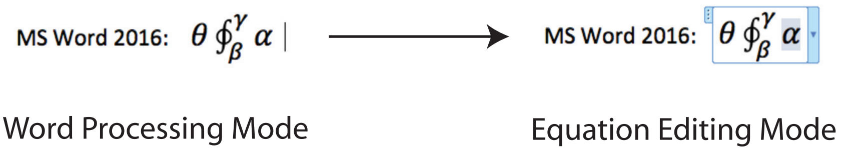
3 Conquering Mac Word

You either love it or you hate it, but you can’t get away from Microsoft Word, even on a Mac! This chapters goes through all recent versions of Word and how they work with *PrintMath*.

3.1 The Re-Editable Math Expression

The dream of the math expression in Word has been: double-click on the expression, and the expression becomes editable; change something, click off, and the expression is updated, magically.


MS Word 2016: $\theta \oint_{\beta}^{\gamma} \alpha$
MS Word 2016: $\theta \oint_{\beta}^{\gamma} \alpha$
Word Processing Mode
Equation Editing Mode

Of course, the real dream of working with math expressions is that the Word processor and the math expression maker are one in the same: you don’t leave the word processing program at all to make your math expressions. This is, however, a daunting dream to realize. It is like saying you should have the power of Photoshop inside of Word to manipulate images; there is only so much a word processor can do before it hands back to a specialized tool like Photoshop which is dedicated to image manipulation.

OLE - Object Linking and Embedding - on the Windows side, fullfills the dream, with some limitations. But the Windows solution is not cross-platform: any OLE MS Word .doc or .docx files brought over from Windows to Macintosh lose their OLE-ness. In fact, they become quite unfunctional indeed.

The Macintosh version of Microsoft Word has a pseudo-OLE (Object Linking and Embedding) feature that uses Apple Events to mimick the

OLE "edit in place" functionality for its built-in Equation Editor (EE), which is the free version of MathType (MT).

If you only need a few equations, and you are not too picky, then stick with the free version of Equation Editor. Don't race out and buy MathType unless you really really need it! Once you install MathType on your Mac, it overwrites the stock (free) Equation Editor, and then you may be in for some fun as new versions of Microsoft Word & Office appear, and then don't work with MathType. Copy your Applications folder over to a new Mac, or install a new hard drive? Goooooooood luck to ya! Everytime I have installed MathType, then sure enough, when something like this changes on my install disk, I have tremendous trouble trying to get MathType to start working again, and if I try to open up Word documents that have MathType equations embedded, then the recursive error messages are quite annoying indeed.

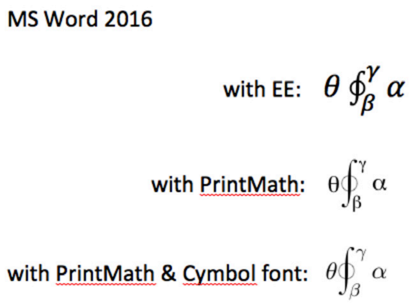
Don't get me wrong. MathType is very slick with its mountains of Word Scripts that automatically vertically center an expression based upon every conceivable situation. If you use Word in a workplace, this script usage might be problematic as many workplaces have Word Scripting turned off for security and anti-virus reasons.

Personally, I think it is just easier and more refined to do the placement of the images myself, manually. Then I have the control that I want, and the images go in exactly as I wish, without having to fight a placement script that thinks it knows better.

Unfortunately, this Mac psuedo-OLE feature is locked into only being available to the MathType/Equation Editor app. Just another in a long string of examples of how Microsoft does not play well with others.

Even without OLE, we can still get our more beautiful *PrintMath* expressions into Microsoft Word, with a minimal amount of extra work, and, I believe, more control than the MathType/EE option.

As for the quality of output, that is the real question. If you like MathType's output, then stick with it. But perhaps this very simple comparison of quality makes you think:



MS Word 2016

with EE: $\theta \oint_{\beta}^{\gamma} \alpha$

with PrintMath: $\theta \oint_{\beta}^{\gamma} \alpha$

with PrintMath & Cymbol font: $\theta \oint_{\beta}^{\gamma} \alpha$

If you like *PrintMath*'s output better than MathType/EE, you will be willing to go the extra mile to get the higher quality output.

Over the past 20 years, Word has changed drastically. To understand the whole Word story and how we need to work with it, some background information is required, as well as some history.

3.2 Mac Image File Formats

In general, there are two *general types* of image formats:

- Bitmaps
  Bitmap images are pixel-based images. All GIF, JPG, and PNG images are pixel-based. When you zoom-in on a bitmap, the image will "get fuzzy" due to a limitation of resolution quality. Nowadays, it is common (thank you iPhone) to have very high resolution bitmap images (like iPhone photos) that have astonishingly high resolutions, and large filesizes. It is no uncommon for a new iPhone to take a 10-25 megabyte (MB) photo!

  Simple Example: a *circle* in a bitmap image is a collection of pixels that are in the shape of a circle.

- Vector
  Vector images are drawn from text-based drawing commands. Example formats are PostScript, PDF, and SVG (Scalable Vector Graphics). Vector images do not suffer from "fuzziness" if you zoom-in (or out) on them - they have "infinite high resolution" as they are not "painted". When a software program draws a vector based file, it then interprets various conditions to generate an image, but if these conditions change, the image is regenerated, so a vector image "always looks nice".

  Simple Example: a *circle* is a command like:
  circle 3.4 1.2 5.33 1.0
  which might mean "draw a circle centered at (3.4, 1.2) of radius 5.33 with stroke size 1.0".

On the Mac, the original graphic file format was PICT, which stood for, of course, "Picture". PICT is *both* a vector and bitmap format.

Unfortunately, the vector side of PICT has only a limited vector drawing primitive set - likes Lines and Circles and shadings - but lacks the most important drawing primitive for truly beautiful rendering: Bézier curves.

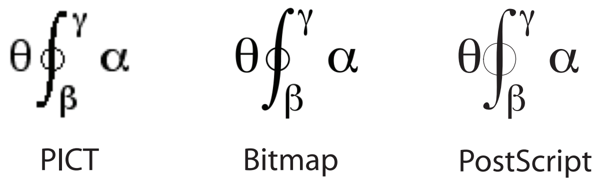
You can insert bitmap images inside of the PICT container. So if you have a PICT file, you might have some vector image commands, and some bitmap pieces, which will not scale well.

As a result, PICT's vector images are rudimentary and don't look very nice, and they are limited to the natural Mac resolution of 72dpi (dots per inch). To add further insult to injury, the PICT images do not interchange with other operating systems like Windows nor iOS nor Linux.

When Apple moved from OS9 to OSX, Steve Jobs knew he had to embrace a more modern core drawing format.

The ubiquitous vector drawing format is Postscript. Even 40 years after its creation by Adobe, it continues to be the print standard. If you have something in PostScript format, your output will look beautiful at any resolution, and print beautifully as expected - if you have a PostScript printer, that is. At least the vector portions will be beautiful: PostScript, too, can have clunky bitmaps inserted into its vector drawing commands.

Here is an integral made in *PrintMath*, showing the 3 different levels of Mac image beauty:



Hopefully you may see the visual differences, especially on the integral sign: the PICT is clunky and jagged, the Bitmap is smoother, but still a bit jagged, and the PostScript is ... amazing - the curved integral ends, the fineness of the circle through the integral sign, the crispness of the greek letters.

If all three of these images "just look like the same integral to you", then you will not be interested in doing the little extra bit of work to get your expressions into PostScript beauty. Probably you should plan to use the Bitmap images and be plenty happy.

But likely you are drawn to *PrintMath* because you appreciate the higher quality images in PostScript that *PrintMath* is capable of creating for you in your Word documents.

So why didn't Apple just use PostScript as their native drawing format? Well, after a number of years, they did.

PDF – Portable Document Format – is a little sibling of PostScript. PDF can be thought of as "super simplified PostScript".[1] When Apple

1 *If you open up a PDF file and look at it, you might think at first that it is actually a binary format, like PNG or GIF or JPG. In actuality, PDF is broken up into blocks just like*

switched from OS9 to OSX, Apple also changed its native drawing format from PICT to PDF.

But before PDF became the core drawing standard, the folks at Apple knew they had to give developers some way to overcome the clunky PICT format: one of the big selling points of Mac computers were the beautiful output from (expensive) Apple LaserWriters.   These first general usage PostScript printers did not like PICT either – they were driven by pure PostScript, and their output looked fantastic.

Apple's solution was a hack.

In the PICT file format specification, there is a structure for text comments, known as **PicComment**.   I suppose originally the idea was some developer might put some comments into the PICT to do something like "Photo taken in New York". Whatever is in PicComment does not get drawn, and other programs are supposed to leave these PicComments alone. The other bitmap images have these Comment fields as well for similar purposes.

The idea from Apple was that for an image in the PICT format, a develper could inject higher quality PostScript into these hidden PicComments, creating the format PICT+PS.

On the screen, the PICT drawing is used, as that is the native drawing on a Mac (or used to be).   When this PICT is sent to a PostScript printer, the *printer driver* would ask the image: do you per chance have any PostScript riding piggyback? If so, the printer driver would grab the PostScript instead and send the higher-refined PostScript to the printer instead of the PICT commands.

This PICT+PS became the standard for high quality printouts of images on the Mac.   A monumental hack.   But those LaserWriters had to look impressive while we waited for the real solution of PDF.

This same trick of hiding secret code in the PicComment area was

---

Postscript, and the "binary looking" code is simply compressed postscript code. If you looked at that code "deflated", then you would recognize how PDF and PS are nearly the same.

the basis of the EGO idea - Editable Graphical Object – pioneed by the creator of *Expressionist*, Allan Bonadio. Injected into the PicComment area could also be the native *Expressionist encoding* to be able to re-edit the graphical object in the program that created the graphical object – in this case, *Expressionist* (the forerunner to *PrintMath*). This encoding is hidden to all PICT readers and renders, but when *Expressionist* would open the PICT, it would see *Expressionist's* encodings, and instead open an the original editable expression in *Expressionist*.

So just like the printer drivers scan the PICT+PS for secret hidden PostScript code, *PrintMath* scans these PicComments to its secret hidden encoding to *reconstruct* the *PrintMath* objects that originated the graphical object.

Nowadays, the EGO idea is used in a variety of softwares, most notably Adobe Fireworks, which outputs PNGs and GIFs and JPGs for web usage, but those PNG and GIF and JPG files have their own comment structures which hold the secret encodings of the source files so that Fireworks can re-edit these images. That's why Fireworks appears to be "magical" in that it can edit PNG files as if they are not really PNG (bitmap) files.

Luckily, the newer PDF versions also have a Comment-type feature that we can exploit to hide our *PrintMath* encodings for re-editability.

Regarding the goal of getting the best possible quality output of *PrintMath* expressions in MSWord, the wide swath of versions of MS Word for Mac fall into 3 main groups:

MSWord Version:	≤ 2004	2008	≥ 2011
Best Solution:	PICT+PS	PDF	PDF

With this in mind, we start with the MS Word ≤ 2004 versions.

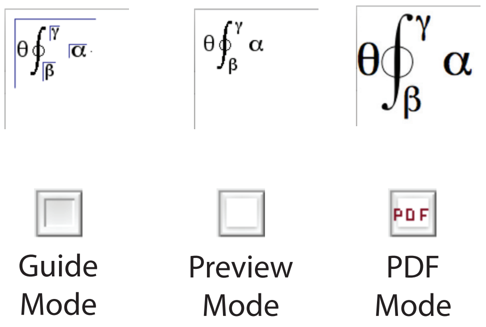
3.3 The +PS Word Options for MS Word ≤ 2004

Let’s start the task of getting our sample expression from *PrintMath*

$$\theta\oint_{\beta}^{\gamma}\alpha$$

into MSWord 2004 or earlier. Even if you are using MSWord 2008 or later, this is still a good section to read to understand the issues.

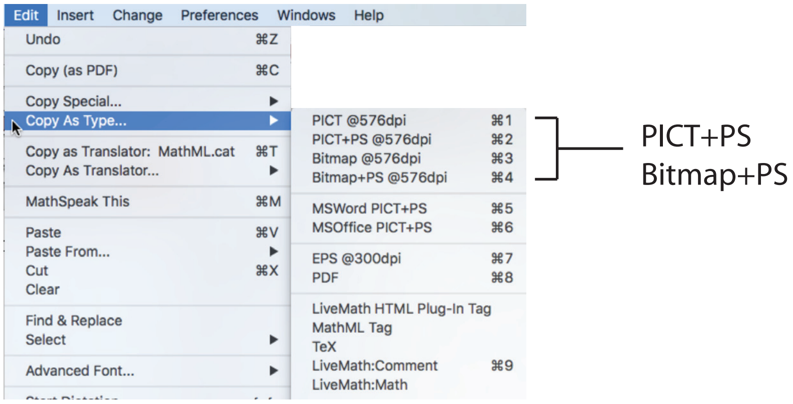
When we make this expression in *PrintMath*, we will see these 3 preview images by clicking on the Preview button:

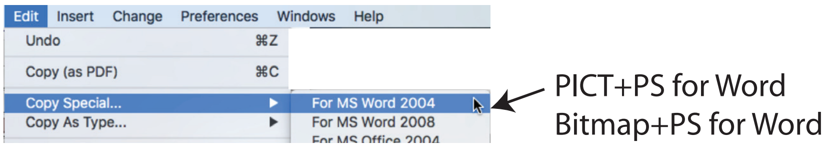


When we copy/paste this expression from *PrintMath* to MS Word, we have the following +PS options to try:

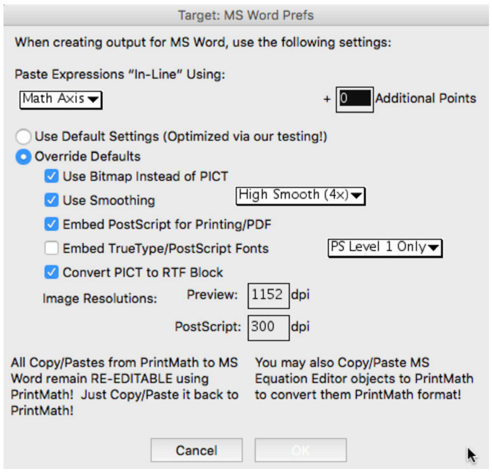
- PICT
- PICT+PS
- PICT+PS for Word
- Bitmap
- Bitmap+PS
- Bitmap+PS for Word

from these menus:





To toggle into to changing the Copy Special for MS Word 2004 from PICT to Bitmap, we have to do a little bit of extra work and go to the **Preferences ▶ Target Prefs ▶ MS Word** and change the Use Bitmap Instead of PICT option:


Target: MS Word Prefs

When creating output for MS Word, use the following settings:

Paste Expressions "In-Line" Using: Math Axis + 0 Additional Points

( ) Use Default Settings (Optimized via our testing!)
(●) Override Defaults
- ☑ Use Bitmap Instead of PICT
- ☑ Use Smoothing: High Smooth (4x)
- ☑ Embed PostScript for Printing/PDF
- ☐ Embed TrueType/PostScript Fonts: PS Level 1 Only
- ☑ Convert PICT to RTF Block

Image Resolutions: Preview: 1152 dpi; PostScript: 300 dpi

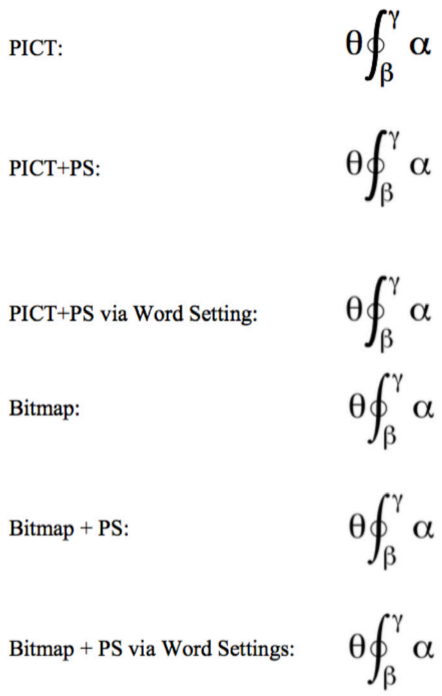
All Copy/Pastes from PrintMath to MS Word remain RE-EDITABLE using PrintMath! Just Copy/Paste it back to PrintMath!

You may also Copy/Paste MS Equation Editor objects to PrintMath to convert them PrintMath format!

Our goal is to get the best possible output, which is from PostScript.

Let's see what happens if we get these 6 options pasted into MS Word.

First, we will see what the screen looks like:



The bitmap versions look the best, but they come with a cost: file size will increase with the more bitmaps you paste into a Word document. If you only have a few expressions, this is no big deal. But if you have 1000s of expressions, this can become an issue.

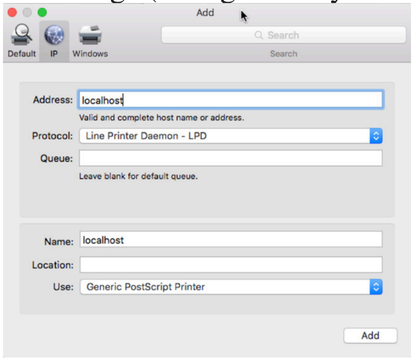
Once we get the *PrintMath* expression into Word, we need to output from Word to ... something.   Maybe you just hit “Print” and that is your output, but you won’t necessarily get the beautiful PostScript output from just hitting Print without doing a little extra work.

Our output goal will be to “export” the MS Word document to PDF. Once you document is in PDF, you can then print it, or whatever you want to do with it (share it with someone, post to a website, etc).

<ins>Warning</ins>: *if you send your .doc or .docx document to a Windows computer with your +PS images, the PrintMath re-editing encoding will be*

lost! Thank you, Microsoft.

To get the “Print To PDF” feature of the Mac to actually use the embedded PostScript code, you need to have a PostScript-capable printer as part of the process. You may have a PostScript-capable printer, and so this is no big deal; but you may not have a PS printer (e.g. InkJet printers), and you have to be a little more clever: you may add a “Generic Postscript Printer” by selecting Add Printer in the Print dialog box, with these settings (and ignore any error messages).



Add

Address: localhost

Valid and complete host name or address.

Protocol: Line Printer Daemon - LPD

Queue:

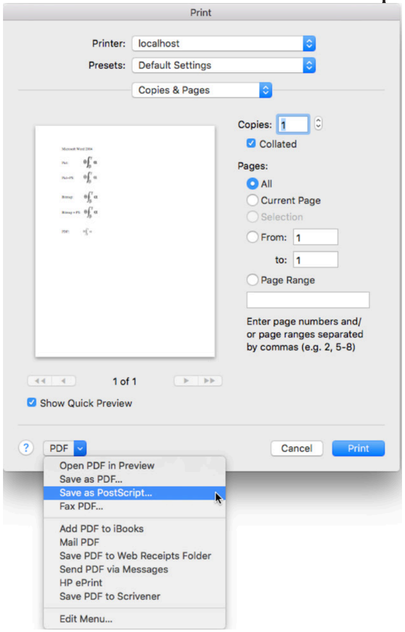
Leave blank for default queue.

Name: localhost

Location:

Use: Generic PostScript Printer

Once you have the Print dialog box pointing to a PostScript printer of some kind, you may then choose Save as PostScript... :



Print

Printer: localhost

Presets: Default Settings

Copies & Pages

Copies: 1

☑ Collated

Pages:

● All

○ Current Page

○ Selection

○ From: 1

to: 1

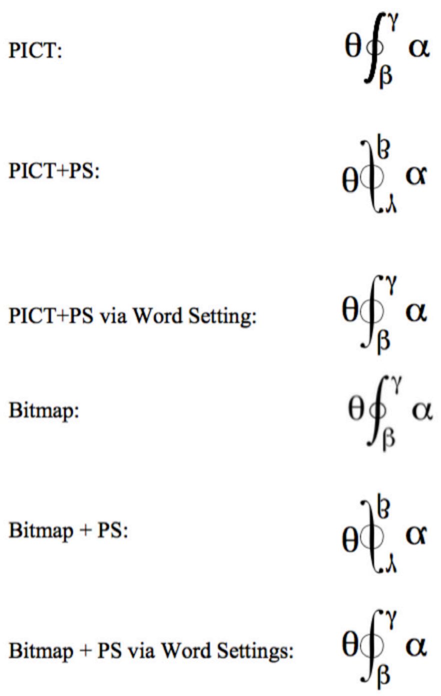
○ Page Range

Enter page numbers and/ or page ranges separated by commas (e.g. 2, 5-8)

1 of 1

☑ Show Quick Preview

This will output a “.ps” PostScript file, running the output routine through a PostScript printer driver, thus, theoretically, sticking to PostScript the entire way.   If we double-click on the ".ps" file that was created, Preview will convert that PostScript file to PDF, and voilà:



Oh myyyyy!   What happened?   The problem was that Microsoft, sometime after MS Word 5.1, required a "flip" of the PostScript coordinates in this PICT+PS trickery.   Yet other programs that also supported this PICT+PS format, like Adobe FrameMaker, Nisus Writer, etc., all used the "correct" PostScript coordinates. So if we did this same experiment in one of those other programs, the PICT+PS that looks correct for MSWord would be inverted in these other programs.

Thus necessitated the creation of the *PrintMath* Target Application menus: we had to know which program you were creating PICT+PS for, and correctly invert the PS if you were working with MS Word.

Now that you have your output in the form of a PDF file, even if you now print to an InkJet (non-PS) printer, your output will look .... fantastic, at least as good as your inkjet printer can do.

If you try to use just the "Open in Preview" or "Save to PDF..." options, the printer driver may not actually force the PostScript issue, requiring Word to actually give it the unlying higher quality PostScript.

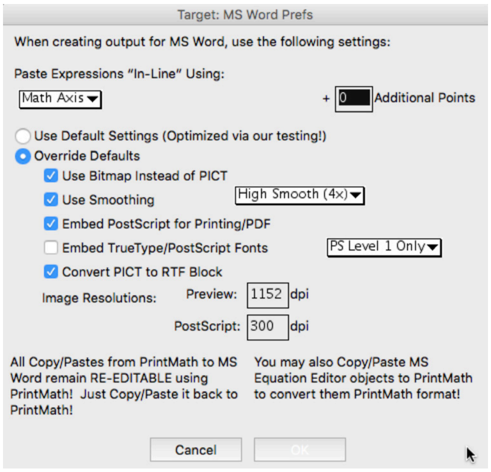
The lesson to be learned is: if your expressions do not look like they are being created from the higher quality PostScript code, you may need to a little work to coax Word and MacOS to give out the PostScript versions as described here.

In Microsoft Word ≤ 2004, when you copy/paste the *PrintMath* expression, you can go back to *PrintMath* and do a paste, and be able to re-edit the *PrintMath* expression in *PrintMath*. Not as slick as OLE, of course, but functional.

It is of note that this feature broke in MS Word 2008. There is a work-around, described in §3.7.

3.5 Vertical In-Line Centering

When pasting a *PrintMath* expression into MS Word 2004 or earlier, you may employ the Convert PICT to RTF Block option:


Target: MS Word Prefs

When creating output for MS Word, use the following settings:

Paste Expressions "In-Line" Using:

Math Axis + 0 Additional Points

( ) Use Default Settings (Optimized via our testing!)
(● Override Defaults)

- ✓ Use Bitmap Instead of PICT
- ✓ Use Smoothing — High Smooth (4x)
- ✓ Embed PostScript for Printing/PDF
- ☐ Embed TrueType/PostScript Fonts — PS Level 1 Only
- ✓ Convert PICT to RTF Block

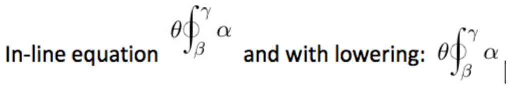
Image Resolutions: Preview: 1152 dpi; PostScript: 300 dpi

All Copy/Pastes from PrintMath to MS Word remain RE-EDITABLE using PrintMath! Just Copy/Paste it back to PrintMath!

You may also Copy/Paste MS Equation Editor objects to PrintMath to convert them PrintMath format!

which will take the PICT+PS, and wrap it up into Microsoft's Rich Text Format (RTF), which is, in some sense, the native language of

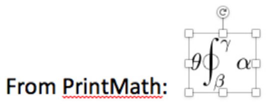
Word. The effect here is provide semi-automated expression lowering for inline equations:



Of course, MathType/EE does this automatically for you. But then you lose the control to be able to set the lowering amount to how much you want.

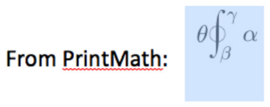
But even with this RTF lowering, you may want to fine-tune the lowering to your tastes. So it is important to know how to do this (and required for MSWord ≥ 2008 as the RTF option is not available).

When you paste a *PrintMath* expression into MS Word, you will see a graphical object in Word:

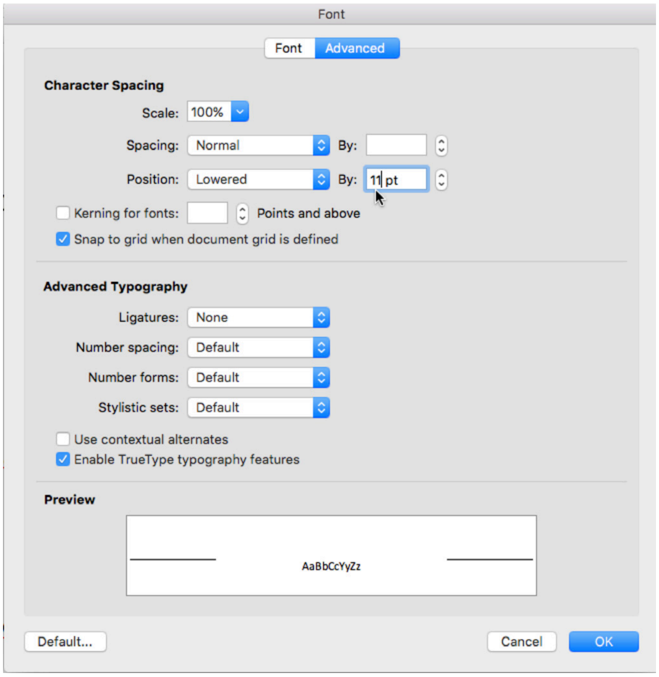


In Word, inexplicably, you cannot lower a graphical object by itself.

So what you must do is put a space (or two) on either side (or just one side) of the expression, and drag select the extra space with the expression:



Now, in Word, you will do menu **Format ▶ Font**, which can be quickly executed with keystrokes: $\boxed{\text{⌘}}$ $\boxed{D}$



Font

Font | Advanced

Character Spacing

Scale: 100%

Spacing: Normal By:

Position: Lowered By: 11 pt

Kerning for fonts: Points and above

[✓] Snap to grid when document grid is defined

Advanced Typography

Ligatures: None

Number spacing: Default

Number forms: Default

Stylistic sets: Default

Use contextual alternates

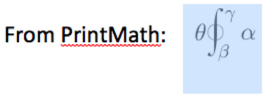
[✓] Enable TrueType typography features

Preview

AaBbCcYyZz

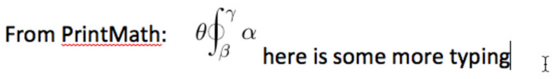
Default... Cancel OK

Simply change the Postion: Lowered to the amount you want to lower. In this case, 11pt is about right.



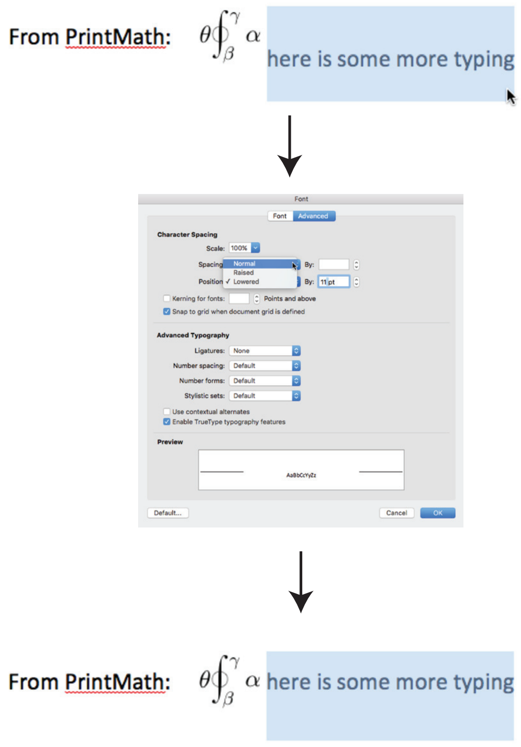
From PrintMath: $\theta\oint_{\beta}^{\gamma}\alpha$

One thing to watch out for after you do a lowering like this is the new cursor position may still be affected by the lowering. So if you put the cursor directly after the expression object and start typing, you may not get what you intended:



From PrintMath: $\theta\oint_{\beta}^{\gamma}\alpha$ here is some more typing

There are many ways to combat this problem. One approach is to put an extra buffer space (or text) after the object so the lowering is restricted to the expression only. You can correct the lowered text by selecting the text and making it normal again:


From PrintMath: $\theta\oint_{\beta}^{\gamma}\alpha$ here is some more typing

Font

Font | Advanced

Character Spacing

Scale: 100%

Spacing: Normal — By:

Position: ✓ Lowered (Normal, Raised, Lowered) — By: 11 pt

☐ Kerning for fonts: Points and above

✓ Snap to grid when document grid is defined

Advanced Typography

Ligatures: None

Number spacing: Default

Number forms: Default

Stylistic sets: Default

☐ Use contextual alternates

✓ Enable TrueType typography features

Preview

AaBbCcYyZz

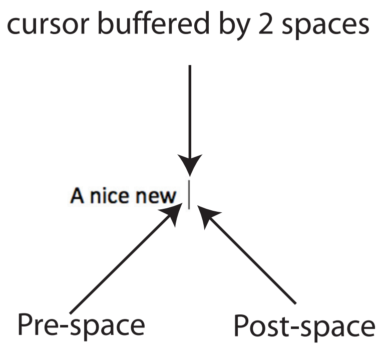
Default... Cancel OK

From PrintMath: $\theta\oint_{\beta}^{\gamma}\alpha$ here is some more typing

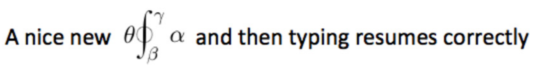
Sometimes the "lowered state" in Word can get maddening, where you may have a whole paragraph or more of text and objects that are all lowered, but since they look correct next to each other, you don't notice until you select the text and do a $\boxed{\text{ ⌘}}$ $\boxed{D}$ to see.

Probably the best practice to get into is:

1. Copy expression in *PrintMath*.
2. In Word, place cursor where you want to insert *PrintMath* expression.
3. Insert 2 or more space characters (or extra text words).



4. Place cursor in between these newly inserted spaces.
5. Paste.

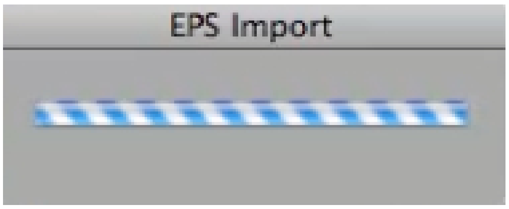

A nice new $\theta\oint_{\beta}^{\gamma}\alpha$ and then typing resumes correctly

The trouble starts in Word when you paste a graphics object onto the end of a line, and then lower that graphics object.

3.4 MS Word 2008

Starting in MS Word 2008, you can copy/paste PDF expressions from *PrintMath*. We will talk about this in a moment. First, the fun that started in MS Word 2008 with PICT+PS.

When you copy PICT+PS in *PrintMath* and then paste into Word 2008, this version of Word senses the hidden PostScript code and converts that PostScript into PDF format, and pastes in the resultant PDF into Word.



While that may have seemed like really cool feature from Microsoft, the next result - not just for *PrintMath* but for a whole host of other programs that used the PICT+PS trick - was that the PostScript to PDF conversion completely wiped out the other important information hidden in the PicComments: the re-editability encoding is obliterated by this new conversion feature in Word 2008!

Since we want to be able to re-edit our *PrintMath* expressions, we need to stop using the PICT+PS format as of MS Word 2008.

Thankfully, you may Copy as PDF in *PrintMath* and paste the resulting PDF into MS Word 2008.

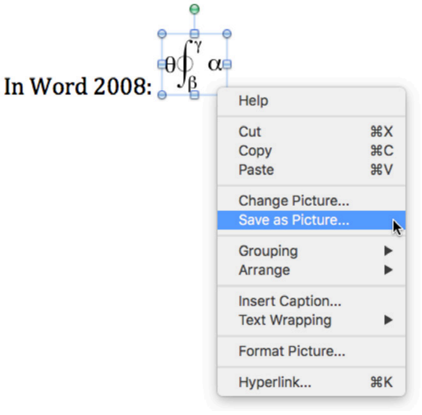
Unfortunately, what MS Word 2008 does to this PDF is ... heartbreaking.

Word 2008 takes this pasted PDF file and stores it in one of its internal directories. The new ".docx" format is actually just a ".zip" archive of files and directories: you may look at these files by changing the ".docx" extension to ".zip", and the double-click to have Archive Utility unzip this file.

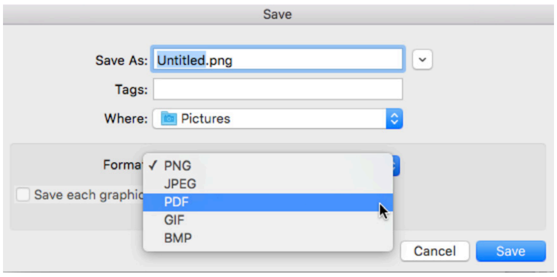
While storing away this PDF, Word 2008 creates a screen bitmap PNG (and rather low resolution as well!). So when you go to Copy/Paste this PDF expression in Word 2008, what you are copying is actually this low resolution bitmap PNG file. Of course, this PDF→PNG conversion obliterates the stored re-editability encoding hidden in the PDF file. So when you try to Paste this copied bitmap PNG file back into *PrintMath*, there is no re-editability encoding to be found.

The only workaround is to do the following:

1. Select the *PrintMath* PDF expression that was pasted into MS Word 2008.
2. Right-click (option-click) and choose Save As Picture:



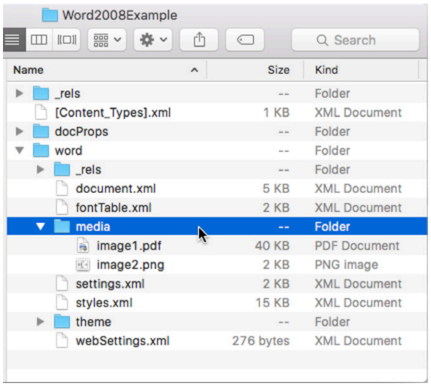
3. In the resulting Save as Picture dialog, choose format PDF:



4. The resulting saved output PDF file (named whatever you like), is actually a copy of the stored PDF, which is the original PDF file from *PrintMath* - with its re-editability encoding intact!
5. In *PrintMath*, open this output PDF file from Step 4, and it will be re-editable. Change as you prefer, then Copy as PDF and paste back into Word 2008.

Not the most ideal workaround, but at least there is some way to save your expressions. This method also works with all pasted PDFs into Word ≥ 2008 as well, from the *PrintMath* program, or other programs that produce re-editable PDFs, so it is a useful trick indeed.

The stored PDFs inside of the .docx structure are all renamed “Image 1”, “Image 2”, etc., which is how Word keeps track of pasted in image objects. So if you really need to hunt around and/or do batch conversions of your pasted-in graphics, you can open up the .docx files (by unzipping them) and gain access to all of these files in a single location.



Name	Size	Kind
_rels	--	Folder
[Content_Types].xml	1 KB	XML Document
docProps	--	Folder
word	--	Folder
_rels	--	Folder
document.xml	5 KB	XML Document
fontTable.xml	2 KB	XML Document
media	--	Folder
image1.pdf	40 KB	PDF Document
image2.png	2 KB	PNG image
settings.xml	2 KB	XML Document
styles.xml	15 KB	XML Document
theme	--	Folder
webSettings.xml	276 bytes	XML Document

Be careful with Copy/Paste inside of Word 2008. If you Copy/Paste in a PDF (from, say, *PrintMath*), and then inside of Word 2008, you select that PDF expression and do a Copy, and Paste to another location in the Word document (or, gasp - copy to another Word document), what you are actually doing is grabbing the bitmap PNG that Word 2008 made for you, and making a copy of that. So if your final output from Word has a nice PDF expression output, followed by lousy PNG bitmap copies of the original, you know why.

All of this behavior has been cured in MS Word 2011 and 2016, so unless you are forced to use Word 2008 for some reason, upgrading will correct these problems.

3.5 MSWord ≥ 2011 Saves Us

Thankfully, with MSWord 2011, and with the current (as of this writing) MSWord 2016 versions, we can safely Copy/Paste the PDF expressions from *PrintMath*, preserving editability.

Starting with MSWord 2016, pasting in PICT+PS does not work at all any longer! So... PDF it is!

What does not work so well (perhaps yet) is the PDF+RTF combination to achieve the “expression lowering” for in-line equations. We will have to do those manually using the technique described in §3.5.

The problem is that when you paste in a PDF into Word ≥ 2011, that PDF is preserved, which is good, but an RTF copy of the PDF is created in “EMF+” format, which is Microsoft’s Enhanced MetaFile format from Windows. The functions necessary to create this EMF+ with embedded PDF are not available on the Macintosh side (so far as I know) in any format other than internal to MS Word ≥ 2011.

So, the vertical lowering of inline expressions must be done manually, using the ⌘ D method.

§

§

§

§

§

§

§

§

§

§

§

§

§

§

§