16 Editing – comment handling

The comment facility can be useful for keeping track of author queries and/or notes for the typesetter and/or the desk editor. This section has a macro that provides a quick-and-easy way of adding comments, a macro for creating an author-query list, a macro for creating a list of all the comments in a set of files, and one that lists the names of those files that contain comments (maybe some don't), and lists all the comments in each.

And don't forget that you can jump easily from comment to comment with the *CommentNext* and *CommentPrevious* macros (in the earlier Navigation section).

However, since the advent of so-called "modern comments", many editors have felt the need to revise how we work in terms of generating author queries and creating lists of them. The old comments system worked well for this purpose, but many feel that the new modern comments are less than helpful. So here, in addition to the original system (usable if you are not yet having to use modern comments) we present an alternative system for handling author queries.

For historic reasons, the original set of comment handling macros have been retained, and are placed here, below the alternative system.

(First, the moving from comment to comment macros still work, but the cursor ***must*** be in the text, not in the comment.)

Jump from comment to comment

(Video: youtu.be/f7jb6zoh8l4)

Running these two macros allows you to skip up and down through the comments. When you get to the top or bottom (first or final comment), it will bong at you to indicate there aren't any more.

```
Sub CommentNext()
```

```
Sub CommentPrevious()
```

Creating (modern) comments using macros

(Video: https://youtu.be/Nmy4t2IxciU)

Yes, I know that you can't use macros inside modern comments, but this macro allows youo to do the next best thing...

CommentCompose (and the menu-based version below) allows you to create a comment within a separate Word file (which it sets up for you automatically); then you run the macro a second time and it creates a new comment in the file you were working on, and pastes in the text.

Composing the comment text within a Word file means that you automatically get curly quotes and can use macros such as MultiSwitch and SpellingSuggest (which I use a lot), and you can add features such as highlighting, bold, italic, etc (but not font colours).

So, in your current document, you select the range to be covered by the new comment and run the macro. If there is no suitable document open, the macro creates a new file which might be titled, say 'Document3', types in the filename of your working file, for identification purposes, and leaves you then to compose your new comment text.

When you have finished typing and/or copying and pasting text for the comment – and you can dip in and out of other files if need be – you simply run *CommentCompose* again. It realises this is a comment-composition file, and it knows the name of the target file, so it selects and copies the comment text, reverts to the target file, creates a new comment and pastes into it the new text.

Next time you try to create a comment in this way, the macro finds 'Document3', or whatever, sees that the filename matches your working file, and selects the old text in there, so you can easily delete it, and off you go again creating the text for your latest new comment.

Users of non-English-based computers can change the first line in the macro to, say:

```
stdName = "Dokument"
```

```
Sub CommentCompose()
```

Menu-based creation of (modern) comments

(Video: https://youtu.be/Nmy4t2IxciU)

This works in the same basic way as *CommentCompose*, but when you run it, it looks for a file in which you have placed a series of standard comments that you want to choose from. The list of available comments is displayed in a menu on screen, and you select, by key-press, the one you want.

It then types that text into the composition file (as used above), and you can then add extra text, or edit the standard comment in some way.

The system allow you to include, in the comment, the text range that was selected in your working file when you ran the macro. For example, you might select, say, "Brown 2004", and select a standard comment that comes out as:

> "Brown 2004" is not in the references list. (*But "Brown 2004" is*.)

You will find that the cursor is next to the "4" of the second date, so you could delete it and type 5:

> "Brown 2004" is not in the references list. (*But "Brown 2005" is.*)

The macro has various set-up options. For example, the above comment could be:

> **PB: p 300, ln 52.** "Brown 2004" is not in the references list. (B*ut "Brown 2004" is.*)

This option is set at the beginning of the macro:

```
' myPrefix = ""
myPrefix = "PB:  "
prefixBold = True

' refText = ""
' refText = "p <p> "
refText = "p <p>, ln <l>.  "
refBold = True

listName = "zzSwitchList"
```

The formatting of the prefix and/or the page and/or line number insertion is up to you. Adjust to taste.

So, to be clear, if you use `myPrefix = ""`, then there is no prefix. And you can have just the page number and not the line number, if you like, by using `refText = "p <p> "`. And they can be bold, or n ot, to taste.

The `listName` is the name of the file containing your list of standard comments. The list must be at the *end* of the file, and it would take this form shown below, followed by a view of the menu the macro creates on screen.

Note that it's the fourth item that generates the comment as above. So the <> is replaced by the text that was copied from the selected range of the comment. Then the "|" (vertical bar) in that item determines where the cursor end up, ready to delete and replace the date.

(You need to use <> if you want to carry the formatting from the original text range – such as bold and italic – into the comment, but use {} where you want the text to take on the formatting of the comment, e.g. the italic in the example above.)

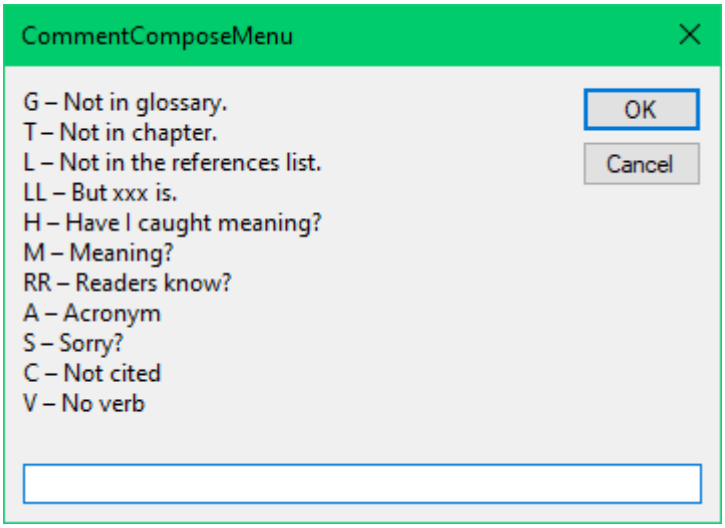
The list must start with some text inside "[[...]]" so that the macro knows where the list starts.

The text in square brackets (only in bold so that it's clearer to see what's what) is used for creating the menu; the text in front of the first "[" is what's actually copied and pasted to make up the comment text.

The letter(s) after the first "[" are the letters to use to choose that comment. It's only shown in uppercase to make it easier to see on screen; you can just press the lowercase letter(s), followed by pressing Enter.

[[comments menu]]
"<>" is not defined in the *glossary*. **[G – Not in glossary.]**
"<>" does not appear in this chapter. OK? **[T – Not in chapter.]**
"<>" is not in the **references list**. **[L – Not in the references list.]**
"<>" is not in the references list. (*But "{}"| is.*) **[LL – But xxx is.]**
"<>" – Have I caught the intended meaning? **[H – Have I caught meaning?]**
"<>" – Sorry but I'm having difficulty working out what this means. What does "<>" mean? **[M – Meaning?]**
"<>" – Will readers know what "|" refers to? If so, that's fine. **[RR – Readers know?]**
"<>" – Will the readers know this acronym? If so, fine. (It's not defined anywhere that I can see.) **[A – Acronym]**
"<>" – Sorry, but I can't work out the intended meaning here. Is it something like "<>"? **[S – Sorry?]**
Doesn't seem to be cited in the text. **[C – Not cited]**
"<>" – This sentence has no principal verb, so I can't work out the meaning, sorry. **[V – No verb]**

And here's menu screen this set of comments would produce:




```
Sub CommentComposeMenu()
```

List all modern comments

(Video: https://youtu.be/Nmy4t2IxciU)

This macro lists all the comments in a file, and it adds the comment(at)or's initials in colour and bold, as set by:

```
myInits = "PB"
myColour = wdBlue

myInits2 = "NH"
myColour2 = wdPink

' For any other initials
```

```
myColour3 = wdGreen
```

So you can specify two initials to have their own colours, and all the rest will be coloured in `myColour3`.

```
Sub CommentsModernCollect()
```

Delete modern comments

(Video: https://youtu.be/Nmy4t2IxciU)

If you place the cursor in the text covered by the comment range (i.e. inside the word 'covered', in this case) or anywhere in front of that range, then this macro will delete that comment.

If, however, you select some text, it will then ask you if you want to delete ALL the comments. What you do next is up to you!

```
Sub CommentsModernDelete()
```

Author queries / comments system

(This is an entirely new system that you would have to learn; I'm not convinced it's a realistic way to go, and I don't use it myself, but having cretaed it, I can't bear to just throw it away. :-)

Here we use the term 'QQcomments' to refer to the new queries/comments system, i.e. neither the traditional comments, or the "modern comments" that are gradually replcing them.

This system places the text of the QQcomments at the very end of the document, and uses the existing endnotes facility to 'index' these notes to the point in the main text to which they relate.

(If your text already uses endnotes for the document itself, then under the References tab is a command to convert the endnotes to footnotes; you can then convert them back later.)

Add a QQcomment

(*Video:* https://youtu.be/XShh1XtFPEc)

To add a QQcomment, place the cursor at the required point in the text and run *QQAddSimple*. If you want to associate the QQcomment with a selected area of text, you can use *QQAdd*. (And, once it's written!...) If you want to add a QQcomment from a menu of items, you can run *QQAddMenu*.

Because the QQcomments are stored in a different part of the document (at the end, after the main text), you need to be able to switch between each QQcomment and the text to which it relates. For this, you can use *QQSwitch*.

However, you may find it convenient to work in a split screen mode, or to have two separate windows open on the same document, so that both the QQcomments and the text are visible at the same time. *QQSwitch* simply selects the appropriate window/split, and scrolls the text to bring the target text into view.

As you add QQcomments, the links in the endnote area are automatically updated; however, when you delete the text of one of the QQcomments, the endnote link in the text is not deleted. Also, if you move pieces of the main text around, Word ensures that the endnote links change accordingly and are renumbered; however, the QQcomments section remains in the same order.

QQSwitch will still take you back and forth between text and the related QQcomment, but if you want to 'tidy up', you can run *QQUpdate*. This re-indexes all the QQcomments. While it doesn't change the order of the QQcomments, it does renumber them, so that, from the numbering, you can see which QQcomments are now out of order, and you can move them to match the order of the endnote links in the main text. (This is harder to explain than to use – watch the video.)

The only options in *QQAddSimple* are to set how 'visible' the endnote markers are. Normally, the markers would look like this,[12] but you can make them stand out, like this,[12] if you feel that helps idenify them:

```
qqBold = True
qqColour = wdBrightGreen
' qqColour = wdNoHighlight
qqSizeAdd = 4
' qqSizeAdd = 0
```

So `qqBold` could be switched off (`False`), and a different highlight colour could be used, or highlighting could be switched off with `wdNoHighlight`, and the amount by which the size is boosted could be changed (as you can see, using the larger size disturbs the line spacing).

When you've finished and want to delete all QQcomments, you can just delete the text of the QQcomments (and the title, 'QQcomments') and run *QQUpdate*; all the endnotes will be removed.

```
Sub QQAddSimple()

Sub QQSwitch()

Sub QQUpdate()
```

Add a QQcomment linked to a text range

(*Video:* https://youtu.be/XShh1XtFPEc)

When you apply an ordinary (including modern) comment, it is linked to and indicates an associated range of text, like the range of text here.

To provide a similar indication that a given QQcomment relates to a particular text range, you can use *QQAdd,*

The same macro also has a number of other optional features which you can select, according to taste.

But if you feel that sometimes you **want** to use a given feature, such as copying the selected text range, and at other times **don't** then you can create two copies of the same macro so that, say, *QQAdd* copies the selected text, ready to paste into the comment, while *QQAdd2* doesn't (so that you can paste some other previously copied text into the QQcomment).

Here are the option settings available:

```
doCopyText = True

addHighlight = wdYellow
' addHighlight = wdNoHighlight

addColour = wdColorBlue
' addColour = wdColorBlack

addUline = True

addPNum = True
addLNum = True
```

The first, if set as True means that the text range that is selected when you run the macro will be copied into the clipboard, ready to paste into the QQcomment.

You can set the highlight colour and/or the font colour and/or whether the range is underlined, according to taste; with all three, it would look like this section of text.[12]

When a given QQcomment text is first created, the final two options decide whether or not the comment starts with a page number and/or line number. Here's an example:[13]

> QQcomments
>
> [qq005] [p.3] [ln.35] This is the comment relate to "Here's an example".

The [qq005] is the unique QQcomment identifier, to link it, through the endnotes, to the relevant area in the main text. (More difficult to explain than to use – see the video.)

The endnotes section might look like this:

> [1] [qq001]
> [2] [qq004]
> [3] [qq005]
> [4] [qq002]
> [5] [qq003]

Another useful macro is *SplitScreen*, which splits the screen, at a particular percentage of the way down the window.

```
Sub QQAdd()
```

```
Sub SplitScreen()
```

Author query list compilation

(Video: youtu.be/Eiw1AE0hffs)

The idea of this macro, based on the CommentAdd macro above, is to copy all the comments out of one file, and put them in a separate file with an 'Answer:' line in between each query. For example, the commentsmight come out as follows (where the 'HB' is the author's initials, as set at the beginning of the macro):

> PB1: p.189 – "the comment I've added to this paragraph" – the first bit of this comment was added automatically, then I typed in this bit.
>
> Answer [HB1]:
>
> PB2: Another comment goes here. OK?
>
> Answer [HB2]:
>
> PB3: 'If you select some text' – Have I caught the intended meaning?
>
> Answer [HB3]:
>
> PB4: 'first character of the text that was selected' – Have I caught the intended meaning?
>
> Answer [HB4]:
>
> PB5: 'attached to the whole of the selected text' – Have I caught the intended meaning?
>
> Answer [HB5]:
>
> PB6: 'Gobblegook is gooble gobble.' – Sorry, but I can't work out the intended meaning here. Is it something like 'Gobblegook is gooble gobble.'?
>
> Answer [HB6]:
>
> PB7: (p. 212, line 19) 'the quoted text' Have I caught the intended meaning?

Answer [HB7]:

This macro also creates another file which is a list of any and all paragraphs that contain one or more comments. This therefore gives the author the context of all the queries for the chapter. It can optionally provide the page number (and line number), in the same way as CommentAddMenu.

You don't actually have to create a context file. This is set using: `createContextFile = True`.

When it finishes you can get it to pull the queries file, not the context file, to the front. Set `queriesFileOnTop = True` at the beginning of the macro.

Sub CommentCopier()

List all (old) comments

To create a list of modern comments,

Sub CopyComments()

If you have to do this for a number of files, and want to save the comments in a file, the following macro will save the comments that occur in a file called 'ThisFile' in a file called 'ThisFile_Cmnts'.

Sub CopyComments2()

List all comments – with numbers

(Video: youtu.be/f7jb6zoh8l4)

Someone suggested that it would be a help, when referring back to the text, to have the list of comments with the same commenter's initials plus index numbers as those in the file. This macro gives a list looking like this:

> [PB1]: 'Chapter 12.' – I can't work out the structure.
>
> [OU2]: I had a go at numbering – see comments below
>
> [OU3]: As comment below
>
> [PB4]: More than one technology so ideally subsections are divided again into subsections.

And I've added the 'Answer:' facility from the *CommentCopier* macro above. This facility is switched on and off with `addAnswers = True/False`.

Also, I want to delete from the list of comments all the comments containing 'T/S:' because these are comments for the typesetter, not the author, so I have:

```
deleteTScomments = True
TScode = "T/S:"
```

So any comment that contains 'T/S:' will be deleted from the list.

TIP: If, when you get the query answers back from the author, you use the macro *CommentNumbering* on the main text with its comments (= queries), it will add the numbers into the text of the comments:

> [PB1]: **[PB1]** 'Chapter 12.' – I can't work out the structure.
>
> [OU2]: **[OU2]** I had a go at numbering – see comments below

[OU3]: **[OU3]** As comment below

[PB4]: **[PB4]** More than one technology so ideally subsections are divided again into subsections.

Why is that useful? Because, when you implement the answer to the query 1 and delete it, the label numbering is automatically updated, giving:

[OU1]: **[OU2]** I had a go at numbering – see comments below

[OU2]: **[OU3]** As comment below

[PB3]: **[PB4]** More than one technology so ideally subsections are divided again into subsections.

All the comments are now renumbered – confusing! But the original numbering is preserved within the text of the comments. (This is much more difficult to describe than it is to do. Try it, and you'll see what I mean!)

```
Sub CommentListNumbered()
```

List all comments – with 'all sorts of stuff'

(Video: youtu.be/f7jb6zoh8l4)

This version brings together a whole range of different aspects of each comment, and puts the whole lot in a table. The idea is that the macro, as it stands, puts in ***all*** the available information, and you can edit the lines in the macro to cut it down to just the items you want (I'll tell you how, in a minute).

The information you can have is:

page number
line number
commenter's initials, e.g. [PB]
...with index number, [PB1] – [PB8], say.
scope (i.e. region of text covered by a comment's highlighting)
and, of course, the content of the actual comment, including its formatting, fonts, bold, italic, super/subscript etc.

New addition: `hd` is the heading of the section in which the comment occurs. However, do you want it to show the nearest sub level, or sub-sub-level? This is set at the beginning of the macro by: `maxHeading = "Heading 3"`

Also, you can split (some of) the comments, and the two 'halves' of the comment will appear in two (adjacent) column.

Finally, you can sort everything into the final table on alphabetical order based on one of the columns.

Sample output:

Author queries – Chapter 1

		Existing text	Who	Comment/query	Response
PB1	p3 l.39	*Cole's Management Theory and Practice*		Have we a ***references list*** anywhere that gives details?	Please insert response in this column.
PB3	p9 l.4	To contribute to the meeting when considering project acceleration		Have I caught the intended meaning?	
PB5	p9 l.15	This person is one who has held a similar position on a previous contract.		In which case, how can anyone start?	
PB7	p11 l.30	a temperature of −5°F		Does he really mean −20°C? And will readers nowadays have a feel for °F? I don't.	

PB9	p12 l.31	These notes have been abstracted from a publication by the Institute of Clerk of Works and Construction Inspectorate – version 2 March 2010.		**Do you need permission?** If not, fine.	
PB6	p11 l.29	"It becomes a cultural shock	PE	Again, who is this a quote from?	
PB8	p11 l.34	Illustrating the engineer's role	PE	Have I caught the intended meaning?	
PB2	p5 l.40	"The overriding feeling is that it successfully works."	RS	*Who is this quoting?*	
PB4	p9 l.7	the blue-arse fly syndrome where one spends time flitting between sites	RS	Are you happy that the term 'blue-arse' is appropriate for this level of book? If so, fine.	

How is this achieved? OK, the key is that I use the vertical bar '|' as the delimiter to show column changes, and there's a line in the macro for the text *before* the comment...

```
cmnt.Range.InsertBefore Text:=inits & itemNo & "|" & "p." & pn & " l." & ln &
"|" & scp & "|"
```

then a line for the text *after* the comment...

```
  cmnt.Range.InsertAfter Text:="|"
```

then a line for the headings...

```
Selection.TypeText Text:="||Existing text|Who|Comment/query|Response" & vbCr
```

Sorting is done on the basis of...

```
sortOnColumn = 4
```

If you don't want it sorted, use `sortOnColumn = 0`. (The sample above is sorted on column 4, in this case the 'Who' column.)

Splitting the comments in two is done, again using the vertical bar, so for one author I've added 'PE|' at the beginning of the comment, and 'RS|' at the beginning of others. If a comment has no '|' then the column is left blank.

To make the split happen, use...

```
splitComment = True
```

```
Sub CommentCollectTabulated()
```

Add a comment

(Video: youtu.be/f7jb6zoh8l4)
N.B. This does *not* work with modern comments!
*(This is an **old** macro. The next two macros, which add standard comments from a menu, are newer and, I think, better. And a newer still* CommentAddSimple *is a very simple way to open a comment with a standard initial text.)*

(The latest version, Dec 2019, adds the line `copySelectedText = True`. If you change it to 'False' then the selected text will not be copied into the comment.)

If, like me, you use Word's comment facility to report queries to the author, then you will constantly be marking bits of text and transferring them into a Comment bubble. Why not do it automatically with a macro? This is the format I use, and I've tried to put in lots of options at the beginning of the macro (*CommentAdd*) so that you can do it in different ways.

What I like to do is demonstrated by the comment I've added to this paragraph, and you can see that it has changed the section of text that I'm highlighting for the author into blue font.

So, you just select the piece of text that you want to quote in the comment, and run the macro. If you don't like the text being in blue, just change that line at the beginning of the macro. And if you prefer it to be highlighted instead (or as well!) you can adjust the various `True` and `False` items in the first few lines. The automatic adding of the page number is optional as well, as is the optional addition of a line number.

If, however, you just want to add a comment without a quote, place the cursor at the point you want to indicate, and run the macro. If no text is highlighted, then it just produces the comment box as here with the 'PB:' added.
But if you don't want 'PB:' at the beginning, but something different then that too is set up in one of the first few lines of the macro.

I've also added a feature because some people prefer to type their comments into the Comments pane, and not into a comment bubble. This is set with `keepPaneOpen = True` at the beginning of the macro.

And I also use a second copy of the same macro (*CommentAdd2*), which I use (a) to say, 'Bloggs 2001 is not on the references list' and (b) to add a blank comment addressed to the typesetters: 'T/S: '. The way this works is that if some text is selected (the missing reference), it does (a), but if nothing is selected, is does (b). I achieve the necessary effect for (a) by using another of the options (`postText = " is not on the refs list"`).

And then for a way of collating all these comments into an author query sheet, see 'Author Query Lister' below.

(But don't forget that Word's own shortcut, Ctrl-Alt-M, adds a plain, ordinary, empty comment.)

```
Sub CommentAdd()
```

```
Sub CommentAdd2()
```

Add a comment from a menu

(Video: youtu.be/Eiw1AE0hffs)
N.B. This does *not* work with modern comments!

(N.B. As this feature is so flexible, it's really hard to explain effectively how it works by just using words, so I do urge you to watch the video.)

(Also, as it's very flexible, it's harder to understand the instructions, and harder to set up than the very latest macro; see below.)

Having used *CommentAdd* and *CommentAdd2* a lot, I decided that it would be useful to have a selection of different standard comments such as '*Have I caught the intended meaning?*' or '*This citation is not on the references list.*' or '*Will the readers know this acronym? (It's not defined anywhere.)*' So I've set these up as a menu and stored them as a list in the zzSwitchList file that I use with MultiSwitch. (If you don't yet use MultiSwitch, it's really worth looking at; it saves me a lot, lot of time and avoids mistyping. But anyway you can create a zzSwitchList file just for this application.)

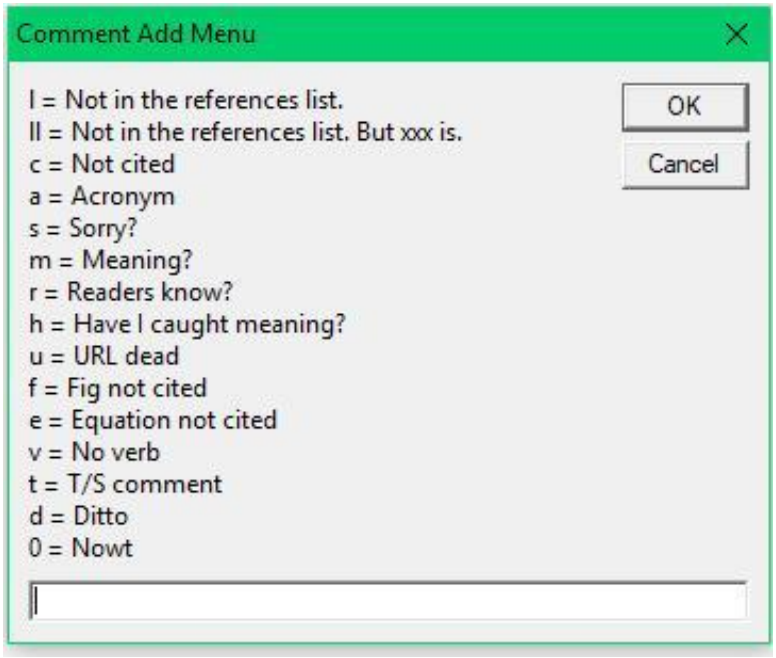
If you select some text in the working file and run this macro, then choose an option, the selected text will be copied into the comment bubble, along with your choice of comment. The formatting of the comment, including any possible highlighting, font colouring and underlining is set in the list in zzSwitchList.

Here's a sample list, illustrating the sorts of arrangements and formatting you can achieve (it's not meant to be realistic, but just to illustrate the various features):

[l] '<>' is **not** in the *references list*. {Not in the references list.}
[ll] '<>' is not in the references list. (But **][** is.) {Not in the references list. But xxx is.}
[c] '<>' – Not cited in the text? Do you want to cite it somewhere or delete it? {Not cited}
[a] '<>' – Will the readers know this acronym? If so, fine. (It's not defined anywhere that I can see.) {Acronym}
[s] '<>' – Sorry, but I can't work out the intended meaning here. Is it something like '<>'? {Sorry?}

[m] '<>' – Will the readers know what this means? If so, fine. {Meaning?}
[r] '<>' – Will readers know what ***'//'*** refers to? If so, that's fine. {Readers know?}
[h] '<>' – Have I caught the intended meaning? {Have I caught meaning?}
[h] '<>' – Have I caught the intended meaning? {Have I caught meaning?}
[u] '<>' – This URL does not seem to work. {URL dead}
[f] '<>' – This figure is not cited in the text. Do you want it cited? If so where? {Fig not cited}
[f] '<>' – Not cited in the text. Where do you want it cited? {Figure is not cited}
[f] '<>' – This is not cited in the text. Where do you want it cited? {Fig not cited}
[e] '<>' – Equation citation missing; please supply. {Equation not cited}
[v] '<>' – This sentence has no principal verb, so I can't work out the meaning, sorry. {No verb}
[t] [T/S:] {T/S comment}
[t] [T/S: This needs formatting in MathType, please.] {T/S: format}
[d] Ditto {Ditto}
[0] {Nowt}

The on-screen menu for the above list looks like this:



As you can probably see, the menu is made up by using the letter(s) inside square brackets plus the text in curly brackets from the end of the line. What text you use (and what menu letters) is entirely up to you. You'll see that there are multiple entries for some of the letter-options, but the macro just uses the first version of any given letter-option that it finds – this makes it very easy for you to change the wording you use for different jobs. Indeed, you could have, in your zzSwitchList file, two or more complete sets of comments; the macro starts from the first square-bracketed letter, and stops at the first blank line, so you can move an alternative list of comments up from lower down in the zzSwitchList file, so that this alternative list is the first list that the macro finds.

The setup for comments is very flexible, including being able to use formatting, highlighting and font colouring within the comments. However, any highlighting and font colouring (and underlining) applied the square-bracketted text is what the macro then applies to the actual text back in the chapter file, not the text in the comment bubble. If you use any of these three attributes, the comment bubble is only attached only to the very first character of the text that was selected, while the attribute shows which text was selected. If there is no formatting on the square brackets then the comment bubble is attached to the whole of the selected text.

The text that was selected is pasted into the comment, at a position indicated by the '<>' symbols, and you can even use it more than once, as in my option [s] above, where it quotes the text and then says, "Do you mean something like..." and then pastes the text in again, so that I can edit it into what I ***think*** the author meant to say.

Gobblegook is gooble gobble.

And as you can see, by applying some formatting to the '<>', the quoted text then takes on that formatting.

On top of that, you can also bring over some of the formatting of the text you're quoting, over in the comment. I wanted this because in my technical documents, I might want to quote something like 'The acceleration is 12 ms^{-2} and blah blah'. So if you want this feature, use `copyAsPureText = False`. In other words, don't just copy pure text, but do allow some formatting to be copied across. The macro currently brings across super- and subscript, plus bold and italic. If you want other features, let me know, but I have put in an option to copy across the actual font name; this is set by setting `keepTextFont = True`.

Another point to note is that, as the macro is currently set up, if no text is selected when you run it, it automatically selects the whole of the current sentence, which I find quite a useful feature.

If you look at the [ll] option, you'll that see I've used '][' . The idea is that once the comment bubble text has been created, the cursor is then placed at that particular point, and the square brackets are deleted; that is then ready for you to type in some specific text. And that typed-in text can, if you want, have some different formatting, say bold. However, it's no good just making the back-to-back brackets bold because they get deleted, and the bold disappears. You have to also apply the bold (or whatever) to the character *before* the][as well as to the][themselves.

You can also choose to have every comment showing (a) a prefix, (b) the page number (and also (c) the line number) of the quoted text, as in this bubble here. These use the settings:

```
myPrefix = "PB: "
addPageNum = True
addLineNum = True
```

(But note that this is the page and line number *at the time when the comment is created*. This means that if you later add/subtract chunks of text, then these numbers will no longer be incorrect. Probably more helpful is to use the page (and line) numbers that are generated by the macros CommentCopier and CommentCollectTabulated below.)

```
keepPaneOpen = False
```

This final other option decides whether the Comments pane is left open after the macro finishes. (It has to be open while the comment is being compiled, but you may or may not want it to stay open.)

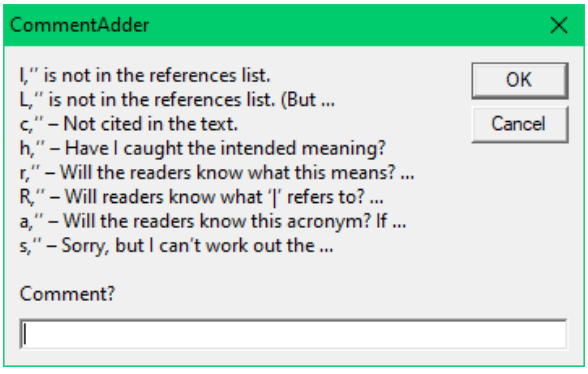
```
Sub CommentAddMenu()
```

Add a comment from a menu

(Video: https://youtu.be/4Ln95a1Cqyc)
N.B. This does *not* work with modern comments!

The idea of this macro, like the previous one, is to add a comment by selecting it from a list that you create, but in this case, the setting up is much simpler – you include your own comments at the beginning of the macro.

To use it, you select the area of text to which the comment relates and run the macro. You then select the comment you want by letter (or you could use numbers):



it then opens a new comment box and types in your chosen text.

Within that comment, the macro is able to (a) type a copy of the selected text and (b) finish by leaving the cursor at a particular position within the comment, ready for you to type something.

Here's a sample line in the macro; it defines one comment:

```
cm = cm & "L,'[]' is not in the references list. (But '[]|', is.)\"
```

So to activate this comment, at the prompt, you would type 'L' and press Return.

If the selected text was, say 'Brown, 2016', then the new comment in the box would contain:

> 'Brown, 2016' is not in the references list. (But 'Brown, 2016|', is.)

And the cursor would be placed where I've left the '|' character (but that character will have been deleted). The idea is that you might then delete the '6' and type '5' instead, or whatever, i.e. there is a 'Brown' reference, but it's the wrong year. That comment is then ready to be lifted out of the document – along with the other comments – by using, say, *CommentCopier*, to generate a queries list for the author.

Here's the start of the macro:

```
' myDefaultSelect = "sentence"
myDefaultSelect = "word"

useDoubleQuotes = False
' useDoubleQuotes = True

cm = "Start\"
cm = cm & ",Dummy\"
cm = cm & "l,'[]' is not in the references list.\"
cm = cm & "L,'[]' is not in the references list. (But '[]|', is.)\"
cm = cm & "c,'[]' - Not cited in the text.\"
cm = cm & "h,'[]' - Have I caught the intended meaning?\"
cm = cm & "r,'[]' - Will the readers know what this means? If so, fine.\"
cm = cm & "R,'[]' - Will readers know what '|' refers to? If so, that's fine.\"
cm = cm & "a,'[]' - Will the readers know this acronym? If so, fine. (It's not
defined anywhere that I can see.)\"
cm = cm & "s,'[]' - Sorry, but I can't work out the intended meaning here. Is
it something like []?\"
```

The first two lines determine whether, if no text is selected when you run the macro (i.e. you've just clicked in a word), it assumes you want to select the current word, or the whole of the current sentence.

The second two lines decide whether the macro uses single or double quote marks – your choice. (It also changes spaced hyphens into spaced en dashes in the comment text.)

Each line that defines your choice of comments has to be punctuated exactly as I have done:

```
cm = cm & "h,'[]' - Have I caught the intended meaning?\"
```

So the 'h' is the letter that you have to type to select it; then a comma; then the comment; finally, a backslash to define the end of the comment. The '`[]`' is the marker to show the macro where you want to place a copy of the selected text, and you put a vertical bar, '|', where you want the cursor to end up.

Obviously, you would use your own wording, but the 'Dummy' line makes it easier for you to add extra lines while strictly maintaining the punctuation pattern of each line. So copy the whole of the 'Dummy' line, and add your chosen letter, and replace 'Dummy' with your chosen text:

```
cm = "Start\"
cm = cm & ",Dummy\"
cm = cm & "l,'[]' is not in the references list.\"
cm = cm & "L,'[]' is not in the references list. (But '[]|', is.)\"
```

```
cm = cm & "c,'[]' - Not cited in the text.\"
cm = cm & "h,'[]' - Have I caught the intended meaning?\"
cm = cm & "r,'[]' - Will the readers know what this means? If so, fine.\"
cm = cm & ",Dummy\"
cm = cm & "R,'[]' - Will readers know what '|' refers to? If so, that's fine.\"
cm = cm & "a,'[]' - Will the readers know this acronym? If so, fine. (It's not
defined anywhere that I can see.)\"
cm = cm & "s,'[]' - Sorry, but I can't work out the intended meaning here. Is
it something like []?\"
```

```
Sub CommentAdder()
```

Add a comment – simple version

N.B. This does *not* work with modern comments!

This macro opens a new comment, and adds a standard text – maybe your initials or some such – and then leaves the cursor in the comment, ready for you to add some text.

You can choose to do this in the bubble or in the comment pane. This is set by:

```
keepPaneOpen = False
```

Or use True if you perfer to be in the Comments pane.

```
Sub CommentAddSimple()
```

Open the comment pane

(There might be a non-macro way of doing this; if so, please let me know. Ta!)

```
Sub CommentsPane()
```

List all commented paragraphs

This macro creates a new file consisting of any and all paragraphs that contain one or more comments.

```
Sub CommentContextCopier()
```

Collect all comments in a set of files

(Video: youtu.be/f7jb6zoh8l4)

(***Mac users!*** Visual Basic on Macs has 'interesting' file handling. Mac users say that this macro seems to work OK provided that the **filenames are short**, say, less than 18 characters.)

This macro produces a single file containing all the comments copied from all the files in a folder, giving the file name of each and a list of the comments therein. It is basically a multifile version of *CommentCopier* above.

As with my other multifile macros, the macro gets you to identify the folder containing the files by bringing up the Open File window. Navigate to the required folder and click Cancel. The macro then asks whether you want to work on *all* the Word files in that folder. If you say 'Yes', it uses the complete list of files that it has created and works through them all one by one.

If you say you *don't* want to work on all the files, the macro just stops. The list will look then something like this:

C:\Program Files\VirtualAcorn\VirtualRPC-SA\HardDisc4\MyFiles2\WIP
Macro Jobs.doc
Roman cats.doc
Stats.docx

The point is that you can then edit this list, either deleting files you don't want included, or putting a vertical bar ('|') in front of any you want it to ignore.

If you now run the macro again, it recognises that this Word document is a file list and so it proceeds to work through the listed (and not the ignored) files, opening each one and creating the list of comments.

The insertion of the "Answer:" lines is optional, by using `addAnswerLine = True/False`.

```
Sub MultiFileComment()
```

Add or remove fixed comment initials and numbers

(Video: youtu.be/f7jb6zoh8l4)

When comments are displayed in bubbles, you see things like:

> **[PB1]:** 'Chapter 12.' – I can't work out the structure.
>
> **[OU2]:** I had a go at numbering – see comments below
>
> **[OU3]:** As comment below
>
> **[PB4]:** More than one technology so ideally subsections are divided again into subsections.

The initials are those of the person who added the comment, and the numbers are just an index from 1 to however many comments there are in total by that person.

However, the initials and numbers aren't actually part of the comment text, and therefore aren't extracted by the macros above. So someone suggested that it would be helpful to be able insert the initials and index number into the text – a sort of 'hard' numbering. Then, as you deal with author queries and delete the comments, you will still have the original numbering there for reference. So the macro below produces:

> **[PB1]:** [PB1]: 'Chapter 12.' – I can't work out the structure.
>
> **[OU2]:** [OU2]: I had a go at numbering – see comments below
>
> **[OU3]:** [OU3]: As comment below
>
> **[PB4]:** [PB4]: More than one technology so ideally subsections are divided again into subsections.

Then, if you delete, say, the first two comments you end up with:

> **[OU1]:** [OU3]: As comment below
>
> **[PB2]:** [PB4]: More than one technology so ideally subsections are divided again into

But then I reckoned that you might add or subtract comments for other reasons and therefore might need to redo the hard numbering. No worries. The macro first of all checks whether the first comment has a hard number. If so, it deletes the hard numbering from all the comments.

Run the macro again and it adds a new set of hard comment numbering.

```
Sub CommentNumbering()
```

Add comment initials and numbers to text

(Video: youtu.be/f7jb6zoh8l4)

For comments such as the ones listed above, this macro adds [PB1], [OU2], [OU3], [PB4] etc into the text at the points to which each comment bubble points. Then if the comments are deleted, we still know where they came from, and which number they were.

`Sub AddCommentMarkersInText()`

Edit the initials within comments

If you need to change the initials assigned to the comments in a file, you can either change every single one to a specific initial or do it selectively, like doing a find and replace.

`Sub CommentInitialReplaceAll()`

`Sub CommentInitialFandR()`

Delete all comments

Does what it says on the tin.

`Sub DeleteAllComments()`

Add a comment from a list to a PDF

Someone asked for the equivalent of *CommentAddMenu* for a PDF. Unfortunately macros only work in Word. What I suggested, however, is to create a list of comments as a Word file, click in the list, run the macro, and it will go through the list and pick up the comment you wanted and copy it to the clipboard. Then all you have to do is go back to the PDF and paste it.

The list needs to look like this:

1
This is the first comment.
2
This is the second one – and each comment can be as long as you like, and can have things like **bold** and *italic*, and even super/subscript, e.g. H_2O.
a
Another thing I say often. And this is two sentences – wow!

So when you run the macro, it asks for the code, and you just type in whatever letter/number/symbol you want to use (just a single character).

`Sub CommentPicker()`

Add a comment from a list into the main text

Extending the idea of the macro above, you can create a list of comments, as before, and when you run the macro and type a letter to select a comment from the list, it will then type it into the text of the main Word file, where the cursor was placed.

The file containing the comments must have the file name specified at the beginning of the macro:

```
commentDoc = "CommentList"
```

One extra feature is that you can insert the comment, but specify where, within the comment, the cursor is to be left. For example, the comment might be “This is used on page [page number], where it has a different meaning.” So in the comments list, you would put:

This is used on page ><, where it has a different meaning.

And after typing the comment into the text, the cursor will be at the position of the ‘><’.

Possible error: if it generates an error, saying “Bad file name” then you’ll need to change:

```
commentDoc = "CommentList"
```

to

```
commentDoc = "CommentList.docx"
```

or `"CommentList.doc"`, if you, like me, generally use .doc files.

Sub CommentPickerInserter()

Change the scope associated with a comment

(This macro gets a mention in video: youtu.be/)
Suppose you’ve added a comment about something, and then you realise that you meant to extend the selection to *include* the comma, before adding the comment. Is there a way of adjusting the scope of the comment, or do you have to copy the text of the comment, delete the comment, and add a new comment, and paste back the text? Yes, you do! (Well, I haven’t found a way to actually *change* the scope of a comment.) However, you can of course, get a macro do that for you. So you would select ‘something’ *plus* the comma, then run the macros, and it would become:

“Suppose you’ve added a comment about something, and then...”

Here’s the macro:

Sub CommentChangeScope()

Solve problems when typing text into comment bubbles

You sometimes get strange effects in comment bubbles, such as weird use of open and close quotes, or odd text size or even the inability to type text in, except v-e-r-y--s-l-o-w-l-y. This is sometimes associated with the file being sourced from another language group. The following macro sometimes helps.

Sub CommentTextNormalise()