**ÉVÈNEMENTS POUR LE COMBOBOX**

Ne pas oublier de changer le nom *ComboBox1*

Exemple : changer ***ComboBox1*** par ***LIST\_PRODUITS***

Private Sub **ComboBox1\_Change**()

 intTopIndex = Me.ComboBox1.TopIndex

End Sub

'------------------ComboBox MouseWheel----------------------------

Private Sub **ComboBox1\_MouseDown**(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)

 ' Définir les noms des objet à l'ouverture de l'USF

 ' sont utilisés dans le code du hook

 Set ObjUSF = Me: Set ObjList = Me.**ComboBox1**

 'Store the first TopIndex Value

 intTopIndex = Me.**ComboBox1**.TopIndex

 '

 Hook\_Mouse

End Sub

' Check to see if focus is lost

Private Sub **ComboBox1\_Exit**(ByVal Cancel As MSForms.ReturnBoolean)

 UnHook\_Mouse

End Sub

**A METTRE DANS UN MODULE**

'-----Allows use of MouseWheel on designated ListBox/ComboBox on a form or, sheet if modified.--------

Option Explicit

Declare Function FindWindow Lib "user32" Alias "FindWindowA" \_

 (ByVal lpClassName As String, ByVal lpWindowName As String) As Long

Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long) As Long

Declare Function GetForegroundWindow Lib "user32" () As Long

Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" \_

 (ByVal Destination As Long, ByVal Source As Long, ByVal Length As Long)

Declare Function SetWindowsHookEx Lib \_

 "user32" Alias "SetWindowsHookExA" (ByVal idHook As Long, ByVal lpfn As Long, \_

 ByVal hmod As Long, ByVal dwThreadId As Long) As Long

Declare Function CallNextHookEx Lib "user32" (ByVal hHook As Long, \_

 ByVal nCode As Long, ByVal wParam As Long, lParam As Any) As Long

Declare Function UnhookWindowsHookEx Lib "user32" (ByVal hHook As Long) As Long

Declare Function GetLastError Lib "kernel32" () As Long ' Used this one to crack the problem.

Type POINTAPI

 X As Long

 Y As Long

End Type

Type MSLLHOOKSTRUCT 'Will Hold the lParam struct Data

 pt As POINTAPI

 mouseData As Long ' Holds Forward\Bacward flag

 flags As Long

 time As Long

 dwExtraInfo As Long

End Type

Const HC\_ACTION = 0

Const WH\_MOUSE\_LL = 14

Const WM\_MOUSEWHEEL = &H20A

Dim hhkLowLevelMouse, lngInitialColor As Long

Dim udtlParamStuct As MSLLHOOKSTRUCT

Public Const GWL\_HINSTANCE = (-6)

Public intTopIndex As Integer

Public ObjUSF As UserForm, ObjList As Object

Function GetHookStruct(ByVal lParam As Long) As MSLLHOOKSTRUCT

' VarPtr returns address; LenB returns size in bytes.

 CopyMemory VarPtr(udtlParamStuct), lParam, LenB(udtlParamStuct)

 GetHookStruct = udtlParamStuct

End Function

Function LowLevelMouseProc \_

 (ByVal nCode As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

'Avoid XL crashing if RunTime error occurs due to Mouse fast movement

 On Error Resume Next

 ' \\ Unhook & get out in case the application is deactivated

 If GetForegroundWindow <> FindWindow("ThunderDFrame", ObjUSF.Caption) Then

 UnHook\_Mouse

 Exit Function

 End If

 If (nCode = HC\_ACTION) Then

 If wParam = WM\_MOUSEWHEEL Then

 '\\ Don't process Default WM\_MOUSEWHEEL Window message

 LowLevelMouseProc = True

 '\\ Change Sheet&\DropDown names as required

 With ObjList

 '\\ if rolling forward increase Top index by 1 to cause an Up Scroll

 If GetHookStruct(lParam).mouseData > 0 Then

 .TopIndex = intTopIndex - 1

 '\\ Store new TopIndex value

 intTopIndex = .TopIndex

 Else '\\ if rolling backward decrease Top index by 1 to cause \_

 '\\a Down Scroll

 .TopIndex = intTopIndex + 1

 '\\ Store new TopIndex value

 intTopIndex = .TopIndex

 End If

 End With

 End If

 Exit Function

 End If

 LowLevelMouseProc = CallNextHookEx(0, nCode, wParam, ByVal lParam)

End Function

Sub Hook\_Mouse()

' Statement to maintain the handle of the hook if clicking outside of the control.

' There isn't a Hinstance for Application, so used GetWindowLong to get handle.

 If hhkLowLevelMouse < 1 Then hhkLowLevelMouse = SetWindowsHookEx(WH\_MOUSE\_LL, AddressOf LowLevelMouseProc, \_

 GetWindowLong(FindWindow("ThunderDFrame", ObjUSF.Caption), GWL\_HINSTANCE), 0)

End Sub

Sub UnHook\_Mouse()

 If hhkLowLevelMouse <> 0 Then

 UnhookWindowsHookEx hhkLowLevelMouse

 hhkLowLevelMouse = 0

 End If

End Sub