How to Easily Comment Blocks of Code in VBA

Authored by

stats writer

November 19, 2025

Writing clean, maintainable, and understandable code is paramount for any developer, and this is especially true when working within the ecosystem of Visual Basic for Applications (VBA). Commenting your code is not merely a suggestion; it is a fundamental practice that facilitates long-term project viability and collaborative development. When applied correctly, comments serve as internal documentation, providing immediate insight into the functionality and logic of complex code blocks. This clarity drastically simplifies the process of maintenance, future modification, and, most critically, debugging.

While VBA offers straightforward mechanisms for commenting individual lines of code, the lack of a native, dedicated function for commenting out large sections simultaneously presents a significant hurdle for programmers. Fortunately, the flexibility of the Visual Basic Editor (VBE) allows users to implement a highly efficient workaround. By customizing the toolbar and assigning a dedicated keyboard shortcut, developers can streamline the process of commenting and uncommenting entire code sections, dramatically improving workflow and productivity.

This detailed guide walks you through the precise steps required to configure a custom keyboard shortcut key, enabling instant block commenting functionality within the VB Editor. Implementing this customization is essential for anyone who regularly works with extensive VBA projects, as it ensures that documentation remains current and that large sections of test code can be disabled and re-enabled efficiently without manual intervention on every line.

The Critical Role of Code Commenting in VBA

Effective code commentary is arguably one of the most undervalued skills in programming, particularly in environments like VBA where code is often inherited or revisited after long periods of inactivity. Comments act as signposts, explaining not just **how the code works**, but perhaps more importantly, **why the code is structured that way**. This contextual information is invaluable when attempting to trace logical errors during the debugging process or when integrating new features into an existing application.

In a typical corporate setting, VBA projects often involve complex interactions with Microsoft Office applications, manipulating large datasets or automating critical business processes. Without robust internal documentation provided by well-placed comments, these complex procedures quickly become opaque, requiring significant reverse engineering every time a modification is needed. Good commenting standards thus directly translate into reduced maintenance costs and fewer errors deployed in production environments, reinforcing the code's longevity and reliability.

Furthermore, commenting is crucial for temporary code management. Developers frequently need to disable a block of code--perhaps for testing alternative logic, isolating a fault, or temporarily removing obsolete features. The ability to quickly comment out dozens of lines of code and then uncomment them when ready is a powerful development tool. This necessity is precisely why

implementing an efficient block commenting mechanism within the VB Editor is highly recommended, despite VBA's native limitations.

Understanding Single-Line Comments in VBA

The standard method for commenting in VBA utilizes the single quotation mark ('). This simple character, placed at the beginning of a line, instructs the compiler to ignore everything that follows on that specific line. If the apostrophe is placed midway through a line of code, the compiler executes the code preceding the apostrophe and ignores the text that follows it. This is useful for providing inline documentation for complex statements or variable declarations.

While the single quotation mark is highly effective for individual lines or short annotations, manually adding an apostrophe to the beginning of every line within a long block of code is tedious and error-prone. Imagine having a fifty-line Sub procedure that needs temporary disabling; performing this task line-by-line is inefficient and distracts the developer from the core task of programming or debugging. This limitation highlights the need for a dedicated, multi-line solution.

For context, here is how a single-line comment appears in standard VBA syntax:

```
Sub CalculateData()
Dim i As Integer ' Variable used for loop counter
For i = 1 To 10
' Code to process data goes here
Cells(i, 1).Value = i * 2
Next i
End Sub
```

The functionality to quickly toggle large sections of code is not integrated natively, forcing developers to look for customization options within the IDE. The following sections detail the customization steps required to overcome this deficiency and introduce a robust block commenting mechanism.

The Challenge of Commenting a Code Block

As established, VBA, unlike many other modern programming languages, does not include a native multi-line comment delimiter (e.g., `/* ... */`). This architectural choice forces developers to rely solely on the single-line apostrophe, which is inadequate for managing large sections of code during development cycles. When testing requires bypassing a significant portion of application logic, the time cost associated with manual commenting rapidly accumulates.

The standard alternative to manual commenting is often deleting the code temporarily, which

introduces severe risks of data loss or version control nightmares. A safer, non-destructive method is essential for high-quality software development. Therefore, integrating the 'Comment Block' feature, which is hidden within the VB Editor's customization options, becomes a critical configuration step for serious VBA development.

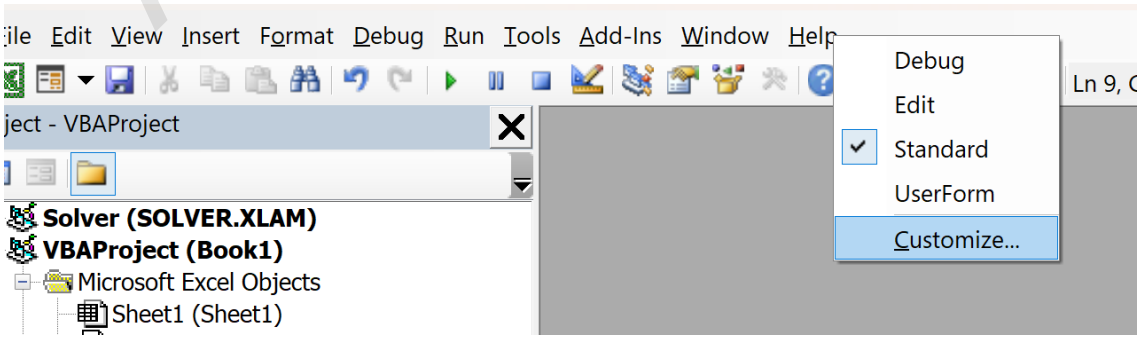
The customization process involves exposing a built-in command that exists solely within the IDE's command set, but is not displayed on the default menus or toolbars. By making this command accessible via the toolbar, and subsequently assigning a keyboard shortcut key, we transform a cumbersome manual task into a quick, two-key operation, drastically accelerating iterative development and testing procedures.

Customizing the VB Editor Toolbar (Step 1)

The first step in creating this essential block commenting feature is accessing the customization settings within the Visual Basic Editor (VBE). The VBE environment is highly configurable, allowing users to tailor the interface to their specific needs, including adding custom commands to existing toolbars. Initiating this process is straightforward, ensuring that even novice users can quickly implement this powerful modification.

To begin the customization process, navigate to the VB Editor interface. Once there, locate any gray area on the existing toolbar (typically found near the top of the window, beneath the main menu) and perform a right-click action. This context menu will present several options, including the crucial **Customize** command, which serves as the gateway to modifying the environment's command settings.

Select the **Customize** option from the menu. This action opens the primary customization dialog box, where you can manage toolbars, menus, and commands. This dialog box is where we will locate and expose the specific block commenting command that is necessary for our desired functionality. This step is foundational to integrating the advanced feature into your daily coding workflow.

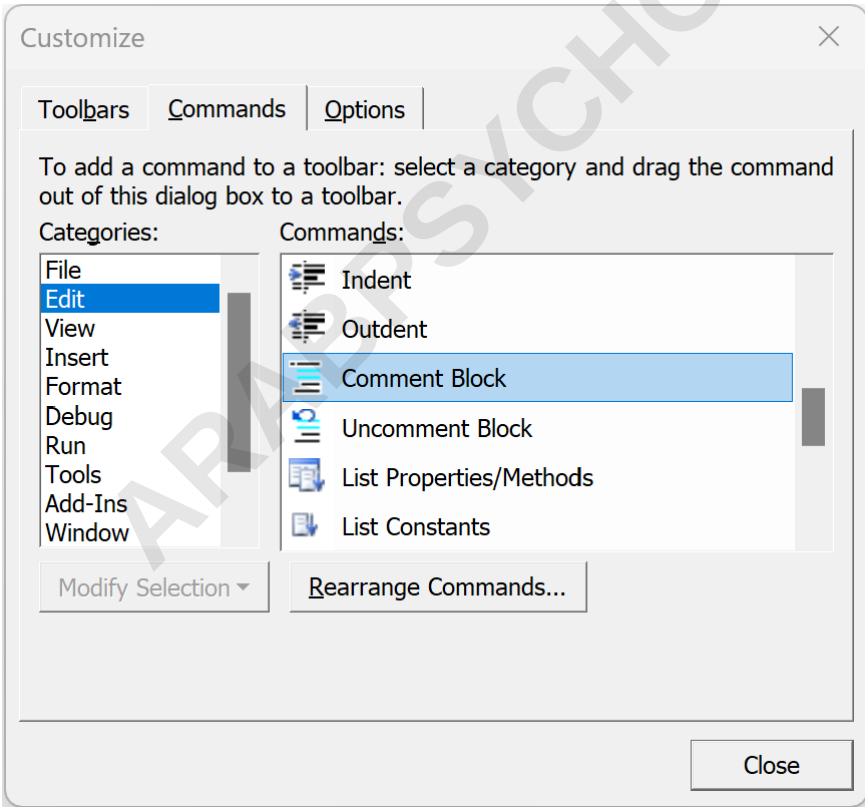


Introducing the Comment Block Feature (Step 2)

Once the **Customize dialog box** is open, the next task is to locate the specific command that handles block commenting. This command is typically nested within the list of editing functions, as it directly manipulates the textual structure of the code. Transition to the **Commands tab** within the dialog box to view all available, but perhaps hidden, commands that can be added to the IDE's interface.

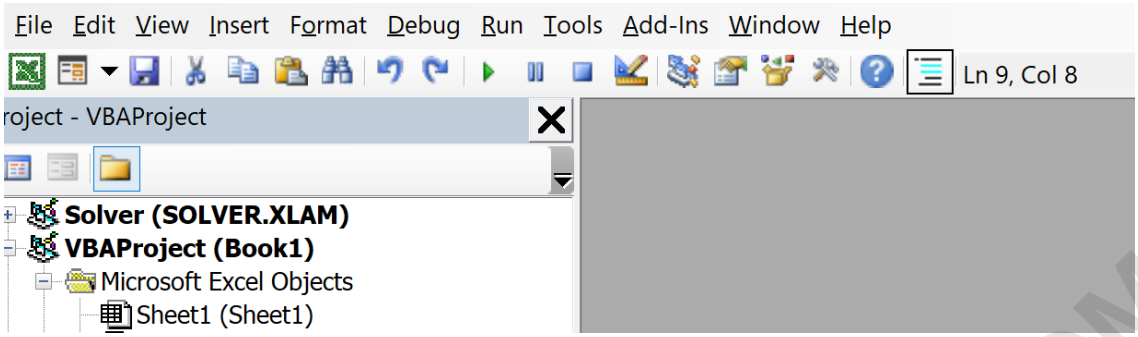
Within the **Commands tab**, observe the pane labeled **Categories**. Scroll through this list until you locate and click on the **Edit** category. Selecting the **Edit** category filters the list of commands shown in the adjacent **Commands pane**, revealing all actions related to code manipulation, searching, replacing, and, crucially, commenting.

In the **Commands pane**, scroll down until you find the **Comment Block** command. This is the exact function we need to enable multi-line commenting. Once located, click and drag the **Comment Block** option directly onto one of the existing visible toolbars in the VB Editor window. A new icon, typically representing lines of code with a comment symbol, will instantly appear on the toolbar, confirming the successful addition of the function.



After dragging and dropping the command, the toolbar should visually update, showing the new icon. This icon provides a clickable interface for block commenting, but to achieve maximum

efficiency, we must proceed to the next step: assigning a keyboard shortcut key.

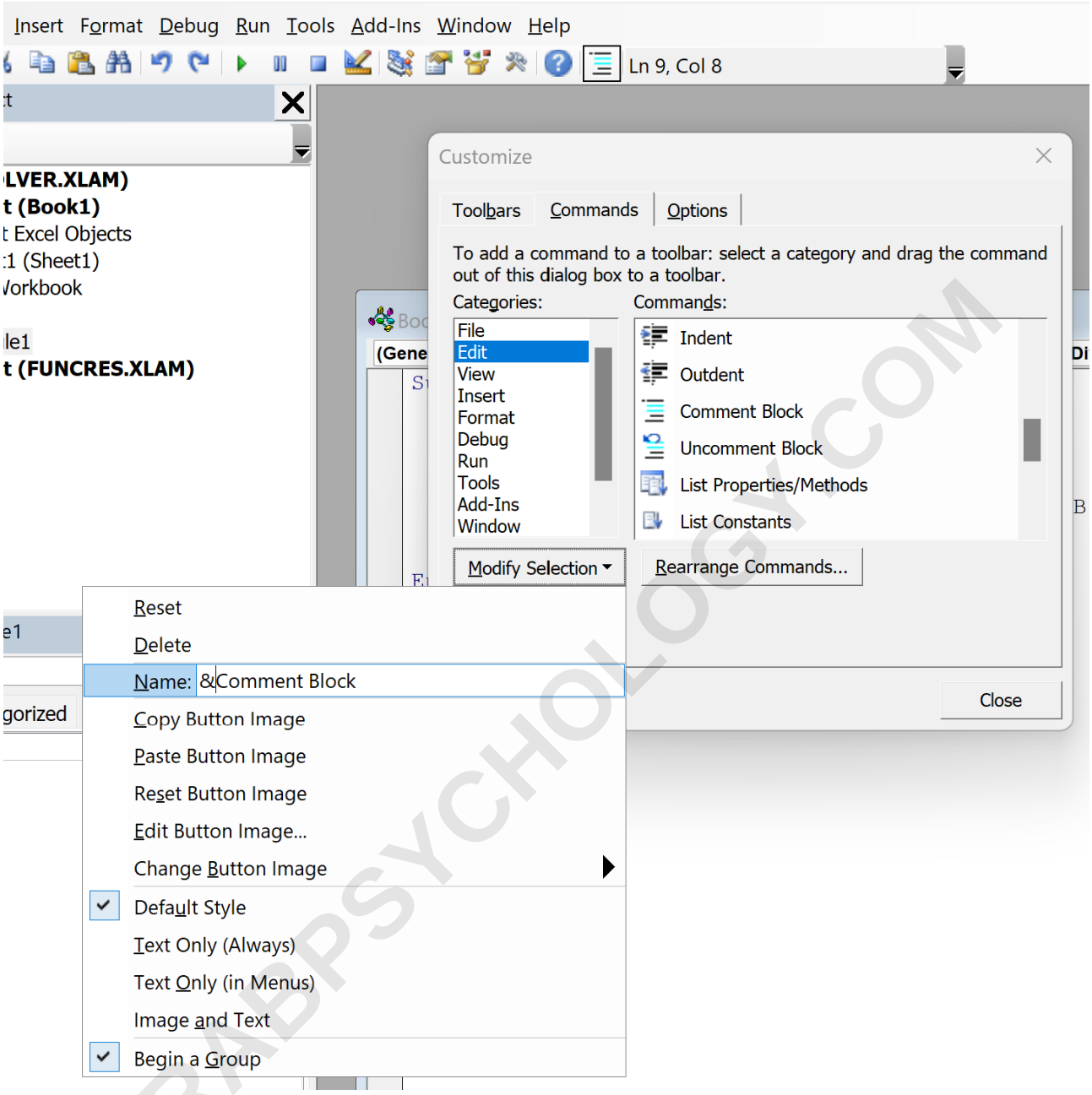


Assigning a Custom Shortcut Key (Step 3)

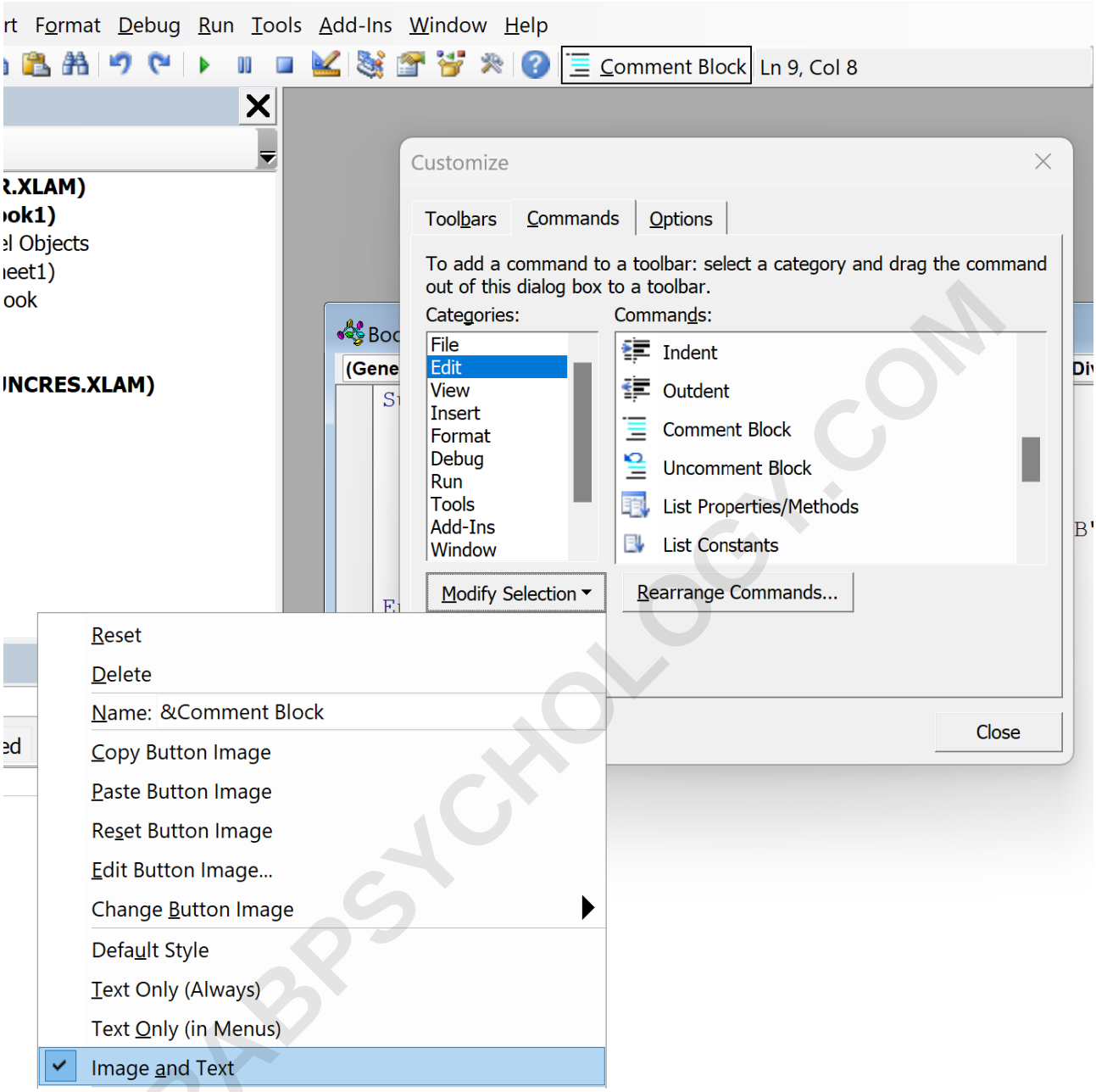
While having the Comment Block icon on the toolbar is functional, the true power of this customization lies in assigning an easily memorable keyboard shortcut key. VBA allows the use of the **Alt key** combined with a letter to create custom access keys for toolbar functions. The letter chosen will be the one immediately following an ampersand (&) symbol in the command's name field.

Ensure the **Customize dialog box** is still open. Click directly on the newly added **Comment Block icon** on the toolbar. A black border will momentarily appear around the icon, indicating it is selected for modification. Next, click the **Modify Selection dropdown** button located within the Customize panel.

Within the options presented, locate the **Name field**. The current name should read "Comment Block." To define the shortcut, insert an ampersand (&) symbol immediately preceding the letter you wish to designate as the shortcut key. For convention, we will use 'C' for Comment, resulting in the new name: **&Comment Block**. Type this modified name into the field and press **Enter** to confirm the change.



After defining the shortcut key via the ampersand, you may notice the text label associated with the icon disappears. To restore the icon and ensure the shortcut remains active, click the **Modify Selection dropdown** again. From the menu, select **Image and Text**. This finalizes the customization, ensuring the icon is visible while activating the Alt + C shortcut for the function. Close the Customize dialog box to return to the active coding environment.



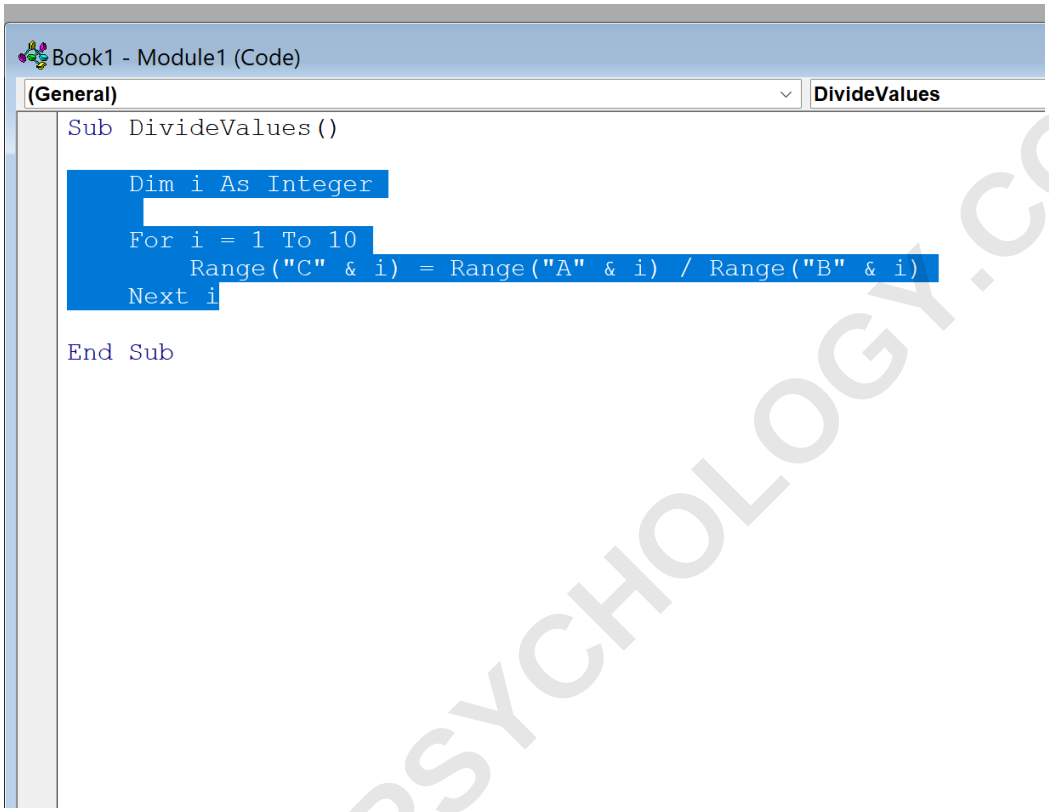
Implementing the Block Comment Shortcut (Step 4)

With the custom shortcut key now fully configured, the process of commenting out a large block of code is dramatically simplified. This new functionality is immediately available across all your VBA projects managed within the current installation of the VB Editor. To utilize this feature, the procedure is intuitive and fast, greatly enhancing the efficiency of testing and iterative development.

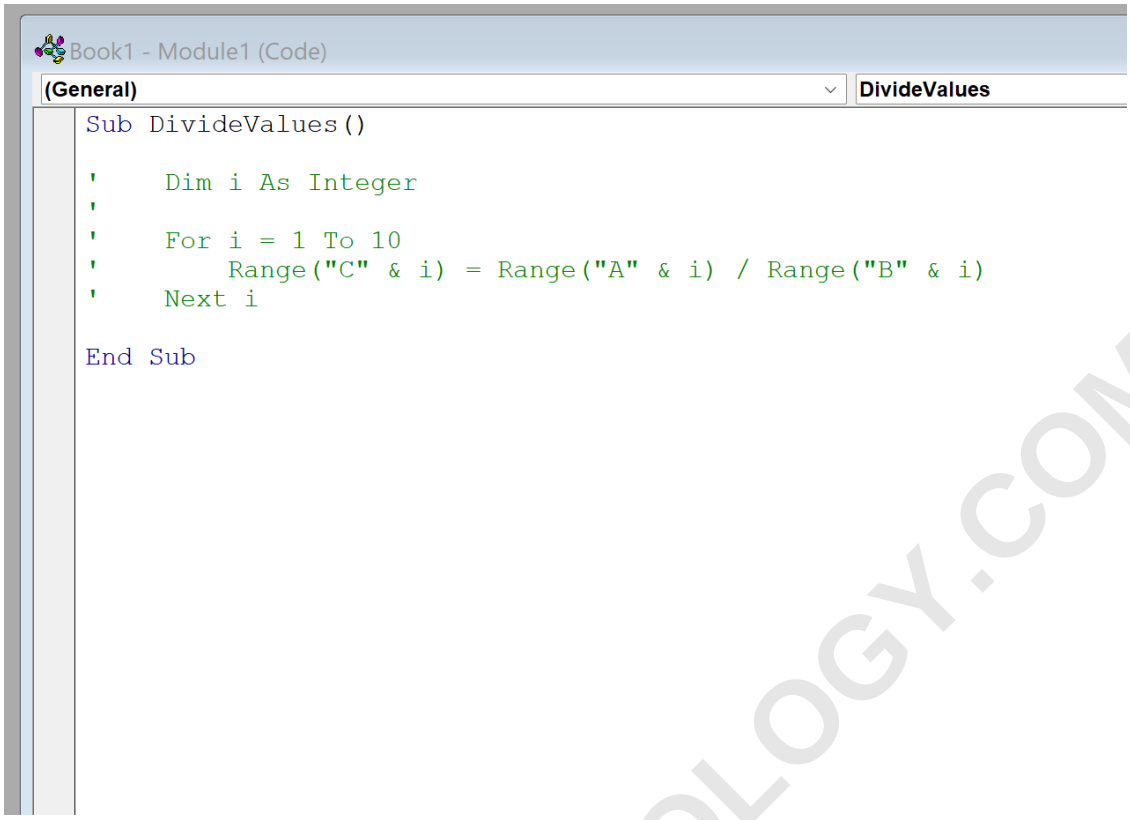
Consider a scenario where a complex calculation or a series of loop operations defined within a Sub procedure needs to be temporarily deactivated for testing purposes. Before this customization,

the developer would have to manually prefix each line with an apostrophe. Now, the process requires only selection and execution of the shortcut.

First, use your mouse to select and highlight the entire section of code that you intend to comment out. This selection can span across multiple lines, including variable declarations, conditional statements, and loops. For example, if we have the following code:



Once the desired block of code is highlighted, simply press the configured keyboard shortcut: **Alt + C**. The VB Editor instantly iterates through the highlighted lines, inserting the single quotation mark (') comment prefix at the beginning of every selected line. This action effectively transforms the entire block into non-executable documentation, ready for testing or temporary disabling.



This method ensures that large procedural sections, complex calculations, or entire test cases can be disabled and enabled at will, significantly accelerating the iterative development and debugging cycles, which are common in real-world VBA projects.

Extending Functionality: Uncommenting Code Blocks

The ability to comment out a block of code efficiently is only half the solution; equally important is the ability to easily restore that code to its executable state. Fortunately, the Visual Basic Editor contains a complementary built-in command called **Uncomment Block**, which can be configured using the exact same procedure outlined above.

To implement the uncommenting functionality, repeat Steps 1 through 3, but substitute **Comment Block** with **Uncomment Block**. When assigning the keyboard shortcut key, it is highly recommended to choose a logical, adjacent key. A common convention is to use 'U' for Uncomment. This involves renaming the command to **&Uncomment Block** in the Customize panel.

Once configured, selecting a commented block of code and pressing **Alt + U** will instantly remove the apostrophe prefix from every line, restoring the code to an active, executable state. Having both the Comment Block (Alt + C) and Uncomment Block (Alt + U) shortcuts fully integrated provides a complete and powerful solution for managing large code sections efficiently.

Best Practices for Effective VBA Commenting

While the mechanical ability to comment blocks of code is important, the true benefit comes from applying smart, disciplined commenting practices. Comments should explain the **intent and assumptions** behind the code, not simply restate what the code is doing. For example, instead of commenting `'Adds 1 to x'`, which is obvious, a better comment would be `'Increment x to handle zero-based index for worksheet array.'`

Always document the purpose of every major Sub procedure and function, including explanations for input parameters and expected output or side effects. If a piece of code is complex, reference any external formulas, business rules, or sources used to derive the logic. This makes the code self-sufficient and easier for future developers (or your future self) to understand.

Finally, utilize the block commenting feature primarily for temporary disabling of test code or major sections currently under reconstruction. Avoid using large blocks of commented-out code as permanent storage for old logic; this is best handled by proper version control systems. Keeping your comments clear, concise, and focused on underlying logic ensures your VBA projects remain robust and easy to maintain.