Code Review Using a Checklist

Conduct a review of the given source code using items from the checklist below.

What the code is used for:

The code is written in Microsoft's Visual Basic for Applications and implements a macro for the Microsoft Word Office application. This is used to make it easier to record a description of a change made to the active Word document.

The macro prompts the user for a description of the change (this is called the 'change text') and appends this to a spreadsheet within a workbook that is in a folder named 'CM'. The CM folder has the same parent folder as the one in which the active document resides. So the macro uses the pathname of the active document to construct the pathname of the changes workbook.

The workbook is named 'AAA Changes.xlsx' where AAA is some string of 3 upper case letters. The same letters appear in the name of the active document so these are copied from there. A version number is also contained within the pathname of the active document so this too is extracted to identify the particular spreadsheet in the workbook that is to be updated. If the workbook does not contain a spreadsheet of the correct name, one is created.

Id	Question	Result
Structure		
1.1	Does the code completely and correctly implement the design?	
1.2	Does the code conform to any pertinent coding standards?	
1.3	Is the code well-structured, consistent in style, and consistently formatted?	
1.4	Are there any uncalled or unneeded procedures or any unreachable code?	
1.5	Are there any leftover stubs or test routines in the code?	
1.6	Can any code be replaced by calls to external reusable components or library functions?	
1.7	Are there any blocks of repeated code that could be condensed into a single procedure?	
1.8	Is storage use efficient?	
1.9	Are symbolics used rather than "magic number" constants or string constants?	
1.10	Are any modules excessively complex and should be restructured or split into multiple modules?	
Documentation		
2.1	Is the code clearly and adequately documented with an easy-to-maintain commenting style?	
2.2	Are all comments consistent with the code?	
2.3	Does the documentation conform to applicable standards?	
Variables		
3.1	Are all variables properly defined with meaningful, consistent, and clear names?	
3.2	Are there any redundant or unused variables?	
Arithmetic Operations		
4.1	Does the code avoid comparing floating-point numbers for equality?	
4.2	Does the code systematically prevent rounding errors?	

Id	Question	Result
4.3	Does the code avoid additions and subtractions on numbers with greatly different magnitudes?	
4.4	Are divisors tested for zero or noise?	
Loops and Branches		
5.1	Are all loops, branches, and logic constructs complete, correct, and properly nested?	
5.2	Are the most common cases tested first in IF-ELSEIF chains?	
5.3	Are all cases covered in an IF-ELSEIF or CASE block, including ELSE or DEFAULT clauses?	
5.4	Does every case statement have a default?	
5.5	Are loop termination conditions obvious and invariably achievable?	
5.6	Are indices or subscripts properly initialized, just prior to the loop?	
5.7	Are all statements that are enclosed within loops correctly within the loop?	
5.8	Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?	
Defensive Programming		
6.1	Are indices, pointers, and subscripts tested against array, record, or file bounds?	
6.2	Are imported data and input arguments tested for validity and completeness?	
6.3	Are all output variables assigned?	
6.4	Is the correct data element operated on in each statement?	
6.5	Is every memory allocation released?	
6.6	Are timeouts or error traps used for external device access?	
6.7	Are files checked for existence before attempting to access them?	
6.8	Are all files and devices left in the correct state upon program termination?	

The result column in the table below highlights the checklist items that have helped reveal an issue (this may not be all the issues!).

Id	Question	Result
Structure		
1.1	Does the code completely and correctly implement the design?	No design.
1.2	Does the code conform to any pertinent coding standards?	No standards.
1.3	Is the code well-structured, consistent in style, and consistently formatted?	Inconsistent style / formatting at 48.
1.4	Are there any uncalled or unneeded procedures or any unreachable code?	No.
1.5	Are there any leftover stubs or test routines in the code?	Debug.Print statements at lines 28 and 158.
1.6	Can any code be replaced by calls to external reusable components or library functions?	No.
1.7	Are there any blocks of repeated code that could be condensed into a single procedure?	No.
1.8	Is storage use efficient?	Not applicable.
1.9	Are symbolics used rather than “magic number” constants or string constants?	Numbers used at 41, 106, 107, 108, 205, and 215.
1.10	Are any modules excessively complex and should be restructured or split into multiple modules?	No.
Documentation		
2.1	Is the code clearly and adequately documented with an easy-to-maintain commenting style?	No. More information on the function of the sub-procedures and their interfaces is required.
2.2	Are all comments consistent with the code?	No. At line 174 inappropriate reference to ‘slash character’.
2.3	Does the documentation conform to applicable standards?	No standards.
Variables		
3.1	Are all variables properly defined with meaningful, consistent, and clear names?	At 144 ‘LeftPart’ first used but not defined. At 11 ‘Home’ – what does this mean?.
3.2	Are there any redundant or unused variables?	Unused variables defined at 12 and 55.
Arithmetic Operations		
4.1	Does the code avoid comparing floating-point numbers for equality?	None.
4.2	Does the code systematically prevent rounding errors?	Not applicable.

Id	Question	Result
4.3	Does the code avoid additions and subtractions on numbers with greatly different magnitudes?	Not applicable.
4.4	Are divisors tested for zero or noise?	Not applicable.
Loops and Branches		
5.1	Are all loops, branches, and logic constructs complete, correct, and properly nested?	Yes.
5.2	Are the most common cases tested first in IF-ELSEIF chains?	Not applicable.
5.3	Are all cases covered in an IF-ELSEIF or CASE block, including ELSE or DEFAULT clauses?	Not applicable.
5.4	Does every case statement have a default?	Not applicable.
5.5	Are loop termination conditions obvious and invariably achievable?	Yes.
5.6	Are indices or subscripts properly initialized, just prior to the loop?	Yes.
5.7	Are all statements that are enclosed within loops correctly within the loop?	Yes.
5.8	Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?	Index correctly updated within loop at 178.
Defensive Programming		
6.1	Are indices, pointers, and subscripts tested against array, record, or file bounds?	Yes.
6.2	Are imported data and input arguments tested for validity and completeness?	No, sub-procedure at 118 assumes active document pathname has a specific syntax and elsewhere (at 64) the code assumes Changes spreadsheet always exists.
6.3	Are all output variables assigned?	Yes.
6.4	Is the correct data element operated on in each statement?	Yes.
6.5	Is every memory allocation released?	Not applicable.
6.6	Are timeouts or error traps used for external device access?	Not applicable.
6.7	Are files checked for existence before attempting to access them?	No, at 64.
6.8	Are all files and devices left in the correct state upon program termination?	Only if successful – risk of leaving Excel process running and workbook open if an error occurs within sub-procedure AddUpdateToChangesFile at 48.

