

VBA SOURCE CODE BOOK

AUTO TAB, LOAD & SAVE

Auto Tab, Load & Save

Customer Name: David Breakman Address: 45414 Maple St.
City: San Diego Email: David@Gmail.com
Home Phone #: (454) 154-1545 Notes: Customer Pays On time

Customer Name	Address	City	Email	Home Phone #	Notes
John Abners	12345 Main St	Los Angeles	John@Gmail.com	(310) 805-4545	Test Notes
David Breakman	45414 Maple St.	San Diego	David@Gmail.com	(454) 154-1545	Customer Pays On time
Fred Calderon	1248 Sandstone Ave.	Santa Monica	Fred@Gmail.com	(484) 578-5487	Test
Jimmy Dean	12983 Park Street	Sacramento	Jimmy@Gmail.com	(454) 545-7878	Customer Was Late
Frank Fredders	4578 Canoga Ave.	Woodland Hills	Frank@Gmail.com	(425) 274-8723	These are test notes
Jeff Johns	12151 Wilshire Blvd.	West LA	Jeff@Gmail.com	(410) 062-6752	On Vacation
Lisa Lanie	45787 Davidson St.	Granada Hills	Lisa@Gmail.com	(394) 850-4781	Test
Debbie Lorenzos	4551 Oak Ridge Ave.	Ventura	David@Gmail.com	(555) 555-5561	Test Friendly Customer
Fred Fredder	15487 Fairview Ln.	Thousand Oaks	Fred@Gmail.com	(555) 555-5555	Test Notes 4
Alex Sampson	2154 Dabney Rd	Los Angeles	Jimmy@Gmail.com	(555) 555-5557	Test Notes
Mary Sampson	4581 Hollands Dr.	San Diego	Frankenstien@Gmail	(555) 555-5560	
Sally Smith	2155 Dabney Rd	Santa Monica	Jeff@Gmail.com	(555) 555-5563	Test Notes
Mark Thompson	4582 Hollands Dr.	Sacramento	David@Gmail.com	(555) 555-5566	
Barry Yeager	2156 Dabney Rd	Woodland Hills	Fred@Gmail.com	(555) 555-5569	
Mary Zingers	4583 Hollands Dr.	West LA	Jimmy@Gmail.com	(555) 555-5572	



Excel For Freelancers

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](#)



Microsoft®
Most Valuable
Professional



OUR COURSES & PRODUCTS



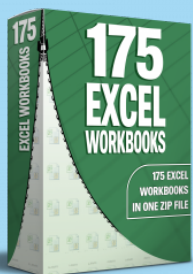
This comprehensive program will take you through a 12-phase process that will turn your enthusiasm for Excel into passive income.

[Click here to learn more](#)



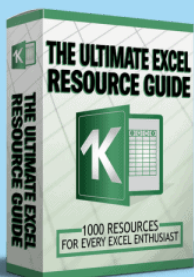
16 hour masterclass that will teach you the tips, tricks and techniques on how to create a dynamic single-click dashboard, and a ton more

[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](#)

Projects	2
VBAProject	2
Documents	2
Sheet1	2
Worksheet_Activate [Sub]	2
Worksheet_Change [Sub]	2
Worksheet_Deactivate [Sub]	2
Worksheet_SelectionChange [Sub]	2
Sheet2	4
(Declarations)	4
ThisWorkbook	6
(Declarations)	6
Workbook_Activate [Sub]	6
Workbook_Deactivate [Sub]	6
Workbook_Open [Sub]	6
Modules	8
CodeReset_Macs	8
(Declarations)	8
ResetCode [Sub]	8
StopCode [Sub]	8
Form_Load	10
(Declarations)	10
Load_Customer [Sub]	10
TabMacros	12
do_nothing [Sub]	12
GetColLtr [Function]	12
GetTabOrder [Function]	12
SetOnkey [Sub]	12
TabRange [Sub]	12
UpOrDownArrow [Sub]	13

```
1 Private Sub Worksheet_Activate()  
2     SetOnkey (True)  
3 End Sub  
4  
5 Private Sub Worksheet_Change(ByVal Target As Range)  
6     If Not Intersect(Target, Range("F3:F9" )) Is Nothing Then  
7         Cells(Range("B1" ).Value, Range("A" & Target.Row).Value).Value = Target.Value  
8     End If  
9  
10    If Not Intersect(Target, Range("I3:I9" )) Is Nothing Then  
11        Cells(Range("B1" ).Value, Range("B" & Target.Row).Value).Value = Target.Value  
12    End If  
13 End Sub  
14  
15 Private Sub Worksheet_Deactivate()  
16     SetOnkey (False)  
17 End Sub  
18  
19 Private Sub Worksheet_SelectionChange(ByVal Target As Range)  
20     If Not Intersect(Target, Range("E13:J26" )) Is Nothing Then  
21         Range("B1" ).Value = Target.Row  
22         Load_Customer  
23     End If  
24 End Sub
```

C

Cells, [2](#)

I

Intersect, [2](#)

L

Load_Customer, [2](#)

R

Range, [2](#)

Row, [2](#)

S

SetOnkey, [2](#)

T

Target, [2](#)

V

Value, [2](#)

W

Worksheet_Activate, [2](#)

Worksheet_Change, [2](#)

Worksheet_Deactivate, [2](#)

Worksheet_SelectionChange, [2](#)

1 Option Explicit

2

E

Explicit, 4


```
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
```

A

ActiveSheet, [6](#)

E

Explicit, [6](#)

N

Name, [6](#)

S

SetOnkey, [6](#)

W

Workbook_Activate, [6](#)

Workbook_Deactivate, [6](#)

Workbook_Open, [6](#)

```
1 Option Explicit
2
3 Sub ResetCode()
4     With Application
5         .Calculation = xlCalculationAutomatic
6         .ScreenUpdating = True
7     End With
8 End Sub
9
10 Sub StopCode()
11     With Application
12         .Calculation = xlCalculationManual
13         .ScreenUpdating = False
14     End With
15 End Sub
16
```

A

Application, 8

C

Calculation, 8

E

Explicit, 8

R

ResetCode, 8

S

ScreenUpdating, 8

StopCode, 8

X

xlCalculationAutomatic, 8

xlCalculationManual, 8

```
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
```

C

Cell, [10](#)
Cells, [10](#)
CustCol, [10](#)
CustRow, [10](#)

E

Explicit, [10](#)

L

Load_Customer, [10](#)

R

Range, [10](#)
ResetCode, [10](#)

S

Sheet1, [10](#)
StopCode, [10](#)

V

Value, [10](#)

```
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
32 Sub SetOnkey(ByVal state As Boolean)
33   'Compiled By Randy Austin
34   'Workbook Provided By www.ExcelForFreelancers.com
35   If state Then
36     With Application
37       .OnKey "{TAB}" , "'TabRange xlNext'"
38       .OnKey "~" , "'TabRange xlNext'"
39       .OnKey "{ENTER}" , "'TabRange xlNext'"
40       .OnKey "{RIGHT}" , "'TabRange xlNext'"
41       .OnKey "{LEFT}" , "'TabRange xlPrevious'"
42       .OnKey "+{TAB}" , "'TabRange xlPrevious'"
43       .OnKey "{DOWN}" , "'UpOrDownArrow xlDown'"
44       .OnKey "{UP}" , "'UpOrDownArrow xlUp'"
45     End With
46   Else
47     'reset keys
48     With Application
49       .OnKey "{ENTER}"
50       .OnKey "{TAB}"
51       .OnKey "~"
52       .OnKey "+{TAB}"
53       .OnKey "{RIGHT}"
54       .OnKey "{LEFT}"
55       .OnKey "{DOWN}"
56       .OnKey "{UP}"
57     End With
58   End If
59 End Sub
60
61 Sub TabRange(Optional iDirection As Integer = xlNext)
```

```

1
62 Dim vTabOrder As Variant, m As Variant
63 Dim lItems As Long, iAdjust As Long
64 On Error GoTo ExitSub
65 '--get the tab order from shared function
66 vTabOrder = GetTabOrder
67 lItems = UBound(vTabOrder) - LBound(vTabOrder) + 1
68
69 On Error Resume Next
70 m = Application.Match(ActiveCell.Address(0, 0), vTabOrder, False)
71
72
73 '--if activecell is not in Tab Order return to the first cell
74 If IsError(m) Then
75     m = 1
76 Else
77     '--get adjustment to index
78     iAdjust = IIf(iDirection = xlPrevious, -1, 1)
79
80     '--calculate new index wrapping around list
81     m = (m + lItems + iAdjust - 1) Mod lItems + 1
82 End If
83
84 '--select cell adjusting for Option Base 0 or 1
85 Application.EnableEvents = False
86 Range(vTabOrder(m + (LBound(vTabOrder) = 0))).Select
87
88 ExitSub:
89 Application.EnableEvents = True
90 End Sub
91
92 Sub UpOrDownArrow(Optional iDirection As Integer = xlUp)
93
94     Dim vTabOrder As Variant
95     Dim lRowClosest As Long, lRowTest As Long
96     Dim i As Long, iSign As Integer
97
98     Dim sActiveCol As String
99     Dim bFound As Boolean
100
101     '--get the tab order from shared function
102     vTabOrder = GetTabOrder
103
104     '--find TabCells in same column as ActiveCell in iDirection
105     '-- rTest will include ActiveCell
106
107     sActiveCol = GetColLtr(ActiveCell.Address(0, 0))
108
109     iSign = IIf(iDirection = xlDown, -1, 1)
110     lRowClosest = IIf(iDirection = xlDown, Rows.Count + 1, 0)
111
112     For i = LBound(vTabOrder) To UBound(vTabOrder)
113         If GetColLtr(CStr(vTabOrder(i))) = sActiveCol Then
114             lRowTest = Range(CStr(vTabOrder(i))).Row
115
116             '--find closest cell to ActiveCell in rTest
117             If iSign * lRowTest > iSign * lRowClosest And _
118                 iSign * lRowTest < iSign * ActiveCell.Row Then
119                 '--at least one cell in iDirection of same column
120                 bFound = True
121                 lRowClosest = lRowTest
122             End If

```

1 2 3


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

→, 13

A

ActiveCell, 12-14
Address, 13
Application, 12-14
Array, 12

B

bFound, 13, 14

C

Cells, 14
Column, 14
Count, 13

D

do_nothing, 12

E

EnableEvents, 13, 14
ExitSub, 13

G

GetColLtr, 12, 13
GetTabOrder, 12, 13

I

i, 13, 14
iAdjust, 13
iDirection, 12, 13
If, 13
iPos, 12
IsError, 13
iSign, 13
IsNumeric, 12

L

LBound, 13
lItems, 13
lRowClosest, 13, 14
lRowTest, 13

M

m, 13
Match, 13
Mid, 12

O

OnKey, 12

R

Range, 13
Row, 12, 13
Rows, 13

S

sActiveCol, 13
sAddr, 12
SetOnkey, 12
state, 12
sTest, 12

T

TabRange, 12

U

UBound, 13

UpOrDownArrow, 13

V

vTabOrder, 13

X

xlDown, 13
xlNext, 12
xlPrevious, 13
xlUp, 13

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