Design Inspection Checklist

General Requirements and Design

- [ ] Has review of the design identified problems with the requirements, such as missing requirements, ambiguous requirements, extraneous requirements, untestable requirements, or implied requirements?
- [ ] Is the design consistent with the requirements? For example, are there: missing functions, extraneous functions, imprecise, ambiguous, or incorrect functions
- [ ] Are deviations from the requirements documented and approved?
- [ ] Are all assumptions documented?
- [ ] Have major design decisions been documented?
- [ ] Is the design consistent with these decisions?
- [ ] Does the design adequately address the following: real-time requirements, performance issues (memory and timing), spare capacity (CPU and memory), maintainability, understandability, database requirements, loading and initialization, error handling and recovery, user interface issues, software upgrades

Functional and Interface Specifications

- [ ] Is the P-spec for each process accurate and complete?
- [ ] Is it specified in precise, unambiguous terms? Does it clearly describe the required transformations?
- [ ] Are dependencies on other functions, Operating system kernel, hardware, etc., identified and documented?
- [ ] Are human factors considerations properly addressed in those functions that provide the user interface?
- [ ] Are design constraints, such as memory and timing budgets, specified where appropriate?

- [ ] Are requirements for error checking, error handling and recovery specified where needed?
- [ ] Are interfaces consistent with module usage? Missing interfaces? Extra interfaces?
- [ ] Are the interfaces specified to a sufficient level of detail that allows them to be verified?

Conventions

- [ ] Does the design follow the established notation conventions?

Requirements Traceability

- [ ] Does the detailed design of this module or interface fulfill its part of the requirements?
- [ ] Has the inspection of this module or interface identified problems in the SRS? For example, missing requirements, ambiguous requirements, conflicting requirements, untestable requirements, implied requirements?
- [ ] Does the detailed design of this module or interface meet its high level design requirements?
- [ ] Has the inspection of the detailed design identified problems in the high level design?
- [ ] Are all functions completely and accurately described in sufficient detail?
- [ ] Are all interfaces completely and accurately described, including keyword or positional parameters, field descriptors, attributes, ranges, and limits?
- [ ] Are the detailed design documents complete and consistent within themselves; data with logic; all internal data defined; no extraneous data?

Structure and Interfaces

- [ ] At a system and subsystem level, have all components or modules been identified on a System Architecture Model?
- [ ] Is the level of decomposition sufficient to identify all modules?
- [ ] Will further decomposition result in identifying more modules?
- [ ] Have all interfaces between system/subsystem elements and modules been clearly and precisely identified?
- [ ] Do successive levels of decomposition result in successive levels of detail?
- [ ] Are modules performing more than one specific function?

Logic

- [ ] Are there logic errors?
- [ ] Are... all unique values tested?, all positional values tested?, increment and loop counters properly initialized?, variables and data areas initialized before use?
- [ ] Has the module been inspected for... correct begin and end of table processing?, correct processing of queues across interrupts?, correct decision table logic?, correct precision/accuracy of calculations?
- [ ] Are message priorities allocated properly to ensure the correct execution of code?

- [ ] Is the message processing sequence correct?
- [ ] Are there errors in handling data, data buffers, or tables, incorrect field updated, conflicting use of data areas, incomplete initialization or update, inconsistent or invalid data attributes?
- [ ] Are procedure call and return interfaces correctly defined; Call and return parameters defined correctly; Correct syntax?

Performance

- [ ] Are memory and timing budgets reasonable and achievable?

Error Handling and Recovery

- [ ] Is there adequate error condition testing?
- [ ] Are error conditions tested where the probability of an error is high or results of an error would be fatal to the system?
- [ ] Are return codes documented?
- [ ] Are return messages understandable?
- [ ] Does the program allow for successful error recovery... across module or process failures?, across operating system failure?, across interrupts?, across hardware failures?

Testability, Extensibility

- [ ] Is the design... understandable (i.e., easy to read, follow logic)?, maintainable (i.e., no obscure logic...)?, testable

Coupling and Cohesion

- [ ] Evaluate the design using the standard coupling and cohesion criteria, if appropriate.