# 15 Editing – navigation

(From Word 2007 onwards, the functionality of *InstantFind* is probably redundant because of the more advanced find facilities now available. However, the *FindFwd*, *FindBack*, *FindFwdCase* and *FindBackCase* may still be very useful.)

When working with text, you want to be able to move around it, looking at various bits, checking them and changing them, so in this section there are tools to allow you to jump instantly to another heading of the same type, to another occurrence of the selected text, to another comment – plus a whole load of other ways of jumping around the text. I find that using these speeds me up considerably.

(And the section above gave you tools to allow you to jump around among the highlighting.)

## Navigation pane customization

I was asked if it was possible, in a macro, to set the width of the navigation pane. It is, but you can customize it in various ways – see below.

**Sub NavPaneWidth()**

## Open navigation pane your size and position

Word's navigation pane can be placed in different positions and at different sizes. This macro switches the pane on and off in the position/size you want.

To help you decide on a suitable size, you can change `doSetUp = False` into `doSetUp = True`, then open the navigation pane at a size you like, then run the macro and it will tell you the current height and width of the pane. You can then put whatever numbers you want in the macro, and change back to `doSetUp = False`.

When putting the pane in different places (here offered are left, right and floating – there are others), you can only specify height of the floating version of the pane.

**Sub NavPaneCustomize()**

## Temporary bookmarks

Before you start navigating around, it's helpful to leave a temporary bookmark behind, so that you can quickly get back to where you started. So there's a macro to place a temporary bookmark at the cursor, and then one that jumps you back to your temporary bookmark.

There's also one to clear the temporary bookmark if you're worried about sending off a file that has a stray bookmark in it.

But there is also a macro that uses the temporary bookmark to make it easier to select a long section of text: Use *BookmarkTempAdd* to place a marker at one end of the section, then go up or down through the text, by whatever means, to find the other end. Place the cursor at the end and run *BookmarkToCursorSelect* to select from marker to cursor.

Note that *InstantFindUp*, *InstantFindDown* and *InstantFindDownWild* also lay down a temporary marker before dashing off to find what you've told them to find, but it's a separate marker. So if you use *BookmarkTempFind* once, it will find the usual temporary marker, but then if you use it a second time, without moving the cursor, it realises it's already at the first marker and will jump you to the *InstantFind* marker instead – so you don't need to learn a new keystroke.

**`Sub BookmarkTempAdd()`**

**`Sub BookmarkTempFind()`**

**`Sub BookmarkTempClear()`**

**`Sub BookmarkToCursorSelect()`**

## Selective F&R at speed

(Find functions: *https://youtu.be/guPVq57Vgm4*)

**Concept**: (1) ***FRedit*** gives you global F&R, as determined by your list of F&Rs. (2) ***MultiSwitch*** gives you local (topical) one-off F&R, again according to your list of F&R items. (3) ***FRselective*** gives you selective F&R through (an area of) your document according to your list of F&R items.

(N.B. *FRselective* has functions *other than setting up selective F&R*, but it takes a while to explain this main functionality, so please don't stop reading when you feel, "Ah, yes, I see how it works!" – there's more to discover!)

So you keep a file of F&Rs that you might want to do selectively (you could use your zzSwitchList – mine is open all the time). Click in the F&R item, run *FRselective*, click in the document you're working on and use any of the following macros (choose your own keystrokes, but mine are shown):

*FRskip* – jumps to the next occurrence of your Find (Alt-→)
(the equivalent of 'Find' in the Find and Replace window)

*FRgo* – makes the Replace and jumps to the next Find (Ctrl-Alt-Shift-→)
(the equivalent of 'Replace' in the Find and Replace window)

*FRstay* – makes the Replace, but *doesn't* jump to the next Find (Ctrl-Alt-Shift-←)
(no equivalent in the Find and Replace window)

Using *FRstay* means that you can check that the Replace is in fact doing what you intended; then you can *FRskip* to the next Find.

My way of thinking of the keystrokes is that using Ctrl-Alt-Shift says "Make a change" (and either stay or go), whereas the 'lighter' Alt-→ is just 'move on'.

An example might help to explain how *FRselective* works. Say your job requires no serial comma, but the author has used a few. What a serial comma actually IS depends on the context, hence the use of *selective* F&R.

(N.B. Understanding the wildcard F&R is not the point here, just the set-up and execution of it.)

Here's the F&R (coloured only for clarity):

```
(, [a-zA-Z]@), and>|\1
```

FRedit users will recognise the format:

```
<Find item>|<Replace item>
```

So here's the sequence of actions:

Use *FRselective* to 'pick up' the F&R

Click in the target document
Click *FRskip* to find the first occurrence
If the first Find need changing, run *FRstay* (if you want to check it) or *FRgo*
If you've done *FRstay*, then you can use *FRskip* to move to the next,
etc, etc.

## Other functionality

Because this is a Find and **Replace** macro, you can also use it just for **finding** things, and then use *FRskip* to look through them (if you want to skip backwards, you can use: *FindBack*).

For example:

```
acronym
[A-Z]{2,}
e.g. BBC

BC/AD/CE/BCE in caps
[ABC][DCE]{1,2}>
e.g BC/AD/CE/BCE in caps

bc/ad/ce/bce in small caps
[abc][dce]{1,2}>
e.g. bc/ad/ce/bce in small caps

[0-9]{1,2}^32[A-S][a-z]@^32[0-9]{2,4}
dates, e.g. 22 Sept 1948, 7 Nov 88

[0-9]{1,2}.[0-9]{1,2}.[0-9]{2,4}
dates, e.g. 22.09.48 or 22.09.1948

[A-Z]{1,2}[0-9]{1,2} [0-9][A-Z]{2}
UK postcodes, e.g. NR8 6TR
```

## Use with the Macro Menu

If you open the Word file version of the Macro Menu, click at the top of the file and type something like:

list acronym

and run *FRselective*, it will search the file for:

list<some text>acronym

or

acronym<some text>list

so it will find:

*AcronymAlyse* – Lists all acronyms, with frequency

If you Ctrl-Home back to the top of the Macro Menu file, the cursor will now be at the **start** of the line, 'list acronym', so if you run *FRselective* again, it knows you that want the opposite search:

acronym<some text>list

and, it will find:

*AcronymAlyse* – Lists all acronyms, with frequency

### Finding word pairs

In fact, *FRselective* can search for any phrase starting and ending with specific words, so if you were to click in 'search' in this sentence and then shift-click in 'ending' *FRselective* would look for 'search… *some text or other* …ending'.

If you now click in a blank line (or to the left of the first word in a paragraph), *FRselective* will look for 'ending… some text or other …search'. Clicking again in a blank line *FRselective* switches the order back again.

Difficult to describe in words on paper. Please watch the video! 😊

**Sub FRselective()**
**Sub FRskip()**
**Sub FRgo()**
**Sub FRstay()**

## Instant find

"Super-Searching 1" (9:52) https://youtu.be/EJSB13x8QMU
(Word's own search facilities, as a background to...)

*N.B. WildcardLoader has been superseded by* ***FRselective (see above)***

"Super-Searching 2" (9:16) https://youtu.be/gTX6Z3uWp8k
*WildcardLoader* simple use

"Super-Searching 3" (6:19) https://youtu.be/o50lMd7LUtA
*WildcardLoader* advanced use

"Super-Searching 4" (7:49) https://youtu.be/m4gVuqrl83w
More about *WildcardLoader*

As an editor, you want to be able to move around the text looking for things, quickly and easily. For example, you're reading through the text, and you read:

'**Step 1**: Put the cat out'

and you think to yourself, 'Hang on, did the lists earlier in the text have a colon after the number or not – and were they separated with a space or a tab?' So, you want to be able to quickly look back – but how far back was it, and will you be able to find it if you rely on scrolling back and scanning by eye?

Using the macros below, there's a quick and easy way: (*You can use whatever keystrokes you like, but for the sake of the explanation, let's assume you are using my keystroke suggestions.*)

Click in the word 'Step' and press Ctrl-Shift-Alt-Up to run *InstantFindUp*. This loads 'Step' into the F&R, and jumps up to the *previous* occurrence. (If you had done Ctrl-Shift-Alt-Down, it would have run *InstantFindDown* and jumped on to the *next* occurrence.)

(By default, the macro assumes that you're looking for the single word at the cursor, so if you want, say, to look for 'Step 1', then just select it before running the macro.)

Now that the F&R is loaded, Alt-Left (*FindBack*) and Alt-Right (*FindFwd*) will jump you to the previous and the next occurrence of 'Step', so you can go up and down each 'Step' to your heart's content.

But then you see that it's finding 'step' as well as 'Step'. Bother!

No worries! We can soon ‘control’ that. Pressing Ctrl-Alt-Left (and -Right), will jump you through *case sensitively*, so that it will take you through the occurrences of ‘Step’, but ignore ‘step’. (Word’s own Ctrl-PageUp and Ctrl-Page-Down also jump you from one occurrence to the next.)

**New feature**: If the word you’re looking for is, say, ‘jump’ and you don’t want it to find ‘jumped’, ‘jumps’ or ‘jumper’, then just add a wildcard ‘>’, and select ‘jump>’ and then when you run *InstantFindDown*(*Up*), and it will see the angle bracket and switch to wildcard mode and only find ‘jump’ as a single word. OK, I admit that it’ll still find ‘longjump’, but you could also add a ‘<’, so search for: ‘<jump>’. (But you could alternatively use the macro *FindDownWild*, below.)

(There’s more to come, but just try these and get used to them before memorising more keystrokes.)

I also find it useful to instantly find the selected text by *first* jumping to the top of the document and *then* searching downwards for it – that’s *InstantFindTop*.

Someone asked to do the opposite: first jump to the bottom of the document and then search upwards for it – that’s *InstantFindBottom*.

**New feature**: When you go off searching for things, you can sometimes lose track of where you were in the document when you decided to search for something. I’ve therefore made it so that you can use my bookmarking macros. So when you run *InstantFindDown(Up)* it will leave behind a bookmark, so you can then get back to this starting point by using *BookmarkTempFind*.

However, this annoys me if I search for something in one of the files I regularly access but may not actually have edited, such as this file you’re reading now(!). If the macro adds a bookmark, then when I try to close the file, it will ask if I want to save it. So I have to think, “Have I actually made any changes to this file or was I just searching for something?” So, even if, at the top of the macro, you’ve set `addBookmark = True`, it will NOT add a bookmark if the current file is one of these named files:

```
butNotTheseFiles = "zzSwitchList,ComputerTools4Eds,TheMacros,5_Library"
```

You can obviously add to, or subtract from, this list the filenames of the files you search regularly.

**Sub InstantFindDown()**

**Sub InstantFindUp()**

**Sub InstantFindTop()**

**Sub InstantFindBottom()**

**Sub FindFwd()**

**Sub FindBack()**

*FindFwd* will find the next occurrence of the search text in the main text or in the footnotes or in the endnotes or in the comments. However, one editor wanted *FindFwd* to search to the end of the text and then, if nothing was found, jump into the notes (if there are any, of either type) and from there into the comments.

I started using it, to replace *FindFwd*, but it didn’t suit the way I work – I like to hear the ‘bong’ to tell me that it has reached the end of the text and can find anything, rather then jumping unannounced into the notes. So I’m sticking to *FindFwd* – you have the option.

**Sub FindFwdAll()**

And I’ve realised that, for *DocAlyse*, it’s useful to be able to ‘find down’, but with wildcards switched on. This is so that you can select, for example, ‘[a-zA-Z]@, [a-zA-Z]@, and’ in the *DocAlyse* summary to search through the text for serial commas. So I’ve added another one:

Sub InstantFindDownWild()

Finally, being a totally nutty speed-freak, I decided it would be helpful to be able to jump up and down, from word to (the same) word, but without losing the previous item that I was trying to find. So, say I was looking for the word *FRedit*, I'd use *InstantFindDown*, (or *Up*) and then *FindFwd* and *FindBack*; but then *FRedit* is mentioned a lot in this book (currently over 300 times!). So then if I decided it was near the earlier reference to 'speed-freak', I could select that term and use *InstantJumpUp* to move to it, without losing 'FRedit' from the Find function.

Sub InstantJumpUp()

Sub InstantJumpDown()

## Instant find – case sensitive

The next pair of macros (referred to above) improve on Word's jumping to the next and previous find. By holding the Ctrl key down – i.e. Ctrl-Alt-Right and Ctrl-Alt-Left (assuming you use the same shortcuts as I do) – it only finds matches that are exactly the same case as the Find text.

Sub FindFwdCase()

Sub FindBackCase()

## Instant replace

Working along the same lines as the macros above, once you've got used to skipping around the text with Find, why not speed up the replacing too?

There are two macros designed for this. The first (which I put on Ctrl-Alt-Shift-Right, in line with Alt-Right being a forward Find) simulates the Replace action of the normal F&R dialogue box (which you still have to set up in F&R dialogue box as normal†). So you click Alt-Right to take you through until you find an occurrence that *does* need replacing, and then click Ctrl-Alt-Shift-Right to change it, and jump to the next occurrence.

But what's the point?! With the normal F&R dialogue box, pressing the F key finds the next, and pressing the R key replaces that occurrence and finds the next, so what do you gain?

First of all you gain from the fact that you are 'in the text', i.e. while you are doing this selective F&R, you can easily stop along the way and change something else that catches your eye without going in and out of the F&R dialogue box. But the second reason makes learning a new keystroke much more worthwhile.

The second of the two macros below does the replace, but it does *not* move you on to the next occurrence. This means that you can stick around and see exactly what effect the Replace has had – i.e. you can double-check that you have set up the F&R properly – especially useful if the F&R is a wildcard one. After all, the next occurrence is only an Alt-Right click away.

(†N.B. To set up the Find and the Replace, you could use *PrepareToReplaceDown*. That's fine, but if you do it 'manually', then after you've typed in the Find text and the Replace text, you must then click 'Find Next', otherwise the F&R box 'forgets' the text you've just typed in. Grr!)

Sub FindReplaceGo()

Sub FindReplaceStay()

## Instant find clipboard

And another macro that's useful is where you've copied (<Ctrl-C>) a word or phrase in *one document* and want to find it in *another*. This macro simply looks down through the current document and jumps straight to the first occurrence it can find of whatever is stored in the clipboard.

**`Sub FindClip()`**

If I'm constantly flipping back and forth, say to find references in a list in a separate file, I find it useful also to have a version that jumps to the top of the file first, and *then* looks for the contents of the clipboard – i.e. it's the same macro as the one above, but with the addition of the first line:

**`Sub FindClipTop()`**

## Clear odd find conditions

If you're using a range of different (perhaps clever) find formats, you sometimes find that a simple find, or FindFwd/Back won't find things that you *know* are there! This may be because some parameter has been set in the Find function. This macro therefore 'cleans down' all the 'funny' Find parameters.

**`Sub FindResetParameters()`**

## Instant find particular formats

(N.B. This is very similar in functionality to *FindStyle* (below), which I created after these macros, but forgetting that they were available! And, I prefer the way that *FindStyle* functions.)

If you've discovered the huge speed-up value of the *InstantFind* macros, then you might also like *InstantFindFormat*.

The idea started when I noticed in a book that I was reading that the note numbers (which are not electronically linked notes, but simple subscripted numbers) weren't sequential, so I wanted to be able to jump from one superscripted number to the next (and back again).

What *InstantFindFormat* does is (a) if some text is selected that has an attribute (so far just super/subscript, italic or bold), it will look for that exact text, but *only* when it occurs with that attribute. So, if you select a superscript '4', it will only find any other superscript '4' in the document. However, if nothing is selected, but the character to the right of the cursor is, say, a superscript '4', then it will jump to the next (or previous) character – any character – that is superscripted. So that will jump me back and forth between all the superscripted numbers.

Or you might use it to jump between all the different items of bold text, or italic text, or perhaps all the subscripts (say if you want to check whether the subscripts are italic or not).

As with the normal *InstantFind*, you use this macro to set up the F&R (either up or down), and then you can use the normal *FindFwd* and *FindBack* to jump up and down through the various finds.

**`Sub InstantFindFormatDown()`**

**`Sub InstantFindFormatUp()`**

## Find in another file

*(Video: youtu.be/BueNLL8uyKE and youtu.be/PB0hXA_1tRo and https://youtu.be/EaFJuKTbhGw and **the most recent**: https://youtu.be/sNUwH1ECFjU)*

If you are working on two similar files, it can be very useful to be able to move from somewhere in the first file over to the equivalent place in the second file. For me, the main application is where someone has added some (hopefully tracked!) changes to one version of a file, and I want to update my master version of the file.

So if you select a bit of the text and run the macro, it will then look through the other file (in fact, it looks at all the other files currently open in Word) until it finds the exact same piece of text. And there you are, at the same place in the other file – almost instantaneously.

However, if that bit of text has been changed, even slightly, then obviously it won't find it, so the macro just beeps at you and returns you to the first file, so that you can select a more suitable bit of text.

But you don't actually have to select some text; you can also just put the cursor, say, in the middle of the line *above* the one you're interested in comparing (because the target line has probably been changed in some way). Then, because no text is selected, the macro knows to use a different search technique: It selects the whole of the line and searches for that text in the other open file(s). This time, if it can't find it, it progressively shaves bits off the ends of the search text until it eventually finds *something* that matches (or maybe not). (See note † below.)

However, it occurred to me that if, say, you've got a list of names or a references list, and all you want to do is search the other file for an individual word or name, it might be quicker to be able to put the cursor in that word, ***and not select it***, and the macro would know to search just for that word, and not the whole line, as mentioned above. So if you want: "no text selected" to mean "just look for a word" then, at the start of the macro, set:

```
selectCurrentWord = True
```

but if you want: "no text selected" to mean "look for the whole line", then, at the start of the macro, set:

```
selectCurrentWord = False
```

Another selection feature now available is if you set, at the start of the macro:

```
wholeWordsSelect = True
```

then if you select some text, it will round off the beginning and end of the selection to the nearest whole word.

**Application idea**: Suppose you're working on, say, four different files that you want to compare – say A, B, C and D – then *FindSamePlace*, will search in the order A, B, C, D. However, if you use *FindSamePlaceBack*, it will search in the reverse order: D, C, B, A. So, if you are in file **C** then *FindSamePlace* will search in the order D –> A –> B, whereas *FindSamePlaceBack* will search in the order B –> A –> D.

(N.B. The ***order*** that these macros use when searching a set of more than two files is ***alphabetic on filename***.)

†In case it helps your use of the macro, I'll explain how it decides which bits to chop off. If the cursor is nearer to one end of the line than the other, it takes 10 characters (the number set by `myStep`, at the beginning of the macro) off the long end and tries again. When that end is shorter than the other, it takes some off the other end, so if it still can't find this shorter text, it gets nearer and nearer to the words around the place where you put the cursor, eventually giving up when there are less than `minLength` characters.

(*Recent upgrade feature*: If you have a file open that has been created by MultiFileText or MultiFileWord then if you click in a line and run this macro, it loads up the relevant original file and then finds the same line, so that you can look at the context.)

(*Another recent upgrade feature*: If you have to work with lots of files open at once, but you don't want the macro to go all the way round through all the files, there are two 'exceptions' methods, to make these macros *ignore* one or more of the files.

```
notThisFile = "zzSwitchList"
notThisFileEnd = "XX"
```

You can probably guess the idea for the first one. I *always* have my switch list file open, so I want this macro to ignore it. But if you made that line `notThisFile = "zz"`, then it would skip past *any* file with 'zz' in its name.

The second one works by typing 'XX' right at the end of the files you *don't* want it to search.)

(Yet *another recent upgrade feature*: If you have to work with lots of files open at once, but you ONLY want the macro search in certain files that have a common element in their filenames, say '_PB', then here's the line to change:

```
onlyLookInTheseFiles = ""
```

So change it to:

```
onlyLookInTheseFiles = "_PB"
```

and it will only searching in files such as 'Chap_01_PB', 'Chap_02_PB', 'Chap_03_PB', etc.

But don't for get to change it back to:

```
onlyLookInTheseFiles = ""
```

when you've finished, otherwise it won't find anything, if you don't have any '_PB' files!)

**Sub FindSamePlace()**

**Sub FindSamePlaceBack()**

## Find any of a set of searches

***N.B. This macro is based on (yet another!) idea by Howard Silcock – Thanks, Howard!***

*(This is a 2014 macro and is probably now redundant in light of the macros above.)*

This macro searches for <this text> OR <that phrase> OR <the other bit of text> – whichever comes next, starting from the current cursor position.

You have two macros, the first of which is used to set up the search, so you would run it and enter:

```
this text,that phrase,the other bit of text
```

or if you happen to want to search for a comma, you can enter:

```
this text|that, phrase|the other bit of text
```

The setup macro stores these search criteria in a variable in the file. Then each time you run the main searching macro, it reads this variable and executes the search, so you can use it over and over again without retyping.

For example, maybe you would want to look through some text to find when and where various people have been mentioned, so you might enter: `Jones,Green,Smythe`

The macro then looks for each of these three names as they occur.

Unfortunately, it will also find 'winter**green**', '**green**wood' etc. No problem, just use: `Jones,$Green,Smythe`. In other words, the $ says that this word should be searched case-sensitively.

But then it would still find '**Green**grass', so you decide you want to use a wildcard search. For that, you would use, say, `Jones,~<Green>,Smythe`. In other words, the tilde "~" (familiar to *FRedit* users) indicates that the following search pattern is to be wildcarded.

Or you might use: `Jones|~<Green>|~Sm[iythe]{3,4}` which will find Smith, Smyth, Smythe (and even Smtyh!), but note that, I've used '|' and not comma as a separator because the wildcard search is `Sm[iythe]{3,4}`, which itself contains a comma!

I'm sure you'll think of ways you can use this tool, but I can see one immediate use of the case sensitive function – searching for a commonly occurring word where it's the ***exceptions*** that you want to find. So, for example, you could ignore 'the' by searching for: `$The,$THE` or maybe even use wildcard for it: `~The>,~THE>` (remember that wildcard searches are always case sensitive).

N.B. Because each macro calls the other macro, it is important not to change the names of these macros. Well, if you do, you'll also have to edit the place in the other macro so that it calls the macro by your new name. For example, instead of `Call FindThisOrThat` you'd change it to `Call MyNewName`.

**`Sub FindThisOrThat()`**

**`Sub FindThisOrThatSetUp()`**

## Copy text into the F&R box

Here's the scenario: You're working on a document and find, say, 'Bloggs & Sons Ltd', and you decide to use F&R to change it to something slightly different, say 'Bloggs & Sons plc'.

To do this manually, you have to select the text, open the F&R window, paste the text into both Find and Replace, and make the necessary changes to the Replace version. However, with this macro, you simply select the text, press Ctrl-Alt-H (or whatever key you want to assign to it) and up pops the F&R window with the text already in both Find and Replace.

**`Sub PrepareToReplaceDown()`**

I also have a version that switches the track changes off as I start it up:

**`Sub PrepareToReplaceDownTCoff()`**

As it stands, this macro trims the final space off the selection. This is useful if you select a single word by double-clicking it, because you may not want the trailing space. However, if you prefer it to transfer *exactly* what you select into Find and Replace, then at the beginning of the macro, just use myText$ = Selection instead of myText$ = Trim(Selection).

You might want to allocate that macro to, say, Ctrl-Alt-H (cf. Ctrl-H for normal F&R) and then have another version (*PrepareToReplaceFromTop*) set to Ctrl-Alt-shift-H which picks up the selected text but, instead of starting the search from just after the selected text, it starts from the top.

**`Sub PrepareToReplaceFromTop()`**

Then if, like me, you want to leave a place-marker behind (I use '[[['), the next macro leaves a marker and then goes to the top and does the F&R from there. Then you can jump back to your marker using the *FindMyMarker* macro.

**`Sub PrepareToReplaceWithMarker()`**

## Find within deleted text

Someone said they wanted to "search through the track changes". What they meant was that they wanted to search through for text that they had deleted (but track changed).

This macro searches through deleted text within track changes, starting from the current cursor position.

It's just a proof of concept really, i.e. it's only a case-insensitive search at the moment, but I can refine it you're interested.

**Sub FindInDeletedText()**

## Multi-purpose find

*(Video: youtu.be/Jm3xUnpYcSo)*

The idea of this macro is that one keystroke, Alt-F (or whatever you choose), allows you to find a range of different things. It does this in a number of ways, but many of them are done by setting up a wildcard find. Once that's set up, you can use

– Type in a word or phrase and it uses the normal F&R to find it
– p17 searches for **p**age 17.
– f17 (or n17) searches for **f**oot**n**ote 17.
– e17 searches for **e**ndnote 17.
– c17 searches for **c**omment 17.

If, when you press Alt-F, some text is selected it will display that text in the input box and, if you just press Enter, it will go and find it, but you can, of course, edit the text before pressing Enter, and also:

– a searches for **a**cronyms: BBC, TTFN etc (see also 'x').
– B searches for **B**C/AD/CE/BCE.
– b searches for BC/AD/CE/BCE in small caps.
– d searches for **d**ates of the form 13 Jan 2030.
– D searches for **D**ates of the form 13.6.09 or 1.6.2010
– e gets the **e**xisting text from Word's Find box, so that you can change it in some way.
– E see under 'z' for zipcode.
– f searches by the **f**ont attribute(s) that have been applied to the text, e.g. superscript, bold, font size.
– h tries to work out which is the next **h**eading, i.e., if it's *not* Normal style, it searches by style; otherwise, it tries to go by whether it's italic and/or bold and the font's size, name and colour, but if that's no good, it looks for a code symbol such as '<A>'.
– i searches for people with initials and surname, *including* full points (periods), e.g. J.L.B. Matekoni.
– I searches for people with initials and surname, *without* full points (periods), e.g. JLB Matekoni.
– m searches for e**m**ail addresses.
– n searches for any **n**umber
– p searches for UK **p**ostcodes.
– s searches for **s**ection numbers, i.e. lines beginning with 2.3 or 14.9.1.1.
– u searches for **u**nits: mA, kg, cm, MN etc.
– w searches for **w**eb addresses.
– x e**x**pands an acronym, i.e. searches for words that might be the expansion of the acronym.
– X e**x**pands an acronym, but looks for uppercase ***or*** lowercase. However, for acronyms longer than three characters it only searches the first three, otherwise Word complains that the wildcard F&R is too complicated!
– y searches for **y**ears, e.g. 1066, 1948 or 2030.

– z searches for US **z**ip codes (five-digit numbers).
– Z searches for Canadian **z**ip (postal) codes (e.g. V9M 3T7).
– E searches for **E**uropean postal codes of the form N-1234, or E 34568.

– numbers 1 to 5 find headings (paragraphs) that start with the same 1 to 5 characters, e.g.:
(a) If the line is "<A>Figure 2.3 Blah blah" then using 2 will find all <A> headings, while 1 finds *any* <A, <B, <Fig etc.
(b) If the line is "Figure 2.3 Blah blah" then using, say, 4 will find all headings (paragraphs) starting "Figu...", but will *ignore* cases where, say, "However, in Figure 3.2 we can see..."

– ( searches for **(**3.16), (12.2), e.g. equation numbers – and even '(2.4.1.7)'

– ‘-’ (a hyphen) searches for a word pair with or without spaces and/or hyphen/en dash. If you select ‘borehole’ or ‘bore hole’ or ‘bore-hole’ or ‘bore–hole’, it will find any/all of those combinations throughout the text.
However, as with *HyphenSpaceWordCount*, you don’t actually need to select any text; just put the cursor in the first word, and it will beep at you and ask you to confirm, “First word? bore” – just press Enter.
But then if you click in ‘borehole’, it will say, “First word? borehole”, so you just delete ‘hole’ (in the input box) and press Enter.

– Custom searches – there’s a section in the macro where I set up two-letter codes for words I often need to search for so, for example, if I type ‘se’ and press Return, it searches for ‘| Spelling Errors’ (which you’ll recognise, if you use SpellingErrorLister and SpellingErrorHighlighter). Look in the middle of the macro, and you’ll see:

```
If Len(myText) = 2 Then
  myCase = True
  Select Case myText
    Case "re": myText = "References^p": myCase = False
    Case "fi": myText = "Figure"
    Case "se": myText = "| Spelling Errors"
    Case "mm": myText = "Macro Menu"
    Case "ap": myText = "Appendix"
  End Select
End If
```

I’m sure you can add and subtract your own special searches. Note that I search for ‘references’ in any case, but the rest are searched case sensitively.

You can also use Word’s own special find features:
– A searches for **A**ll word forms, e.g. if it’s got ‘go’ it will find ‘going’, ‘gone’, ‘went’.
– W searches with the ‘**W**hole words’ feature enabled, so ‘the’ does *not* find ‘other’ or ‘thesis’.
– S searches with the ‘**S**ounds like’ feature enabled.

Note: When you are searching for some text, the macro searches downwards as you would expect. However, if it cannot find the searched-for text before the end of the file, it does ***not*** then jump to the top of the text and start searching from there. Instead, in beeps at you and then searches upwards from the current cursor position. I thought this was more useful, but do ask me for an up-and-over-the-top version if you prefer. In any case, if you hear a beep, but the cursor stays where it is then that text ain’t nowhere in the file. What’s more, before doing the search, the macro switches off all the special features: wildcards, match case, sounds like and whole words, so if it’s there, it’ll find it.

The macro also loads the text into the Find box of Word’s F&R, so if you then want to jump around the file looking at the various occurrences then, you can use the *InstantFind* macros above, as usual.

## *Where was I?*

If you are doing any of searches above and you add a ‘[’ at the end of the line then, before the macro goes off to wherever you want it to go, it leaves behind a ‘[[[’ as a marker so that you can then use the ‘[’ option to quickly and easily get back to where you started. (Or, if you find it easier, you can use *FindMyMarker* macro – a single keystroke to refind the ‘[[[’.)

## *Make up your own F&R*

One of the powers of this macro is that, for a specific job, you can add your own specific wildcard F&R. For example, someone wanted “a macro to change the note numbers in a list of endnotes from a full point (period) following by a space into no full point (period) and a tab”. So, if you use *SmartFinder* and press ‘.’, the Find and Replace boxes will be set up accordingly, and then you can use *FindReplaceGo* and/or *FindReplaceStay* to go through the notes one by one (or on autorepeat!) changing them all. (Or just use Ctrl-H once the wildcard F&R is set up.)

I think the format is reasonably straightforward:

```
  Case ".": myFind = "^13([0-9]{1,2}). ": myReplace = "^p\1^t"
     ' Full point (period) off note number
```

### Clearing up

After you have used some of the features such as the All word forms feature, other F&Rs may get a bit confused. So if you use the Customize Keyboard window to assign, say on Ctrl-Alt-F to the *FindOptionsDialog* command you can clear down all these special features to give yourself a ‘clean’ F&R.

**Sub SmartFinder()**

## Versatile searching via wildcards

“Super-Searching 1” (9:52) https://youtu.be/EJSB13x8QMU
(Words own search facilities, as a background to...)

“Super-Searching 2” (9:16) https://youtu.be/gTX6Z3uWp8k
*WildcardLoader* simple use

“Super-Searching 3” (6:19) https://youtu.be/o50lMd7LUtA
*WildcardLoader* advanced use

“Super-Searching 4” (7:49) https://youtu.be/m4gVuqrl83w
More about *WildcardLoader*

Please don’t worry! You don’t need to learn wildcarding yourself – just use wildcard searches that other people have worked out. You can get some of them from the **Resources** tab on wordmacrotools.com.

*WildcardLoader* is a versatile macro with a number of features, but even if you use the first and most obvious facility, it will enable to you do more versatile and effective searching.

If you have a file with some wildcard searches in it, simply put the cursor in one of them, and *WildcardLoader* will load the search criterion into Word’s Find box and search for the first occurrence in that file. But then click in your target file and use *FindFwd* or *FindBack* to look through all the occurrences that the wildcard finds for you.

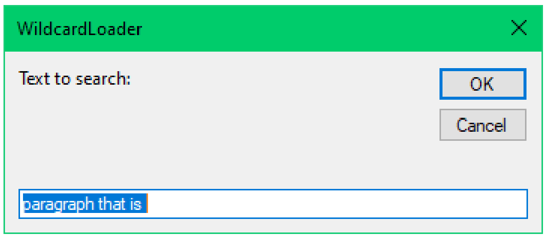
**Find two words in proximity** (i.e. in the same paragraph): If you have the two words you want to find, e.g. ‘According to Smith et al (2016)...’ then you simply click in ‘Smith’ and then Shift-click in ‘2016’, run the macro and it will take you straight to the next ‘Smith et al (2016)’ citation, or to the reference: Smith, P.Q., Jones T.A. & Benson, Z. (2016) How to blah, blah, etc.’ (Note, to *set up* the search, the words don’t have to be in the same paragraph – just click and shift-click any two words anywhere.)

The WC it generates (in case you are interested) is: [sS]mith[!^13]@2016. Because wildcard finds are case *sensitive*, it is offering to find the text with or without an initial capital. The [!^13]@] in the middle just means, ‘any text that is *not* a new paragraph, i.e. find the two words *only* if they occur in the same paragraph.

In the demo video, I search within the ‘Alice in Wonderland’ book for Alice and rabbit: [aA]lice[!^13]@[rR]abbit, so whether it’s ‘the rabbit’ or ‘the White Rabbit’, it will still find it.

However, note that it only finds ‘[aA]lice’ *followed by* ‘[rR]abbit’ in a given paragraph. But then you can tell the macro to switch it round for you: double-click any short word (or just select a couple of characters), run the macro, and it will know that you want it to do the switch to: [rR]abbit[!^13]@[aA]lice.

If you want to see what the current WC search is, just click in any blank paragraph, run the macro, and it will type out the search for you, like this:
[A-Z]{2,}
The final feature is, if you click in the middle of a paragraph that is not a WC search, it will respond by opening up an input box into which you can type or paste some text to be searched (it also picks up the words near your cursor, in case you want to use them).



The list of WC finds on the website starts as follow, but this macro, like FRedit and MultiSwitch, is a ‘content-free’ macro, i.e. you can give it any text you want.

```
acronym
[A-Z]{2,}
e.g. BBC

BC/AD/CE/BCE in caps
[ABC][DCE]{1,2}>
e.g BC/AD/CE/BCE in caps

BC/AD/CE/BCE in small caps
[abc][dce]{1,2}>
e.g. BC/AD/CE/BCE in small caps

[0-9]{1,2}^32[A-S][a-z]@^32[0-9]{2,4}
dates, e.g. 22 Sept 1948, 7 Nov 88

[0-9]{1,2}.[0-9]{1,2}.[0-9]{2,4}
dates, e.g. 22.09.48 or 22.09.1948
```

The fonts, colours and highlights are totally irrelevant! They are only there to help me quickly locate the particular WC I want – and, of course, at any stage, I can move the WCs around in the file.

What I have done, however, is to put an *example* of the target text on the line immediately *following* the WC. So when I run the macro, it will select the first example it can find, i.e. on the next line. This encourages me that it is actually working.

Enjoy!

And if you want to save a wildcard find that you’ve just tried out, you can type it out with *WildcardTyp*e, ready to put it in with your list of other wildcard finds. Place the cursor on a blank line and run the macro.

Another idea is that, for a given file you’re working on, say a whole book, you want to save some wildcard finds just for that book. So if you use *WildcardLoader* to, say, do a search for ‘Brown’ and ‘2016’, and you think you might want it again, or perhaps for a different year, run *WildcardSave*, and it will go to the top of the document, place a copy of the find there, show it to you and then, when you click ‘OK’, it will take you back to where you were in the text.

**Sub WildcardLoader()**

**Sub WildcardType()**

**Sub WildcardSave()**

## Versatile find tool – ListOfFound

This is a simple but really versatile finding device – a solution looking for a problem.

The one-line descriptor in the Macro Menu says, ‘Creates a list of paragraphs containing specific word(s)’ – but that hides a wealth of possibilities.

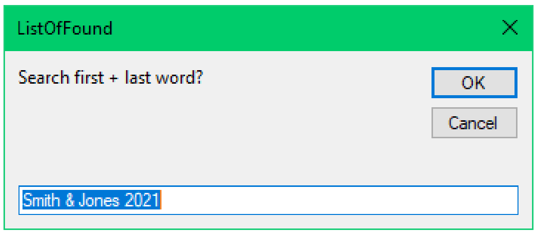
Basically, the macro searches through the current document and makes a list – in a new document – of what it finds. It was created originally for the ever-present problem (for me, anyway) of searching through a list of 1200+ items in the Macro Menu to find out: ‘Is there a macro for...?’

Suppose you say you want a macro to ‘add comments’. The trouble is that the Macro Menu has 131 macros mentioning the word ‘comment’ and 195 with ‘add’. However, if you run ListOfFound with ‘add comment’, it will list just the 16 relevant macros.

Having created the macro for that purpose, I could see many wider applications, and new features began to sprout.

Suppose you have various ‘Smith’-related citations and/or references and there’s a query over ‘Smith & Jones 2021’ – but should that be 2022? Or maybe ‘Smith **et al.** 2021’?

So, just click in ‘Smith’, then shift-click in ‘2021’, run the macro, and up comes a box:


ListOfFound

Search first + last word?

Smith & Jones 2021

Now, click the right arrow (to unselect the text in the input box), delete the ‘1’, and click ‘OK’ or press Enter. In seconds you’ll have a list of all the references and citations that contain ‘Smith’ and also ‘202’. Helpfully, it could also have the paragraph with, for example, ‘In 2024, Smith and co-workers researched...’

The macro’s search features include:
1) Find a single word
2) (as above) Find two words (the first and last of the selection)
3) Find all the selected text (rounded to whole words at each end of the selection)
4) Case sensitive (Find: ‘Smith’, but not ‘Nasmith’ or ‘blacksmith’)
5) Whole words only (Find: ‘Smith’, but not ‘Smithson’ or ‘blacksmith’)
6) You can do a combination of (4) and (5)!

Here are the mechanics for the above options:
1) Click in a word and run the macro (or just run the macro and type in the word).

2) Click in the first word, shift-click in the other word and run the macro:
e.g. Smith & Brown 2021

3) To search for a phrase, do (2), but then click the right arrow (to clear the selection in the input box) and type a double quote mark (" or ”), then press Enter or click ‘OK’. (*The single QM symbolises that text as being a quotation.*)

4) To search case-sensitively, do (1) or (2), click the right arrow (to clear the selection in the input box) and type a hash (#), then press Enter or click ‘OK’. (*The # was chosen as it’s near the Enter key!*)

5) For a whole-word search, do (1) or (2), but click the right arrow (to clear the selection in the input box) and type a greater-than character (>), then press Enter or click ‘OK’. (*The > was chosen, as it’s used in wildcard searches to signify a word end.*)

6) You can combine any of the three limiting factors, (" or ”), # and/or >.

## More tips

1) If it makes it easier, e.g. with the Macro Menu, you can type your search into a separate blank line – I left a blank line at the top the Macro Menu, but any blank line will do. When you run the macro, it senses this is a short line and offers it to you as the search.

2) Remember that the default is insensitive, i.e. it finds any form of the word(s), but then (" or ”), # and/or > limit the number of finds.

3) If the macro generates a list of found that is very long, you could, of course, use any of the other find-type macros to look through the finds for specific words. You could even do a ListOfFound of your ListOfFound!

4) N.B. And when you’ve found what you’re looking for, you can jump straight back and to the original context with FindSamePlace – or FindSamePlaceBack, which is sometimes quicker.

5) If the text is a list, such as the Macro Menu, you probably don’t the ‘paragraphs’ double spaced. To achieve this, you can add an exclamation mark to the end of the search text. But for a list that you use regularly, you can add the file name to the list at the beginning of the maco:

```
listFiles = ",Macro_Menu,BirdList,"
```

Be careful to maintain the pattern of commas to delimit the filenames.

**Sub ListOfFound()**

## Find highlighted text

Each time you run this macro (*HighlightFindDown*), it looks for the next piece of text that is highlighted.

However,  if you want to only find text in a particular highlight colour then select a bit of text in that colour and run the macro from there.

And then the following macro (*HighlightFindUp*) does the same thing but upwards. So, if you are going down through the document looking at highlights using the first macro, and want to go back up and look at the previous, bit of highlighted text, run this macro instead.

You could, for example, assign the first macro above to Alt-PageDown and following one to Alt-PageUp.

Searching for any highlighted text is much quicker, so please be patient if you’re looking for a specific colour on long documents. Indeed, if it’s taking to long, the macro beeps and displays a counter, to reassure you that it’ll get there eventually. If you get bored, you can stop the macro with Ctrl-Break.

However, if you want to actually DO SOMETHING with the next bit of highlighted text then *SelectNextHighlight* will actually select the next bit of highlighted text, whereas *HighlightFindDown* leaves the cursor at the beginning of the highlighted text. And *SelectPreviousHighlight* does the reverse, of course.

**Sub HighlightFindDown()**

**Sub HighlightFindUp()**

**Sub SelectNextHighlight()**

**Sub SelectPreviousHighlight()**

## Find coloured text

This works in the same way as the previous macro but it finds text that is in a particular ***font colour***, say red or blue or whatever. Each time you run the macro, it jumps to the next bit of text in your chosen colour. If no text is selected, it remembers the font colour you looked for last time and finds the next bit of text in that same colour.

If you want to find text in a different font colour, simply select a piece of text in the desired colour and it will look for the next bit of text in *that* colour.

The other way to change the text search colour is to select a piece of text that is black. The macro then jumps to the next piece of non-black text, and you can say ‘yes’ or ‘no’ according to whether this is the colour of text that you want to search for.

If, however, you click ‘Cancel’, it will assume that you want to look through all the sections of non-black text.

**Sub FindColouredText()**

**Sub FindColouredTextUp()**

## Jump from (heading) style to style

(This is an old pair of macros, and the macro that follows them might be considered to be an improvement on it.)

The next two macros allow you to move up and down through the text on the basis of the style. So, for example, you might want to look through the various headings in ‘Heading 1’ style. Just place the cursor anywhere on a line in the given style and run the appropriate macro – up or down.

**Sub FindStyleOld()**

**Sub FindStyleOldUp()**

## Jump from (any) style to style

This *FindStyle* macro is, I think, an improvement on the previous one because I’ve used a different way of doing it. What this macro does is to assess the bit of text that has been selected by the user, checking if it’s italic or bold or small caps or super- or subscript, and then checking whether the font size is different from that of the Normal style (ditto the font colour and the font name) and then setting those attributes into the F&R. Then all you do is to use your normal *FindFwd* and *FindBack* keystrokes.

This means that, for example, you can select a superscripted character and then skip from one superscript to the next and back again. Or you could select a bit of a Level 2 heading and jump back and forth between the various Level 2 headings, or select a bit of small caps text and find other bits in small caps.

**Practicalities**
You can run *FindStyle* as a separate macro (see TheMacros), but I ‘attach’ it to the *FindOnly* macro. What I do is add a few extra lines:

**Sub FindOnly()**
```
' Version 16.12.10
' Calling FindStyle from the FindOnly macro

If Selection.Start <> Selection.End Then
  Application.Run MacroName:="FindStyle"
  Exit Sub
End If

myFind = InputBox("Find?", "Smart finder", Selection.Find.Text)
If Len(myFind) = 0 Then Exit Sub
etc, etc
```

What happens is that if no text is selected then it just goes into the main part of the *FindOnly* macro, but if some text is selected (which *FindStyle* expects) then it runs the *FindStyle* macro instead, so one keypress fulfils two functions.

**Sub FindStyle()**

## Jump to next applied style

If you're interested in paragraphs that are ***not*** in Normal style, but have an applied style then this macro jumps you to the next non-Normal paragraph.

**Sub JumpNextAppliedStyle()**

## Jump back to table of contents

After you have hyperlinked from an item in a table of contents (ToC) to that point in the text, it is good to be able to go back. You can use the command *WebGoBack* which, using the Customize Keyboard dialog, you can allocate to a keystroke. However, I found that it wasn't altogether reliable, so I created a macro that reads the current line, jumps to the top of the file and finds the first occurrence of that line, i.e. the item in the ToC.

The other advantage of this method is that if you are using the macro above to jump from heading to heading (style to style) then, whatever heading you have now reached, running this macro will jump you back to exactly the right place in the ToC.

**Sub ToCback()**

## Jump from comment to comment

*(Video: youtu.be/f7jb6zoh8l4)*

N.B. These macros are redundant as Word has them built in. From the Customize Keyboard dialogue box (call it up with the *CustomKeys* macro!) and select 'All Commands' in the left-hand list and then, in the right-hand list, first find *NextComment* and give it a keystroke, and then *PreviousComment* and do the same.

Yes, I know that you can use the browser arrows down at the bottom right of the window (below the scroll bar down arrow), but the same browser can be used for jumping between lots of different things. Having set up these two macros, I can leave the browser for jumping between other things, and know that any time I press Ctrl-Alt-Up or Ctrl-Alt-Down I'll be able to skip straightaway between comments.

**Sub CommentNext()**

**Sub CommentPrevious()**

## Jump into and out of a comment

Someone wanted a keystroke to move the cursor into and out of the comment box – this macro does that.

**Sub CommentJumpInOut()**

## Jump from edit to edit

*(Video: https://youtu.be/EaFJuKTbhGw )*

If you're wanting to look through all the track changes in a file – comments and changes – then you don't actually need a macro. Yes, I know that there are icons on the toolbar that allow you to go to the next and previous tracked edit, but if you, like me, prefer to use keystrokes rather than clicking icons, all you need to do is to use the Customize Keyboard dialogue box (call it up with the *CustomKeys* macro!) and select 'All Commands' in the left-hand list and then, in the right-hand list, find *PreviousChangeOrComment* and assign a keystroke to it. And then do the same for the *NextChangeOrComment* command.

Similarly for the *NextComment* command.

However, there is no such thing (well, not in Word 2010) as a *NextChange* command. (There's a *NextChangeOrComment* and a *NextComment*, but not a *NextChange*). So if you want to jump to the *changes* without having to go through the comments, here's a pair of macros.

```
Sub ChangeNext()
```

```
Sub ChangePrevious()
```

## Jump from footnote to footnote

Nothing too earth-shattering, but I just decided it would be useful to look back at the previous (or the next) footnote. If you're in the text next to the footnote citation number then the two macros below will jump you between the citations (e.g. to check that the citation numbers are outside the punctuation – here,[13] and not here[14].) If you then double-click the citation number, it will, of course, jump you to the footnote text. From there, the same macros will move you to the text of the previous or next footnote.

```
Sub FootnoteNext()
```

```
Sub FootnoteNextUp()
```

## Jump between main text and footnotes/endnotes

When someone asked for this, I thought it was hardly worth the effort, but having used it, I can see the value. Put the cursor anywhere in front of a note call-out, like this,[16] and press Alt-N (or whatever) and it looks through to find the next note and jumps straight into the note text; read and/or edit this note and then, without having to move the cursor or the mouse, click Alt-N again, and it jumps you straight back into the main text where the note is cited. So there's no need to carefully place the mouse pointer and double-click the note number.

[16]Here's the note text (in the footnote area in real use). The cursor can be anywhere in here if you want to jump back.

```
Sub NoteJumper()
```

## Jump from (section) number to number

(This is fairly basic – The next section gives a more comprehensive system.)

It suddenly dawned on me that it might be useful to be able to jump through a document via the numbered (sub)sections. So, the first two new macros (*NextNumber* and *NextNumberUp*) allow you to put the cursor next to, say, 3.1.1 (just to the left of the 3, I mean) and click (whatever keystroke) and it'll jump you to '3.1.2', then click again to '3.1.3', '3.1.4' etc – and the other macro takes you back up through the sequence, of course.

The same macros also work for 5.6 –> 5.7 –> 5.8 (and back again), and even for numbers without full points (periods): 467 –> 468 –> 469 (and back again).

And then I thought that it would be useful if you could jump from Table 3.4 –> Table 3.5 etc, or ditto for Figure 2.7 –> Figure 2.8, or Fig 1.2 –> Fig. 1.3 etc, etc – I'm sure you get the idea!

So the second two macros also look at the four characters before the cursor, and makes those part of the search.

So, basically, you again put the cursor just to the left of the number, and when you run *NextNumberPlus* (or *NextNumberPlusUp*) and it tracks up and down through the numbered tables/figures/equations etc.

However, you can also use this second pair of macros for section numbers, because if the number is the first item on the line:

**2.3 This is the title of my next section**

then it clocks that and only looks for 2.4 (or 2.2 if you're going up) where it occurs at the beginning of the line. This means that it does not then jump from the heading mentioned above to, say, 'Figure 2.4'.

Here are the four macros:

**Sub NextNumber()**

**Sub NextNumberUp()**

**Sub NextNumberPlus()**

**Sub NextNumberPlusUp()**

## Jump down through (section) numbers

This is a development of the idea of the previous macro, providing much more flexibility, with the aim of using it to check the continuity of the numbering in a document. Basically, you place the cursor in the section number or in the caption for a figure ('Figure' or 'Fig.'), a table ('Table') or a box ('Box'), and each time you run it, it will jump to the next numbered item in the sequence.

With captions, it detects if there's one item missing in the sequence. When it gets to the end of a chapter, it tries to find the first caption in the next chapter. So, for example, at 3.4, it looks for 3.5; if it can't find it, it looks for 3.6 and, if it finds it, it tells you 3.5 is missing. But if there's no 3.6 either, it looks for 4.1.

If the section numbering is hierarchical, it performs the following checks, stopping if it gets an affirmative:

1) Add '.1' to see if the section has a subsection (e.g. 4.2.3 –> 4.2.3.1).
2) Look for the next number up (e.g. 4.2.3 –> 4.2.4).
3) Check for the next number up (e.g. 4.2.3 –> 4.2.5), alerting you if there's a missing number.
4) It also checks for the next number up (e.g. 4.2.3 –> 4.2.6) in case two in a row are missing.
5) Check the next number down the hierarchy (e.g. 4.2.3 –> 4.3)
6) Jump to the next chapter (e.g. 4.2.3 –> 5.1)

**Sub FindNextNumber()**

As an extra aid, *FindNextNumber* remembers 'where it has come from', so if you want to jump back to the previous number, just run this extra macro. (I have *FindNextNumber* on Shift-Alt-Right, and *FindPreviousNumber* on Shift-Alt-Left as a way of thinking moving forward through the text and then one step back again.)

**Sub FindPreviousNumber()**

## Jump to an item in an auto-numbered list

The editor who requested this had a document with a long numbered list, each item consisting of one or more paragraphs. Then through the rest of the text were citations of, for example, 'see item 32'. The editor had to check that item 32 really ***was*** the item being referred to, but the trouble was that the list was auto-numbered, so you couldn't just search for, say, '32.' or '32)' or '32^t' because auto-numbers can't be searched with normal F&R. So a macro was needed. Here's one possible solution.

You open a second window on the file. Click on the citation number (say, 32) in one window and run the macro. It will take you straight to that item in the list, but in the other window.

**Sub ListItemFinder()**

In complex files that contain a number of different lists, the macro needs to know which list you're referring to, so you place the cursor at the beginning of the list you actually want to search, and run a second macro:

**Sub ListCheckStart()**

This macro suggests a value for `myOffset` at the beginning of the other macro. For example,

```
myOffset = 12
```

## Jump to next/previous table

If you want to look through all the tables in a document, these two macros jump you back and forth through them, one by one.

**Sub TableNext()**

**Sub TablePrevious()**

## Jump to next/previous paragraph that has borders

If you want to look through all the text in a document that is surrounded by a border, these two macros jump you back and forth through them, one by one. *(If these macros don't work properly, please let me know. They haven't been fully tested, sorry.)*

**Sub BordersNext()**

**Sub BordersPrevious()**

## List all figures/tables/boxes

*(Video: youtu.be/DnG1XCuOUlk)*

This looks down through the text and finds every single reference to Figure/Fig/Fig., Table and Box, and lists them in a separate file. You then end up with a list such as:

```
	Table 3.1
	Table 3.1
Table 3.1:
	Table 3.2
Table 4.1:
Table 4.2:
	Table 4.2
	Table 4.3
	Table 4.3
Table 4.3:
		Table 4.3
Table 5.1
Table 6.1
		Table 6.1:
	Table 6.2.
Table 6.2:
	Table 6.2
Table 6.3:
Table 6.4:
Table 6.5:
		Table 6.5
		Table 6.6
Table 6.6:
		Table 7.1
		Table 7.2
Table 7.1:
Table 7.2:
		Table 7.3
Table 7.3
		Table 8.1
Table 8.1
		Table 8.2
		Table 8.2
Table 8.2
		Table 8.3
Table 8.3
		Table 10.1
Table 10.1:

Plurals
Tables 4.5 and 4.6
Tables 6.3–6.5
```

To make it easier to see what's what, if the reference is in the middle of a paragraph, it's presumably a citation, not a caption, so the macro adds a tab in front of it. However, if (e.g. in 4.3 and 6.1 above) the reference appears at the start of a paragraph (i.e. looks like a caption), and yet there already seems to be a caption for that item, it adds two tabs – just as a way of making it stand out.

Then after creating the list, it checks through to see if any of the items has a caption but is not cited, or vice versa, in which case it highlights it.

But sometimes the citation is something like "Tables 6.3–6.5 show..." in which case it lists those citations at the end under "Plurals", so you can check whether those citations cover the (apparently) missing items.

**N.B.** As a quick way to check the consecutiveness of the numbers, you can use *FindNextNumber* to run through from the first item against the left margin, auto-repeating the macro (assuming it's assigned a keystroke). If there's a number missing, it'll bleep at you and tell you.

Another way in which consecutiveness can be checked is that the macro lists any lines starting "<Cap>" (or whatever you set at the beginning of the macro code):

```
<Cap>Figure 12.1 A detailed battery model applicable to NiMH
<Cap>Figure 12.2 Generic battery model.
<Cap>Figure 12.3 Typical battery powered electric vehicle dr
<Cap>Figure 12.4 First-order equivalent circuit model of the
<Cap>Figure 12.5 OCV versus SOC.
<Cap>Figure 12.6 Lumped parameter model of parallel-connected
<Cap>Figure 12.7 Ohmic resistance Rt identification result.
<Cap>Figure 12.7 Ohmic resistance Rp identification result.
<Cap>Figure 12.9 Ohmic resistance Cp identification result.
<Cap>Figure 12.10 MATLAB Simulink implementation of the mode
<Cap>Figure 12.11 Hybrid battery model.
```

You can immediately see inconsistencies.

Finally, when you are running this macro from within FRedit, make sure that, at the beginning of the macro it says:

```
mainDocBackToFront = True
```

which means, “bring the file you’re testing to the front” at the end of the listing process. Otherwise, the other items in the FRedit list will be executed on the fig/tab/box list instead of the actual chapter text.

**Sub FigTableBoxLister()**

## Create a list of all fig/tab/box captions

This looks down through the text and finds all occurrences of lines that have ‘Fig’, ‘Tab’ or ‘Box’ in **bold**, i.e. captions for said items. It creates a list of them in a new file.

If they aren’t in bold it will miss them! So change the line:

captionsAreBold = True

to False. That’s fine, but any paragraphs starting “Figure 4.3 shows...” will also be included.

**Sub CaptionsListAll()**

## Find short paragraphs (lists)

This looks down through the text and finds the next short paragraph. Useful if you’re trying to look through at all the lists in a long document.

**Sub ParaShort()**

## Highlight all lists

If you want to go through a text looking at all the lists and doing things to them, it might be helpful to highlight them all so that you don’t miss any. This macro adds a green (or whatever you prefer) highlight to every line/paragraph longer than 10 characters (`minLength`) and shorter than 150 characters (`maxLength`) (or whatever you prefer).

**Sub ListHighlighter()**

## Go to page

If you want to jump to a specified page number, you can, of course, use <Ctrl-F> and then select Go To Page, but this is much quicker. Instead, just press Alt-G (or whatever) to run the macro, type in the page number and press Enter.

**Sub PageHopper()**

And this more complex-looking one puts the first line of the selected page right up near the top of the screen, so that you see as large a section of the selected page as possible. This makes it easier to find the part of the selected page you want.

**Sub PageHopper2()**

## Easier scrolling

If you are reading the text on screen and editing as you go, how do you move the text up the screen as you read? I guess most people use the scroll-wheel on the mouse. I used to, but sometimes found that, as the screen display jumped up, I lost my place.

Here's the tool I now use. If you click somewhere in the line you are reading and run this macro, that line moves up the very top of the screen – so it's very easy to find.

And you can set it so that the macro leaves the cursor-line as the first, second, third line of the screen – your choice.

For example,

```
endLineNumber = 2
```

means it leaves the cursor-line as the second line of the screen.

**Sub JumpScroll()**

## Lost cursor – where was I?!

It may sound silly, but on a complex page, I sometimes put my cursor in a paragraph, get distracted by something and then can't find where I was up to. The solution is easy – use *JumpScroll*. This brings your cursor position up to the top of the page and therefore makes it easier to find.

Actually, I find it even more useful when I'm reading and reading and reading through some text. I put the cursor in the bit of text I've got up to and run *JumpScroll*, which brings the page up further with the cursor line at the top of the page.

## List of headings

(Some of the other macros now available in this section, such as *FindStyle* and *SmartFinder*, may have made this redundant, but it might still be useful for some purposes.)

When you run it, it creates, in a separate file, a list of the various headings. You can use this list to give you an overview of your document if you want to move about the document a fair bit.

So you can move to a specific line (heading) in the list-of-headings file, run the macro, and it drops you at that heading in the text. Run the macro again, and you're back in the list-of-headings file. You can save the list-of-headings file for future use. However, if you want to create a new list, simply select some text – any text – and the macro takes that as a sign that you want to create a new list, although it does ask if that's what you want to do, just in case that's not what you meant to do.

You can set up the style names that the macro will look for at the top of the macro – currently, it's set to find three styles: 'Heading 1', 'Heading 2' and 'Heading 3'. However, there is another way of setting it up which overrides these names.

At the very end of the file whose headings you want to list, put the word 'Heading', several times, in the various different heading styles that you want it to deal with:
**Heading**
**Heading**
*Heading*

The macro will then read these and copy them across to the ListOfHeadings file which means that the list will use the styles as defined in the document you're working on.

**Sub ListofHeadings()**

## List of headings by style

This macro creates a list, in a separate document, of all the headings in a file. It does so according to the style: Heading 1,Heading 2,Heading 3 etc. You could use it to list other style-based sections of text by putting the style names in the list at the head of the macro.

**Sub ListAllHeadings()**

## Show paragraph style colour + applied colour

If you want to know what the font colour of the current style is, and whether there is any other font colour applied on top of that style colour, this macro will tell you. It can display the colour numbers in hex and/or decimal.

**Sub FontColourReader()**

## Display and/or speak the style of a paragraph

If you want to know what the style of the current paragraph, this macros speaks it and/or pops up a message box.

**Sub StyleDetector()**

## Find any of these words

This macro allows you to specify a number of different words and to jump straight to the next occurrence of any of the words from your list.

The words to search are set in the line:

```
myWords = ":and:or:but:so:yet:if:"
```

you can add/subtract/change the list accordingly.

**Sub SearchTheseWords()**