

Ads by Google
[Excel Code](#)
[Excel VBA Macro](#)
[Analyze Excel](#)
[Excel Developer](#)
[VBA](#)



FREE *Excel* STUFF

SEARCH

EXCEL **HELP**. Popular
NEW! MULTIPLE EXCEL
SEARCH & LINKS

EXCEL **FORMULAS**

EXCEL **MACROS**

EXCEL **NEWSLETTER**

PRODUCTS

Up to \$139.00 FREE!

CATEGORIES & **SEARCH**

EXCEL **TEMPLATES**

EXCEL **ADD-INS**

EXCEL **TRAINING**

MORE....

OTHER

EXCEL **DEVELOPMENT**

MICROSOFT EXCEL VBA MACROS. HOW TO CREATE AN EXCEL ADD-IN FOR THEM



NEW!
MORE
BOOKS..

Add to Google

Se

Spreadsheet Mod

Quantrix: Used by Top Financial Modelers in M
www.Quantrix.com

Ads by Google - Advertise on this s

Current Special! Complete Excel EXCEL TRAINING COURSE for Excel 97 - Excel 2003, only ~~\$145.00~~. \$59.95 Instant **BUY/DOWNLOAD**, **30 Day Money Back Guarantee** & Free EXCEL HELP for LIFE!

Back to: [EXCEL VBA](#) . Got any Excel/VBA Questions? **Free EXCEL HELP**

Creating Excel Add-ins

I am often asked by users 'what is the best way to distribute their macros?' My answer, is without doubt via an **Excel Add-in**. After all, this is what Add-ins are for. For those that are not sure what an Excel add-in is, it's is nothing more than an Excel Workbook that has been saved as an Add-in, **File>Save as \ Microsoft Excel Add-in (*.xla)**. Once saved and re-opened the Workbook will be hidden and **can only be seen** in the "**Project Explorer**" via the Visual Basic Editor. It is **NOT** hidden in the same way as the Personal.xls as this can be seen (and made visible) via **Windows>Unhide**.

Once completed users can easily install your Add-in like below

- Save A Copy To **C:\WINDOWS\Application Data\Microsoft\AddIns** . If Not Any Location, Just Take Note Of It For Step 4.
- Open Any Workbook.
- On The **Tools** Menu, Point To **Add-Ins** And Click **Browse**. Locate Your Add-In From Where You Saved It, Select It And Then Click **OK**.
- Ensure Your Add-In Is In The **Add-Ins Available:** Box And It Is Checked.
- Now Click **OK** And The Add-In Is Installed.

Most code can be be saved to an Excel Add-in without too many changes. Some things to be aware of are

Ads by Google

Mini-graphs - free trial

Bissantz
SparkMaker for
Excel
dashboards and
reports - free trial

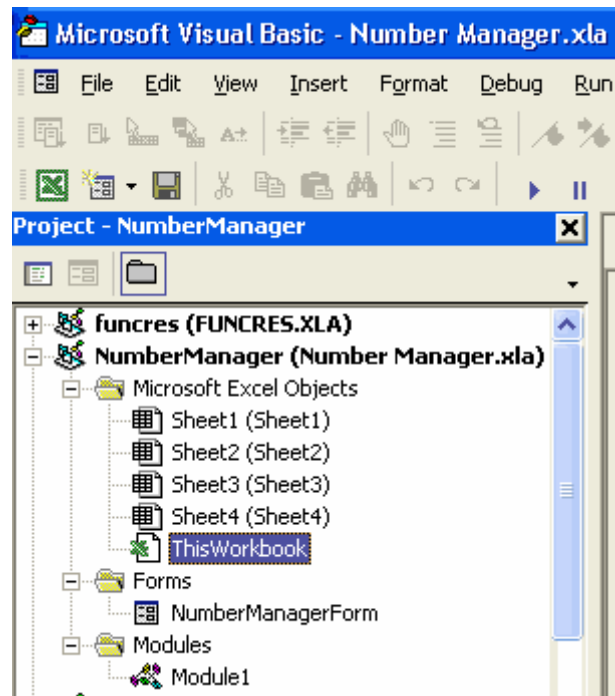
www.bissantz.de

Advertise on this site



- **ThisWorkbook** Will Always Refer To The Add-In, Not The Users Workbook. Use **ActiveWorkbook** Instead.
- We Cannot Refer To Sheets In The **ActiveWorkbook** Via [CODENAMES](#) .
- We Should Always Put Toolbars Etc Back To How The User Had Them Originally. There Is Nothing Worse Than An Add-In That Changes All Your Excel Settings Without Your Knowledge.
- Always Include Some Sort Of Error Handling (Yes, Most Add-Ins Will Error At Some Time).
- Be Very Aware That The User May Have Many Sorts Of Protection On. NEVER Use Code To Uprotect Any Part Of The Users Workbook. Simply Display A Message Asking Them To Unprotect.
- Make Full And Good Use Of The Worksheet (Sometimes More) You Have In The Add-In. I Use The Worksheet(S) To Store User Setting Like Toolbars Etc.
- Holding Down The **Shift** Key Will **NOT** Prevent Any Add-In Workbook Events Running Like It Can In A Normal .Xls
- If You Need To See The Add-In Workbook Again, Eg Updates Modifications Etc. Go Into The **VBE** While The Add-In Is Installed And From The **Properties Window** Select The **IsAddin** Property And Set It To False. Saving The Workbook As An Add-In Sets This Property To True.
- Apply Protection To The Modules Of You Add-In, Via **Tools>VBAProject Properties-Protection**.

Ok, once you have created your Add-in you will need to make the macros within it easy for the user to run. This is best achieved by making full use of both the **Workbook_AddinInstall** and the **Workbook_AddinUnInstall** Events in the Private Module of the **ThisWorkbook** Object. Simply double click on **ThisWorkbook** for the *.xla file and Excel will take you into the Private Module where the code is placed.



Let's look at a simple example of this.

Option Explicit

```
Dim cControl As CommandBarButton
```

Private Sub Workbook_AddinInstall()

```
On Error Resume Next 'Just in case
```

```
    'Delete any existing menu item that may have been left.
```

```
    Application.CommandBars("Worksheet Menu Bar").Controls("Super Code").Delete
```

```
    'Add the new menu item and Set a CommandBarButton Variable to it
```

```
    Set cControl = Application.CommandBars("Worksheet Menu Bar").Controls.Add
```

```
    'Work with the Variable
```

```
        With cControl
```

```
            .Caption = "Super Code"
```

```
            .Style = msoButtonCaption
```

```
            .OnAction = "MyGreatMacro"
```

```
            'Macro stored in a Standard Module
```

```
        End With
```

```
On Error GoTo 0
```

```
End Sub
```

```
Private Sub Workbook_AddinUninstall()
```

```
On Error Resume Next 'In case it has already gone.
    Application.CommandBars("Worksheet Menu Bar").Controls("Super Code").Delete
On Error GoTo 0
End Sub
```

This code will be all you need to add a single menu item (called **Super Code**) to the end of the existing **Worksheet Menu Bar** as soon as the Add-in is installed by the user via **Tools>Add-ins**. When the **Super Code** menu item is clicked a macro (that is within a standard module of the add-in) is run. As mentioned earlier, the above code **MUST** be placed in the Private Module of **ThisWorkbook** for the Add-in.

If you want the **Super Code** menu item added, say before the **Format** menu item, you could use some code like this.

Option Explicit

```
Dim cControl As CommandBarButton
```

Private Sub Workbook_AddinInstall()

```
Dim iContIndex As Integer
```

```
    On Error Resume Next 'Just in case
    'Delete any existing menu item that may have been left
    Application.CommandBars("Worksheet Menu Bar").Controls("SuperCode").Delete

    'Pass the Index of the "Format" menu item number to a Variable.
    'Use the FindControl Method to find it's Index number. ID number _
    is used in case of Customization
    iContIndex = Application.CommandBars.FindControl(ID:=30006).Index

    'Add the new menu item and Set a CommandBarButton Variable to it.
    'Use the number passed to our Integer Variable to position it.
    Set cControl = Application.CommandBars("Worksheet Menu Bar").Controls.Add(Befor

    'Work with the Variable
        With cControl
            .Caption = "Super Code"
            .Style = msoButtonCaption
            .OnAction = "MyGreatMacro"
            'Macro stored in a Standard Module
        End With
    On Error GoTo 0
```

End Sub

There would be no need to change the **Workbook_AddinUninstall()** code in this case.

We have covered ID numbers while working with CommandBars etc in a **Prior Newsletter Issue** The link to the Microsoft site that has a **BIG** list of all the ID numbers for working with CommandBars can be [FOUND HERE](#)

The above examples actually have the all the menu item code in the **Workbook_AddinInstall** and **Workbook_AddinUninstall** Not a problem when the code is only adding one menu item. If however, you will be adding more than one and perhaps even Sub menus, you should place it in a Procedure (or 2) inside a standard Module. Then use some code as shown below

```
Private Sub Workbook_AddinInstall()  
Run "AddMenus"
```

End Sub

```
Private Sub Workbook_AddinUninstall()  
Run "DeleteMenu"  
End Sub
```

Then in the **standard module** put some code perhaps like this

```
Sub AddMenus()  
Dim cMenu1 As CommandBarControl  
Dim cbMainMenuBar As CommandBar  
Dim iHelpMenu As Integer  
Dim cbcCutomMenu As CommandBarControl  
  
    '(1)Delete any existing one.We must use On Error Resume next _  
        in case it does not exist.  
    On Error Resume Next  
    Application.CommandBars("Worksheet Menu Bar").Controls("&NewMenu").Delete  
  
    '(2)Set a CommandBar variable to Worksheet menu bar  
    Set cbMainMenuBar = Application.CommandBars("Worksheet Menu Bar")  
  
    '(3)Return the Index number of the Help menu. We can then use _  
        this to place a custom menu before.
```

```

iHelpMenu = cbMainMenuBar.Controls("Help").Index

'(4)Add a Control to the "Worksheet Menu Bar" before Help
'Set a CommandBarControl variable to it
Set cbcCutomMenu = cbMainMenuBar.Controls.Add(Type:=msoControlPopup, Before:=if

'(5)Give the control a caption
cbcCutomMenu.Caption = "&New Menu"

'(6)Working with our new Control, add a sub control and _
    give it a Caption and tell it which macro to run (OnAction).
        With cbcCutomMenu.Controls.Add(Type:=msoControlButton)
            .Caption = "Menu 1"
            .OnAction = "MyMacro1"
        End With
'(6a)Add another sub control give it a Caption _
    and tell it which macro to run (OnAction)
        With cbcCutomMenu.Controls.Add(Type:=msoControlButton)
            .Caption = "Menu 2"
            .OnAction = "MyMacro2"
        End With

'Repeat step "6a" for each menu item you want to add.

'Add another menu that will lead off to another menu
'Set a CommandBarControl variable to it
    Set cbcCutomMenu = cbcCutomMenu.Controls.Add(Type:=msoControlPopup)
' Give the control a caption
    cbcCutomMenu.Caption = "Next Menu"

'Add a control to the sub menu, just created above
    With cbcCutomMenu.Controls.Add(Type:=msoControlButton)
        .Caption = "&Charts"
        .FaceId = 420

.OnAction = "MyMacro2"

End With

On Error GoTo 0

End Sub

Sub DeleteMenu()

```

On Error Resume Next

```
Application.CommandBars("Worksheet Menu Bar").Controls("&NewMenu").Delete  
On Error GoTo 0
```

End Sub

You can read some similar, less detailed, text on Excel add-ins on the [MICROSOFT DEVELOPER NETWORK HERE](#)

EXCEL DASHBOARD REPORTS & EXCEL DASHBOARD CHARTS 50% Off

Special! Free Choice of COMPLETE EXCEL TRAINING COURSE **OR** EXCEL ADD-INS COLLECTION on all purchases totaling over \$64.00. ALL PURCHASES totaling over \$150.00 **gets you BOTH!** **Purchases MUST be made via this site.** Send payment proof to special@ozgrid.com 31 days after purchase date.

Excel Password Cracker

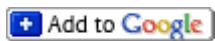
for Microsoft Excel documents. Recovers lost passwords. More... www.intelore.com

Ads by Google

Advertise

Instant Download and Money Back Guarantee on Most Software

EXCEL TRADER PACKAGE Technical Analysis in Excel With \$139.00 of **FREE software!**



SEARCH TIPS

Search

FREE EXCEL HELP

Microsoft  and Microsoft Excel  are registered trademarks of Microsoft Corporation. OzGrid is in no way associated with Microsoft