Generic Checklist for Code Reviews

Structure

✓ Does the code completely and correctly implement the design?
❏ Does the code conform to any pertinent coding standards?
✓ Is the code well-structured, consistent in style, and consistently formatted?
❏ Are there any uncalled or unneeded procedures or any unreachable code?
❏ Are there any leftover stubs or test routines in the code?
❏ Can any code be replaced by calls to external reusable components or library functions?
❏ Are there any blocks of repeated code that could be condensed into a single procedure?
✓ Is storage use efficient?
✓ Are symbolics used rather than "magic number" constants or string constants?
❏ Are any modules excessively complex and should be restructured or split into multiple routines?

Documentation

✓ Is the code clearly and adequately documented with an easy-to-maintain commenting style?
❏ Are all comments consistent with the code?

Variables

✓ Are all variables properly defined with meaningful, consistent, and clear names?
✓ Do all assigned variables have proper type consistency or casting?
❏ Are there any redundant or unused variables?

Arithmetic Operations

✓ Does the code avoid comparing floating-point numbers for equality?
✓ Does the code systematically prevent rounding errors?
✓ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?
❏ Are divisors tested for zero or noise?

Loops and Branches

✓ Are all loops, branches, and logic constructs complete, correct, and properly nested?
✓ Are the most common cases tested first in IF- -ELSEIF chains?
✓ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?
❏ Does every case statement have a default?
✓ Are loop termination conditions obvious and invariably achievable?
✓ Are indexes or subscripts properly initialized, just prior to the loop?
❏ Can any statements that are enclosed within loops be placed outside the loops?
✓ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Defensive Programming

✓ Are indexes, pointers, and subscripts tested against array, record, or file bounds?
✓ Are imported data and input arguments tested for validity and completeness?
✓ Are all output variables assigned?
✓ Are the correct data operated on in each statement?
✓ Is every memory allocation deallocated?

❑ Are timeouts or error traps used for external device accesses?
✓ Are files checked for existence before attempting to access them?
✓ Are all files and devices are left in the correct state upon program termination?