Code - General Issues

This checklist covers issues that relate to any type of coding activity. See the more specialized C*xCheck_Code-Xxx* checklists for further detail on specific areas.

Before You Begin

CDG-1 Do you understand the design you are about to construct?

CDG-2 Does the design provide an appropriate level of detail for coding to begin?

CDG-3 Is the design to be implemented straightforward and feasible, or should it be revisited before attempting to construct it?

CDG-4 Do you understand the language and technology well enough to implement the design?

General

CDG-5 Is the code written in terms of the problem domain as much as possible rather than in terms of computer-science or programming-language structures?

CDG-6 Is the code traceable to upstream design and requirements artifacts?

CDG-7 Does the code have documented test cases and/or unit tests as defined by the project?

CDG-8 Does the code adhere to *CxStand_Code* or other designated coding standard?

CDG-9 Does the code adhere to designated coding styles and / or templates?

CDG-10 Does the code compile with no warnings from the compiler?

Understandability

CDG-11 Does the code read from top to bottom?

CDG-12 Are implementation details hidden as much as possible?

CDG-13 Is the code straightforward and does it avoid "cleverness"?

CDG-14 Has tricky code been rewritten rather than commented?

CDG-15 Do you thoroughly understand your code?

CDG-16 Is it easy to understand?

Performance

CDG-17 Are the affects of resource constraints on the technology and operational environment understood and within the bounds of the performance requirements?

CDG-18 Are the affects of system load on the technology and operational environment understood and within the bounds of the performance requirements?

CDG-19 Is the expected priority of efficient or highly optimized code clear for various areas of the system?

CDG-20 Has profiling support been planned for areas of the code at risk for performance issues?

Assertions and Tracing

CDG-21 Are assertions used to document assumptions?

CDG-22 Are assertions used to aid debugging?

CDG-23 Are tracing statements used to document events and aid debugging?

CDG-24 Assertions are not being used to handle errors that should be handled in the code?

Error Handling

CDG-25 Is it easy to differentiate between nominal path processing and error processing?

CDG-26 Are error conditions handled appropriately as per the requirements and design? (i.e., the level of robustness in detecting and responding to errors matches the needs of the system: not too little and not too much).

CDG-27 When attempting recovery from error conditions, are assumptions reasonable?

Code Changes

CDG-28 Is the change part of a systematic change strategy?

CDG-29 Has the change been reviewed as thoroughly as initial development would be?

CDG-30 Does the change enhance the program's internal quality rather than degrading it?

CDG-31 Have you improved the system's modularity by breaking routines into smaller routines, when possible?

CDG-32 Have you improved the programming style--variable names, routine names, formatting, comments, and so on?

CDG-33 If changes cause you to look for ways to share code, have you considered putting the shared code at a higher level as well as considered putting it at a lower level?

CDG-34 Does this change make the next change easier?

Layout

CDG-35 Does the program's layout show its logical structure?

CDG-36 Is formatting done primarily to illuminate the logical structure of the code?

CDG-37 Is the formatting scheme used consistently?

CDG-38 Are related statements grouped together?

CDG-39 Are blank lines used to separate code elements including functions, control sequences, related blocks of code, etc?

CDG-40 Does the formatting scheme result in code that's easy to maintain?

CDG-41 Does the formatting scheme improve code readability?

CDG-42 Have relatively independent groups of statements been moved into their own routines?

CDG-43 Are references to variables as close together as possible, both in total live time and from each reference to a variable to the next?

Individual Statements

CDG-44 Are continuation lines indented sensibly?

CDG-45 Are groups of related statements aligned?

CDG-46 Are groups of unrelated statements unaligned?

CDG-47 Does each line contain one statement?

CDG-48 Is each statement written without side effects?

CDG-49 Are data declarations aligned?

CDG-50 Is there one data declaration per line?

Self-Documenting Code

Routines

CDG-51 Does each routine's name describe exactly what it does?

CDG-52 Does each routine perform one well-defined task?

CDG-53 Is each routine's interface obvious and clear?

Data Names

CDG-54 Are names of data types descriptive enough to help document data declarations? Are they used specifically for that purpose?

CDG-55 Are variables named well?

CDG-56 Are variables used only for the purpose for which they're named?

CDG-57 Are well-named enumerated types used instead of makeshift flags or boolean variables?

CDG-58 Are named constants used instead of magic numbers or magic strings?

Data Organization

CDG-59 Are extra variables used for clarity when needed?

CDG-60 Are references to variables close together?

CDG-61 Are data structures simple so that they minimize complexity?

CDG-62 Is complicated data accessed through abstract access routines (abstract data types)?

Control

CDG-63 Are related statements grouped together?

CDG-64 Have relatively independent groups of statements been packaged into their own routines?

CDG-65 Does the normal case follow the if rather than the else?

CDG-66 Are control structures simple so that they minimize complexity?

CDG-67 Does each loop perform one and only one function, like a well-defined routine?

CDG-68 Is nesting minimized?

CDG-69 Have boolean expressions been simplified by using additional boolean variables, boolean functions, and decision tables?

Dependencies

CDG-70 Does the code make dependencies among statements obvious?

CDG-71 Do the name and parameters of routines make dependencies obvious?

CDG-72 Do comments describe any dependencies that would otherwise be unclear?

Good Commenting Technique

CDG-73 Are comments up to date, clear, and correct?

CDG-74 Can someone pick up the code and immediately start understanding it?

CDG-75 Does the source listing contain enough information to understand the program?

CDG-76 Do comments explain the code's intent or summarize it, rather than just repeating it? (i.e., the why rather than the how)

CDG-77 Is the PDL-to-code process used?

CDG-78 Is the distinction between major and minor comments clear?

CDG-79 Are the comments indented the same as the code?

CDG-80 Is the commenting style easy to maintain and allow for easy modification of comments?

CDG-81 Does the code avoid endline comments?

CDG-82 Do comments prepare the reader's mind for what is to follow?

CDG-83 Does every comment count? (i.e., have redundant, extraneous, or self-indulgent comments been removed or improved?)

CDG-84 Is code that works around an error or uses an undocumented feature commented?

CDG-85 Is each control statement commented?

CDG-86 Are the ends of long or complex control structures commented?

Data Declarations

CDG-87 Are units on data declarations commented?

CDG-88 Is the range of values on numeric data commented?

CDG-89 Are coded meanings commented?

CDG-90 Are limitations on input data commented?

CDG-91 Are flags documented to the bit level?

CDG-92 Has each global variable been commented where it is declared and where it is used?