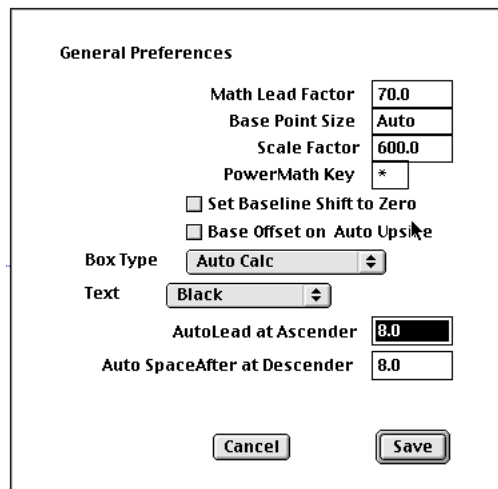


VI. Basic Specification of Type

*Base Point Size
default to “Auto”*

We pointed out earlier (III. **Basic Concepts**) that the most basic of all type parameters in PowerMath is the Base Point Size, because this is the one constant to which all relative values are scaled. No more than one base point size can be in effect for any one equation. There are three ways in which this value can be specified.

The first option is to call the General Preferences dialog under Supersets/Edit Preferences and enter “Auto” in the Base Point Size edit text field. This sets a default preference for all subsequently created equations, such that the base point size will automatically be set to the point size at the current insertion point in InDesign when the equation is created. This seems to be the most useful default preference, as it automatically sets up equations to match type sizes in main text, exercise sections, footnotes, etc.



*Base Point Size
default to absolute
value*

The second option is also to call the General Preferences dialog under Supersets/Edit Preferences, but this time to enter an actual value in the Base Point Size edit text field. This will result in the same base point size for all subsequently created equations, no matter the current InDesign point size, until this default is changed again.

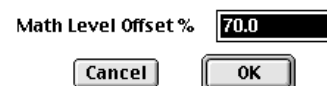
*Override the Base
Point Size default*

The third option is to override the default value for a specific equation by calling the Base Point Size dialog from the PowerMath “Style” menu while the main editing window is open for that equation.



*Set the Math Level
Offset*

Another general typographic preference is the Math Level Offset (III. **Basic Concepts**). This also can be specified as an overall default in the General Preferences dialog, or as a one-time override from the PowerMath “Style” menu while the main editing window is open.



Font, Face, and Scale

Most of the time, the basic typographic parameters of font, face, and scale (relative to the base point size) will be accessed automatically by means of “Style” calls, which will be discussed in the next section. Sometimes, however, automatic style access is disabled by the user so that these parameters can be selected individually. Style access is disabled by choosing “No Style” from the PowerMath “Style/Alt Style” menu.

NOTE: Access to the individual parameters of font, face, and scale under the Powermath “Style” menu is only enabled when styles have been turned off by selecting “No Style”. Otherwise, these menu items are “grayed out” because they would conflict with the parameters which are automatically enabled by styles.

Text Color

The “Color” menu item is always available because color is not currently part of the automatic Style parameter set. This item is meant to select text color only. Color for various kinds of rules is selectable under the options setups for procedures which use rules (see the section on accessing the main data setups).

VII. Accessing the Supersets

*PowerMath™
SuperSets folder*

Legacy documents
and Supersets

This section describes where the superset files (**III. Basic Concepts**) are stored on your system and how they are associated with a InDesign document. The next section will describe how the superset main data setups for styles, pi characters, and procedures for composing mathematical structures are accessed and modified.

The supersets are stored in a “PowerMath Super Sets” folder inside the Preferences folder, which is located in the user’s library folder. If you launch InDesign with PowerMath.pln installed, the program will load supersets from the PowerMath SuperSets folder.

Legacy documents and supersets from PowerMath for OS 8/9 are fully supported in PowerMath for OS X: legacy equations in Ascii format and legacy supersets “as is” without modification.

Custom Pi fonts for oversized glyphs which were used very extensively throughout these legacy books and documents are also currently supported by PowerMath (uniquely among related applications, we believe) in OS X, including drawing and accurate derivation of any and all individual oversized glyph metrics, with no special concerns or workarounds.

Section VII

VIII. Accessing the Main Data Setups from the Superset" (A General Overview)

The PowerMath “supersets” include sets of typographic styles, automatically mapped pi characters, and composition procedures which are oriented toward the construction of complex mathematical structures (**III. Basic Concepts**). Prior to PowerMath 4.0, the elements of these sets were accessed simply by choosing a numeric index to whichever set was to be selected or, in the case of Pi characters, by choosing one of four precedence codes followed by an “input character” which had been set up to map to a pi character data setup. Although the various data setups can still be accessed in the old way, mostly for compatibility with the old Ascii language and for the convenience of experienced PowerMath users, there were a number of compelling reasons to look for a better access method:

- *It was hard to remember index numbers, precedence codes, and arbitrarily chosen “input characters”. It took too long to learn, and it was too easy to forget where everything was.*
- *Referring to things in this cryptic way leads to the evolution of an Ascii coding language which is cumbersome and difficult to read.*
- *The user of the Ascii coding language had to know not only what he wanted, he also had to know where everything was in a specific implementation of a superset. This meant that any given instance of Ascii input, whether typed by a keyboard or generated by a translation program, was tied to a specific implementation of a superset, and would almost certainly not work with any other superset.*

“Tags”: naming the elements

A much better way was implemented as of PowerMath 4.0 in OS 9. A user interface was developed which allowed the designer of a superset to assign names, or “Tags”, to the stored elements of that superset. This single step leads to the elimination of the problems listed above:

- *Tags can be mnemonic, easy to learn and easy to remember.*
- *Mnemonic tags in conjunction with braces delimiting command “scope” have led to an Ascii coding language which is clear, readable and very easy to learn.*
- *Availability of tags leads to naming conventions for the elements of a superset. These names can be made to resemble the tags and keywords used in external standards such as TeX. Now a given instance of Ascii input can be expected to work across a wide range of well-designed supersets, allowing the code generator to refer to what is wanted without regard to where data elements were actually assigned.*
- *PowerMath is now much better suited to be interfaced to translation software for batch export/import of equations to and from external document standards such as MSWord and TeX, and this has in fact been done as we shall see later.*

Accessing Procedures

This section will show you how to access the data setups in order to use mathematical composition Procedures, character styles (“Alternate Styles”), and mappings to Pi Characters. At the same time, the actual design of supersets, and the assignment of tags along with the actual parameters of the various data setups, will be discussed. We will begin with an overview of using the tags interactively, and then we’ll take a closer look at the new dialogs for accessing the procedures, style, and pi character options.

The use of tags in Ascii coding will be explained in a later section. All procedures, both old and new, can still be invoked using the existing system of keyboard shortcuts, which can be seen under the “PowerMath/Procedures” menus.

The various data setups for the procedures can still be accessed by calling up the “PowerMath/Procedures Options” dialogs and tabbing to the “Index” field to access

Section VIII

an option by its indexed position. Note that, as in the old program, a consistent relationship between the cmd-key for invoking the procedure and the cmd-key for calling a new option for that procedure is attempted. If division is < -d>, then its options are accessed as < -opt-d>.

NOTE: In the old program, an option was called and then the procedure was invoked as a separate step. Now, setting the option also invokes the procedure.

Calling an “options” dialog from the “PowerMath/Procedures Options” dialog does not allow changes to be made to the options. The “Edit” button is always grayed out. In order to make these changes, the dialogs must be called from the Supersets/Edit Procedures menu. The cmd-key relationship is still maintained, only now the shift key must also be pressed, as in < -sh-opt-d>. There are a number of reasons for separating editing from calling:

- *It's important to segregate the superset editing function. This should only be done in an orderly fashion by authorized administrators.*
- *The Edit Procedures dialogs may soon be password protected.*
- *When changes are not required, it's considerably faster to call the procedures options dialogs because they do not have to load system resources like font and color menus. Such parameters are merely displayed.*

Tagging

Looking at the new “Tagging” feature, using the Math Rule dialog as an example we first open the Selector window. Key < -opt-y>.

Note that there is a new scrolling list of tags, and just above it an edit text “hot-key” field called “Tag”. You can either **double click** on a tag in the list, or start keying a tag name in the tag field.

Partial matches to any items in the list will cause the scrolling list to go to the item as you key. This helps make tag selection fast and easy, and it eliminates any need to take your hands off the keyboard to reach for a mouse.

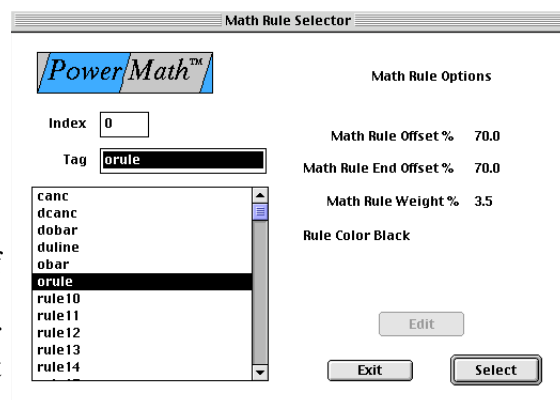
As you locate a tag, the parameter display in the dialog changes, so you can see what options you are selecting. Click <Select> or press <ret> to select the option associated with the tag *and* to invoke the procedure under the new set of options.

Note also that the old “Index” field is still available if you prefer to locate options that way.

There are now twenty options available per procedure, instead of the old ten.

Unused option sets, or those for which no mnemonic tag has yet been designed, have default tags assigned by the program. Every option and pi character has a tag, either assigned or default!

As noted, the “Edit” button is grayed out because this is an access dialog only.



Section VIII

Improved Procedures

Some previously existing procedures have been enhanced as follows:

Case Fraction

The case fraction options now include a field for optical space over/under rule. This space, if a non-zero value is specified, is unconditionally maintained by PowerMath. The Atsui (and a 3rd party) interfaces are used to gain the information needed here.

Built-up Divisions

The division (buildup) procedure has been similarly enhanced, but in a more complex way. A *range* of acceptable optical space can be defined by specifying min/max space above, as well as below, the division rule. This eliminates many problem situations in buildups that required manual intervention (“nudging”) in the older versions.

Styles

The number of styles has been increased to ten, and tagging enables calling any of the ten from the new Ascii language, as well as calling any interactively without having to use the mouse.

Matrix

A significant improvement to the matrix procedure in version 3.0 of PowerMath was the introduction of “Computed” column widths, which automatically set the width of any column to the widest element it contained. Just click on computed columns in the options dialog, and in this case the column width field functions as a gutter width field.

An essential improvement to the matrix function is that you can now nest multi-level built-up procedures like division and summation with limits within a matrix. You can also key across/down or down/across.

Superior/Inferior

Superiors and Inferiors will now stack automatically when called sequentially, with no intervening full-size character (a feature that works more like TeX). In example 8.1, key **v< g>2< =>< u>0**

Example 8.1

v_0^2

If you do not wish to stack the superior and inferiors, place a space between the superior and inferior callouts.

The Stacking Procedure

This procedure, through its options, allows vertical stacking and horizontal aligning of all three of, or any two of:

a main baseline element

an “over” element

an “under” element

Example 8.2:

$\lim_{x \rightarrow \infty}$

Example 8.2 used the stack option with the tag “lim” to set up a procedure which actually used to be quite tedious to do in PowerMath. (The “roman” style tag and the “rarr” pi character tag were also used). The stacking procedure is very general in nature and can be used very effectively in many situations.

Macro Procedure

Another powerful procedure in PowerMath is macroizing. Any number of macros can be defined. The contents of the macros are keyed in the Ascii language. Each macro can contain up to 256 characters.

Using Macros

Macros can be accessed either interactively, or through the Ascii importation process. Exportation back to Ascii will show the macro content, not the original macro call.

A powerful feature of the macro capability is the “merge” code, which acts as a “transfer input source” code for variable input.

Within the macro itself, the \$ acts as a merge code (literal \$ is keyed as \\$. In Ascii coding, the merge is also a \$. Interactively, the merge code is accessed as **< i>**.

Section VIII

Multiple merge codes can be used within a macro.

NOTE: a macro cannot call itself or any other macro.

A detailed explanation of Macros can be found in Section XI.

IX. Character Pi Interface

One of the most powerful features of PowerMath has always been the Character Pi interface. This interface was designed to enable the user to automatically access a character, along with a range of optional typographic parameters, and to automatically return to the set of parameters which was in effect before this access took place. Each such access is associated with a keyable character, of which there are 256, and with a precedence code, of which there are 4, for a total of 1024 automatic accesses of this kind.

Note: The interface is called the “Pi Interface”, although the accessed character can come from any font and need not be a pi character.

The basic typographic parameters are font, output character, baseline offset, space left and right, and horizontal and vertical scale. Additional parameters are available, depending on the Character Type selection, which indicates the purpose for which a particular setup is intended. Character Type selection can be used to associate a mapping setup with any one of the following:

- *Simply access a character, and indicate how to place superior and inferiors next to it if it is an upsized character.*
- *Associate the character with a “summation” procedure and indicate how to place limits*
- *Associate the character with a “integral” procedure and indicate how to place limits*
- *Associate the character (a radical character) with a “radical” procedure and adjust automatic rule placement if necessary*

PowerMath has simplified the process of creating and choosing pi characters by the use of “tags” and “palettes”.

When the pi interface is first called with `< -opt-j>`, and then the “show palette” button is clicked, the mapped character palette appears as below. Notice also the location of the “Key” popup, presently set to level one.

Accessing the Pi
Character Interface

HOTKEY:
`< -opt-j>`

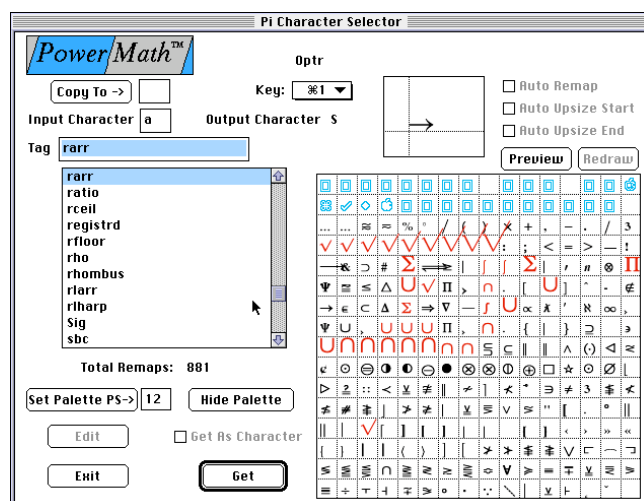
When you look at this palette on a color monitor, you will notice that the entries are color coded. Black characters are mapped to simple character access setups, red characters are mapped to procedural entries such as summation, integration and radical, while the blue outline characters are unmapped and available for mapping by the user.

The palette shows 256 mappings at a time. The “key” popup is used to change the display to any of the four sets of 256 characters.

Show Palette

Clicking on a palette entry causes the corresponding setup to become “current” in the interface. At this point, the “tag” for the current entry also becomes current. It is useful to remember these tags, because access by tag is the easiest and most direct access method. If the “Get” button is now clicked, the character setup will be inserted into the main PowerMath editing window.

If the setup that was chosen in this way was for a procedure, the character will be placed in the editing window and the procedure will be turned on and will be waiting for the operator to complete key-



Section IX

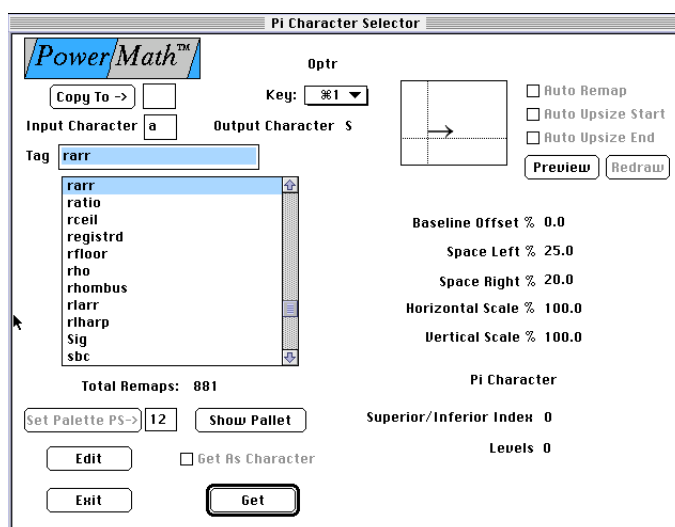
boarding for that procedure. Alternatively, in this case, the the “get as character” check-box will become enabled in the mapping interface and by checking it before clicking on the “get” button, the character can be accessed without turning on the procedure.

Access by tag is just a matter of calling up the dialog and either scrolling to the tag and double clicking on it or of using the “hot keys” entry field just above the scrolling list to get to it. Just hit <ret> when the desired tag is “current”

From the main PowerMath equation window, key < -opt-sh-j>. Essentially, the same window appears except that the <Edit> button is now enabled.

Editing a
Pi Character Setup

The Pi Selector
Window



When you select a tag the preferences for that tag become active. Tags are listed alphabetically in the list window. The current tag is in the *Tag* window.

The *Copy To* button will copy the active tag and its preferences to another keyboard placement in the same level.

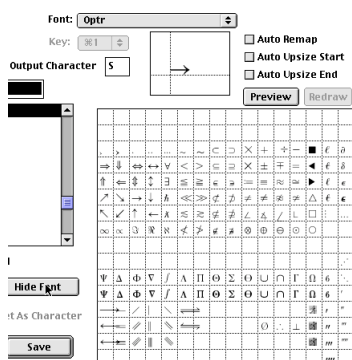
The *Key* window indicates in which pi level the active setup resides.

The *Input Character* window shows the keyboard character which is mapped to the current pi character setup. Our example shows a right

arrow “rarr”. It can be input from the Apple keyboard by accessing pi level 1 and the character “a” Notice that the font the right arrow has been created in is “Optpr” and the output character in Optpr is “s”. We also can see a preview of the character. And just under the scrolling list we can see that we have 881 total mapped characters in our superset.

If we choose <Edit> the preference fields for the active setup become editable. From the *Font* menu we can change the font for the setup we are editing. We can also change the *Output Character*. All changes are reflected in the preview window.

In the editing window, we can see the output font



keyboard layout without leaving PowerMath. If you click the <Show Font> button, the keyboard table for the current font will be displayed. Double clicking on a character in the table will change the output character for the setup we are editing.

When an output character has been chosen, the character preferences can be edited. The fields are interactive with the preview window. Changes made to the preferences will be automatical-

ly reflected.

The Preview Window

Section IX

The Preview window display is proportional to the Base Point Size. All characters previewed reflect a correct percentage of both baseline deflection and horizontal/vertical scaling.

Auto Remap

- ☐ Auto Remap
- ☐ Auto Upsize Start
- ☐ Auto Upsize End

Auto Remap: Some characters are keyed so often that a shortcut method of accessing Pi setups was desirable. For example, the equal sign used in standard math equations is a pi font character. The equal sign created from the keyboard is a text font character. PowerMath has allowed setups mapped in the Pi Character interface to be automatically retrieved from the normal keyboard. The pi equal sign is mapped to the text “=”. When editing the pi setup, if you check the Auto Remap box, then any time you key the associated input character it automatically remaps to that pi setup .

Note: This feature was called *AutoPi* in old versions of PowerMath.

Note: A character that is usually accessed from the text font should not be Auto Remapped

Auto Upsize

Auto Upsize Start: Characters which can expand to encompass more than one level (fences, radicals, etc.) can be created to automatically remap to a larger size when multiple levels must be enclosed. For example, when you create a parenthesis “(” with AutoUpsize Start checked, PowerMath will remap in accordance with the greatest number of levels encountered between that character and a “)” mapped with AutoUpsize End (or between the AutoUpsize Start character and the end of the line).

(1+2)(< -d>1<ret>2<ret>)

Notice that parenthesis access the same initial setup for the linear “1+2” and for the following buildup.

The result will be as follows:

$$(1 + 2)\left(\frac{1}{2}\right)$$

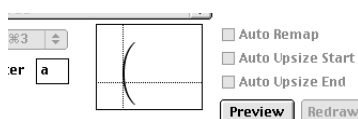
Now, again keying interactively:

[< -r>V< d>1<ret>2<ret><ret>]

the square braces and the radical will automatically upsize to fit the built-up division. The resulting be as follows:

$$\left[\sqrt{\frac{1}{2}}\right]$$

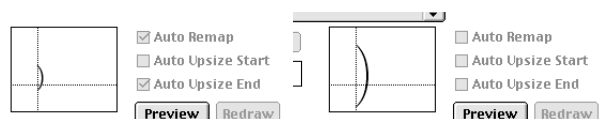
Specification of a character to be an “AutoUpsize” character is always done to the setup on level one, where we start by editing the setup and checking “AutoUpsize Begin”. On level three *for the same input character*, a setup must be



created to fit a three level equation, on level five a five level setup must be created, and the same for level seven with a seven level setup.

If there is a matching end character (fences, etc.), map that character similarly, but check AutoUpsize End.

For example, our level 1 open parenthesis “(” is sized to fit a one level equation. Notice it is



Section IX

also Auto Remapped. The AutoUpsize Start button is checked. The same input character “(“ on level three is set up to fit a three level equation. If the equation following the opening “(“ grows to three levels, PowerMath will automatically access the level three setup. Notice that on level three the Auto Remap and Auto Upsize buttons are grayed and not editable.

Baseline Offset %	0.0
Space Left %	25.0
Space Right %	20.0
Horizontal Scale %	100.0
Vertical Scale %	100.0

Compare the setups for the AutoUpsize End for the close parenthesis “)”.
Auto Upsize End: Ends the containment of levels for AutoUpsizing.

Note: Not all characters created with AutoUpsize Begin require an AutoUpsizeEnd. Radical characters will auto upsize and the nesting will end when the procedure ends.

Other editable fields for a pi character include

Baseline Offset % The vertical baseline deflection of the character as it related to the level baseline. The percentage is of the base point size.

Space Left % The space desired before a character is rendered.

Note: Math Space is 30% of full space.

Space Right % The space desired after a character is rendered.

Note: Math Space is 30% of full space.

Horizontal Scale % Horizontal scale of the character.

Vertical Scale % Vertical scale of the character.

When creating or editing a setup, you may specify an association with a procedure. **These choices are detailed for each type of setup in Section X.** The choices determine placement of limits, number character repetitions, and sup/inf index level. The editing window changes and give different options for each choice.

Character: Pi	Character: Summation
Superior/Inferior Index: 0	Main Character Count: 0
Levels: 0	Levels: 0
Character: Integral	Limits Index: 0
Main Character Count: 0	Character: Radical
Levels: 0	Rule Weight %: 0.0
Lower Limit Kern: 0.0	Rule Offset %: 0.0
Limits Index: 0	Rule Kern %: 0.0

The choices are simple Pi Character, Summation, Integral, and Radical.

To save changes, click **<Save>**. To retrieve the character, **<Get>**. Click **<Exit>** to return to the PowerMath equation window.

Superiors and Inferiors can be accessed from the PowerMath “Special” menu. Under this menu, the “Baseline” item always returns to the main baseline. Once in superior or inferior, nesting can be accomplished by continuing to select superior or inferior again. The “Down Base” item in the special menu steps out of these nesting levels one level at a time.

Superior/Inferior can be vertically “stacked” simply by invoking superior, keying the superior character(s), invoking “baseline”, invoking “inferior”, keying the inferior character(s), and invoking “Baseline”.

Character
Specifications

Optional Association
with Procedures

Superiors and
Inferiors:
Overview

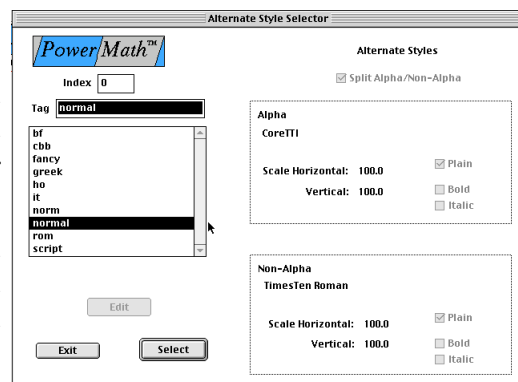
X. Character Styles

Accessing Character Styles

Character Styles set up the font, face, and scale which are automatically accessed at any given time. The PowerMath default superset defines the normal Alpha (alphabet) math font as CoreTTI, which is based on TimesTenItalic with special spacing around troublesome characters (the lowercase “f” and “d”, for example). Non-Alpha characters (such as numbers) are accessed from the TimesTenRoman font. PowerMath has enabled ten Character Styles.

From the the main PowerMath equation window, key < -opt-s> or choose “Style Selector” from the “PowerMath/Procedure Options” menu. This activates the selector window.

Double click on (or key) an existing tag. The preferences for that style will appear in the Alpha and Non-Alpha windows. Choose <Select> to begin the style.



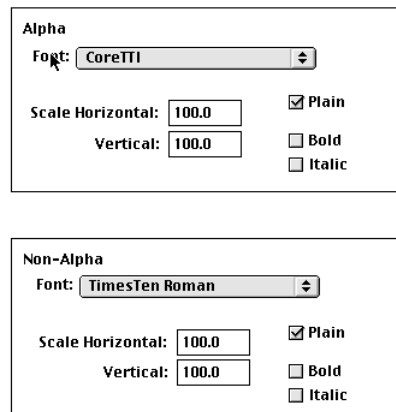
Notice that an index window exists which can still be used to access the styles. Users of older versions of PowerMath will recognize that roman text is still set at index “6” and now is also tagged as “rom”. Boldface roman text is still indexed to level “4” and is also tagged as “bf”. Normal style is set by either index “0” or index “8”. The tags for normal (italic alpha and roman non-alpha) are tagged as both “norm” and “normal”.

Editing Character Styles

To edit a style, key < -opt-sh-s>. The same Alternate Style Selector window is opened, except that now the <Edit> button is active. Click on it to begin editing. If not already done, change the style’s default tag name. If the style attribute will have separate values for alpha and non-alpha characters, check the *Split Alpha/Non-Alpha* button. If the values will be the same, leave the button unchecked.

In the *Alpha* window, choose the font from the *Font* menu. Only the fonts active on the machine will be available. Scale the style based on the Base Point Size. Check the attribute for the style if needed. Plain, bold, and italic are available if the font family does not have a native version of the attribute.

If the *Split Alpha/Non-Alpha* button is not checked, the Non-Alpha window is grayed out. If it is checked, set the values for the non-alpha characters as above.



Choose <Save> to write the changes to the superset. To activate the style, choose <Select>. Choose <Exit> to return to the main PowerMath equation window.

Note: Once a style has been activated, it will remain until changed back or until the end of the equation.

Within PowerMath ASCII delimiters [&&] key the unique tag name within Style delimiters “~~”. The text “from now on” will reflect that style change until changed again or the end of the equation. The ASCII string

[&abc~bf~def~norm~ghi&]

when imported through the PowerMath ASCII filter will become

abc**def**ghi

Accessing Character Styles in Ascii coding

XI. Macros

A useful feature of PowerMath is the ability to set repetitive keystrokes as a macro. A macro is a series of commands which the user sets and controls. For example a common mathematical string such as:

$$\lim_{n \rightarrow \infty}$$

could recur throughout a chapter. Instead of rekeying the entire string, a macro could be written to repeat the process.

Creating a Macro

Inside the PowerMath equation box, key < -**opt-Sh-l**>. This will open the editable Macro window.

Click the <New> button and key in a unique name for your macro. For example “MyLimit”. (Remember that case sensitivity is important.) We tab to the open window and using the ASCII language, key in the expression:

***lim*{~rom~lim~norm~}{x|rarr||inf}}**

NOTE: This section assumes you are using the default superset and that you are familiar with PowerMath Ascii4.0 (see Appendix A).

Choose <Save> and <Exit>. The macro is saved to the superset.

Key < -**opt-l**>, key or click on “MyLimit” and click “Select”. The macro will insert the expression you stored and return control to the operator.

Macros are delimited with exclamation points in the ASCII language.

Inside PowerMath ACSII delimiters [&&], key the string **!MyLimit!**. When the string is imported, PowerMath will read the key strokes inside the existing macro function.

NOTE: When the string is exported, it will not read the macro, but the full string.

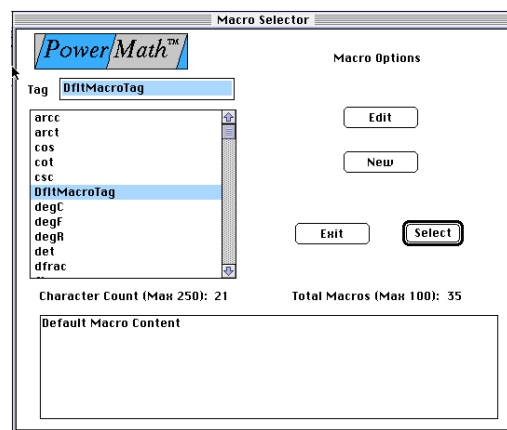
Sometimes we have common expressions with consistent differences in each. For example we might have a series of build ups that are generally repetitive, but different in one or a few specific places. The PowerMath 4.0 macros have enabled a merge function whereby the operator can switch control of input to and from parts of macros both through ASCII filter importation and interactively.

When keying a macro inside the editing window, insertion of a dollar sign signifies that control is to be switched back to the standard input at that point. For example:

***frac*{ab}{c\$}**

When PowerMath executes the macro, it will write “ab” in the numerator, write “c” in the denominator and then return the control of the input to the operator who can now key any character, style, procedure, or function in available in PowerMath.

To return to a pending macro segment, key < -i>. PowerMath will continue executing the macro from the merge code.



Accessing a Macro Interactively

Macros in the PowerMath ASCII Filter

SYNTAX:

!MacroName!

Using the wildcard (merge key) for macros

Using merge codes interactively

Section XI

Using merge codes in ASCII

A dollar sign “\$” keyed following a macro callout instructs PowerMath to return to the macro until it comes across another “\$” or the end of the equation. For example, we have created a macro named “NewMacro” inside the macro editing window. The macro string is `*frac*{pd|b}{pd|$}`. Within PowerMath ASCII delimiters [&&] we key the string `!NewMacro!x$`. When importing, PowerMath will start building the fraction from the macro. When it comes across the \$, it gives control of the equation back to ASCII input until it comes across another “\$” or the end of the equation. In our example, the character “x” would be inserted at the end of the fraction’s denominator.

We’ve created a macro called “NewMacro2” (Remember that the macro name can be anything unique). This macro string is as follows: `x^{2}= *obar*{$}-3$` (Notice two wildcards)

NOTE: Any number of wildcards can be placed in a macro.

Within PowerMath ASCII delimiters [&&] we key the string

`!NewMacro2!a|beta|c|Del|x`

and in the next line we key

`!NewMacro2!z|nab|w$_{ij}$`

After PowerMath imports the equations the result would be

$$x^2 = \overline{a\beta c\Delta} - 3^x$$

$$x^2 = \overline{z\nabla w} - 3_{ij}$$

Macros are a powerful tool which can be used for many purposes. Remember these rules

- * *You cannot run a macro within a macro.*
- * *In Ascii, the merge codes are placed outside the macro delimiters “!macroname!”.*

XII. Mathematics Composition Procedures

Division

Example:

$$\frac{1}{2}$$

Accessing the Division Procedure Interactively

HOTKEY:
< -d>

This section will detail creating, editing, and accessing the basic procedures of PowerMath. There will also be illustrated instructions for specific dialogs and explanations for new features. This section will also explain the process of accessing the procedures both interactively and through the PowerMath 4.0 ASCII import language.

The Division procedure is used to typeset multilevel buildup math division structures. The baseline deflection for the rule, the rule weight, and the rule color can all be set as preferences in twenty different setups. The default division tag name is “frac”. This will set a horizontal black bar, 3.5% the weight of the base point size, with a deflection 30% above baseline. Because the Math Level Offset (III. **Basic Concepts**) has been set to 70%, the top field (numerator) has a deflection value of 70% of the Base Point Size above the base line, while the bottom field (denominator) has a deflection value of 70% below the baseline.

Note: The default superset has set up a number of Division setups, including Division procedures with rules which are heavier or tinted or even white rules.

See Appendix A for a more detailed presentation of available Division setups.

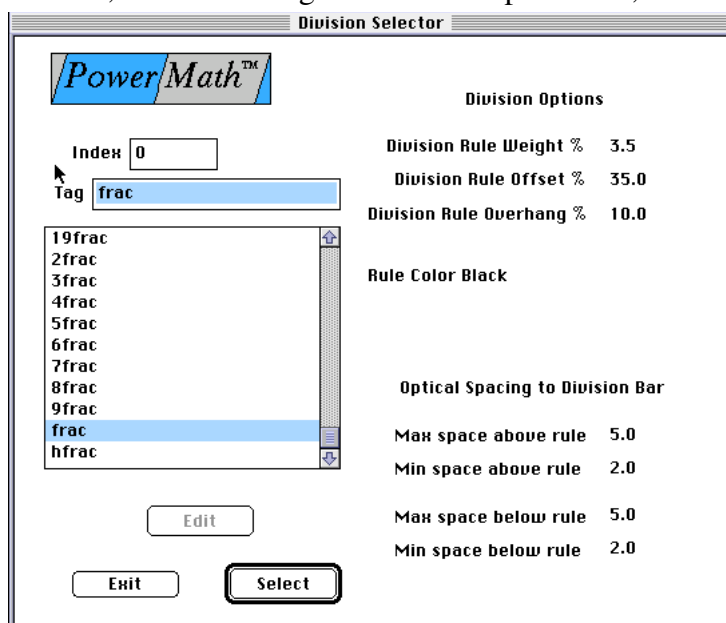
In the main PowerMath equation window, at the insertion point, key < -d> (or < -opt-d> to access this procedure from the Division options dialog). You can also select “Division” from the PowerMath/Procedures menu bar.

When using the options dialog, key in (or double click) an existing Division tag. Click <Select>. This setup becomes current and the procedure now automatically begins. Hereafter, the hot key < -d> will automatically access this setup.

If you just use the hot key < -d>, without accessing the dialog, the procedure likewise automatically begins. Notice that the Division procedure is turned on in the nested procedures section of the main PowerMath window.

The cursor is automatically moved to the numerator field. You then insert the character(s) you want inside the numerator. When you have finished with the numerator, press <ret>. The cursor automatically moves to the denominator. You then insert the character(s) you want inside the denominator. When you have finished with the denominator, press <ret>. The procedure is ended. You will notice that the division procedure has disappeared in the nested procedure section of the main PowerMath equation window.

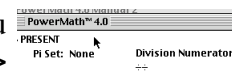
Note: You can nest a Division procedure inside a Division procedure by accessing one procedure, click <Select>, then access another. For example: key < -opt-d>, key in (or double click on) “frac”, click <Select>, key < -opt-d>, key in (or double click on) “frac”, click get. You will see two Division procedures active on the nested pro-



PowerMath™ 4.0
ENT
Set: None Division Numerator
+

Section XII

cedure section in the main PowerMath equation window. As you key the contents of each element, terminate that element with **<ret>** in order to move to the next element.



To place a fraction above a fraction, key **< -d>< -d>**, and two division procedures will appear in the nested procedure section of the main PowerMath equation window. The cursor will move to the numerator of the top fraction. Key the character(s) in the numerator of the top fraction and press **<ret>**. The cursor then moves to the denominator of the top fraction. Key the character(s) in the denominator of the top fraction and press **<ret>**. This will end the top fraction. One of the Division procedures will disappear in the nested procedure section of the main PowerMath equation window. You are still in the numerator of the original fraction. To move to the denominator, press **<ret>** again. Key the character(s) in the denominator of the of the overall fraction and press **<ret>**. This will end the entire buildup.

Editing Division setups

HOTKEY:

< -opt-sh-d>

While inside the main PowerMath window, key **< -opt-sh-d>** or choose “Division” from the “Superset/Edit Procedures” menu. This opens the Division editing dialog.

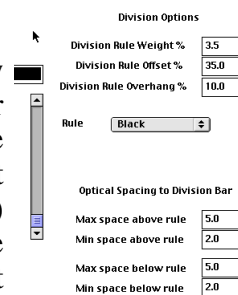
To edit a setup, key in (or double click on) an existing Division tag. Click the **<Edit>** button. This will enable you to change any of the preference values.

Set the division rule weight. PowerMath 4.0 default is 3.5% of the base point size.

Set the division rule offset. PowerMath 4.0 default setting is 35% deflection off the baseline

Set the division rule overhang. This value is the amount of rule that will extend before and after the longest set of character(s) in either the numerator or denominator. The default is 10%.

Next set the optical space over and under the rule. This is a new feature in PowerMath 4.0. The setting will be the tolerance for optical spacing. The default setting is minimum 2.0% of the base point size and maximum 5.0% of the base point size. Most character(s) will set within this tolerance, but if the character(s) exceed either the minimum or the maximum tolerance in the numerator or denominator PowerMath vertically moves that field to comply. This eliminates all instances of characters over-setting division rules, and allows for a globally editable overall compliance with your stylesheet specification.



After editing new changes, click the **<Save>** button. You can then access that setup by clicking the **<Select>** button or you can leave the Division dialog by clicking on the **<Exit>** button.

Creating a new Division setup

While inside a main PowerMath window, choose **< -opt-sh-d>** or “Division” from Superset/Procedures Options menu. This will open the Division editing window.

Choose an unused setup by keying in (or double clicking on) the PowerMath default tag name. Click the **<Edit>** button. Key in a unique tag name (case sensitivity is important). You can then tab through and make the changes to the appropriate fields. After setting the new preferences, click the **<Save>** button. You can then access that setup by clicking the **<Select>** button or leave the Division window by clicking on the **<Exit>** button.

Section XII

*Accessing Division
through the
PowerMath ASCII
filter language*

SYNTAX:

***frac*{}{}$\frac{abc}{def}$**

Inside PowerMath delimiters **[&&]**, and at the point of insertion, key the unique Division tag name inside the procedure delimiters **“**”**. The PowerMath default Division tag is ***frac*{}{}$\frac{abc}{def}$**. So as an example, to create a simple fraction key this text:

[&*frac*{abc}{def}&].

The result will be:

$$\frac{abc}{def}$$

To nest Divisions inside Divisions, key the second tag inside the first tag’s delimiters.

The string **[&*frac*{*frac*{abc}{def}}{ghi}&]** will result in:

$$\frac{\frac{abc}{def}}{ghi}$$

Example:

Summation

The summation procedure basically places editable fields above and below a baseline character. The baseline character is usually predefined in the pi interface, but any keyboard character could be used as the baseline character in the summation procedure. Products, co-products, and variant integrals are some examples of functions that can be created with the summation procedure.

*Editing a
Summation
character*

HOTKEY:

< -sh-op-j>

Creating a character within the Pi Character set has previously been explained. We will assume you have read and that you understand that section. Key **< -sh-opt-j>**. This should bring up the main Pi Character editing window. Key in (or double click on) the unique tag name for the Summation character you wish to edit. Click **<Edit>** This will enable the preference fields to be edited.

Three unique features of a Summation setup are the Main Character Count, Levels, and Limits Index.

*Main Character
Count*

The *Main Character Count* allows the user the ability to create a repeating character as the main character. For example, to get a triple summation, you can create a new character based on a single summation with the Main Character Count set a “3”. The resulting setup will be produced as

$$\sum_a^a \sum_b^b \sum_c^c$$

Example:

$$\sum_a^a$$

Keeping in mind the summation procedure places limits above and below the main character, the *Level* at which those limits sit must be specified. For example, a display summation is approximately 180% larger than base point size. The character itself is a three level character. We must set the Level to 3 or the limits will only extend to the top and bottom of the first level.

Setting the level incorrectly will result in the output

$$\sum$$

Section XII

Accessing a sup/inf setup for the limits

You can set the sup/inf *Index* for limit characters. The size and placement of limits for a summation may be different from those for text. The Limit Index allows the user to specify a predefined sup/sub setup as the default setting for these limit characters.

Note: The default superset has already set up a number of Summation procedures in the Pi Character Interface. See Appendix A for a detailed layout of Summation setups.

Accessing Summation setups Interactively

HOTKEY:

< -e><E>

Math™ 4.0	
1	Summation Character
	Σ

You can access a tag for a summation in the usual way from the Pi Character interface accessed by < -opt-j>, in which case the main summation character is accessed and inserted automatically. Or, within the main PowerMath equation window, at the insertion point, choose < -e> or “Summation” from the PowerMath/Procedures file menu. Notice that the Summation symbol appears in the nested procedure section of the main PowerMath equation window.

If you are using the keyboard shortcut < -e>, you will notice that the Pi Character set has automatically changed to “1”. The next character you enter will be the main summation character. The default for a normal Summation is mapped to the input character “E”. Key in a capital “E”.

Once the main Summation character has been accessed, the cursor automatically appears at the top limit of the summation. You then insert the character(s) you want inside the top limit. When you have finished with the top limit, press enter. The cursor automatically moves to the bottom limit of the summation. You then insert the character(s) you want inside the bottom limit. When you have finished with the bottom limit, press enter. The procedure is ended. You will notice that the summation procedure has disappeared from the nested procedure section of the main PowerMath equation window.

Accessing Summations with the PowerMath ASCII filter language

The default PowerMath ASCII tags for Summation are ***sum*{lim}{lim}** for a normal display Summation, ***smsum*{lim}{lim}** for inline Summation or small Summation, and ***vsum*{lim}{lim}** for Summations which have their limits on the right side of the main character.

SYNTAX:

***sum*{}{}**

Within PowerMath delimiters [&&] at the insertion point you can key [**&*sum*{abc}{def}&**] The resulting equation after importation with the PowerMath ASCII filter would be:

$$\sum_{def}^{abc}$$

The Summation limit alignment

The summation procedure allows the fields above or below the baseline character to be center aligned, left aligned, or right aligned. The PowerMath default is center aligned. You do not have to specify this setting every time you create a summation, but that default setting can be changed to affect subsequent functions created by the summation procedure.

When you change the current setting for the summation limit alignment, you set that value for any summations that will be created while the current PowerMath editing window remains open.

Setting the current Summation limit alignment interactively

Within the main PowerMath equation window, key < -opt-e>. A summation limit alignment dialog will appear. Choose the alignment you want. Click <Select>. The current setting will now be made “from now on” until the current editing window is closed.

Section XIII

HOTKEY:

< -op-e>

or

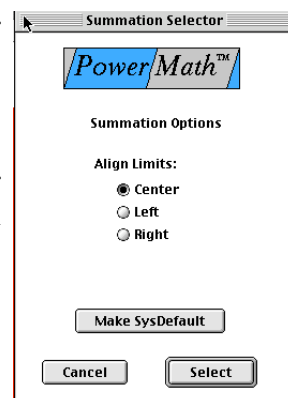
< -op-sh-e>

*Setting the
Summation Limit
Alignment Default
through the
PowerMath ASCII
filter language.*

To “permanently” change the default, open this dialog for editing with < -opt-sh-e> and click on the “Make SysDefault” button.

Within the PowerMath delimiters [&&] key the string ***SA*[]**. Inside the straight brace delimiters enter L for left align; C for center align; R for right align. This string is effective “from now on” during the current importation process.

[&*SA*[L]x+*sum*{y}{z}&]



Integral

Example:

$$\int_b^a$$

*Editing a Integral
character*

HOTKEY:

< -opt-sh-j>

The Integral procedure basically places editable fields to the right of baseline character. The baseline character is usually predefined in the pi interface, but any keyboard character could be used as the baseline character in the Integral procedure.

Creating a character within the Pi Character set has previously been explained. We will assume you have read and understand that section. Key < -opt-sh-j>. This will bring up the main Pi Character editing dialog. Key in (or double click on) the unique tag name for the Integral character you wish to edit. Click <Edit> This will enable the preference fields to be edited.

Four unique features of an Integral character are the Main Characters Count, Levels, Lower Limit Kern, and Limits Index

Main Character Count allows the user the ability to create a repeating character as the main character. For example, to get a triple Integral, you can create a new character based on a single Integral with the Main Character Count set a “3”. The resulting setup will be rendered as:

$$\iiint_b^a$$

Integral	
Main Character Count	1
Levels	3
Lower Limit Kern %	35.0
Limits Index	2

*Accessing a sup/inf
setup for the limits*

Keeping in mind the Integral procedure places Limits stacked to the right of the main character, the *Level* at which those limits sit must be set. For example, a display Integral is approximately 180% larger than base point size. The character itself is a three level character. We must set the level to 3 or the limits will only extend to the top and bottom of the first level.

Example: Setting the Level incorrectly will result in the output

$$\int_b^a$$

The integral is an oblique character. Sometimes the lower limit appears to have too much space. *Lower Limit Kern* allows the user to “tuck” the limit back to the right.

You can set the sup/inf *Index* for limit characters. The size and placement of limits for an integral may be different from those for a summation. The Limit Index allows the user to use a predefined sup/sub setup as the default setting for limit characters.

Note: The default superset has already set up a number of Integral procedures. See Appendix A for a detailed layout of Integral setups.

Section VII

Accessing Integrals interactively

HOTKEY: < -f>

Accessing Integrals with the PowerMath ASCII filter language

SYNTAX:
***int*{lim}{lim}**

Within the main PowerMath equation window, at the insertion point key < -f> to launch the procedure (or choose “Integral” from the PowerMath/Procedures file menu). Notice that the Integral symbol appears in the nested procedure section of the main PowerMath equation window.

Also notice that the “Pi Character set” field has automatically changed to “1”. The next character you enter will be the base Integral character. The PowerMath default for a normal Integral is mapped to “I”. Key in a capital “I”. The cursor automatically appears at the top limit of the Integral. You then insert the character(s) you want inside the top limit. When you have finished with the top limit, press <ret>. The cursor automatically moves to the bottom limit of the Integral. You then insert the character(s) you want inside the bottom limit. When you have finished with the bottom limit, press <ret>. The procedure is ended. You will notice that the Integral procedure has disappeared in the nested procedure section of the main PowerMath equation window.

The default ASCII tags for Integral are ***int*{lim}{lim}** for a normal display Integral, ***smint*{lim}{lim}** for inline Integral or small Integral, and ***vint*{lim}{lim}** for Integrals which have limits on the right side of the main character.

Within PowerMath delimiters [&&] at the insertion point you can key [&*int*{abc}{def}&] The result after PowerMath import would be

$$\int_{def}^{abc}$$

Stack Limits

Example:

$$\sum_{f \atop de}^{ab \atop c}$$

HOTKEY:
< -L>

You can stack limits by invoking the Stack Limit procedure in PowerMath. While inside a limit field within the summation or integral procedure you can invoke a command < -L> which will automatically stack limits above (if you are in a top limit) or below (if you are in a bottom limit). You can invoke this command indefinitely. To end the stacked limits, press <ret> as you would normally to transition to the next field. The cursor moves on to the bottom field (if you are in the top limit) or ends the procedure (if you are in the bottom limit).

Note: The stacked limit uses the same superior/inferior setup which was used for the other limits.

Note: Stacking limits is not an editable change. If you set the procedure without stacking, you cannot go back and edit in stacked limits. The procedure must be rekey-boarded in this case.

To change the appearance of the stacked limit, invoke a new sup/inf index. You will have to return the sup/inf value when you move to the next field or the procedure will use that setting instead of the preference saved originally.

Accessing Stacked Limits through the ASCII filter

SYNTAX:
L

Within PowerMath delimiters [&&] at the insertion point inside the limit field delimiters of a summation or integral procedure key ***L*** without delimiters. Everything from that point to the end delimiter of the limit field will be considered a stacked limit. To key a second stacked limit, key in another ***L*** and so on.

Example:

Section XII

The string:

[&*sum*{ab*L*c}{de*L*f}&]

would result in:

$$\sum_{ab}^c \frac{de}{f}$$

Radical

Example:

$$\sqrt{f}$$

Editing a Radical character

The Radical procedure is mostly used for square root structures.

The Radical procedure can also be used any time you set a character that is followed by a rule. El-Hi division and angle definitions are two examples of this.

Note: The default superset has already set up a number of Radical procedures. See Appendix A for a detailed layout of Radical setups.

Creating a character within the Pi Character set has previously been explained. We will assume you have read and understand that section. Key < **-opt-sh-j**>. This will bring up the main Pi Character editing dialog. Key in (or double click on) the unique tag name for the radical character you wish to edit. Click <Edit> This will enable the preference fields to be edited.

The unique feature of a radical character is the rule. When you resize an existing radical character (for example, changing the normal radical from 100% to 105%), the rule position and kerning value must also be adjusted. Keep in mind you are trying to match the character corner with the rule.

Character	
Radical	
Rule Weight %	3.5
Rule Offset %	-1.0
Rule Kern %	4.0

The PowerMath default *Rule Weight %* for radicals is 3.5.

The *Rule Offset %* moves the rule vertically along the main character.

The *Rule Kern %* moves the rule horizontally along the main character.

Note: When adjusting the rule offset, the higher the positive number, the lower the rule sits off the baseline; the higher the negative number, the higher the rules sits off the baseline.

Note: When adjusting the rule kern, the higher the positive number, the closer to the character the rule begins; the higher the negative number, the farther away from the character the rule begins.

Accessing Radicals interactively

HOTKEY:
< **-r**>

Within the main PowerMath equation window, at the insertion point key < **-r**> or choose “Radical” from the PowerMath/Procedures file menu. Notice that the radical symbol appears in the nested procedure area.

Also notice that Pi Character set has automatically changed to “1”. The next character you enter will be the character you have mapped to the radical character. The default for a normal radical is mapped to “V”. Key in a capital “V”. You then insert the character(s) you want inside the radical. When you have finished, press enter.

Note: The radical setups have been created with Auto-Upsize, which means if the contents of the radical include multiple “levels”, PowerMath will automatically remap the radical character.

Note: The default radical character was created to fit a full sized character with a superscript. You can also choose a small radical (which fits a lowercase character

Section XII

without a superscript by substituting **<opt-v>** for the capital “V” in the above example.

The default PowerMath ASCII tags for radical are ***rad*{content}** for a normal radical, ***smrad*{content}** for lowercase characters without superscripts, and ***cfrad*{content}** for radicals which contain case fractions.

Within PowerMath delimiters **[&&]** at the insertion point you can key **[&*rad*{abc}&]**. The result after PowerMath import would be

$$\sqrt{abc}$$

*Accessing Radicals
with the PowerMath
ASCII filter
language*

SYNTAX:

***rad*{ }**

Note: To key a multi-level radical, you still use the same syntax. PowerMath automatically recognizes multi-level and chooses the appropriate radical character. For example the single level radical ***rad*{abd}** uses the same syntax as the multi-level ***rad*{frac*{abc}{def}}**.

Note: To nest a radical inside a radical see Appendix A.

Case Fraction

Example:

$$\frac{1}{2}$$

*Accessing the Case
Fraction procedure
interactively*

HOTKEY:

< -h>

The case fraction procedure is used mostly to insert small, “in-line” numeric fractions. The default case fraction is 65% of the base point size and will fit within most paragraph leadings (9/11, 10/12) without oversetting the previous or following lines.

Note: The default superset has a number of Case Fraction setups, including Case Fraction without rules, and Case Fractions with tighter leading for specialized purposes. See Appendix A for a more detailed layout of Case Fraction options.

In the main PowerMath equation window, at the insertion point key **< -h>** or choose “Case Fraction” from the Superset/Procedures menu bar.

OR key in (or double click) an existing Case Fraction tag. Click **<Select>**. Notice that the Case Fraction procedure is turned on in the nested procedures section of the main PowerMath window.

The cursor is automatically moved to the numerator field. You then insert the character(s) you want inside the numerator. When you have finished with the numerator, press enter. The cursor automatically moves to the bottom field (denominator). You then insert the character(s) you want inside the denominator. When you have finished with the denominator, press **<ret>**. The procedure is ended. You will notice that the Case Fraction procedure has disappeared in the nested procedure section of the main PowerMath equation window.

While inside a main PowerMath window, key **< -opt-sh-h>** or choose “Case Fraction” from Superset/PowerMath Options file menu. This will open the Case Fraction editing dialog. PowerMath has enabled twenty different case fraction setups.

To edit a setup, key in (or double click on) an existing Case Fraction index. Click the **<Edit>** button. This will enable you to change any of the preference values.

Set the height and width separately. Keep in mind that 65% of the base point size

Case Fraction Options	
Point Size %	65.0
Set Width %	65.0
Rule Weight %	3.0
Rule Offset %	35.0
Numerator Offset %	45.0
Denominator Offset %	22.5
Optical Space Over/Under Rule %	0.0
Rule Color	Black

*Editing Case
Fraction
setups*

Section XII

would mean that if the base point size were 10 points, the characters in a case fraction would be 6.5 points.

Set the weight and offset for the rule. The default setting is 35% deflection off the baseline

Set the offset for the numerator and denominator. The default setting is 70% deflection off the baseline.

Set the optical space over and under rule. This setting will be for optimal spacing. The PowerMath 4.0 default setting is 14% of the base point size. If the character(s) in either the numerator or denominator do not meet that value, PowerMath vertically moves that field proportionately. This automatically maintains an absolute optimal spacing, unless the feature is turned off by specifying a value of zero.

After editing new changes, click the **<Save>** button. You can then access that setup by clicking the **<Select>** button or leave the Case Fraction dialog by clicking on the **<Exit>** button.

Creating a new Case Fraction setup

HOTKEY:

< -opt-sh-h>

While inside a main PowerMath window, key **< -opt-sh-h>** or choose “Case Fraction” from the “Superset/Edit Procedures Options” menu. This will open the Case Fraction editing dialog.

Choose an unused setup by keying in (or double clicking on) the PowerMath default tag name. Click the **<Edit>** button. Key in a unique tag name (case sensitivity is important). You can then tab through and make the changes to the appropriate fields.

Accessing Case Fraction through the PowerMath ASCII filter language

Inside PowerMath delimiters **[&&]**, and at the point of insertion, key the unique Case Fraction tag name inside procedure delimiters **“**”**. The PowerMath default Case Fraction tag is ***cf*{}{}&[]**. So as an example, to create a simple fraction key this text: **[&2*cf*{1}{2}&[]]**. The result will be:

$$2\frac{1}{2}$$

Matrix

Example:

<i>a</i>	<i>b</i>	<i>c</i>
<i>d</i>	<i>e</i>	<i>f</i>
<i>g</i>	<i>h</i>	<i>i</i>

To create a matrix with PowerMath you must first define the parameters of that particular matrix. The PowerMath default superset defines a default matrix as keyboarded Across/Down, Center Aligned, Computed Column Widths, and 3 rows by 3 columns. Of course, not all matrices are set up this way. So it is important to define each matrix just before each one is built.

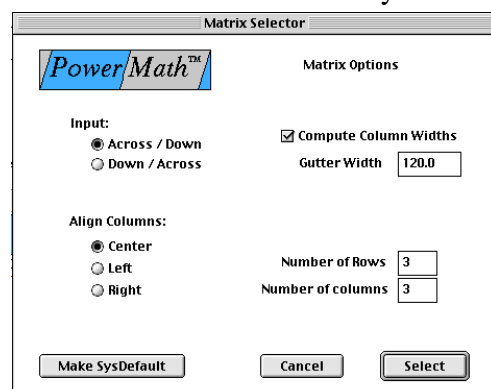
Note: The Matrix preference selector is “from now on”, so if you have a series of matrices that share the same preferences, you would not have to define every matrix.

In the main PowerMath equation window key **< -opt-m>**. This will open the Matrix Selector dialog.

Select the desired keyboard input sequence. There are two: down and across or across and down)

Select the desired alignment of the columns: Center, Left, or Right.

Defining a Matrix interactively



Section XII

Next you have the option to allow PowerMath to automatically “Compute Column Widths”. PowerMath finds the widest field in any given column and adjusts the width of the column accordingly. Unchecking “Compute Column Widths” changes the “Gutter Width” field to a column width field for specific values. This is rarely used.

Define the number of columns and rows.

You have a choice to keep these preferences as the System Default. To do so, open this dialog with **< -op-sh-m>** and click on **<Make SysDefault>**.

When you click the **<Select>** button, you have invoked the Matrix procedure. Notice that the Matrix symbol appears in the nested procedure area.

NT	column 1, row 1
et: None	

Here is a sample matrix:

$$\begin{matrix} 2 & 0 & 0 \\ 0 & 2 & i \\ 0 & -i & 2 \end{matrix}$$

Notice this matrix has three rows, two columns, and is center aligned.

We set the Matrix Selector exactly as above and clicked **<Select>**

PowerMath has moved the cursor to the first row, first column field. Remember, we set the Input preference to Across and Down. We key in 2<ret>0<ret>0<ret> (this moves the cursor to the second row of the first column) 0<ret> 2<ret> i<ret> (this moves the cursor to the third row of the first column) 0<ret> -i<ret>2<ret>. This moves us out of the matrix and ends the procedure. Notice that the Matrix symbol has disappeared from in the nested procedure section of the main PowerMath equation window.

Note: You can nest any procedure while in a column/row field.

Creating a Matrix in ASCII

The syntax for an ASCII matrix reflects the syntax used in the interactive dialog. Just as in the interactive matrix, you need to set up the matrix.

The setup and usage syntax is as follows (see Appendix A for details):

~MAT~	<i>calls out the matrix setup</i>
[A%B%C%D%E%F]	<i>reflect the fields within an interactive matrix setup window</i>
	<i>A=Rows</i>
	<i>B=Columns</i>
	<i>C=Alignment</i>
	<i>D= column width if fixed columns, gutter width if computed columns. (value = % of base point size)</i>
	<i>E=Calculated or Fixed columns</i>
	<i>F=Input series (A= across and down, D=down and across)</i>
MAT	<i>begins matrix</i>
{{}}	<i>reflect the fields and cells</i>
%	<i>separates the fields within a row or column</i>

Section XII

A two by two center aligned matrix would be keyed like this

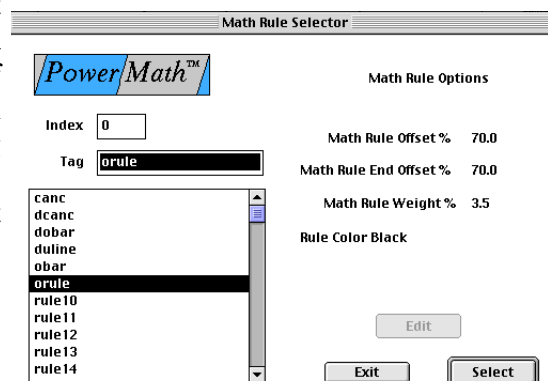
[&~MAT~[2%2%C%120.0%C%A]*MAT*{a%b}{c%d}&]

The result will be

$$\begin{matrix} a & b \\ c & d \end{matrix}$$

Math Rule

The Math Rule procedure sets a rule beginning from the insertion point and which continues until the procedure is ended by the operator or at the end of the equation. The baseline deflection, rule weight, and color can all be set as preferences in twenty different setups. Underline, cancellation and vector arrows can all be set with the Math Rule procedure. The default Math Rule tag is 'obar'. This will set a horizontal black bar, 3.5% the width of the base point size, with a deflection 60% above the baseline. While setting 10 pt math, this would mean 6 points above the baseline (or 2 pts above most lowercase alpha and greek characters)



Accessing the Math Rule procedure interactively

HOTKEY:

< -opt-y>

In the main equation window, at the insertion point, choose < -opt-y> or "Math Rule" from the "PowerMath /Procedures" menu bar.

Key in (or double click) an existing math rule tag. Click <Select>. Notice that the Math Rule procedure is turned on in the nested procedures section of the main PowerMath window.

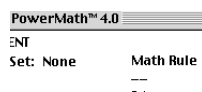
Editing Math Rule setups

HOTKEY:

< -opt-sh-y>

To end the procedure, key <ret>. You will see the procedure disappear from the nested procedure section.

Note: You can nest Math Rules inside Math Rules by accessing one setup, click get, then access another. Example: Key < -opt-y>, key in (or double click on) "orule", click <Select>, key < -opt-y> again, key in (or double click on) "dorule", click get, add character(s). You will see two Math Rule procedures active on the nested procedure section in the main PowerMath equation window.



To end the procedure, press <ret>. You will see one of the Math Rule procedures disappear from the nested procedure section in the main PowerMath equation window. Press <ret> again to end the second procedure.

The string

< -y>agc<Enter>

will result in the equation

$$\overline{agc}$$

Note: You can nest Math Rules inside Math Rules, but if you choose the same procedure, the rule will appear as only one rule.

Note: The default superset has a number of Math Rule setups. See Appendix A for a more detailed layout of Math Rules.

Creating a new Math

Section XII

Rule setup

While inside a main PowerMath window, choose **< -opt-sh-y>** or “Math Rule” from the “Superset/Edit Procedure Options” menu. This will open the Math Rule editing dialog. PowerMath has enabled twenty different Math Rule setups.

To edit a setup, key in (or double click on) an existing Math Rule index. Click the **<Edit>** button. This will enable you to change any of the preference values.

After editing new changes, click the **<Save>** button. You can then access that setup by clicking the **<Select>** button or leave the Math Rule dialog by clicking on the **<Exit>** button.

While inside a main PowerMath window, choose **< -opt-sh-y>** or “Math Rule” from the “Superset/Edit Procedure Options” menu. This will open the Math Rule editing window.

Choose a unused index by keying in (or double clicking on) a PowerMath default tag name. Click the **<Edit>** button. Key in a unique tag name (case sensitivity is important). You can then tab through and make the changes to the appropriate fields as explained above.

After making new changes, click the **<Save>** button. You can then access that setup by clicking the **<Select>** button or leave the Math Rule dialog by clicking on the **<Exit>** button.

Cancellation Rules

Accessing Math Rules through the ASCII language

Note that there are two fields for vertical offsets for the Math Rules. When these are different, the rule sets on a slope. See Appendix A for existing Cancellation setups.

Inside PowerMath delimiters **[&&]**, and at the point of insertion, key the unique Math Rule tag name inside procedure delimiters **“***”**. The PowerMath default Math Rule tag for underline is ***uline*{}**. So as an example, to create an underline under character(s) key this text:

[&*uline*{abc}&]

The result will be

abc

To nest Math Rules inside Math Rules key the second unique PowerMath ASCII tag inside the first PowerMath ASCII delimiters.

[&*obar*{*dobar*{abc}}&]

will result in

abc

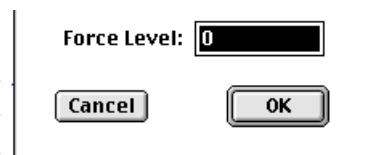
Force Level

Using Force Level Interactively

There may be times when you need to force part of the equation to be either raised or lowered to another full level. When the procedure is called either through the PowerMath ASCII language or interactively, the change is “from now on” or until the end of the equation.

At the insertion point key **< -opt-;>** or “Force Level” from the “Powermath/Procedures” menu bar. A dialog box will appear.

Insert the positive or negative value you wish to move the baseline. For example, if you wish to move the baseline one full level above the original baseline, you

A dialog box titled "Force Level:" with a text input field containing the number "0". Below the input field are two buttons: "Cancel" and "OK".

Force Level: <input type="text" value="0"/>
<input type="button" value="Cancel"/> <input type="button" value="OK"/>

Section XII

would insert 1. If you wish to move the baseline two full levels under the original baseline, you would insert -2.

Note: If the equation ends after the force level, we do not have to reset the force level back to the original baseline. But if the equation had continued with characters on the original baseline we would have opened the force level window again < **-opt-;**> and inserted “0” as the value.

Within PowerMath ASCII delimiters [**&&**], at the point you wish to move the baseline key ***FL*[]** (case sensitive). Inside the straight base delimiters, key the positive or negative value you wish to move the baseline. Remember force level is “from now on”. When you want to move the baseline back to the original position, you must key ***FL*[0]**.

Using Force Level
though the
PowerMath ASCII
filter language

SYNTAX:
***FL*[]**

Note: When moving from level to level, you do not have to reset to 0. Keying with the ASCII filter (and interactively with < **-opt-;**>) you could actually key: ***FL*[-3]abc*FL*[3]def** and PowerMath would move the baseline from three full levels under the original baseline to three full levels above the original baseline.

Note: When keying with the ASCII language, when PowerMath comes across “***FL***” it expects to see the delimiter “[” as the next character followed by a value (positive or negative) then the closing “]”. If it does not, PowerMath will error. Once the ***FL*** delimiters are satisfied, “[” becomes a straight brace character.

Stacking

Stacking is a new feature for PowerMath 4.0. This procedure allows characters to be positioned and horizontally aligned directly above or below the baseline including or excluding content along the baseline.

HOTKEY:
< **-opt-t>**



In the main PowerMath equation box key < **-opt-t>** or “Stack” from the PowerMath/Procedures file menu bar. The Stacking index dialog will appear. The values for the procedure are set in this order.

Baseline

Over

Under

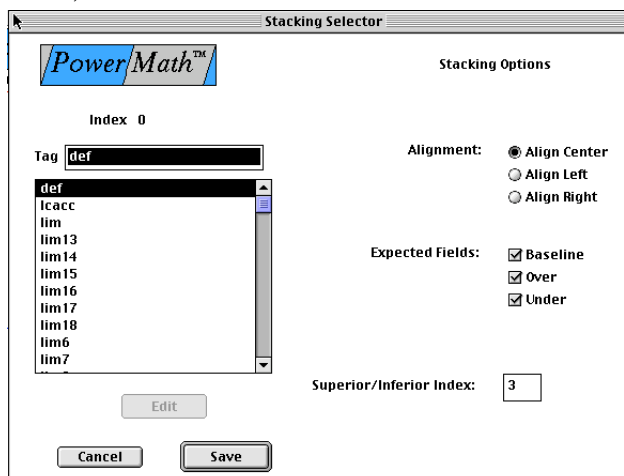
You can create a stack that enables all three or any combination of two.

If all three fields are checked, the stacking procedure will place the *Baseline* field, then the *Over* field, and then the *Under* field. If only over and under are checked, the procedure will begin with the *Over* field, then move to the *Under* field. If *Baseline* and one other field are checked, the program will begin setting in the *Baseline* field first.

Choosing a Stacking
procedure index
interactively

In the main PowerMath equation window, at the insertion point, key < **-opt-t>** or “Stack” from the PowerMath/Procedures file menu bar.

Key in (or double click) the PowerMath stacking tag. For our example, we choose “oscr”. The preferences for that unique tag will appear on the right side of the window



Section XII

Click the **<Select>** button. Notice the procedure and index level will appear in the nesting procedure section of the main PowerMath equation window

Beginning from the main field, (assuming that the baseline field is active) insert character(s). To move to the next field, key **<Return>** and insert the next set of character(s). To end the procedure, key **<Return>**.

Note: The default superset has a number of stacking setups. See Appendix A for a detailed layout of stacking setups.

Note: You must define a stacking setup in conjunction with a Superior/Inferior setup in order to set the size and position of the “over” and “under” elements.

Within PowerMath delimiters **[&&]** at the insertion point you can access the stacking procedure by keying the unique tag name within the procedure delimiters **“***”**. For our example, we key

[&*oscr*{a+b}{~rom~always|sp|positive~norm~}&]

The result will be

always positive
 $a + b$

*Accessing Stacking
with the ASCII
Language*

Superiors and Inferiors

*The default
Superior/Inferior
index settings*

HOTKEYS:

< -g> and
< -u>

The superior and inferior function of PowerMath allows the baseline to move a percentage above or below the original baseline. A Superior/Inferior setup determines the size and percentage of deflection off the baseline. The default superior/inferior preference is set for super- and subscripts.

PowerMath comes with a default setting for superior/inferiors. The values are set at 65% base point size, 40% superior baseline deflection, 20% inferior baseline deflection. When choosing a superior, key **< -g>**. Notice the procedure and index level in displayed in the main PowerMath window.

Beginning at the insertion point, everything entered will appear 65% of the base point size and 40% above the baseline. To return to the previous values (or the status at the original insertion point) key **< ==>**

ber 18B PRESENT
12.0 PI Set: None
Sup/Inf 0

x< -g>2< ==>

will become

x^2

When keying a inferior, key **< -u>**. Beginning at the insertion point, everything entered will appear 65% of the base point size and 20% below the baseline. To return to the previous values (or the status at the original insertion point) key **< -j>**

H< -u>2< -jO>

will become

H_2O

*Accessing a
Superior/Inferior
setup*

While inside a PowerMath editing window, key **< --opt==>** or “Superior/Inferior” from the “PowerMath/Procedures” menu. This will open the Superior/Inferior dialog.

Key in an existing tag (or double click on it), The preferences will appear for that setup along the right side of the box.

Click **<Select>**. Superiors or inferiors keyed from that point on will be set using the values of that setup.

Editing

Section XII

Superior/Inferior
index values

< --opt-sh==>

Creating a new
Superior/Inferior
setup

Nesting the
Superior/Inferior
Procedure

To edit a setup, key in (or double click on) an existing Superior/Inferior tag. Click the **<Edit>** button. This will enable you to change any of the preference values. After editing new changes, click the **<Save>** button. You can then access that setup by clicking the **<Select>** button or leave the Superior/Inferior dialog by clicking on the **<Exit>** button.

Note: The default superset has a number of Superior/Inferior setups. See Appendix A for a more detailed layout of Superior/Inferior options.

While inside a main PowerMath window, key < --opt-sh==> or “Superior/Inferior” from the “Superset/Edit Procedures Options” menu. This will open the Superior/Inferior editing dialog. PowerMath has enabled twenty different setups.

Choose an unused setup by keying in (or double clicking on) a PowerMath default tag name. Click the **<Edit>** button. Key in a unique tag name (case sensitivity is important). You can then tab through and make the changes to the appropriate fields.

The *Point Size %* is based on the Base Point Size. For example with these preferences, if you were setting 10 point math, the sup/inf characters will be 6.5 points.

The *Set Width %* is a horizontal scale based on the Base Point Size.

Superior Offset % default is set at 40% baseline deflection. The *Inferior Offset %* is set at 20% below baseline deflection by default.

Pi Space is a % of the space left and right around operators, such as ‘=’, when these operators appear in superior or inferior strings, such as limits. The space is usually greatly reduced in these cases.

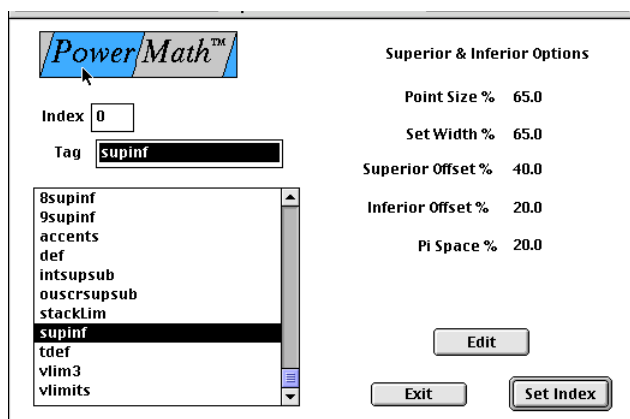
After making new changes, click the **<Save>** button. You can then enable that index by clicking the **<Select>** button or leave the Superior/Inferior window by clicking on the **<Exit>** button.

Once you have accessed Superior/Inferior characters, you can continue to nest Superior/Inferiors. If, for example, you have chosen the default Superior/Inferior index, and have begun a superscript < -g> (recall the preferences are 65% of the base font size and 40% deflection off the baseline) you can begin another superscript < -g>. All the characters keyed thereafter will be 65% the size of the first superior size (which was 65% of the original base point size), and 40% deflection above the first superior baseline (which was 40% above the original baseline).

g< -g>2

becomes

g^2



Superior & Inferior Options	
Point Size %	65.0
Set Width %	65.0
Superior Offset %	40.0
Inferior Offset %	20.0
Pi Space %	20.0

Section XII

g< -g>2< -g>3

becomes

$$g^{2^3}$$

To return to the previous baseline choose < -]>. This will return to the values prior to the insertion of the last < -g>. To return to the original baseline choose < ==>.

The sequence: **xy< -g>2< -g>y< -]>2< ==>**

becomes:

$$xy^{2^2}$$

and **xy< -g>2< -g>y< ==>2**

becomes:

$$xy^{2^2}2$$

*Stacked Super-
and Subscripts*

Example: 2_2^2

*Multi-level Super-
and Subscripts for
oversized characters*

Example: $\left(\frac{1}{n}\right)_i^2$

*Keying
Superior/Inferior
Procedure with the
PowerMath ASCII
filter language*

SYNTAX:
^{}_{}>

*Accessing Superior/
Inferior setups in the
ASCII language*

Example:
 ab^2

PowerMath includes a feature which will automatically stack super- and subscripts. Simply begin a superscript < -g>, key the character(s), return to the baseline < ==>, begin a subscript < -u>, key the character(s), and return to baseline < ==>. This can involve any Superior/Inferior setup.

Note: If the super- subscript feature is invoked to the right of a Pi character that has a multi-level value in its setup, the sup/sub characters will go to that level. And these multi-level superiors and inferiors will also stack automatically.

Use the “Levels” field for a “Pi Character” in the Pi Interface (< -opt-sh-j>).

This is particularly useful, for example, for setting exponents for multi-level parens.

ASCII coding for superscripts **^{content}** and subscripts **_{content}** can be invoked anywhere within PowerMath delimiters **[&&]**.

[&x^{2}|sp|~rom~or~norm~|sp|y_{3}&]

becomes

$$x^2 \text{ or } y_3$$

To nest super- subscripts, insert the string inside the curly brace delimiters “{}”.

[&ab^{x^{2}}|sp|~rom~or~norm~|sp|y_{3^{z}}&]

becomes

$$ab^{x^2} \text{ or } y_{3^z}$$

To key stacked super- subscripts key **^{}** followed by **_{}>**.

[&x^{2}_{y}&]

becomes

$$x_y^2$$

Prior to invoking a super- or subscript, set the index by placing the Superior/Inferior index tag within “***” delimiters. Remember that these tags must all be unique and case sensitive.

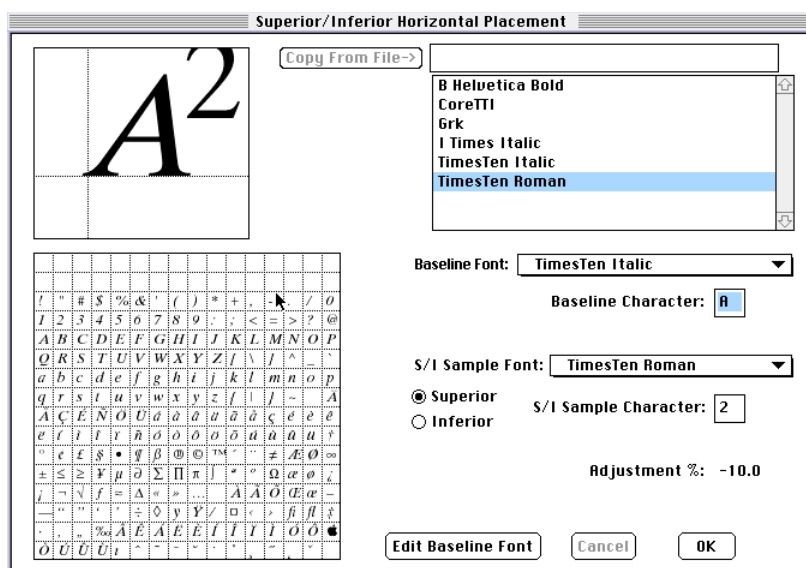
In the equation **[&ab*supinf*^{2}&]** the superscript will have the preferences of the setup associated with “supinf” in the Superior/Inferior procedure setup dialog.

XIII. Customized Horizontal Placement of Superior and Inferiors

A powerful new interface has been designed which lets the user set up customized horizontal positioning of superior and of inferior characters next to any baseline character in any font. Once specified, this space remains permanently in effect (or until changed). Furthermore, data entered through this interface can be changed and then a global edit to the document will pick up the changes everywhere!

This interface is called from the PowerMath “File/Place SupInf...” menu item.

The preview window offers an interactive view of what effect values have as they are specified.



The “Baseline Font” popup menu is used to specify which font is being edited.

The Baseline Character field is used to specify which character from the baseline font is being dealt with. You can select a character for this field either by typing one in, or by clicking on one in the palette box.

The “S/I Sample font” popup is used to specify which font is being used

The Interface

to view a sample, a possible superior or inferior character.

The S/I Sample character field is used to select trial characters for the S/I functions.

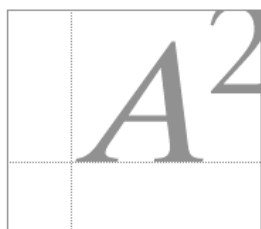
NOTE: You use this interface to specify setups for individual characters from the Baseline Font. The setups relate the baseline character to the superior or inferior *function*, not to specific superior or inferior characters from specific fonts.

The values specified in this interface are stored in a file, one such file for each font. The files are named using the font name, and are kept in the “Powermath Font Metrics” folder in the preferences folder in the System folder.

The scrolling list at the top of the dialog is a list of these stored metrics files. This list is not used to select a baseline font for editing. It is used if you wish to select an existing metrics file in order to copy its values into the currently selected baseline font file (selected from the “Baseline Font” popup menu) for storage and/or subsequent editing.

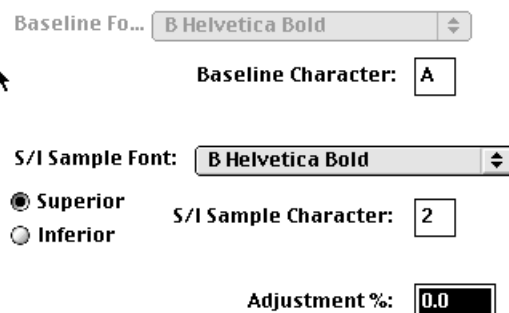
Edit Baseline Font

Note that various fields in this dialog are enabled/disabled by the “Edit baseline font” button, as appropriate.



Note: Only fonts active on the system will be available to edit.

Choose the font you wish to edit, and select



Section VIII

the **<Edit Baseline Font>** button. This will enable the “Adjustment” field. Your baseline font character will appear in the preview window. Choose an appropriate S/I sample font. The trial superior or inferior character will be displayed from this font. As values are keyed into the “Adjustment” field, the preview window will reflect changes. For each currently selected baseline character, you can change the adjustment % to a positive or negative value. The percentage is of the Base Point Size. A negative number will draw the sup/inf closer to the character being edited. The values for a capital “A” in the TimesTen Italic family will obviously be different from values for a lowercase “d” in the same family. And of course, for many baseline characters, the spacing to the superior function will be entirely different from that for the inferior function.

Once a font has been edited, click on **<OK>**. The values will be saved to the file in the “Powermath Font Metrics” folder in the System folder.

Note: The files in the “Powermath Font Metrics” folder in the System folder must reside on any system on which edits or updates to the equations which use them will be done!!

Note: If you choose not to use a font’s values for a document, remove the file from the “Powermath Font Metrics” folder. Moving the “Powermath Font Metrics” folder out of the System folder will disable this feature completely for edits, updates, or new equations.