

Re: Is it possible to run a macro in tandem with, or right after,

## Re: Is it possible to run a macro in tandem with, or right after,

---

*Source:*

<http://www.tech-archive.net/Archive/Excel/microsoft.public.excel.programming/2007-10/msg04513.html>

---

- *From:* JLatham <HelpFrom @ Jlathamsite.com.(removethis)>
  - *Date:* Sat, 27 Oct 2007 03:49:01 -0700
- 

Ok, let me try it this way.

Let me start by saying anything you can do from the keyboard you can do in code. That includes the actions performed by [Ctrl]+[C], [Ctrl]+[R], Edit | Paste Special and anything else. What I was trying to say is that everything you're doing with several different operations can be done within a single macro, or by having one macro call several others that each perform one piece of the process. I only mentioned using .OnKey because you appeared to want a way to do it all by pressing [Ctrl]+[C] – but you don't have to use that as the shortcut to your initial macro.

When you choose (select) the cells to be worked with, that becomes the SELECTION within VBA code. So you can start right off with Selection.Copy which is the same as [Ctrl]+[C]

Record a macro while you do it one time and you'll see what objects and methods to use to get the job done.

"meh2030@xxxxxxxxx" wrote:

On Oct 26, 7:16 pm, JLatham <HelpFrom @ Jlathamsite.com.(removethis)> wrote:

I think if you'll do some research on Excel's Application.OnKey method you will come up with the answer you want. Basically you could use the workbook\_open, or a specific worksheet's worksheet\_activate event (one of a couple of others) to reassign the action taken when [Ctrl]+[C] is clicked – you could have it first do a .Copy of the currently selected area, then call your store\_Row\_Heights() routine and then do whatever else is needed. Actually you could put the .Copy part at the beginning of the store\_Row\_Heights routine if it would work to have it there and have the [Ctrl]+[C] run that routine.

Re: Is it possible to run a macro in tandem with, or right after,

Re: Is it possible to run a macro in tandem with, or right after,

Don't forget to reset [Ctrl]+[C] to its normal function when you're done with having it do your special processing: good places for that would be Worksheet\_Deactivate  
Workbook\_Deactivate  
Workbook\_BeforeClose  
events.

"meh2...@xxxxxxxx" wrote:

Code is below:

I'm on a quest to create a macro that performs like a copy, paste special. About two days ago I posted a query entitled "Custom Paste Special – Row Heights" but have since learned that it appears that there is no way to access a copied range (i.e. access the marquee range produced by a ctrl+c). As a result, I have crafted something new below. (FYI: on a first time basis, store\_Row\_Heights must run before output\_Row\_Heights).

I know that if you assign two macros to one keyboard shortcut, the most alphabetically macro will only run. I will eventually assign "output\_Row\_Heights" to a keyboard shortcut (Application.OnKey "+^r", "output\_Row\_Heights"), and if needs be I can just as easily assign "store\_Row\_Heights" to a keyboard shortcut as well. However, I'm looking to see if there is another solution.

If it's possible to have store\_Row\_Heights run promptly after ctrl+c then it will save me from having to assign an additional keyboard shortcut to "store\_Row\_Heights". (Additionally it will be similar to

Re: Is it possible to run a macro in tandem with, or right after,

the paste special operation where you first copy and then paste a particular attribute). Thus, I'm wondering if it's possible to have my macro run right after ctrl+c. (This shouldn't affect the integrity of ctrl+c).

One idea that I have is to somehow programmatically detect when the user presses "ctrl+c" or detect when the user depresses "ctrl+c" and then have "store\_Row\_Heights" run. However, I have yet to program anything related to detecting or "trapping" events. If this is the answer then I can pick up the books and start learning, but I want to get some feedback before I start down this path.

If you have a specific answer please let me know. Thanks in advance.

Option Explicit

```
Dim i As Long 'use as a counter
Dim j As Long 'use as a counter
Dim a 'temporary miscellaneous variable
```

```
Function store_Row_Heights(Optional inputAry) As Variant
'The Optional parameter is so that the output_Row_Heights
'sub procedure can access the rowHgt array from this
'function.
```

```
Dim copyRng As Range
Dim rowRng As Range
Dim numRows As Long
Dim rowHgt() As Single
Dim rowCell As Range
```

Re: Is it possible to run a macro in tandem with, or right after,

```
'Static so that when output_Row_Heights accesses this  
function  
'the stored rowHgts array can be returned back to the  
'output_Row_Heights sub procedure.  
Static rowHgts() As Single
```

```
If Not IsMissing(inputAry) Then 'the optional parameter is  
determined  
by Missing
```

```
store_Row_Heights = rowHgts  
'do I need to delete whatever is in the static variable right  
here?
```

```
Exit Function
```

```
End If
```

```
'create a range object that storse the range that is selected  
Set copyRng = selection
```

```
numRows = copyRng.Rows.Count 'copy the number of rows  
in the copied  
range
```

```
'create a new range object that contains only 1 column  
Set rowRng = copyRng.Range(Cells(1, 1), Cells(numRows,  
1))
```

```
ReDim rowHgts(1 To numRows)
```

Re: Is it possible to run a macro in tandem with, or right after,

```
'store the row heights of the copied range  
i = 0
```

```
For Each rowCell In rowRng.Cells
```

```
    i = i + 1
```

```
    rowHgts(i) = rowCell.RowHeight  
    'Debug.Print i; ":"; rowHgts(i)
```

```
Next
```

```
End Function
```

```
Sub output_Row_Heights()
```

```
    Dim passAry() As Single  
    Dim aryTest As Boolean  
    Dim firstTimeAryTest As Boolean  
    Dim outputRow As Long
```

```
    'Static so that if the user wants to output the row heights in  
    multiple  
    'locations the row heights don't need to be "re-copied". (I'm  
    not sure  
    if  
    'I want this feature yet).  
    Static outputRowHgts() As Single
```

```
    aryTest = IsArrayAllocated(passAry)
```

Re: Is it possible to run a macro in tandem with, or right after,

```
If aryTest = False Then
```

```
outputRowHgts = store_Row_Heights(passAry)
```

```
'check if this sub procedure is being run before  
'the store_Row_Heights function is run (i.e. can't  
'run output_Row_Heights before store_Row_Heights...  
'for the very first time use that is. The Static  
'variable outputRowHgts will allow you to run  
'output_Row_Heights later on, but I'm not sure  
'quite yet if I want it to be this way).  
firstTimeAryTest = IsArrayAllocated(outputRowHgts)
```

```
If firstTimeAryTest = False Then
```

```
MsgBox "No row heights have been copied for pasting."
```

```
Exit Sub
```

```
End If
```

```
End If
```

```
'test if the pasting selection will run into the  
'end of the worksheet  
outputRow = selection.Row
```

```
If outputRow + UBound(outputRowHgts) > 65536 Then
```

```
MsgBox "You are trying to paste more row heights than" &  
vbCr _
```

Re: Is it possible to run a macro in tandem with, or right after,

& "there is room on the worksheet."

Exit Sub

End If

'output the stored row heights in the new selection  
For j = LBound(outputRowHgts) To  
UBound(outputRowHgts)

selection.Cells(1, 1).Offset(j - 1, 1).RowHeight =  
outputRowHgts(j)  
'Debug.Print j

Next

End Sub

Function IsArrayAllocated(Arr As Variant) As Boolean  
'from Chip Pearson's website

'an array handled by the Split function needs to test whether  
'LBound is greater than UBound because the L- and  
UBound functions  
'don't return errors.

'The function below, IsArrayAllocated will accurately return  
'True or False indicating whether the array is allocated.  
'This function will work for both static and dynamic arrays of  
'any number of dimensions, and will correctly work for  
unallocated  
'arrays with valid (non-error-causing) LBound values, such  
as those

Re: Is it possible to run a macro in tandem with, or right after,

'arrays set by the Split function.

On Error Resume Next

```
IsArrayAllocated = Not (IsError(LBound(Arr))) And _  
IsArray(Arr) And _  
(LBound(Arr) <= UBound(Arr))
```

End Function– Hide quoted text –

– Show quoted text –

I appreciate the thoughts, but I'm either not sure if I was clear enough in my explanation or if you didn't read clearly enough. As I mentioned earlier, I don't want to change the integrity of ctrl+c. I want my macro to behave much like a copy, paste special. I don't want to "use" ctrl+c as the .OnKey for my macro. This is because for the operation that I'm doing I typically perform copy -- paste special -- formats; then paste special -- links (no need to copy again); and I want to add paste special -- row heights. I don't want to have to do a separate action to copy my row heights into the store\_Row\_Heights function and then do a separate action to run the output\_Row\_Heights sub procedure. I want to mimic the copy -- paste special. Copy to "load up" the info (i.e. all, formulas, values, formats, etc.) and my custom row heights, then paste special -- formats; paste special -- links; and then output\_Row\_Heights (i.e. Application.OnKey "+^r", "output\_Row\_Heights" as noted earlier; and yes, I would put this into my personal.xls file in the Workbook open event to set the key and then unset the key with the Workbook close event).

If you have an other ideas, I'm all ears.

Matt