Excel Hotkey Upgrade

and the

Improvements to Excel User Productivity

By:

Matthew Cowan

Contents

Executive Summary

Business Description

Zions Bancorporation is bank holding company that operates over 500 full-service banking offices in Arizona, California, Colorado, Idaho, Nevada, New Mexico, Oregon, Texas, Utah and Washington. The Corporate Finance Department produces a variety of management reports including the annual budget, forecasted financial statements, bank stress tests, macroeconomic forecasts, regulatory capital ratios and liquidity analysis.

Business Problem

Each report produced by the Corporate Finance Department must be presented in a clean and professional manner as these reports are sent to key stakeholders including the executive management team, the Board of Directors, common stock investors and the Federal Reserve Bank and other bank regulators. Formatting an Excel file to appear clean and professional can often take longer than retrieving the data from a database or performing calculations within Excel. Improvements in the speed at which Excel users are able to format their documents can, therefore, greatly improve the output of the department as a whole.

Although we do not often think about it, the seemingly small amount of time each user requires to reach over to the mouse, select a cell range, click on the appropriate ribbon and button and move his/her hands back to the keyboard can add up to a significant amount of time throughout the day. This amounts to lost productivity for the user and the department as a whole.

Solution: Excel Hotkey Upgrade

Excel hotkeys are one method to increase the speed at which users format reports in Excel. Some of the most popular hotkeys include ctrl + B to make the font bold, ctrl + C to copy a range and ctrl + V to paste a copied or cut range.

For my final project in MBA 614, I decided to build an Excel Add-In that would create new hotkeys that will speed up other frequently used formatting properties in Excel. The macros in this file perform the following functions and are called by the following keystrokes:

1. Ctrl + Shift + B: Toggles through a variety of different border options for the selected range.
2. Ctrl + K: Toggles through a variety of interior color options for the selected range.
3. Ctrl + Shift + K: Toggles through a variety of formats for the numbers in the cells of the selected range.
4. Ctrl + J: Auto-fits all the rows in the selected range.
5. Ctrl + Shift + J: Auto-fits all the columns in the selected range.
6. Ctrl + Shift + H: Turns on and off the view gridlines property in the selected sheet.
7. Ctrl + N: Decrease the font size of the cells in the selected range.
8. Ctrl + Shift + N: Increase the font size of the cells in the selected range.
9. Ctrl + M: Decrease the column width of the selected range[1].
10. Ctrl + Shift + M: Increase the column width of the selected range.
11. Ctrl + L: Decrease the row height of the selected range.

[1]**Note:** For both the column width and row height macros, if the heights or widths are not equal, the code will reset the height to 15 or the width to 8.43 (the Excel defaults) as the heights/widths must be equal for the code to increase/decrease the height/width.

12. Ctrl + Shift + L: Increase the row height of the selected range.

The remainder of this paper will describe the underlying code used to accomplish each formatting hotkey macro.

Implementation of the Solution

To activate the file on a user's computer, all the user needs to do is download the xlsm file attached with this document, save it as an Excel Add-In, go into the Add-Ins subsection of the Excel Options menu and activate the Add-In once it has been saved. It should be saved in a location where it will not need to be moved as this will stop the functionality of the add-in.

Implementation Documentation

Ctrl + Shift + B

In order to toggle through different border options on the selected range, I declared a static variable ‘a’ and used this variable as a counter for a Select Case loop that toggles through 8 border options:

- Bottom border only
- Top border only
- All borders
- Outside border
- Top and bottom border
- Top border single line, bottom border double line
- Thick outside border
- No border

It should be noted that each case begins by erasing all the previous border formatting for the selected range. Although it was possible to leave the previous border selection intact (unless it was replaced by the macro), it made more sense to erase the previous selection each time the user toggles through a different border format.

By using a static variable, the next time the user pushes Ctrl + Shift + B, the next case will be applied to the selected range (even if the selected range is different from the range originally formatted) as the counter will be saved at the number last used.

After the select case has run, the code contains the statement “a = a + 1” so that the next time the user presses Ctrl + Shift + B, the next case will be utilized. The cases number from 0 to 7 and an if statement at the very bottom states that if ‘a’ > 7, then ‘a’ = 0, so the variable will loop back to 0 and restart.

Ctrl + K

In order to toggle through a set of interior colors for the selected range, I declared a new variable ‘interiorColor’ as an integer. Then I wrote two sets of if statements that are meant to assess what the current interior color of the range is and what it should be changed to based on the current interior color.

The first if statement looks at the selection and if the entire selection’s color index is 2, then interiorColor is set equal to 1, or else if the entire selection’s color index is 6, then interiorColor is set equal to 2, etc until all 7 color indexes in the macro have been tested and if none of the color indexes match, then interiorColor is set equal to 0.

The next if statement assigns a color index to the region based on the number held in the interiorColor variable. If interiorColor = 0, then the interior color index for the entire selection is set to 2, or else if interiorColor = 1, then the color index is set to 6, etc. The final interior color that is utilized once the 7 colors have been toggled through is -4142, which is no color, and if the selection is no color, then the interiorColor variable will be reset to 0 and the macro will begin from the first color index in the list once again.

Ctrl + Shift + K

This hotkey toggles through a variety of possible number formats for the selected cells. The loop that toggles through number formats operates exactly the same as the border toggle macro with a static variable 'b' and a Select Case loop that toggles through the formats.

The following number formats are programmed into the loop:

1. Number with commas - #,##0.00_);(#,##0.00)
2. Currency - $#,##0.00_);($#,##0.00)
3. Accounting - _($* #,##0.00_);_($* (#,##0.00);_($* ""-""??_);_(@_)
4. Percent - #,##0.00%_);(#,##0.00%)
5. Short date - m/d/yy
6. General

Ctrl + J & Ctrl + Shift + J

Both these macros are very simple, but very handy because they save the user having to select the column/row they want to auto-fit and double click with the mouse on the row/column selected. The entire code is simply: Selection.EntireRow.AutoFit and Selection.EntireColumn.AutoFit.

Ctrl + Shift + H

This is also a relatively simple code, but saves quite a bit of effort (aggregated over the total number of times you have to go to the ribbon, go to the view tab and select/unselect view gridlines). There is an if statement that looks if the ActiveWindow.DisplayGridlines = True. If it is true, then it switches to false; if it is false, it switches to true.

Ctrl + N & Ctrl + Shift + N

These two macros decrease and increase the font in the cells of the selected range. One nice feature is that if different cells have different font sizes, the relative differences in font size remain, but all fonts increase/decrease by one point.

The code for both is exactly the same except one has a plus sign and the other a minus sign. First, I wrote in the code 'On Error Resume Next' because I didn't want an error message to pop up if the code tried to make the font larger or smaller than the max/min allowed in Excel. Next I declared 'cell' as a range variable. Then I created a For Each loop that looks at each cell in the selection and makes the font size equal to the current font size minus 1 or plus 1. Thus, the code loops through each cell in the range and decreases/increases the font size by 1.

Ctrl + M & Ctrl + Shift + M

These hotkey upgrades allow the user to increase and decrease the column width of the selected range. I should note a key limitation of this code. I was only able to design the code to increase/decrease the width if all the columns in the selected range are the same width, so if they are not the same width, the code automatically resets them to the default excel width of 8.43.

Once again, the code on both macros is very similar. To begin the code, I typed in 'On Error Resume Next' so there will not be an error message if the user tries to increase/decrease the width beyond the max/min width allowed by Excel. Then I created two if statements: the first statement asks if the selection.columnWidth is >= 0 (for the code to decrease the column width, the if statement asks if columnWidth is <=255, the maximum width of a cell in Excel) and if it is, then it skips to a later spot in the code that increases the column width of the selection by 0.5.

The next if statement asks if selection.columnWidth = Null. If it does, then the column width is set to 8.43 and the sub is exited from. Anytime the column widths of a selection are not equal, the value that debug.print returns for columnWidth is Null. Thus, I was forced to make all the columns equal in width in order to create a code that increases/decreases the column width.

Ctrl + L & Ctrl + Shift + L

These macros increase/decrease the row height of the selected range. The code is exactly the same as the column width code except the Excel default row height is 15 and the increment is 1, not 0.5.

Discussion of Learning and Conceptual Difficulties Encountered

Writing this code was a great refresher on various sections of the class this semester. I relearned how to use the select case loop, which was very handy for the border and number format toggle codes. I also relied heavily on if statements and the if/elseif/else constructs we learned earlier this semester.

The only major difficulty that I was unable to resolve deals with the column width and row height increase/decrease macros. Optimally, I wanted to have each column or row in the selection increase by a fixed amount and keep the relative differences the same. For example, if there were 3 columns with widths 1, 2 and 3, they would all increase to 2, 3 and 4. I was unable to find a coding trick that would accomplish this, even though I believe it is possible.

I tried to dim 'cell' as a range and increase the width of each 'cell' within the range, but the problem with this solution is that if the selected range is a large range, it will increase the width (or height) by the fixed amount more than once. For example, if I had selected a range that was 1x5 cells, the code would have worked fine as each cell width would increase by the fixed amount, but if it was 2x5 or 3x5, the width would increase by the fixed amount 2 or 3 times over. If a large range were selected, the loop would take a very long time to execute and the cells would go to the maximum height/width, which is not what the user would want.

Assistance on the Project

I did not receive any assistance from classmates, the TA or the professor on this assignment. I did consult online blogs/forums on vba programming for several of the more difficult codes, but that is the extent of the assistance I received on this assignment.