VBA SOURCE CODE BOOK

How To Automatically Set The Tab Order, Save & Load Data in Microsoft Excel

DOWNLOAD APPLICATION

VIEW TRAINING

by: Randy Austin

ABOUT THE AUTHOR

A two-time Microsoft MVP & lifetime Excel enthusiast, Randy Austin founded Excel For Freelancers in 2017. Excel For Freelancers quickly became the most prominent resource Excel for developers to learn how to turn their passion for Excel into profits by building & selling their own excel-based applications for passive & recurring income.

With nearly 300,000 YouTube subscribers, 14,000,000 video views, 200+ comprehensive training videos, and a thriving 40,000 member Facebook community, Excel For Freelancers has positioned itself as the #1 Excel developers resource in the world.

Get free content, training, and downloads just by clicking any of the free resources below:

WEBSITE YOUTUBE FACEBOOK TWITTER

INSTAGRAM TELEGRAM RUMBLE

MVP Microsoft® Most Valuable Professional

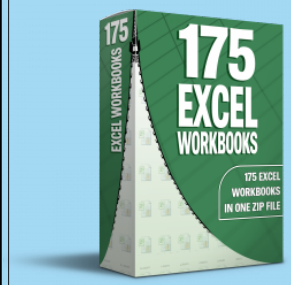
OUR COURSES & PRODUCTS

This comprehensive program will take you through a 12-phase process that will turn your enthusiasm for Excel into passive income.

Click here to learn more

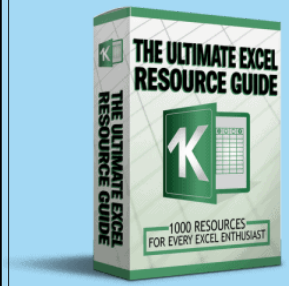
16 hour masterclass that will teach you the tips, tricks and techniques on how to create a dynamic single-click dashboard, and a ton more

Click here to learn more



Incredible Package of 175 of my BEST Applications into a SINGLE ZIP File which also includes the "175 Workbook Library".

Click here to learn more



With 1000 live links, continuously updating content, sort-able and filterable items, you will always have exactly what you need, when you need it.

Click here to learn more

Table of Contents

```
Private Sub Worksheet_Activate()
  SetOnkey (True)
End Sub

Private Sub Worksheet_Change(ByVal Target As Range)
  If Not Intersect(Target, Range("F3:F9" )) Is Nothing Then
    Cells(Range("B1" ).Value, Range("A"  & Target.Row).Value).Value = Target.Value
  End If

  If Not Intersect(Target, Range("I3:I9" )) Is Nothing Then
    Cells(Range("B1" ).Value, Range("B"  & Target.Row).Value).Value = Target.Value
  End If
End Sub

Private Sub Worksheet_Deactivate()
  SetOnkey (False)
End Sub

Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  If Not Intersect(Target, Range("E13:J26" )) Is Nothing Then
    Range("B1" ).Value = Target.Row
    Load_Customer
  End If
End Sub
```

```
1   Option Explicit
2
```

E

```
1   Option Explicit
2
3   Private Sub Workbook_Activate()
4     If ActiveSheet.Name = "Customers"  Then SetOnkey (True)
5   End Sub
6
7   Private Sub Workbook_Deactivate()
8     SetOnkey (False)
9   End Sub
10
11  Private Sub Workbook_Open()
12
13  End Sub
14
```

```
Option Explicit

Sub ResetCode()
    With Application
        .Calculation = xlCalculationAutomatic
        .ScreenUpdating = True
    End With
End Sub

Sub StopCode()
    With Application
        .Calculation = xlCalculationManual
        .ScreenUpdating = False
    End With
End Sub

```

```
1   Option Explicit
2
3   Sub Load_Customer()
4     Dim CustRow, CustCol As Long
5     Dim Cell As String
6     StopCode
7     With Sheet1
8       CustRow = .Range("B1" ).Value 'Customer Row
9       For CustCol = 5 To 10
10        Cell = .Cells(11, CustCol).Value
11        Range(Cell).Value = .Cells(CustRow, CustCol).Value
12      Next CustCol
13    End With
14    ResetCode
15  End Sub
```

```
1
2   Sub do_nothing()
3     'nothing to do
4   End Sub
5
6
7   Private Function GetColLtr(sAddr As String) As String
8     Dim iPos As Long, sTest As String
9     Do While iPos < 3
10      iPos = iPos + 1
11      If IsNumeric(Mid(sAddr, iPos, 1)) Then
12        Exit Do
13      Else
14        sTest = sTest & Mid(sAddr, iPos, 1)
15      End If
16    Loop
17    GetColLtr = sTest
18  End Function
19
20
21
22  Function GetTabOrder() As Variant
23    '--set the tab order of input cells - change ranges as required
24    '  don't include "$" in these cell references
25    If ActiveCell.Row < 12 Then
26      GetTabOrder = Array("F3" , "I3" , "F6" , "I6" , "F9" , "I9" )
27    End If
28
29    ' If ActiveCell.Row > 27 Then GetTabOrder = Array("")
30  End Function
31  Sub SetOnkey(ByVal state As Boolean)
32    'Compiled By Randy Austin
33    'Workbook Provided By www.ExcelForFreelancers.com
34    If state Then
35      With Application
36        .OnKey "{TAB}" , "'TabRange xlNext'"
37        .OnKey "~" , "'TabRange xlNext'"
38        .OnKey "{ENTER}" , "'TabRange xlNext'"
39        .OnKey "{RIGHT}" , "'TabRange xlNext'"
40        .OnKey "{LEFT}" , "'TabRange xlPrevious'"
41        .OnKey "+{TAB}" , "'TabRange xlPrevious'"
42        .OnKey "{DOWN}" , "'UpOrDownArrow xlDown'"
43        .OnKey "{UP}" , "'UpOrDownArrow xlUp'"
44      End With
45    Else
46      'reset keys
47      With Application
48        .OnKey "{ENTER}"
49        .OnKey "{TAB}"
50        .OnKey "~"
51        .OnKey "+{TAB}"
52        .OnKey "{RIGHT}"
53        .OnKey "{LEFT}"
54        .OnKey "{DOWN}"
55        .OnKey "{UP}"
56      End With
57    End If
58  End Sub
59
60  Sub TabRange(Optional iDirection As Integer = xlNext)
61
```

1

```
    Dim vTabOrder As Variant, m As Variant
    Dim lItems As Long, iAdjust As Long
    On Error GoTo ExitSub
    '--get the tab order from shared function
    vTabOrder = GetTabOrder
    lItems = UBound(vTabOrder) - LBound(vTabOrder) + 1

    On Error Resume Next
    m = Application.Match(ActiveCell.Address(0, 0), vTabOrder, False)


    '--if activecell is not in Tab Order return to the first cell
    If IsError(m) Then
      m = 1
    Else
      '--get adjustment to index
      iAdjust = IIf(iDirection = xlPrevious, -1, 1)

      '--calculate new index wrapping around list
      m = (m + lItems + iAdjust - 1) Mod lItems + 1
    End If

    '--select cell adjusting for Option Base 0 or 1
    Application.EnableEvents = False
    Range(vTabOrder(m + (LBound(vTabOrder) = 0))).Select

    ExitSub:
    Application.EnableEvents = True
End Sub

Sub UpOrDownArrow(Optional iDirection As Integer = xlUp)

    Dim vTabOrder As Variant
    Dim lRowClosest As Long, lRowTest As Long
    Dim i As Long, iSign As Integer

    Dim sActiveCol As String
    Dim bFound As Boolean

    '--get the tab order from shared function
    vTabOrder = GetTabOrder

    '--find TabCells in same column as ActiveCell in iDirection
    '--  rTest will include ActiveCell

    sActiveCol = GetColLtr(ActiveCell.Address(0, 0))

    iSign = IIf(iDirection = xlDown, -1, 1)
    lRowClosest = IIf(iDirection = xlDown, Rows.Count + 1, 0)

    For i = LBound(vTabOrder) To UBound(vTabOrder)
      If GetColLtr(CStr(vTabOrder(i))) = sActiveCol Then
        lRowTest = Range(CStr(vTabOrder(i))).Row

        '--find closest cell to ActiveCell in rTest
        If iSign * lRowTest > iSign * lRowClosest And _
           iSign * lRowTest < iSign * ActiveCell.Row Then
          '--at least one cell in iDirection of same columnn
          bFound = True
          lRowClosest = lRowTest
        End If
```

```
1 2 3
123         End If
124       Next i
125
126       If bFound Then
127         Application.EnableEvents = False
128         Cells(lRowClosest, ActiveCell.Column).Select
129         Application.EnableEvents = True
130       End If
131     End Sub
```

Thank You!

This source code was created and made available to help you gain a better understanding of how VBA is used to create amazing Excel-based applications.

Thank you so much for your continued shares, likes and support. It really helps.

Excel For Freelancers