

# **Advanced Find and Replace in Microsoft Word**

**Jack M. Lyon**

© 2002 by the Editorium. All rights reserved.



## Contents

Preface.....	3
Searching with Codes .....	4
Microsoft Word’s Built-in Codes .....	4
Searching for Special Characters .....	5
ANSI Character Codes.....	6
What’s That Character? .....	7
Replacing with “Find What Text” .....	8
Example: Formatting Note Numbers .....	8
Using Wildcards.....	10
The Basics.....	10
Wildcard Combinations .....	11
Wildcard Ranges.....	13
Wildcard Grouping .....	14
Using the “Find What Expression” Wildcard .....	15
Wildcards in the Real World.....	18
Example 1 .....	18
Example 2 .....	18
Example 3 .....	19
Two-Step Searching.....	20
Step 1 .....	20
Step 2 .....	21
Reference .....	22

## Preface

This document is a compilation of articles that originally appeared in my email newsletter, *Editorium Update*. Microsoft Word’s advanced search features are extremely powerful, but they’re also virtually undocumented. Most explanations of their use have been limited to a simple table of various wildcards. I wrote these articles to remedy that situation.

As you read these articles, you may want to actually try the techniques in Microsoft Word, using some junk documents that you no longer need. Doing so will help you learn more than just reading the instructions.

I hope these articles will help you understand how useful Word’s advanced search features can be and how much time they can save you. Using these features, you can quickly fix repetitive problems that would take hours to correct by hand.

Enjoy!

## Searching with Codes

Why should you, as an editor, writer, or publisher, care about something as “technical” as searching with codes? Because they make it possible to find and replace things you ordinarily couldn’t, such as paragraph breaks, dashes, and symbols. This can be a big help in cleaning up all kinds of editorial and typographical problems that you’d otherwise have to fix by hand.

There are actually two different kinds of codes:

1. Microsoft Word’s built-in codes (such as ^p for paragraph breaks and ^t for tabs).
2. ANSI character codes (such as ^013 for paragraph breaks and ^009 for tabs).

Both kinds of codes are useful, but the list of ANSI codes includes every character (not including Unicode characters) you can use in Microsoft Word. Later I’ll provide a list of these codes and explain how to use them.

### *Microsoft Word’s Built-in Codes*

First, I’ll give you a list of Word’s built-in codes, which you can use in Microsoft Word’s Find and Replace dialog (Edit/Replace). For example, if you wanted to find an em dash, you’d enter the following code in the “Find what” box: ^+

To replace it with an en dash, you’d enter this in the “Replace with” box: ^=

You can also insert Word’s built-in codes by clicking the Special button in the Find and Replace dialog and then selecting the item you need. Please note that you can use some of the codes only in finding text, others only in replacing, and others in either one.

You can also use combinations of codes. For example, you could search for tabs followed by paragraph breaks (^t^p) and replace them with paragraph breaks alone (^p).

And now, here’s the list.

### **Note**

For easy reference, all the code lists are also included at the end of this document.

## Codes You Can Use in the “Find What” and “Replace With” Boxes

Character	Find What	Replace With
Annotation Mark (comment)	^a	
Any character	^?	
Any digit	^#	
Any letter	^\$	
Caret character	^^	^^
Clipboard contents		^c
Column break	^n	^n
Contents of the Find What box		^&
Em dash	^+	^+
En dash	^=	^=
Endnote mark	^e	
Field	^d	
Footnote mark	^f	
Graphic	^g	
Line break	^l	^l
Manual page break	^m	^m
Nonbreaking hyphen	^~	^~
Nonbreaking space	^s	^s
Optional hyphen	^-	^-
Paragraph mark	^p	^p
Section break	^b	
Tab character	^t	^t
White space	^w	

### *Searching for Special Characters*

As I said above, Microsoft Word has ANSI character codes you can use to find certain items that are not usually visible in the text:

- For a carriage return, you can use ^013.
- For a section break, you can use ^012.
- For a word space, you can use ^032.

Of course, you can also use Word’s built-in codes from the table above, which you can insert into the Find dialog’s “Find what” box by clicking the “Special” button:

- For a carriage return, you can use ^p.
- For a section break, you can use ^b.
- For a word space, you can use ^w  
(actually, any white space).

So why would you want to use the first codes?

Because if you’re finding something by using wildcards, the second ones won’t work. For example, let’s say that (for some reason) you’re searching for “wh” followed by any other

character (the wildcard for which is “?”), followed by a carriage return. In the Find dialog’s “Find what” box, you enter this: wh?^p

And to make Word search for the wildcard rather than an actual question mark, you put a check in the box labeled “Use wildcards.”

Finally, you click the Find button. What happens? You get an error message:

^p is not a valid special character for the Find What box or is not supported when the Use Wildcards check box is selected.

“Well then, how,” you politely ask your computer, “am I supposed to find what I’m looking for?” As usual, it doesn’t reply, but here’s the answer anyway. In the “Find what” box, you enter this: wh?^013

And that will do the job. (On a PC. On a Macintosh, using numeric codes may not work when using wildcards. Here’s a little trick, however. Try enclosing the code in square brackets, and precede the code with a backslash. For example, to find a carriage return on a Mac, try using [^013].)

Ordinarily, you should probably use Word’s built-in codes, such as ^p and ^b. But when those don’t work, now you’ve got an alternative.

### ***ANSI Character Codes***

You can also search for any characters using numeric character codes (technically ANSI numbers). I’m including the list at the end of this document, with codes for both PC and Macintosh, although I make no guarantees about how the characters themselves will show up.

Also, you’ll notice that I haven’t included the codes for such ordinary characters as letters of the alphabet, since you can search for these by using the characters themselves. No code is needed.

To use the codes for finding or replacing special characters, simply insert them, preceded by a caret and a zero, in the “Find what” or “Replace with” boxes in Microsoft Word’s Find and Replace dialog box.

For example, if you wanted to find a u with an umlaut, you’d enter the following code in the “Find what” box on a PC: ^0252 On a Macintosh, you’d enter this: ^0159

You can also use many of the codes to insert special characters into your documents. To do so:

1. Turn on Num Lock for the numeric keypad.
2. Hold down the ALT key.
3. On the numeric keypad, type a zero followed by the code.
4. Release the ALT key.

The character will be inserted into your document.

### **WARNING:**

Use numeric codes to replace paragraph returns and section breaks only when absolutely necessary, because Word stores formatting information in these characters. Try to stick to Word’s built-in codes when you can. Also, be aware that some fonts assign different characters to the numeric codes. The list below should be accurate for Times New Roman on a PC and Times on a Macintosh.

## What's That Character?

But what if you're trying to find and replace some obscure character in an unusual font? Here's the scenario: You open a giant document from a client and start looking through it. But what's this? The same odd character at the beginning of every paragraph. Must be some kind of file translation error. Odder still, Microsoft Word won't let you paste the character into its Find and Replace dialog, so how are you going to get rid of them all? By hand? Horrors!

If you knew the character's numeric code, you could search for it. But this character isn't on the usual list. How can you find out its numeric code? By using our trusty NextCharacter macro:

### For Microsoft Word 6 or 7 (95):

```
'Macro starts here
NextChar$ = Str$(Asc(Selection$))
MsgBox "The code for the next character is " & NextChar$ & ".", "Next Character"
'Macro ends here
```

### For Microsoft Word 8 (97 or 98) or 9 (2000 or 2001)

```
'Macro starts here
Dim NextChar$
NextChar$ = Str(Asc(WordBasic.[Selection$]()))
WordBasic.MsgBox _ "The code for the next character is " & NextChar$ & ".", _ "Next
Character"
'Macro ends here
```

## To Create the Macro

1. Copy the appropriate macro from this newsletter.
2. Click the "Tools" menu at the top of your Word window.
3. Click "Macro."
4. In Word 97, 98, 2000, or 2001, click "Macros."
5. Make sure "Macros Available In" shows "Normal.dot."
6. Type a name for the macro in the "Macro Name" box--"NextCharacter" should do nicely.
7. Click "Create."
8. Paste the macro at the current insertion point.
9. In Word 6 or 7, click "File," then "Close," then "Yes." In Word 97, 98, 2000, or 2001, click "File," then "Close and Return to Microsoft Word."

## To Run the Macro:

1. Put your cursor in front of the character whose numeric code you want to know.
2. Click the "Tools" menu at the top of your Word window.
3. Click "Macro."
4. In Word 97, 98, 2000, or 2001, click "Macros."
5. Make sure "Macros Available In" shows "Normal.dot."
6. Select the macro (probably "NextCharacter") in the "Macro Name" box.

## 7. Click "Run."

After you run the macro, a message box will appear on your screen with the numeric code you need.

## Replacing with “Find What Text”

If you're faced with a complex task using Microsoft Word's Find and Replace feature, the “Find What Text” replacement code may come in handy. For example, let's say you need to add the HTML italic tags `<I>` and `</I>` around anything formatted with italic. (If you don't understand HTML, don't worry. You'll soon see the point of this example.) You might think you'd need a macro to add the tags, but you don't. You can easily do it like this:

1. Open the document you want to tag.
2. Open the Find and Replace dialog (click on the Edit menu; then click “Replace”).
3. With your cursor in the “Find What” box, turn on italic formatting (CTRL+I) so that the word “Italic” is displayed below the box. Make sure the box itself is empty.
4. In the “Replace With” box, enter “`<I>^&</I>`” (if you want, you can also set this box to “Not Italic” by pressing CTRL+I a couple of times).
5. Click the “Replace All” button. Any italicized text will be surrounded by the HTML italic tags.

The `^&` code in the “Replace With” box represents the text you specified in the “Find What” box. In this case, that's any text with italic formatting. What you're saying is, “Find any text in italic and replace it with itself surrounded by HTML italic codes.” As a specific example, let's take the following line,

“This is a test to *see* what will happen.”

When you use the Find and Replace procedure above, you'll get the following result:

“This is a test to `<I>see</I>` what will happen.”

You can use the same principle to manipulate text in a variety of ways:

- Put quotation marks around the titles of magazine articles that an author has italicized.
- Insert a bullet in front of every paragraph formatted with Heading 3 style. (You knew you could find style formatting, right? In the Find or Replace dialog, click the “More” button [if available], then “Format,” and then “Style.”)
- Insert “Chapter” in front of every number formatted with Heading 1 style.

And so on. Any time you need to add something to unspecified text that's formatted in a specific way, try using “Find What Text.”

### ***Example: Formatting Note Numbers***

I'll show you how to use the “Find What Text” feature to change the format of note numbers. I'm going to use footnotes as an example, but you can do the same thing with endnotes.

When you create footnotes in Microsoft Word (Insert menu | Footnotes | Footnote), the footnote numbers are formatted in superscript, like this (I'm using carats [^] to indicate superscript formatting):

<sup>1</sup> This is the text of note 1.

<sup>2</sup> This is the text of note 2.

And so on. But sometimes you might want your footnote numbers to have regular formatting and be followed by a period, like this:

1. This is the text of note 1.
2. This is the text of note 2.

Microsoft Word has no numbering option that will do this. Nevertheless, there *is* a way to do it, using "Find What Text":

1. Open a document containing footnotes (be sure to keep a backup copy of the document, just in case).
2. Make sure you're viewing the document in Normal mode (View menu/Normal).
3. Open the footnote pane (View menu/Footnotes).
4. Make sure your cursor is at the top of the footnote pane.
5. Open the Find and Replace dialog (Edit menu/Replace).
6. In the "Find what" box, enter "<sup>02</sup>"(don't include the quotation marks). <sup>02</sup> is the code that represents a footnote number.
7. In the "Replace with" box, enter "&." (don't include the quotation marks). Be sure to include the period after the ampersand. Also, in earlier versions of Word, you may need to follow the period with a space. The & code itself represents any text that was found, or in other words, the "Find What Text."
8. With your cursor in the "Replace with" box, click the "Format" button. (You may need to click the "More" button first.)
9. Click "Font."
10. In the Find Font dialog, clear the "Superscript" checkbox so that the replacement text won't be formatted in superscript.
11. Click the "OK" button to close the dialog.
12. In the Find and Replace dialog, click the "Replace All" button.

Your footnotes will now be formatted like this:

1. This is the text of note 1.
2. This is the text of note 2.

Pretty neat! Remember, however, that if you now add another footnote, its number will be formatted in the superscript default, and you'll have to fix it by hand. To do so:

1. Select the number.
2. Press CTRL+SPACE to remove the superscript format.
3. Type a period after the number.

## **WARNING:**

Be careful not to delete a note number or type a note number by hand. Microsoft Word uses a special code to represent a note number, and if you fool around with this code, you risk corrupting your file. You can, however, delete or move a note *reference* number that appears in the *body* of your document, like this,<sup>3</sup> and Microsoft Word will automatically renumber your notes, leaving their new formatting intact.

I ordinarily advise people not to mess around with automatic note numbers, because it's fairly easy to corrupt a document by doing so. If you know what you're doing, however, you can at least change the formatting of the note numbers if you really need to. Now you know how!

# Using Wildcards

## *The Basics*

When I was in the fifth grade in wintry Idaho, rather than venturing out into the cold, some fellow students and I often spent recess playing poker. (Did our teacher know about this? I can't remember.) Being *extremely* sophisticated players, we often designated jokers, deuces, and one-eyed jacks as wild cards—that is, they could represent any card in the deck. With the help of these wild cards, we had plenty of royal flushes, hands with five aces, and so on. Now that was poker!

Microsoft Word, too, has a bunch of “wild cards” (which Microsoft spells as one word) that you can use to find various combinations of characters in a document. Wildcards can get pretty complicated, but this week we'll cover just the basics.

The simplest wildcard is the question mark (?), which represents any single character. If you want to see how it works, try this:

1. Open a document with some text that you can play around with.
2. Click the “Edit” menu.
3. Click “Find.”
4. In the “Find What” box, enter a question mark (?).
5. Put a checkmark in the “Use wildcards,” or “Use Pattern Matching” box. (You may need to click the “More” button first.) Checking this box tells Microsoft Word that you're going to use a wildcard. If you didn't check the box, Microsoft Word would assume you were trying to find a question mark.
6. Click the “Find” button.

Microsoft Word will find the first character after your cursor position. Click the “Find” button again. Microsoft Word will find the next character. And so on.

That doesn't seem very useful, but let's suppose you're editing a document that was scanned from a magazine article and is riddled with typos. You notice that the word “but” shows up in various ways, including “bat” and “bet.” Let's say that this is a technical article with no references to baseball, winged mammals, or games of chance, so you decide to use the ? wildcard to find “bat” and “bet” and replace them in a single pass. Here's the procedure:

1. Click the “Edit” menu.
2. Click “Replace.”
3. Enter “b?t” in the “Find What” box.
4. Enter “but” in the “Replace With” box.
5. Put a checkmark in the “Use Pattern Matching Box.”
6. Click the “Replace All” button.

Both “bat” and “bet” will be replaced with “but.” The problem is, so will “bit.” And, unfortunately, since you can't specify “Find Whole Words Only” when the “Use Pattern Matching” box is checked, Microsoft Word will replace “better” with “butter,” “combat” with “combut,” and who knows what else. So, instead of clicking the “Replace All” button, you should click the “Replace” button for each individual item as needed.

Now you begin to see the power—and the danger—of using wildcards. Like cut-throat poker, they are not for the faint of heart. But if you know what you’re doing, they can be very useful. Unfortunately, they won’t help much in the “Replace With” box. In fact (with one exception that we’ll discuss in the future), you can’t use them there at all. Why? Because Word has no way of knowing what you want them to represent.

Let’s say you want to find “but” and replace it with either “bet” or “bat,” so you put “b?t” in the “Replace With” box and click the “Replace All” button. Word doesn’t know whether you want to replace “but” with “bet” or “bat,” so it just replaces it with the actual text “b?t.” So, basically, the only thing you can use in the “Replace With” box is actual text or certain built-in codes, mentioned earlier.

Here’s a list of wildcards for you to play with (on some junk text—don’t use a real document):

?	Finds any single character: “c?t” finds “cat,” “cut,” and “cot.”
*	Finds any string of characters: “b*d” finds “bad,” “bread,” and “bewildered.”
[ ]	Finds one of the specified characters: “b[ai]t” finds “bat” and “bit” but not “bet.”
[-]	Finds any single character in the specified range (which must be in ascending order): “[l-r]ight” finds “light,” “might,” “night,” and “right” (and “oight,” “pight,” and “qight,” if they exist).
[!]	Finds any single character except those specified: “m[!u]st” finds “mist” and “most” but not “must.” “t[!ou]ck” finds “tack” and “tick” but not “tock” or “tuck.”
[!x-z]	Finds any single character except those in the specified range: “t[!a-m]ck” finds “tock” and “tuck” but not “tack” or “tick.”
{n}	Finds exactly n occurrences of the previous character or expression: “re{2}d” finds “reed” but not “red.”
{n,}	Finds at least n occurrences of the previous character or expression: “re{1,}d” finds “red” and “reed.”
{n,m}	Finds from n to m occurrences of the previous character or expression: “10{1,3}” finds “10,” “100,” and “1000.”
@	Finds one or more occurrences of the previous character or expression, if there are any: “me@t” finds “met” and “meet.”
<	Finds the beginning of a word: “<inter” finds “interest” and “interrupt” but not “splinter.”
>	Finds the end of a word: “in>“ finds “in” and “main” but not “inspiring.”

### ***Wildcard Combinations***

Now let’s talk about how to combine wildcards, which will let you get pretty fancy about the stuff you want to find. Basically, you just need to know that you *can* combine wildcards. Then you can get as crazy as you like.

Earlier we used the “?” wildcard to find every three-letter combination starting with b and ending with t—“bet,” “but,” “bit,” “bat,” and so on—by searching for “b?t” with “Use Pattern Matching” turned on in the Find dialog box.

Now let’s say we wanted to find the same characters but add others as well. For example, we might want to find every three-letter combination starting with b and ending with d—“bed,” “bud,” “bid,” “bad,” and so on—in addition to the combinations ending in t. Can we really do that? Sure!

After bringing up the Find dialog (Edit > Find) and turning on “Use Pattern Matching,” we’ll start by entering the letter b into the “Find What” box, telling Microsoft Word to find that letter.

Next, we’ll enter the ? wildcard, which tells Microsoft Word to find any single character.

Finally, we'll enter a new wildcard: [td]. Microsoft Word will find any one of the characters specified in the brackets.

Altogether, the string of characters looks like this – b?[td] – and there we are, doing wildcard combinations! This particular combination tells Microsoft Word to find the letter b followed by any other single character followed by t or d.

How will something like this help you in editing? Suppose you're working on a manuscript in which the author has misspelled a name in nearly every way possible. You could comb through the manuscript over and over, hoping to catch all the variations. Or, you could be sure to catch them all by searching with wildcards. For example, let's say your manuscript is a book about India and the name in question is Gandhi. Your author has misspelled it as "Ghandi," "Gahndi," and "Ganhdi." (Not possible? Hah!) You can find every last one of them with the following string:

G[andh][andh][andh][andh]i

Then, if you've put the correct spelling, "Gandhi," in the "Replace With" box, you can find and replace each wrong spelling with the right one in a single pass, which is much more efficient than finding and replacing each variation separately.

You may be wondering why you couldn't just use the \* wildcard to represent the whole string of letters, like this: G\*i

You could. But remember, the \* wildcard represents any string of characters—including spaces. It's not limited to characters within a word (and neither are other wildcards). That means, in addition to finding the misspelled names, it will find the first 14 characters of the following phrase: "Go to the officer's hall." So be careful, especially if you're planning to use "Replace All" rather than finding and replacing one item at a time.

There is a way to simplify the wildcard combination, however. Consider this string:

G[andh]{3}i

It's functionally the same as:

G[andh][andh][andh][andh]i.

The {3} tells Word to find exactly three more occurrences of the previous "expression," which is [andh].

But now a complication: Suppose that our slapdash author has also spelled Gandhi's name as "Gandi." Uh-oh. Our original string won't catch that, because this new misspelling is one character shorter than our string specifies. But consider this:

G[andh]{2,3}i

The {2,3} tells Word to find from 2 to 3 occurrences of the previous expression, so this string will catch all of our misspelled variations so far.

What if we want to allow for more or fewer characters, being particularly unsure of our author? We can use this string:

G[andh]@i

The @ wildcard tells Microsoft Word to find one or more occurrences of the previous expression, if there are any. That ought to cover nearly anything our author throws at us. If we want to get a little more specific, we can use {2,}, which tells Word to look for at least two occurrences of the previous expression.

Here's a tip: What would happen if we put a lowercase g rather than a capital G at the beginning of our string? Word wouldn't find the misspelled names. Why? Because with "Use Pattern Matching" turned on, Word automatically matches case—a useful thing to know.

## ***Wildcard Ranges***

Wildcard ranges are fairly simple. You just use the [-] wildcard to tell Microsoft Word what to find. Let's continue with our example from above: b?[td]

As you probably recall, this tells Word to find the letter b followed by either t or d. In other words, it will find "bet," "but," "bit," "bat," "bed," "bud," "bid," "bad," and so on.

But what if we wanted to find "bat," "bad," "bet," and "bed" but *not* "bit," "bid," "bud," and "but"? After bringing up the Find dialog (Edit > Find) and turning on "Use Pattern Matching" (you may need to click the "More" button before this is available), we could use this wildcard combination in the "Find What" box: b[a-e][td]

This tells Word to find the letter b followed by any letter from a to e (in other words, a, b, c, d, or e) followed by t or d. (The range must be in ascending order—in other words, from a "lower" letter [such as a] to a "higher" letter [such as z].)

Here's another way to approach this:

b[!f-z][td]

Notice the exclamation mark at the front of the "range" wildcard. The exclamation mark tells Word to find every character except those specified—in this case, the letters f through z. This wildcard combination, too, will find "bat," "bad," "bet," and "bed" but not "bit," "bid," "bud," and "but."

Here's a range that I use all the time: [0-9] This little beauty finds any occurrence of a digit. What's that good for? Let's say you're editing a document with lots of numbered lists, like this:

1. Lorem ipsum dolor sit amet.
- 2 Ut wisi enim ad minim veniam.
3. Duis autem vel eum iriure dolor.

Did you notice that the number 2 has no period? Good! You must have "the eye." But if you have several long lists, you might want to let Word find these problem numbers for you. To do so, try this wildcard string:

^013[0-9]@[!.]

Pretty cryptic. But if you've been reading so far, you can probably figure this out:

- ^013 is the numeric code for a carriage return. (Note: Using ANSI codes with wildcard searches may not work on a Macintosh.)
- [0-9] represents any digit.
- @ tells Word to find one or more occurrences of the previous expression, if there are any (in this case, any digit). This is necessary in case you have lists with two-digit (or longer) numbers.
- [!.] tells Word to find any character except a period.

Piece of cake! Here are two other wildcard ranges you might find useful:

- [a-z] represents any occurrence of a lowercase letter.

[A-Z] represents any occurrence of an uppercase letter.

Remember, too, that you can use the [] wildcard (without a hyphen) to specify a whole group of characters without using a range. For example, this wildcard will find various kinds of punctuation:

[.,:;\?!\]

You may be wondering about the backslash (\) in front of the question and exclamation marks. The backslash tells Word to treat the following character as a character and not as a wildcard. (Remember, ? is the wildcard for a single character, and ! is the wildcard for “except.”)

Don’t be afraid to try all of these wildcard combinations and ranges for yourself (on some junk text, of course). As you experiment, you’ll better understand what works and what doesn’t. Then, when the need to use wildcards arises (which it will), you’ll be ready.

### ***Wildcard Grouping***

For the past few weeks we’ve been talking about using wildcards to find and replace text in Microsoft Word. This week we’ll discuss wildcard grouping, which is simply a way of telling Word that you want certain wildcards to be used together as a unit.

Continuing with our example from above, let’s say that you’re editing a document with lots of numbered lists, like this:

1. Lorem ipsum dolor sit amet.
2. Ut wisi enim ad minim veniam.
3. Duis autem vel eum iriure dolor.

Now let’s say that you want to replace the space after each number and period with a tab. After calling up the Replace dialog (Edit > Replace) and putting a check in the “Use wildcards” or “Use Pattern Matching” box, you could enter the following string of characters into the “Find What” box:

^013[0-9]@.

(You can’t see it, but there’s a space on the end of that string, and it needs to be included.) As you probably recall, this tells Microsoft Word to do the following:

- (^013) Find a paragraph mark
- ([0-9]) followed by a number
- (@) followed by one or more numbers,  
if there are any
- (.) followed by a period
- ( ). followed by a space

But that still won’t let us replace that space with a tab. Why? Because there’s no way to replace the space independently of the rest of the string—whatever the string finds *includes* the space.

So let’s try this: (^013[0-9]@.)( )

Notice that we’ve grouped the wildcards and other characters together with parentheses. (In case you can’t tell, that’s our uncooperative space between the last two parentheses.) Such groups, for reasons known only to the mathematically minded, are called “expressions,” and in this case there are two of them:

1. (^013[0-9]@.)
2. ()

Grouping things together like this makes it possible to refer to each group independently in the “Replace With” box—a wonderful thing! So in the “Replace With” box, we’ll enter this string:  
\1^t

That “\1” is an example of the little-known “Find What Expression” wildcard, which lives deep in the wilds of Redmond, Washington, and only comes out at night. It’s a backslash followed by the number one, and it tells Word to replace whatever is found by the first expression – (^013[0-9]@.) – with whatever the first expression finds. (Yes, you read that correctly.) In other words, Word replaces whatever the first expression finds with *itself*. That seems strange, but it means we can treat the second expression – () – as an independent unit, which is exactly what we need to do. (By the way, “Find What Expression” wildcards are the only wildcards that can be used in the “Replace With” box. They are simply a backslash followed by a number.) The ^t, of course, is the code for a tab.

You’ll notice that we haven’t included a “\2” code, which would replace something with whatever is found by our *second* expression, the space in the parentheses. Since we haven’t included that code, the space will be replaced by nothing—in other words, it will be *deleted* during the Find and Replace. So the relationship between the wildcards in the “Find What” string and the “Replace With” string is something like this:

Find What:	Replace With:
(^013[0-9]@.)	\1 (followed by a tab: ^t)
()	[nothing]

Now let’s try using them:

1. Start the Replace dialog (Edit > Replace).
2. Put a check in the “Use wildcards” or “Use Pattern Matching” box (you may need to click the “More” button before this is available).
3. In the “Find What” box, enter this:  
(^013[0-9]@.)()
4. In the “Replace With” box, enter this: \1^t
5. Click the “Replace All” button.

Presto! All of the spaces after your numbers will be replaced with tabs, and your list will now look like this:

- 1.<tab>Lorem ipsum dolor sit amet.
- 2.<tab>Ut wisi enim ad minim veniam.
- 3.<tab>Duis autem vel eum iriure dolor.

To me, this is like magic, and it comes in handy more often than you might think. I hope you’ll find it useful

### ***Using the “Find What Expression” Wildcard***

We’ve been talking about using wildcards to find and replace text in Microsoft Word. I introduced the “Find What Expression” wildcard (\n) and now I want to show you how to use it to move things around.

Let’s say you’ve got a list of authors, like this:

Emily Dickinson  
Ezra Pound  
Willa Cather  
Ernest Hemingway

and you need to put last names first, like this:

Dickinson, Emily  
Pound, Ezra  
Cather, Willa  
Hemingway, Ernest

You can use the “Find What Expression” wildcard to do this in a snap.

Start the Replace dialog (Edit > Replace) and put a check in the “Use wildcards” or “Use Pattern Matching” box (you may need to click the “More” button before this is available). Then, in the “Find What” box, enter this:

```
^013([A-z]@) ([A-z]@)^013
```

By now, you’ll probably understand these codes and wildcards:

`^013` represents a paragraph mark.

`[A-z]` represents any single alphabetic character, from uppercase A to lowercase z.

`@` represents any additional occurrences of the previous character—in this case, any single alphabetic character, from uppercase A to lowercase z.

`()` groups `[A-z]@` together as an “expression” representing an author’s first name. (This grouping is the key to using the “Find What Expression” wildcard in the “Replace With” box.)

The space after the first `([A-z]@)` expression represents the space between first name and last name.

The next `([A-z]@)` group represents the author’s last name.

The final `^013` represents the paragraph mark after the name.

Now, in the “Replace With” box, enter this:

```
^p\2, \1^p
```

The `^p` codes represent paragraph marks. “Wait a minute,” you say. “You just used `^013` for a paragraph mark. Why the change?”

Excellent question. The answer has two parts:

1. If we could use `^p` in the “Find What” box, we would. But since Word won’t let us do that when using wildcards (it displays an error message), we have to resort to the ANSI code, `^013`, instead.
2. If we use `^p` in the “Replace With” box, Word retains the formatting stored in the paragraph mark (a good thing). If we use `^013`, Word loses the formatting for the paragraph (a bad thing). In a list of author names, this probably doesn’t matter, but you’ll need to know this when finding and replacing with codes in more complicated settings.

Continuing with our example, `^p\2, \1^p`:

`\2` is the “Find What Expression” wildcard for our *\*second\** expression (hence the 2) in the

“Find What” box—in other words, it represents the last name of an author in our list.

, The comma follows this wildcard because we want a comma to follow the author’s last name.

A space follows the comma because we don’t want the last and first names mashed together, like this: “Pound,Ezra.”

\1 is the “Find What Expression” wildcard for our \*first\* expression (hence the 1) in the “Find What” box—in other words, it represents the first name of an author in our list.

Now click the “Replace All” button. The authors’ names will be transposed:

Dickinson, Emily  
Pound, Ezra  
Cather, Willa  
Hemingway, Ernest

You’ve always wondered how to do that, right? But now you’re wondering about middle initials. And middle names. And Ph.D.s.

All of those make things more complicated. But here, in a nutshell, are the Find and Replace strings you’ll need for some common name patterns (first last, first middle last, first initial last, and so on). First comes the name pattern, then the Find string, and finally the Replace string, like this:

NAME PATTERN  
FIND WHAT  
REPLACE WITH

William Shakespeare  
^013([A-z]@) ([A-z]@)^013  
^p\2, \1^p

Alfred North Whitehead  
^013([A-z]@) ([A-z]@) ([A-z]@)^013  
^p\3, \1 \2^p

Philip K. Dick  
^013([A-z]@) ([A-Z].) ([A-z]@)^013  
^p\3, \1 \2^p

L. Frank Baum  
^013([A-Z].) ([A-z]@) ([A-z]@)^013  
^p\3, \1 \2^p

G. B. Harrison, Ph.D.  
^013([A-Z].) ([A-Z].) ([A-z]@, ) (\*)  
^013  
^p\3 \1 \2, \4^p

J.R.R. Tolkien  
^013([A-Z].)([A-Z].)([A-Z].) ([A-z]@)^013

`^p\4, \1\2\3^p`

That list doesn't show every pattern you'll encounter, but it should provide enough examples so you'll understand how to create new patterns on your own—which is the whole point. Once you've created all of the patterns you need, you could record all of that finding and replacing in a single macro that you could run whenever you need to transpose names in a list.

## Wildcards in the Real World

You might be interested in seeing some of the wildcard combinations I've used recently in a few actual editing projects. Maybe you'll find them useful too.

### *Example 1*

The manuscript I've been working on has lots of parenthetical references like this:

(Thoreau, Walden, p 10.)

You'll notice that there's no period after the p. To fix these references, I used the following string in Microsoft Word's "Find What" box in the Replace dialog (Edit > Replace), with "Use Wild Cards" (or "Use Pattern Matching") turned on: `p ([0-9]@.\))`

That's an odd-looking thing with its double parentheses, but its meaning becomes clear when you consider that the first closing parenthesis represents the closing parenthesis of the reference. The backslash in front of it tells Word to treat it as a character rather than the end of a group "expression." So the whole string says this:

1. Find a p followed by a space.
2. Find, as a group, one or more digits followed by a period followed by a closing parenthesis.

I put this in the "Replace With" box: `p. \1`

And that string says this:

1. Replace the p followed by a space with p followed by a period and a space.
2. Replace the rest of the "Find What" string (the group in parentheses) with itself.

When I was finished finding and replacing, the references looked like this:

(Thoreau, Walden, p. 10.)

### *Example 2*

Here's another example from the manuscript I've been working on:

(Genesis 8:26)

You'll notice that there's no period before the closing parenthesis. Wanting to fix these, I put this string in the "Find What" box: `([0-9]@[0-9]@)\)`

It says:

1. Find, as a group, any number of digits followed by a colon followed by any number of digits.
2. Find a closing parenthesis character.

I put this in the “Replace With” box: \1.)  
And that string says:

1. Replace the group with itself.
2. Replace the closing parenthesis with a period and a closing parenthesis.

When I was finished finding and replacing, the references looked like this:

(Genesis 8:26.)

“Why,” you may be wondering, “did you have to use wildcards? Why didn’t you just find a closing parenthesis and replace it with a closing parenthesis and a period, like this: Find What: )  
Replace With: .)”

I couldn’t do that because the manuscript had other parenthetical items (like this one) that didn’t need a period. Using wildcards makes it possible to find exactly the items you want and ignore those you don’t.

### ***Example 3***

The manuscript had Bible references that looked like this:

II Corinthians  
II John  
II Kings

I wanted them to look like this:

2 Corinthians  
2 John  
2 Kings

I put this in the “Find What” box: II ([A-Z])  
– which says:

1. Find I followed by I followed by a space.
2. Find any capital letter.

And I put this in the “Replace With” box: 2 \1 – which says:

1. Replace the II with a 2.
2. Replace the capital letter with itself.

Worked like a charm.

“Why,” you ask, “didn’t you just replace II with 2 throughout the manuscript rather than use wildcards?” Well, I could have. But I was also thinking about other entries like these:

I Corinthians  
I John  
I Kings

Obviously, I couldn’t just replace I with 1 throughout the manuscript, so I used – I ([A-Z]) – in the “Find What” box and – 1 \1 – in the “Replace With” box and that took care of the problem.

I hope you’re beginning to see how powerful wildcards can be and how much time they can save while you’re editing a manuscript. Using wildcards, you can quickly fix repetitive problems

that would take hours to correct by hand. I highly encourage you to try them, but I also urge you to back up your documents and experiment on some junk text before using wildcards in the “real world.” Also, try finding and replacing items individually before replacing all of them globally. Then you’ll know that the wildcards you’re using actually do what you need to have done.

## Two-Step Searching

While editing in Microsoft Word, I often need to find something that’s *partially* formatted and replace it with something else. For example, let’s say a manuscript has a bunch of superscript note numbers preceded by a space that’s *\*not\** in superscript. Here’s an example (with carets indicating superscript):

Lorem ipsum dolor sit amet. ^1^

I’d like to have Word find all such spaces and replace them with nothing (in other words, delete them), so that the result looks like this:

Lorem ipsum dolor sit amet.^1^

Unfortunately, that doesn’t seem possible. I can open Word’s Replace dialog (Edit > Replace) and set the “Find What” box to superscript, but the space isn’t superscript, and the manuscript has thousands of spaces that *don’t* precede a superscript number. It also has numbers that aren’t superscript (like 2001), so I can’t just find spaces preceding numbers. What’s an editor to do?

Find and replace the spaces in two steps rather than one:

1. Mark the superscript with codes.
2. Delete the spaces and codes.

### Step 1

To mark the superscript with codes, do this:

1. Open Word’s Replace dialog by clicking the “Edit” menu and then “Replace.”
2. Put your cursor in the “Find What” box and make sure the box is empty.
3. Click the “Format” button.  
(You may need to click the “More” button first.)
4. Click “Font.”
5. Put a checkmark in the “Superscript” box.
6. Click the “OK” button. The “Find What” box should now be set to superscript.
7. Put your cursor in the “Replace With” box.
8. Type the following string in the “Replace With” box: <S>^&
9. Click “Replace All.”

All of your superscript numbers will be replaced with themselves, preceded by <S>, which is a code I just made up to indicate superscript. In other words, your sentences will now look like this:

Lorem ipsum dolor sit amet. ^<S>1^

Feel free to make up your own codes for whatever you need (italic, bold, paragraph styles, and so on).

The other code in the “Replace With” box, ^&, is Microsoft Word’s “Find What Text” code (discussed earlier), which represents the text that was found (the superscript numbers).

## ***Step 2***

To delete the spaces and codes, do this:

1. Open Word’s Replace dialog by clicking the “Edit” menu and then “Replace.”
2. Type <S> in the “Find What” box. (You can’t see it very well in here, but there’s a space in front of that code, and it needs to be there.)
3. Click the “No Formatting” button so you’re no longer finding superscript, which is now represented by the <S> code.
4. Put your cursor in the “Replace With” box and make sure the box is empty.
5. Click “Replace All.”

All of the spaces in front of the codes (and thus in front of the superscript numbers) will be deleted, as will the codes themselves, leaving your sentences looking like this:

    Lorem ipsum dolor sit amet.^1^

You can use this little two-step trick any time you need to find and replace partially formatted text. Now that you know how, that will probably be quite often.

## Reference

### Codes You Can Use in the “Find What” and “Replace With” Boxes

Character	Find What	Replace With
Annotation Mark (comment)	^a	
Any character	^?	
Any digit	^#	
Any letter	^\$	
Caret character	^^	^^
Clipboard contents		^c
Column break	^n	^n
Contents of the Find What box		^&
Em dash	^+	^+
En dash	^=	^=
Endnote mark	^e	
Field	^d	
Footnote mark	^f	
Graphic	^g	
Line break	^l	^l
Manual page break	^m	^m
Nonbreaking hyphen	^~	^~
Nonbreaking space	^s	^s
Optional hyphen	^-	^-
Paragraph mark	^p	^p
Section break	^b	
Tab character	^t	^t
White space	^w	

## Wildcards

?	Finds any single character: “c?t” finds “cat,” “cut,” and “cot.”
*	Finds any string of characters: “b*d” finds “bad,” “bread,” and “bewildered.”
[ ]	Finds one of the specified characters: “b[ai]t” finds “bat” and “bit” but not “bet.”
[-]	Finds any single character in the specified range (which must be in ascending order): “[l-r]ight” finds “light,” “might,” “night,” and “right” (and “oight,” “pight,” and “qight,” if they exist).
[!]	Finds any single character except those specified: “m[!u]st” finds “mist” and “most” but not “must.” “t[!ou]ck” finds “tack” and “tick” but not “tock” or “tuck.”
{n}	Finds exactly n occurrences of the previous character or expression: “re{2}d” finds “reed” but not “red.”
{n,}	Finds at least n occurrences of the previous character or expression: “re{1,}d” finds “red” and “reed.”
{n,m}	Finds from n to m occurrences of the previous character or expression e.g. 10{1,3} finds “10,” “100,” and “1000.”
@	Finds one or more occurrences of the previous character or expression, if there are any: “me@t” finds “met” and “meet.”
<	Finds the beginning of a word: “<inter” finds “interest” and “interrupt” but not “splinter.”
>	Finds the end of a word: “in>“ finds “in” and “main” but not “inspiring.”

## Ranges

[a-e]	Finds any of a, b, c, d or e
[0-9]	Finds any digit
[a-z]	Finds any occurrence of a lowercase letter.
[A-Z]	Finds any occurrence of an uppercase letter.
[!x-z]	Finds any single character except those in the specified range: “t[!a-m]ck” finds “tock” and “tuck” but not “tack” or “tick.”

## Groups

( )	Creates a wildcard group
\1	Inserts the contents of the first wildcard group in the “Replace With” text.
\2	Inserts the contents of the second wildcard group in the “Replace With” text; etc.

## Character Codes

Character	Name	Mac	PC
	footnote reference	2	2
	tab	9	9
	line break	11	11
	page/section break	12	12
	paragraph break	13	13
	column break	14	14
-	nonbreaking hyphen	30	30
	optional hyphen	31	31
	space	32	32
,	comma	226	130
f	folio	196	131
„	double comma	227	132
...	ellipses	201	133
†	dagger	160	134
‡	double dagger	224	135
^	caret	246	136
‰	per thousand	228	137
Š	cap S hacek		138
<	open angle bracket	220	139
Œ	capital oe diphthong	206	140
‘	open single quote	212	145
’	close single quote	213	146
“	open double quote	210	147
”	close double quote	211	148
•	bullet	165	149
—	en dash	208	150
---	em dash	209	151
~	tilde	247	152
™	trademark	170	153
š	l/c s hacek		154
>	close angle bracket	221	155
œ	l/c oe diphthong	207	156
ÿ	cap Y umlaut	217	159
	non-breaking space	160	160
¡	inverted exclamation	193	161
¢	cent	162	162
£	pound	163	163
¤	cell	219	164
¥	yen	180	165
	pipe	124	166
§	section	164	167
¨	umlaut	172	168
©	copyright	169	169
ª	ordinal, feminine	187	170
«	left chevrons	199	171
¬	not	194	172
-	soft hyphen	248	173
®	registered	168	174
-	macron	248	175
°	degree	161	176
±	plus or minus	177	177
²	superscript 2	50	178
³	superscript 3	51	179
´	acute accent	171	180
µ	micro	181	181
¶	paragraph	166	182
·	middle dot	225	183

Character	Name	Mac	PC
¸	cedilla	252	184
¹	superscript 1	49	185
º	ordinal, masculine	188	186
»	right chevrons	200	187
¼	one-fourth		188
½	one-half	189	189
¾	three-fourths	190	190
¿	inverted question	192	191
À	cap A, grave	203	192
Á	cap A, acute	231	193
Â	cap A, circumflex	229	194
Ã	cap A, tilde	204	195
Ä	cap A, umlaut	128	196
Å	cap A, angstrom	129	197
Æ	cap AE, diphthong	174	198
Ç	cap C, cedilla	130	199
È	cap E, grave	233	200
É	cap E, acute	131	201
Ê	cap E, circumflex	230	202
Ë	cap E, umlaut	232	203
Ì	cap I, grave	237	204
Í	cap I, acute	234	205
Î	cap I, circumflex	235	206
Ï	cap I, umlaut	236	207
Ð	cap eth		208
Ñ	cap N, tilde	132	209
Ò	cap O, grave	241	210
Ó	cap O, acute	238	211
Ô	cap O, circumflex	239	212
Õ	cap O, tilde	205	213
Ö	cap O, umlaut	133	214
×	multiply	120	215
Ø	cap O, slash	175	216
Ù	cap U, grave	244	217
Ú	cap U, acute	242	218
Û	cap U, circumflex	243	219
Ü	cap U, umlaut	134	220
Ý	cap Y, acute	89	221
Þ	cap thorn		222
ß	sharp s	167	223
à	l/c a, grave	136	224
á	l/c a, acute	135	225
â	l/c a, circumflex	137	226
ã	l/c a, tilde	139	227
ä	l/c a, umlaut	138	228
å	l/c a, angstrom	140	229
æ	l/c ae, diphthong	190	230
ç	l/c c, cedilla	141	231
è	l/c e, grave	143	232
é	l/c e, acute	142	233
ê	l/c e, circumflex	144	234
ë	l/c e, umlaut	145	235
ì	l/c i, grave	147	236
í	l/c i, acute	146	237
î	l/c i, circumflex	148	238
ï	l/c i, umlaut	149	239
ð	l/c eth		240
ñ	l/c n, tilde	150	241
ò	l/c o, grave	152	242
ó	l/c o, acute	151	243

<b>Character</b>	<b>Name</b>	<b>Mac</b>	<b>PC</b>
ô	l/c o, circumflex	153	244
õ	l/c o, tilde	155	245
ö	l/c o, umlaut	154	246
÷	divide	214	247
ø	l/c o, slash	191	248
ù	l/c u, grave	157	249
ú	l/c u, acute	156	250
û	l/c u, circumflex	158	251
ü	l/c u, umlaut	159	252
ý	l/c y, acute	121	253
þ	l/c thorn		254
ÿ	l/c y, umlaut	216	255