Colours in Word 2007

Introduction

In common with many other applications, Word used to have rather limited colour palettes, but the situation was improved in Word 2000 and has, until 2007, remained relatively stable. From Word 2000, with a couple of exceptions, any component of a Word Document which could be coloured, could be given any of the 16 million or so colours available in the so called RGB colour space. Components of documents could take their colour from one specified in a Style or be given a colour directly; some direct colour formatting could also be applied with an early incarnation of a Themes feature.

Whatever the source of a colour, it was specified at that source by picking from a glittering dialogue or by explicitly specifying the amount of red, green, and blue which made it up. However specified, once chosen it was fixed as an RGB value, and stored, both in Word's workspace and in the Document, as a number representing that RGB value. The only way to change a colour was to change that specific number, either by letting Word do it based on selections through the UI or directly in VBA (or other) code.

Word 2007 has a new colour model, which forms part of the new Themes feature. A colour can still be specified as an absolute (RGB) colour but it can also be specified as a colour, or as an adjustment to a colour, belonging to a colour scheme, itself part of a Theme. If a change is made to the colour scheme, that change is reflected immediately in anything coloured with reference to it. The inclusion of this feature has some effects in the User Interface but has a bigger impact on VBA (or other code) solutions that work with Word documents.

This monologue details some aspects of the use of colour in Word Documents, particularly in VBA code. The 2007 Themes feature is common to Word 2007, Excel 2007, and PowerPoint 2007 and, consequently, much of what is written here will also apply to Excel and PowerPoint. Previously, however, the three applications had rather different implementations of colour (Excel had a limited palette, for example, and PowerPoint had a simpler colour scheme system) so this similarity is limited. When I have time and inclination I may expand the scope to include backward compatibility issues with Excel and/or PowerPoint, but currently this is limited to Word.

Themes and Schemes

It must be hard to find terminology for new features that does not clash with existing terminology and yet still conveys some meaning. It is equally hard to find terminology to explain the new features so please accept my apologies in advance if you find what follows to be rather strangely worded; it is just my best attempt at avoiding confusion whilst trying to maintain accuracy.

A Theme is a collection of formatting characteristics, some or all of which can be used in place of absolute formatting, whether applied directly or as part of a Style. The concept of a Theme is open ended but the implementation in Word 2007 consists of just three (fairly wide ranging) components: colours, fonts, and effects. Of interest at the moment are colours, and the colours that belong to a Theme are grouped together as a Colour Scheme.

A Colour Scheme consists of twelve base colours: two background colours, two text colours, six complementary 'Accent' colours, and two hyperlink colours. Despite their names there is no restriction on usage of any of them and, for example text can be coloured with any of the Accent colours or a background colour. As well as just the base colours, it is possible to apply a lightness or darkness to anything coloured with a scheme colour by specifying a percentage lighter or darker than any of the base colours. In this way, a range of co-ordinated colours can be used in a document and, by a change of Colour Scheme within a Theme, or complete change of Theme, they can all be changed at once.

UI Considerations

As it is essentially background reading, details of how Word handled colour in versions prior to Word 2007 has been removed from this page and given its own home. If you are not familiar with earlier releases you may like to read it:

⇒ Colours in Word 2003 (and earlier versions) [*link to 2003.php on this site*]

Conceptually, Word 2007 is much the same as earlier versions of Word, in that it still has a series of progressive dialogues for changing colours (and the same exceptions). Where it differs is in the first dialogue, which no longer has the fixed 40-colour palette; it does have a limited palette of ten, what it calls, standard colours but they are supplementary to the main palette of ten groups of Theme colours:

The Word 2007 Font Color Dropdown

The *Theme Colors* at the top are presented in two blocks. The first row contains ten of the twelve colours in the current colour scheme, the two missing colours being the hyperlink and followed hyperlink colours, which, as far as I can tell, can not be used anywhere in the User Interface, even on their intended targets. It should be noted that, in Word 2007, all documents have a Theme and, thus, a Color Scheme; if one is not specifically assigned, a default one is used.

The second block presents variations on each of the scheme colours – lighter or darker by what is indicated as a percentage. The actual numbers used for the percentages are, to some extent, irrelevant in the UI but it is as well to understand that they are actually percentages of available lightness (or darkness) rather than percentages of the lightness of the base colour itself; 100% lighter means 100% light (i.e. white) and 100% darker means 100% dark (i.e. black). Although it is possible to set a colour to any percentage darker or lighter up to 100%, the UI only supports a small subset – a few shades darker for the Background 1 and Background 2 colours, a few shades lighter for the Text 1 colour, and three lighter and two darker options for the Text 2 and all the Accent colours.

The so-called *Standard Colors* are standard only in as much as they are the same in Word, Excel, and PowerPoint. They are not standard by any other definition I know; they are not Windows colours, they are not basic colours (red, green, blue, etc.), they are not the same as any group of colours ever presented in an Office UI in the past, or ever grouped together in any way in VBA or the object model, and only two of them are considered standard enough to be included in the 88 standard colours in the full colour dialogue. They are built into the User Interface and they cannot be changed in any way; perhaps that's what makes them standard.

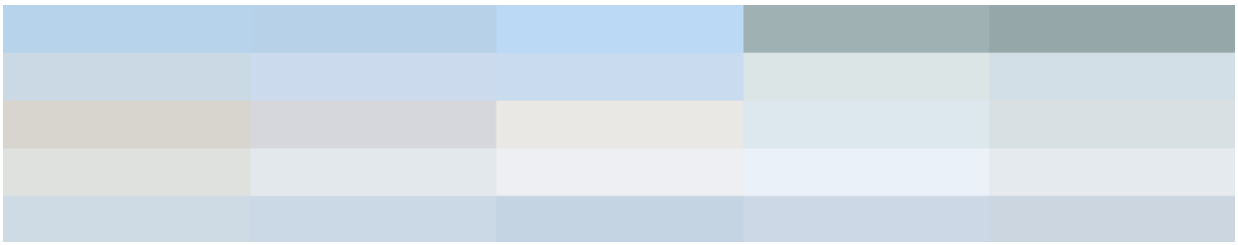
The *Recent Colors* towards the bottom are as they ever were, and only shown when there is at least one recently used non-Theme, non-standard colour to show, and the *More Colors…* choice gives the same expanded colour selection dialogue that was in earlier versions of Word. Some of the dropdowns, Shading, for example, have a *No Color* option instead of the *Automatic* one. It is also possible to have custom colours in a Theme (PowerPoint MVP, Echo Swinford, shows how to set them up, here [*link to Echo's site at http://www.echosvoice.com/2007/customcolors1.htm*]) and, if you have any, they will appear on some of the dropdowns, but there doesn't appear to be any rationale for which ones show them and which don't. The options vary from dropdown to dropdown, but the basic format is always like the Font Color one shown.

When a colour has been chosen, Word 2007 stores it, in its workspace, as selected; that is, if a fixed colour is chosen Word stores the number representing that fixed colour. If, however, a Colour Scheme colour is chosen Word will store a different number referencing that Colour Scheme and any adjustments to its brightness; it will not store the RGB value itself. If the document is later saved in Word 2007 format (.docx or .dotm) then the same values will be stored in the document. If, on the other hand, the document is saved in Word 97-2003 format (.doc) all the scheme colour values will be converted to absolute RGB values for the colours in the scheme active at that moment; this degradation in colour is not picked up by the Compatibility Checker and no warning about it is given.

To summarise, Scheme colours are always available and maintained in Word 2007, even when working in compatibility mode, and are always silently removed when saving in Word 97-2003 format. There are a couple of potential issues. The first, a relatively minor one, is that if people are working on a document both in Word 2007 and an earlier version they will not be working with a consistent colour palette, but perhaps the bigger problem is the potential loss of Scheme colours that may have been carefully and unwittingly chosen.

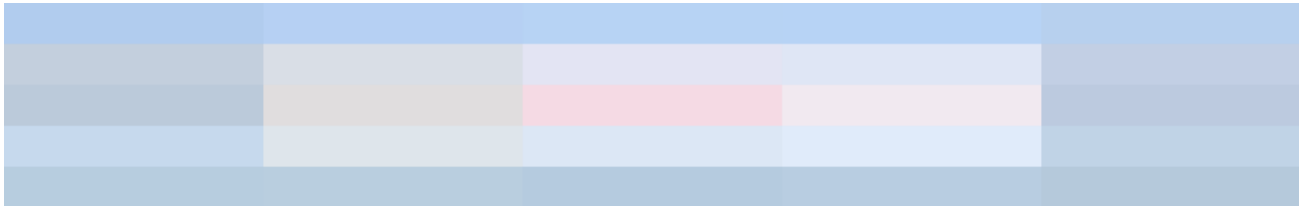
Before leaving the UI behind it is worth noting that Themes are more than just a different colours dialogue; they are integrated into the

Word environment. For example the various Table Styles presented on the *Table Tools* > *Design* Contextual tab reflect the Accent colours in the current Colour Scheme.



The Word 2007 Table Tools > Design Contextual Tab showing Themes in Action

Less immediately obvious is that Styles can be and, by default, are based on Theme elements and that the Theme is reflected in the Ribbon. Go to the *Page Layout* tab on the Ribbon, open the Themes Gallery (from the leftmost icon on the Ribbon), and change your current Theme; for this example, I chose the one called *Verve*. You will immediately see the effect in that the Themes icons themselves change colour. If you then return to the Home tab of the Ribbon you will see that the Styles previewed there now reflect your chosen Theme.



The Word 2007 Ribbon Home Tab showing the *Verve* Theme

VBA: Where The Fun Begins!

ColorFormat Objects

There have been amendments and improvements to *ColorFormat* objects in Word 2007. But it isn't entirely straightforward to work with them. More details coming .....

With the introduction of the new colour model in Word 2007, *ColorFormat* objects have gained an *ObjectThemeColor* property (I assume that this, which really refers to a scheme colour, had to called a theme colour because the name *SchemeColor* was already taken). This does work, and it works with the existing *TintAndShade* property to allow full access from VBA to the new colour settings. (but see above)

Color Properties

Document elements that do not have a *ColorFormat* object have had no changes made in their object model but there have been some changes made to the range of allowable values in their *Color* properties. Enumeration constants are not provided for the new values so, in order to work with these colours, it becomes necessary to understand how they are held. The first thing to know is that, where there is no new interface, Word (in effect, although I cannot be certain of the reality) uses a single storage area for both the old-style RGB values and the newer scheme references; one or other value only, not both at the same time, is stored in the same place. Any setting or checking of the stored value must be able to correctly determine and/or interpret the value and there doesn't appear to be any documentation. What follows is based solely on my own observations.

Although one will need to be somewhat technical to fully understand it, I have tried to explain in some detail so please persevere if you can. One point to note is that I have deliberately used the term 'significant' when describing the positions of bits and bytes to avoid any confusion; if you don't understand the possible source of the confusion, and indeed for most practical purposes, you can consider more significant to mean more to the left.

All colours in Word are stored as 32-bit values, made available to VBA via the object model as *Long* data types, which can be considered as four separate bytes. Exactly what is stored can vary but there are two types of value:

1. Absolute colours, stored as 24-bit RGB values, with the most significant byte (or, more specifically, the most significant bit) set

to zero, equating to positive numbers (or zero) when interpreted as *Long* values. Each of the primary components of the colour uses one byte: red, green, and blue in order from least significant to most significant. There are Word enumeration constants, for example *wdColorBlue* or *wdColorPink*, available for all of the 40 colours in the (pre-2007) basic palette with several extra shades of grey thrown in for good measure.

A representation of an RGB colour (White in this case) as stored by Word

There is a special RGB value that can be returned to VBA, but cannot be set through VBA, and which is never held by Word with any special meaning. This is the colour with the code equating to decimal number 9999999, red 127, green 150, blue 152, a nondescript sort of grey, which indicates that elements being queried do not all have the same colour.

2. Special, referential or relative, values indicating that colours stored, or determined, somewhere else are to be used. These may be in several different formats but they all have the most significant bit set to one, equating to negative numbers when interpreted as *Long* values. The special values are:

    a. In all versions of Word, a value of 0xFF000000 (decimal -16777216) means to use an Automatic (normally contrasting) colour. There is a Word enumeration constant, *wdColorAutomatic*, set to this value to facilitate working with it.

    The "0x" is a techie prefix for a hexadecimal (or just plain hex) number, a number with digits from 0 to 15, represented by the usual 0 to 9 and then the letters A to F for the values 10 to 15. Hexadecimal notation is often used as a short way to represent binary numbers, each hexadecimal digit corresponding to a 4-digit binary number. Relevant here is the hex-digit F which is the same as the binary number 0b1111 (where "0b" is the techie prefix for a binary number).

    Each byte in the code for the colour is represented by two hexadecimal digits, corresponding to 8 binary digits (or *bits*). Here the most significant has a value of 0xFF and each of the others has a value of 0x00.

A representation of Word's storage of the code for Automatic Colour

    b. In ActiveX controls in documents, and in VBA (for UserForm controls, for example) special values for system colours (for some reason lost in the mists of time these are also called *OLE Colors*) ranging from 0x80000000 to 0x80000018. There are VBA enumeration constants, for example *vbTitleBarText* or *vbButtonFace*, available for all of these.

A representation of Word's storage of the code for a system colour (this is the Window Background)

    c. In Word 2007 — at last, the interesting bit — in Word 2007, new values representing scheme colours.

        - The most significant byte of these can be considered in two halves – sometimes, rather horribly, called nybbles. The more significant half is set to 0b1101 (0xD or decimal 208) and the less significant half to a different value (from 0b0100 (0x4) to 0b1111 (0xF)) for each of the scheme colours.

        The second most significant byte is always set to zero

        The remaining two bytes hold two numbers from 255 (0xFF), meaning unchanged, to 0 (0x0), meaning 100%, representing percentages darker (more significant byte) and lighter (less significant byte). The darker and lighter percentages cannot both be specified – the percentage darker is ignored if the percentage lighter is other than 0xFF (lightness unchanged).

A representation of Word 2007's storage of the code for a Theme Colour (Accent 1)

There is a new Enumeration in Word, *wdThemeColorIndex*, with constants for each of the scheme colours, with values as per the less significant half of the first byte. This enumeration also contains constants for the values 0 to 3, called *Dark1, Light1, Dark2*, and *Light2*, which values appear to produce the same colours as *Text1, Background1, Text2*, and *Background2* respectively. There is also a new Enumeration in Office, *msoThemeColorIndex*, again with constants for each of the scheme colours, and again with extra constants for *Dark1, Light1, Dark2*, and *Light2*. The values of the constants in this enumeration, however, are each 1 greater than the less significant half of the first byte and the corresponding Word constant.

The Word and Office constants serve the same purpose but are for use in different circumstances — please don't ask why. Although not their intended purpose, you will see shortly how the Word enumeration constants can also be used to help in working with Color properties. For reference a complete list is presented here.

A complete list of Word and Office *ThemeColorIndex* Enumeration Constants

Colour Scheme Element		**Word Enumeration: wdThemeColorIndex**		**Office Enumeration: msoThemeColorIndex**	
Name	**Number**	**Constant Name**	**Value**	**Constant Name**	**Value**
Dark 1	0	wdThemeColorMainDark1	0	msoThemeColorDark1	1
Light 1	1	wdThemeColorMainLight1	1	msoThemeColorLight1	2
Dark 2	2	wdThemeColorMainDark2	2	msoThemeColorDark2	3
Light 2	3	wdThemeColorMainLight2	3	msoThemeColorLight2	4
Accent 1	4	wdThemeColorAccent1	4	msoThemeColorAccent1	5
Accent 2	5	wdThemeColorAccent2	5	msoThemeColorAccent2	6
Accent 3	6	wdThemeColorAccent3	6	msoThemeColorAccent3	7
Accent 4	7	wdThemeColorAccent4	7	msoThemeColorAccent4	8
Accent 5	8	wdThemeColorAccent5	8	msoThemeColorAccent5	9
Accent 6	9	wdThemeColorAccent6	9	msoThemeColorAccent6	10
Hyperlink	10	wdThemeColorHyperlink	10	msoThemeColorHyperlink	11
Followed Hyperlink	11	wdThemeColorFollowedHyperlink	11	msoThemeColorFollowedHyperlink	12
Background 1	12	wdThemeColorBackground1	12	msoThemeColorBackground1	14
Text 1	13	wdThemeColorText1	13	msoThemeColorText1	13
Background 2	14	wdThemeColorBackground2	14	msoThemeColorBackground2	16
Text 2	15	wdThemeColorText2	15	msoThemeColorText2	15
		wdNotThemeColor	-1	msoNotThemeColor	0
				msoThemeColorMixed	-2

Setting Color Properties

To set a color property of a font, for example, to an encoded value representing a new Theme colour, you must somehow calculate the value. I have written a fairly detailed description of how to build code to do this, more-or-less from scratch but, in order that this page does not become excessively long, I present it on a separate page. If you are comfortable with the code below, then please continue to read here; if not, then I recommend a detour via the link below the code:

```
Sub Colours1()
    Selection.Font.Color = GetThemeColor(ThemeColorIndex:=wdThemeColorAccent2, _
                                         TintAndShade:=0.4)
End Sub
```

```
Function GetThemeColor(ThemeColorIndex As wdThemeColorIndex, _
                       TintAndShade As Double) _
         As Long

    Const HexadecimalPrefix As String = "&H"
    Const UseThemeColor     As String = "D"
    Const UnusedZeroByte    As String = "00"
    Const Unchanged         As String = "FF"

    Dim ThemeColor          As String
    Dim LightnessOrDarkness As String

    ThemeColor = Hex$(ThemeColorIndex)

    If TintAndShade >= 0 Then
        LightnessOrDarkness = Unchanged & Right$("0" & Hex$((1 - TintAndShade) * &HFF), 2)
    Else
        LightnessOrDarkness = Right$("0" & Hex$((1 + TintAndShade) * &HFF), 2) & Unchanged
    End If

    GetThemeColor = CLng(HexadecimalPrefix & _
                         UseThemeColor & _
                         ThemeColor & _
                         UnusedZeroByte & _
                         LightnessOrDarkness)
End Function
```

Code to set the colour of a Font to a Theme colour

⇒ How the code was built [*link to a full explanation of the code on page 2007BuildSet.php on this site*]

Querying Color Properties

You should now be able to set colours to any variation on a theme you care to choose, but that is only part of the picture. Suppose you have set a colour to a variation not shown in the UI, how can you find out what it is? It is possible to work directly with the numeric values of *Color* properties as *Long* data types but it involves some none-too-obvious calculations. It is probably easier, and certainly more understandable, to extract the separate hexadecimal values, those used to set colours in the code above.

As with the code to set a theme colour, the code to do this, to query a colour, the reverse of the code above, is simply presented here and a detailed description of it is presented on a separate page. The whole process of querying is more involved than the process of setting, and the code here just extracts, and reports, the numeric codes relating to the theme, codes which can then be used to obtain more information.

```
Option Explicit

Enum ColourType
    ColourTypeRGB        = &H0
    ColourTypeAutomatic  = &HFF
    ColourTypeSystem     = &H80
    ColourTypeThemeLow   = &HD0
    ColourTypeThemeHigh  = &HDF
End Enum

Type ColourDetails
    ColourType      As ColourType
    ThemeColorIndex As WdThemeColorIndex
    TintAndShade    As Double
End Type
```

```
Sub Colours2()

    With QueryColour(Selection.Cells(1).Shading.BackgroundPatternColor)

        Select Case .ColourType

            Case ColourTypeRGB
                MsgBox "The background of the cell is a standard RGB value"
            Case ColourTypeAutomatic
                MsgBox "The background of the cell is set to Automatic"
            Case ColourTypeSystem
                MsgBox "The background of the cell is set to a Windows system colour"
            Case ColourTypeThemeLow
                MsgBox "The background of the cell is set to Theme colour " & _
                       .ThemeColorIndex & ", with tinting or shading " & .TintAndShade
```

```
            Case Else
                MsgBox "The background of the cell is not in a recognised format"

        End Select

    End With

End Sub
```

```
Function QueryColour(ColourToTest As Long) _
         As ColourDetails

    Dim ColourToTestHex  As String
    Dim ColourTypeByte   As Byte

    ColourToTestHex = Right$(String$(7, "0") & Hex$(ColourToTest), 8)
    ColourTypeByte = CByte("&H" & Left$(ColourToTestHex, 2))

    Select Case ColourTypeByte

        Case ColourTypeRGB
            QueryColour.ColourType = ColourTypeRGB

        Case ColourTypeAutomatic
            QueryColour.ColourType = ColourTypeAutomatic

        Case ColourTypeSystem
            QueryColour.ColourType = ColourTypeSystem

        Case ColourTypeThemeLow To ColourTypeThemeHigh
            QueryColour = QueryThemeColor(ColourTypeByte, ColourToTestHex)

        Case Else
            QueryColour.ColourType = ColourTypeByte

    End Select

End Function
```

```
Function QueryThemeColor(ColourTypeByte As Byte, _
                         ColourToTestHex As String) _
         As ColourDetails

    Const Unchanged      As Byte = &HFF

    Dim LightnessByte    As Byte
    Dim DarknessByte     As Byte

    LightnessByte = CByte("&H" & Mid$(ColourToTestHex, 7, 2))
    DarknessByte = CByte("&H" & Mid$(ColourToTestHex, 5, 2))

    QueryThemeColor.ColourType = ColourTypeByte And &HF0
    QueryThemeColor.ThemeColorIndex = ColourTypeByte And &HF

    If DarknessByte <> Unchanged Then
        QueryThemeColor.TintAndShade = Round(-1 + DarknessByte / &HFF, 2)
    End If

    If LightnessByte <> Unchanged Then
        QueryThemeColor.TintAndShade = Round(1 - LightnessByte / &HFF, 2)
    End If

End Function
```

Code to query the colour of a Table Cell background set to a Theme colour

⇒ How the code was built [*link to a full explanation of the code on page 2007BuildQuery1.php on this site*]

I like numbers, but it's possible that you prefer words; should that be your preference, you might like to see how the numbers returned by the code above can be made more meaningful. As this is not really an essential part of the process being presented here, it is all presented on a separate page, which, should you wish, you can see from the link below. Naturally I encourage you to take a look, but the choice is yours.

⇒ How to translate theme colour indexes to words [*link to 2007BuildQuery2.php on this site*]

Determining the RGB value of a Scheme Colour

Having a code number, or even a name, for a Theme colour is all very well but it would be much better to have the real, RGB, colour. Themes work by keeping the actual colour details within the Theme and having, as you have seen, references to them in the colour properties of various objects. It's now time to look at the Themes themselves.

Every Document edited in Word 2007 has exactly one Theme. If you are working on older documents in compatibility mode and saving them in Word 97-2003 format, that Theme won't be saved with the document but it will exist while the document is being edited. The Theme details in the Document are provided in the *DocumentTheme* Object (note that there is also an *ActiveTheme* property but it has nothing to do do with Office 2007 Themes) and this is where the fun continues.

The *DocumentTheme* Object has a *ThemeColorScheme* Property that takes a single argument to specify the particular theme colour you want details of. This argument is of type *msoThemeColorSchemeIndex*, an enumeration that differs from both the Word, and the Office, *ThemeColorIndex* enumerations you have so far seen. Although there are similarities between the enumerations, there is no logical correlation between them and the only way to translate one into the other is with some hard code, an example of which can be seen below the table.

A comparison of the Word *ThemeColorIndex* and the Office *ThemeColorSchemeIndex* Enumerations

Word Enumeration: wdThemeColorIndex		**Office Enumeration: msoThemeColorSchemeIndex**	
Constant Name	**Value**	**Constant Name**	**Value**
wdThemeColorMainDark1*	0	msoThemeDark1*	1
wdThemeColorMainLight1*	1	msoThemeLight1*	2
wdThemeColorMainDark2*	2	msoThemeDark2*	3
wdThemeColorMainLight2*	3	msoThemeLight2*	4
wdThemeColorAccent1	4	msoThemeAccent1	5
wdThemeColorAccent2	5	msoThemeAccent2	6
wdThemeColorAccent3	6	msoThemeAccent3	7
wdThemeColorAccent4	7	msoThemeAccent4	8
wdThemeColorAccent5	8	msoThemeAccent5	9
wdThemeColorAccent6	9	msoThemeAccent6	10
wdThemeColorHyperlink	10	msoThemeHyperlink	11
wdThemeColorFollowedHyperlink	11	msoThemeFollowedHyperlink	12
wdThemeColorBackground1	12	msoThemeLight1	2
wdThemeColorText1	13	msoThemeDark1	1
wdThemeColorBackground2	14	msoThemeLight2	4
wdThemeColorText2	15	msoThemeDark2	3

You can see from the above table that to find details of the *Background1* colour, for example, you would look at the *Light1* colour scheme in the Theme. In Word, however, although not, as far as I can tell, in any of the other Office applications, this is not guaranteed to be correct. The actual mapping from the codes used in documents to the theme scheme colours is buried deep in the VBA behind the document and there is no access to it from either the User interface or the Word Object Model. Interested readers may care to look at the *clrSchemeMapping* element in the *Settings* Part of the *WordProcessingML* Schema, but, as there is nothing you can do about it in code and as it is unlikely ever to be set to anything other than the default in practice, the possibility will not be considered further here. It is just worth noting that the mappings that can be set do not include the first four, marked with asterisks above: these mappings are hard-wired in Word, and are guaranteed.

Armed with the translation you can now write a simple procedure to return the code you need to gain access to the theme details:

```
Function ThemeColorSchemeIndex(ThemeColorIndex As WdThemeColorIndex) _
         As MsoThemeColorSchemeIndex

    Select Case ThemeColorIndex

        Case wdThemeColorMainDark1:         ThemeColorSchemeIndex = msoThemeDark1
        Case wdThemeColorMainLight1:        ThemeColorSchemeIndex = msoThemeLight1
        Case wdThemeColorMainDark2:         ThemeColorSchemeIndex = msoThemeDark2
        Case wdThemeColorMainLight2:        ThemeColorSchemeIndex = msoThemeLight2
        Case wdThemeColorAccent1:           ThemeColorSchemeIndex = msoThemeAccent1
        Case wdThemeColorAccent2:           ThemeColorSchemeIndex = msoThemeAccent2
        Case wdThemeColorAccent3:           ThemeColorSchemeIndex = msoThemeAccent3
        Case wdThemeColorAccent4:           ThemeColorSchemeIndex = msoThemeAccent4
        Case wdThemeColorAccent5:           ThemeColorSchemeIndex = msoThemeAccent5
        Case wdThemeColorAccent6:           ThemeColorSchemeIndex = msoThemeAccent6
        Case wdThemeColorHyperlink:         ThemeColorSchemeIndex = msoThemeHyperlink
        Case wdThemeColorHyperlinkFollowed: ThemeColorSchemeIndex = msoThemeFollowedHyperlink
        Case wdThemeColorBackground1:       ThemeColorSchemeIndex = msoThemeLight1
        Case wdThemeColorText1:             ThemeColorSchemeIndex = msoThemeDark1
        Case wdThemeColorBackground2:       ThemeColorSchemeIndex = msoThemeLight2
        Case wdThemeColorText2:             ThemeColorSchemeIndex = msoThemeDark2
        Case Else:                          ' This shouldn't really ever happen

    End Select

End Function
```

Once you have this color scheme index, the (base) RGB value is directly accessible and all that remains is to apply the specified tinting or shading. As you probably expect by now, this isn't an entirely straightforward operation, and it requires you to take a walk in colour space and use the HSL (Hue, Saturation, Luminance) colour model. Once again, full details of this are on a separate page:

⇒ Code to convert to and from HSL and apply tinting or shading [*link to ColourSpace.php on this site*]

You have now seen all the elements necessary to determine the red, green, and blue components of a colour applied via a theme. The complete code, built up in pieces over the course of this rather long and rambling monologue, and with a couple of final amendments, is presented below. I am working on providing a downloadable sample document, which should be available soon; please check back in a day or two.

```
Option Explicit

Private Enum ColourType
    ColourTypeRGB = &H0
    ColourTypeAutomatic = &HFF
    ColourTypeSystem = &H80
    ColourTypeThemeLow = &HD0
    ColourTypeThemeHigh = &HDF
End Enum

Private Type ColourDetails
    ColourType      As ColourType
    ThemeColorIndex As WdThemeColorIndex
    TintAndShade    As Double
    RGB             As Long
End Type

Private Type HSL
    H As Double ' Range 0 - 1
    S As Double ' Range 0 - 1
    L As Double ' Range 0 - 1
End Type
```

```
Sub Colours2()

    Dim ResultRGB As Long
    Dim ResultHex As String

    With QueryColour(Selection.Cells(1).Shading.BackgroundPatternColor)

        Select Case .ColourType

            Case ColourTypeRGB
                MsgBox "The background of the cell is a standard RGB value"
            Case ColourTypeAutomatic
                MsgBox "The background of the cell is set to Automatic"
```

```
            Case ColourTypeSystem
                MsgBox "The background of the cell is set to a Windows system colour"
            Case ColourTypeThemeLow
                MsgBox "The background of the cell is set to: " & _
                       ThemeColorName(.ThemeColorIndex) & _
                       TintAndShadeText(.TintAndShade) & vbNewLine & _
                       "The actual colour is: RGB(" & (.RGB And &HFF&) & _
                                    ", " & (.RGB And &HFF00&) / &H100& & _
                                    ", " & (.RGB And &HFF0000) / &H10000 & ")"
            Case Else
                MsgBox "The background of the cell is not in a recognised format"

        End Select

    End With

End Sub
```

```
Private Function QueryColour(ColourToTest As Long) _
                 As ColourDetails

    Dim ColourToTestHex  As String
    Dim ColourTypeByte   As Byte

    ColourToTestHex = Right$(String$(7, "0") & Hex$(ColourToTest), 8)
    ColourTypeByte = CByte("&H" & Left$(ColourToTestHex, 2))

    Select Case ColourTypeByte

        Case ColourTypeRGB
            QueryColour.ColourType = ColourTypeRGB

        Case ColourTypeAutomatic
            QueryColour.ColourType = ColourTypeAutomatic

        Case ColourTypeSystem
            QueryColour.ColourType = ColourTypeSystem

        Case ColourTypeThemeLow To ColourTypeThemeHigh
            QueryColour = QueryThemeColor(ColourTypeByte, ColourToTestHex)

        Case Else
            QueryColour.ColourType = ColourTypeByte

    End Select

End Function
```

```
Private Function QueryThemeColor(ColourTypeByte As Byte, _
                                 ColourToTestHex As String) _
                 As ColourDetails

    Const Unchanged      As Byte = &HFF

    Dim LightnessByte    As Byte
    Dim DarknessByte     As Byte

    LightnessByte = CByte("&H" & Mid$(ColourToTestHex, 7, 2))
    DarknessByte = CByte("&H" & Mid$(ColourToTestHex, 5, 2))

    QueryThemeColor.ColourType = ColourTypeByte And &HF0
    QueryThemeColor.ThemeColorIndex = ColourTypeByte And &HF

    If DarknessByte <> Unchanged Then
        QueryThemeColor.TintAndShade = Round(-1 + DarknessByte / &HFF, 2)
    End If

    If LightnessByte <> Unchanged Then
        QueryThemeColor.TintAndShade = Round(1 - LightnessByte / &HFF, 2)
    End If

    QueryThemeColor.RGB = GetRGB(QueryThemeColor.ThemeColorIndex, _
                                 QueryThemeColor.TintAndShade)

End Function
```

```
Private Function GetRGB(ThemeColorIndex As WdThemeColorIndex, _
```

```
                    TintAndShade As Double) _
                As String

    Dim ColorSchemeIndex    As MsoThemeColorSchemeIndex
    Dim ColorSchemeRGB      As Long
    Dim ColorSchemeHSL      As HSL
    Dim TintedAndShadedRGB  As Long

    ColorSchemeIndex = ThemeColorSchemeIndex(ThemeColorIndex)
    ColorSchemeRGB = ActiveDocument.DocumentTheme. _
                     ThemeColorScheme(ColorSchemeIndex).RGB

    ColorSchemeHSL = RGBtoHSL(ColorSchemeRGB)
    ColorSchemeHSL.L = (ColorSchemeHSL.L * (Abs(TintAndShade))) + _
                       (Abs(TintAndShade > 0) * (1 - TintAndShade))

    TintedAndShadedRGB = HSLtoRGB(ColorSchemeHSL)

    GetRGB = TintedAndShadedRGB

End Function
```

```
Private Function ThemeColorSchemeIndex(ThemeColorIndex As WdThemeColorIndex) _
                As MsoThemeColorSchemeIndex

    Select Case ThemeColorIndex
        Case wdThemeColorMainDark1:         ThemeColorSchemeIndex = msoThemeDark1
        Case wdThemeColorMainLight1:        ThemeColorSchemeIndex = msoThemeLight1
        Case wdThemeColorMainDark2:         ThemeColorSchemeIndex = msoThemeDark2
        Case wdThemeColorMainLight2:        ThemeColorSchemeIndex = msoThemeLight2
        Case wdThemeColorAccent1:           ThemeColorSchemeIndex = msoThemeAccent1
        Case wdThemeColorAccent2:           ThemeColorSchemeIndex = msoThemeAccent2
        Case wdThemeColorAccent3:           ThemeColorSchemeIndex = msoThemeAccent3
        Case wdThemeColorAccent4:           ThemeColorSchemeIndex = msoThemeAccent4
        Case wdThemeColorAccent5:           ThemeColorSchemeIndex = msoThemeAccent5
        Case wdThemeColorAccent6:           ThemeColorSchemeIndex = msoThemeAccent6
        Case wdThemeColorHyperlink:         ThemeColorSchemeIndex = msoThemeHyperlink
        Case wdThemeColorHyperlinkFollowed: ThemeColorSchemeIndex = msoThemeFollowedHyperlink
        Case wdThemeColorBackground1:       ThemeColorSchemeIndex = msoThemeLight1
        Case wdThemeColorText1:             ThemeColorSchemeIndex = msoThemeDark1
        Case wdThemeColorBackground2:       ThemeColorSchemeIndex = msoThemeLight2
        Case wdThemeColorText2:             ThemeColorSchemeIndex = msoThemeDark2
        Case Else:                          ' This shouldn't really ever happen

    End Select

End Function
```

```
Private Function RGBtoHSL(RGB As Long) As HSL

    Dim R As Double ' Range 0 to 1
    Dim G As Double ' Range 0 to 1
    Dim B As Double ' Range 0 to 1

    Dim RGB_Max  As Double
    Dim RGB_Min  As Double
    Dim RGB_Diff As Double

    Dim HexString As String

    HexString = Right$(String$(7, "0") & Hex$(RGB), 8)
    R = CDbl("&H" & Mid$(HexString, 7, 2)) / 255
    G = CDbl("&H" & Mid$(HexString, 5, 2)) / 255
    B = CDbl("&H" & Mid$(HexString, 3, 2)) / 255

    RGB_Max = R
    If G > RGB_Max Then RGB_Max = G
    If B > RGB_Max Then RGB_Max = B

    RGB_Min = R
    If G < RGB_Min Then RGB_Min = G
    If B < RGB_Min Then RGB_Min = B

    RGB_Diff = RGB_Max - RGB_Min

    With RGBtoHSL

        .L = (RGB_Max + RGB_Min) / 2
```

```
        If RGB_Diff = 0 Then

            .S = 0
            .H = 0

        Else

            Select Case RGB_Max
                Case R: .H = (1 / 6) * (G - B) / RGB_Diff - (B > G)
                Case G: .H = (1 / 6) * (B - R) / RGB_Diff + (1 / 3)
                Case B: .H = (1 / 6) * (R - G) / RGB_Diff + (2 / 3)
            End Select

            Select Case .L
                Case Is < 0.5: .S = RGB_Diff / (2 * .L)
                Case Else:     .S = RGB_Diff / (2 - (2 * .L))
            End Select

        End If

    End With

End Function
```

```
Private Function HSLtoRGB(HSL As HSL) As Long

    Dim R As Double
    Dim G As Double
    Dim B As Double

    Dim X As Double
    Dim Y As Double

    With HSL

        If .S = 0 Then

            R = .L
            G = .L
            B = .L

        Else

            Select Case .L
                Case Is < 0.5: X = .L * (1 + .S)
                Case Else:     X = .L + .S - (.L * .S)
            End Select

            Y = 2 * .L - X

            R = H2C(X, Y, IIf(.H > 2 / 3, .H - 2 / 3, .H + 1 / 3))
            G = H2C(X, Y, .H)
            B = H2C(X, Y, IIf(.H < 1 / 3, .H + 2 / 3, .H - 1 / 3))

        End If

    End With

    HSLtoRGB = CLng("&H00" & _
                    Right$("0" & Hex$(Round(B * 255)), 2) & _
                    Right$("0" & Hex$(Round(G * 255)), 2) & _
                    Right$("0" & Hex$(Round(R * 255)), 2))

End Function
```

```
Private Function H2C(X As Double, Y As Double, HC As Double) As Double

    Select Case HC
        Case Is < 1 / 6: H2C = Y + ((X - Y) * 6 * HC)
        Case Is < 1 / 2: H2C = X
        Case Is < 2 / 3: H2C = Y + ((X - Y) * ((2 / 3) - HC) * 6)
        Case Else:       H2C = Y
    End Select

End Function
```

```
Function ThemeColorName(ThemeColorIndex As WdThemeColorIndex, _
                        Optional LanguageId As MsoLanguageID) _
         As String

    If LanguageId = 0 Then
        LanguageId = LanguageSettings.LanguageId(msoLanguageIDUI)
    End If

    Select Case LanguageId

        Case msoLanguageIDDutch

            Select Case ThemeColorIndex
                Case wdThemeColorMainDark1:         ThemeColorName = "Donker 1"
                Case wdThemeColorMainLight1:        ThemeColorName = "Licht 1"
                Case wdThemeColorMainDark2:         ThemeColorName = "Donker 2"
                Case wdThemeColorMainLight2:        ThemeColorName = "Licht 2"
                Case wdThemeColorAccent1:           ThemeColorName = "Accent 1"
                Case wdThemeColorAccent2:           ThemeColorName = "Accent 2"
                Case wdThemeColorAccent3:           ThemeColorName = "Accent 3"
                Case wdThemeColorAccent4:           ThemeColorName = "Accent 4"
                Case wdThemeColorAccent5:           ThemeColorName = "Accent 5"
                Case wdThemeColorAccent6:           ThemeColorName = "Accent 6"
                Case wdThemeColorHyperlink:         ThemeColorName = "Hyperlink"
                Case wdThemeColorHyperlinkFollowed: ThemeColorName = "Gevolgde Hyperlink"
                Case wdThemeColorBackground1:       ThemeColorName = "Achtergrond 1"
                Case wdThemeColorText1:             ThemeColorName = "Tekst 1"
                Case wdThemeColorBackground2:       ThemeColorName = "Achtergrond 2"
                Case wdThemeColorText2:             ThemeColorName = "Tekst 2"
                Case Else:                          ThemeColorName = "Onbekent " & ThemeColorIndex
            End Select

        Case msoLanguageIDFrench

            Select Case ThemeColorIndex
                Case wdThemeColorMainDark1:         ThemeColorName = "Sombre 1"
                Case wdThemeColorMainLight1:        ThemeColorName = "Clair 1"
                Case wdThemeColorMainDark2:         ThemeColorName = "Sombre 2"
                Case wdThemeColorMainLight2:        ThemeColorName = "Clair 2"
                Case wdThemeColorAccent1:           ThemeColorName = "Accentuation 1"
                Case wdThemeColorAccent2:           ThemeColorName = "Accentuation 2"
                Case wdThemeColorAccent3:           ThemeColorName = "Accentuation 3"
                Case wdThemeColorAccent4:           ThemeColorName = "Accentuation 4"
                Case wdThemeColorAccent5:           ThemeColorName = "Accentuation 5"
                Case wdThemeColorAccent6:           ThemeColorName = "Accentuation 6"
                Case wdThemeColorHyperlink:         ThemeColorName = "Lien hypertexte"
                Case wdThemeColorHyperlinkFollowed: ThemeColorName = "Lien hypertexte visit�"
                Case wdThemeColorBackground1:       ThemeColorName = "Arri�re-plan 1"
                Case wdThemeColorText1:             ThemeColorName = "Texte 1"
                Case wdThemeColorBackground2:       ThemeColorName = "Arri�re-plan 2"
                Case wdThemeColorText2:             ThemeColorName = "Texte 2"
                Case Else:                          ThemeColorName = "Inconnu " & ThemeColorIndex
            End Select

        Case Else ' msoLanguageIDEnglishUS

            Select Case ThemeColorIndex
                Case wdThemeColorMainDark1:         ThemeColorName = "Dark 1"
                Case wdThemeColorMainLight1:        ThemeColorName = "Light 1"
                Case wdThemeColorMainDark2:         ThemeColorName = "Dark 2"
                Case wdThemeColorMainLight2:        ThemeColorName = "Light 2"
                Case wdThemeColorAccent1:           ThemeColorName = "Accent 1"
                Case wdThemeColorAccent2:           ThemeColorName = "Accent 2"
                Case wdThemeColorAccent3:           ThemeColorName = "Accent 3"
                Case wdThemeColorAccent4:           ThemeColorName = "Accent 4"
                Case wdThemeColorAccent5:           ThemeColorName = "Accent 5"
                Case wdThemeColorAccent6:           ThemeColorName = "Accent 6"
                Case wdThemeColorHyperlink:         ThemeColorName = "Hyperlink"
                Case wdThemeColorHyperlinkFollowed: ThemeColorName = "Followed Hyperlink"
                Case wdThemeColorBackground1:       ThemeColorName = "Background 1"
                Case wdThemeColorText1:             ThemeColorName = "Text 1"
                Case wdThemeColorBackground2:       ThemeColorName = "Background 2"
                Case wdThemeColorText2:             ThemeColorName = "Text 2"
                Case Else:                          ThemeColorName = "Unknown " & ThemeColorIndex
            End Select

    End Select

End Function
```

```
Function TintAndShadeText(TintAndShade As Double, _
                          Optional LanguageId As MsoLanguageID) _
         As String

    If LanguageId = 0 Then
        LanguageId = LanguageSettings.LanguageId(msoLanguageIDUI)
    End If

    Select Case TintAndShade

        Case 0

            TintAndShadeText = ""

        Case Is > 0

            Select Case LanguageId
                Case msoLanguageIDDutch
                    TintAndShadeText = ", lichter "
                Case msoLanguageIDFrench
                    TintAndShadeText = ", plus clair "
                Case Else
                    TintAndShadeText = ", lighter "
            End Select

            TintAndShadeText = TintAndShadeText & TintAndShade * 100 & "%"

        Case Is < 0

            Select Case LanguageId
                Case msoLanguageIDDutch
                    TintAndShadeText = ", donkerder "
                Case msoLanguageIDFrench
                    TintAndShadeText = ", plus sombre "
                Case Else
                    TintAndShadeText = ", darker "
            End Select

            TintAndShadeText = TintAndShadeText & TintAndShade * -100 & "%"

    End Select

End Function
```

At last, The End!

Tony Jollans. This page last updated: 4 March 2013.