

Les couleurs en VBA Excel: Les équivalences Hex-Long-RGB


par [SilkyRoad \(silkyroad.developpez.com\)](http://silkyroad.developpez.com)

Date de publication : 16/09/2006

Dernière mise à jour : 16/09/2006

Cette page décrit:

La palette de couleurs standard dans la feuille de calcul.

Un exemple de procédure pour retrouver les équivalences de couleur Hex-Long- **RGB** , en VBA Excel.

Un classeur démo est téléchargeable en fin de page.

- I - Introduction
- II - Description
- III - Téléchargement

I - Introduction

Excel dispose d'une palette standard de 56 couleurs pour le remplissage des cellules et la colorisation des polices.



Vous pouvez personnaliser cette palette dans chaque classeur si besoin:

Utilisez le menu Outils

Options

Sélectionnez l'onglet "Couleur".

Sélectionnez une des couleurs standard dans la boîte de dialogue.

Ensuite, cliquez sur le bouton "Modifier" afin de choisir une couleur de remplacement.

Cliquez sur le bouton OK pour valider.

La palette de base est modifiée.

Cette opération peut être automatisée par macro.

Vba


```
'9 est l'index de la couleur dans la palette des 56 couleurs.  
ActiveWorkbook.Colors(9) = RGB(164, 240, 240)
```

Nota:

Cliquez sur le bouton "Par défaut" (Menu Outils/Options/Onglet Couleurs) afin de réinitialiser la palette.

Vba

```
'Réinitialise la palette standard  
ActiveWorkbook.ResetColors
```

Vous constatez que la macro servant à modifier la palette utilise la fonction  **RGB**. Cette fonction est aussi utilisée pour la personnalisation des **UserForm** , des **contrôles** , des formes automatiques et autres objets Excel...

Vba

```
'Exemple pour définir la couleur des objets lors de l'initialisation d'un UserForm.  
Private Sub UserForm_Initialize()  
'Les valeurs Long et Hex renvoient la même couleur dans cet exemple.  
    With Me  
        .BackColor = RGB(224, 120, 243)  
        .CommandButton1.BackColor = 15956192 'Valeur Long  
        .Label1.BackColor = &HF378E0 'Valeur Hex  
    End With  
End Sub
```

L'exemple précédent montre aussi qu'il est possible de spécifier des valeurs Long, Hex ou RGB pour obtenir le même résultat.

Lorsque vous manipulez les couleurs par macro, la question peut se poser: Quelle est la valeur à attribuer pour obtenir une couleur précise souhaitée.

C'est l'objet du classeur démo.

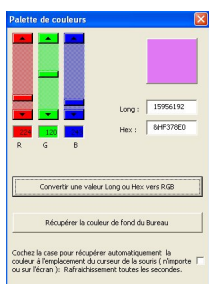
Le chapitre suivant décrit les procédures utilisées.

II - Description

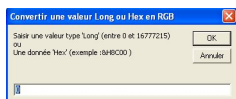
Le classeur en téléchargement permet de visualiser rapidement des couleurs en modifiant les valeurs RGB (rouge, vert, bleu), à partir de 3 ScrollBar.

Les équivalences Hex et Long s'affichent automatiquement.

Vous pouvez ainsi savoir quelle valeur insérer dans vos projets, et au format de votre préférence (RGB, Hex ou Long).



Le classeur vous offre aussi la possibilité de saisir une valeur Hex ou Long afin de retrouver l'équivalence RGB.



Les autres options du classeur:

1. Récupérer la couleur de fond du bureau.
2. Récupérer automatiquement la couleur à l'emplacement du curseur de la souris (n'importe où sur l'écran) lorsque la checkBox est cochée. Un Rafraîchissement est appliqué toutes les secondes.

Configurations testées:

Win98 & Excel97

Win2000 & Excel97

WinXP & Excel2002

La procédure du module standard.

Vba

Option Explicit

```
'GetCursorPos: renvoie la position de la souris sur l'écran.  
Public Declare Function GetCursorPos Lib "user32" (lpPoint As POINTAPI) As Long  
'GetDC: Renvoie le Handle d'un Contexte d'Affichage hDC (Handle of Device Context)  
Public Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long  
'GetPixel: renvoie la couleur du pixel en fonction des coordonnées spécifiées (X et Y)  
Public Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, _  
ByVal Y As Long) As Long
```

```
'Coordonnées d'un point de l'écran.
```

```
Type POINTAPI  
    X As Long  
    Y As Long
```

```
End Type
```

```
Public Cible As Boolean
```

```
Public Function GetDcColor() As Double
```

```
Dim DeskHdc As Long
```

```
Dim Pxy As POINTAPI
```

```
    'GetDC(0): Pour récupérer le hDC de l'écran  
    DeskHdc = GetDC(0)  
    'Récupère la position du curseur de la souris  
    GetCursorPos Pxy  
    'La fonction renvoie la couleur à l'emplacement spécifié  
    GetDcColor = GetPixel(DeskHdc, Pxy.X, Pxy.Y)
```

```
End Function
```

```
'Affichage UserForm
```

```
Sub Lancer()
```

```
    UserForm1.Show
```

```
End Sub
```

```
Sub Demarrer()
```

```
    'Timer qui va déclencher la récupération de la couleur à l'emplacement  
    'du curseur de la souris (toutes les secondes).  
    Application.OnTime Now + TimeValue("0:0:01"), "MiseAJour"
```

```
End Sub
```

```
'Procédure déclenchée par le Timer, qui va permettre la mise à jour du Userform  
'en fonction de la position de la souris.
```

```
Sub MiseAJour()
```

```
    Dim Rouge As Integer, Vert As Integer, Bleu As Integer
```

```
    Dim Couleur As Long
```

```
    'affiche la couleur correspondant à l'emplacement du curseur de la souris
```

```
    UserForm1.CommandButton1.BackColor = "&H" & Hex(GetDcColor)
```

```
    Couleur = UserForm1.CommandButton1.BackColor
```

Vba

```
'--- Convertit la couleur au format RGB -----  
Rouge = Int(Couleur Mod 256)  
Vert = Int((Couleur Mod 65536) / 256)  
Bleu = Int(Couleur / 65536)  
'-----  
  
'--- Affiche les codes RGB dans les TextBox -----  
UserForm1.TextBox3 = Rouge  
UserForm1.TextBox4 = Vert  
UserForm1.TextBox5 = Bleu  
'-----  
  
If Cible = True Then Exit Sub  
  
Call Demarrer  
End Sub  
  
Sub Arreter()  
Cible = True  
End Sub
```

La procédure du UserForm.

Vba

```
Option Explicit  
'GetSysColor: permet de retrouver la valeur des couleurs système.  
Private Declare Function GetSysColor Lib "user32" (ByVal nIndex As Long) As Long  
  
Dim Tableau(1 To 3) As Long  
Const COLOR_BACKGROUND = 1  
  
'CheckBox pour spécifier la récupération de la couleur  
'à l'emplacement de la souris.  
Private Sub CheckBox1_Click()  
If CheckBox1 = True Then  
Cible = False  
Demarrer  
Else  
Cible = True  
Arreter  
Unload Me  
Lancer  
End If  
End Sub  
  
Private Sub UserForm_Initialize()  
'Initialise les contrôles avant d'afficher la boîte de dialogue  
TextBox1 = 0  
TextBox2 = "&H0"  
TextBox3 = 0  
TextBox4 = 0  
TextBox5 = 0
```

Vba

```
Cible = False

End Sub

'Evenement fermeture du UserForm
Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)
    On Error Resume Next
    Cible = True

    'Ferme (si nécessaire) le Timer de récupération des couleurs à l'emplacement de la souris,
    'sinon la procédure continue à fonctionner, même après la fermeture du userForm
    Application.OnTime EarliestTime:=Now + _
        TimeValue("00:00:01"), Procedure:="MiseAJour", Schedule:=False
End Sub

Private Sub CommandButton2_Click()
    Dim Rouge As Integer, Vert As Integer, Bleu As Integer
    Dim Couleur As Long

    On Error Resume Next

    '----- Transforme les valeurs Long & Hex en code RGB -----
    Couleur = InputBox("Saisissez une valeur type 'Long' (entre 0 et 16777215) " & _
        vbCrLf & "ou" & vbCrLf & "Une donnée 'Hex' (exemple :&H8C00 )", _
        "Convertir une valeur Long ou Hex en RGB", 0)

    Rouge = Int(Couleur Mod 256)
    Vert = Int((Couleur Mod 65536) / 256)
    Bleu = Int(Couleur / 65536)
    '-----

    Application.EnableEvents = False
    TextBox3 = Rouge
    TextBox4 = Vert
    TextBox5 = Bleu
    Application.EnableEvents = True

End Sub

'----- Met à jour les autres contrôles lors du déplacement des ScrollBar
Private Sub ScrollBar1_Change()

    TextBox3 = ScrollBar1.Value
    TextBox1 = Val(TextBox1) - Tableau(1) + Val(ScrollBar1.Value)
    Tableau(1) = ScrollBar1.Value
    TextBox2 = "&H" & Hex(TextBox1)
    CommandButton1.BackColor = TextBox1

    'pour utiliser directement le format RGB sans conversion:
    'CommandButton1.BackColor = RGB(ScrollBar1, ScrollBar2, ScrollBar3)
    'TextBox3 = ScrollBar1

End Sub

Private Sub ScrollBar2_Change()
    TextBox4 = ScrollBar2.Value
    TextBox1 = Val(TextBox1) - (Tableau(2) * 256) + (Val(ScrollBar2.Value) * 256)
    Tableau(2) = ScrollBar2.Value
    TextBox2 = "&H" & Hex(TextBox1)
    CommandButton1.BackColor = TextBox1

    'pour utiliser directement le format RGB sans conversion
    'CommandButton1.BackColor = RGB(ScrollBar1, ScrollBar2, ScrollBar3)
```

Vba

```
'TextBox4 = ScrollBar2
End Sub

Private Sub ScrollBar3_Change()
    TextBox5 = ScrollBar3.Value
    TextBox1 = Val(TextBox1) - (Tableau(3) * 65536) + (Val(ScrollBar3) * 65536)
    Tableau(3) = ScrollBar3.Value
    TextBox2 = "&H" & Hex(TextBox1)
    CommandButton1.BackColor = TextBox1

    'pour utiliser directement le format RGB sans conversion:
    'CommandButton1.BackColor = RGB(ScrollBar1, ScrollBar2, ScrollBar3)
    'TextBox5 = ScrollBar3
End Sub

Private Sub TextBox3_Change()
    On Error Resume Next
    ScrollBar1.Value = TextBox3
End Sub

Private Sub TextBox4_Change()
    On Error Resume Next
    ScrollBar2.Value = TextBox4
End Sub

Private Sub TextBox5_Change()
    On Error Resume Next
    ScrollBar3.Value = TextBox5
End Sub

'-----

Private Sub CommandButton3_Click()
    Dim Rouge As Integer, Vert As Integer, Bleu As Integer
    Dim CouleurBureau As Long

    'Récupère la couleur du bureau
    CouleurBureau = GetSysColor(COLOR_BACKGROUND)

    'Convertit la couleur au format RGB
    Rouge = Int(CouleurBureau Mod 256)
    Vert = Int((CouleurBureau Mod 65536) / 256)
    Bleu = Int(CouleurBureau / 65536)

    'Mise à jour des données pour afficher la couleur dans l'UserForm
    TextBox3 = Rouge
    TextBox4 = Vert
    TextBox5 = Bleu

    'Rafraichissement du UserForm
    Me.Repaint
End Sub
```

III - Téléchargement

Téléchargez le classeur démo.

Téléchargez le classeur démo (Miroir).

