*BY ORDER OF THE*
*SECRETARY OF THE AIR FORCE*

*AIR FORCE MANUAL 91-119*

*5 JUNE 2012*

*Safety*

# *SAFETY DESIGN AND EVALUATION CRITERIA FOR NUCLEAR WEAPON SYSTEMS SOFTWARE*

**COMPLIANCE WITH THIS PUBLICATION IS MANDATORY**

**ACCESSIBILITY:** Publications and forms are available for downloading or ordering on the e-Publishing website at **www.e-Publishing.af.mil**

**RELEASABILITY:** There are no releasability restrictions on this publication

OPR: HQ AFSEC/SEWN

Certified by: AF/SE (Maj Gen Feest)

Supersedes: AFMAN 91-119, 1 February 1999

Pages: 30

This manual implements AFPD 91-1, *Nuclear Weapons and System Surety*, and contains the minimum design and evaluation criteria for software requiring nuclear safety certification. It applies to all organizations that design, develop, modify, evaluate, operate or acquire a nuclear weapon system. This publication is consistent with AFPD 13-5, *Air Force Nuclear Enterprise*. This Manual is applicable to Air National Guard and Air Force Reserve units performing nuclear missions. This manual applies to new systems or modified portions of existing systems. Existing certified systems are not required to be modified solely to meet the requirements of this manual. Refer recommended changes and questions about this publication to the Office of Primary Responsibility (OPR) using the AF Form 847, *Recommendation for Change of Publication*; route AF Form 847s from the field through the appropriate (MAJCOM) publications/forms manager. Ensure that all records created as a result of processes prescribed in this publication are maintained in accordance with AFMAN 33-363, *Management of Records*, and disposed of in accordance with the Air Force Records Disposition Schedule (RDS) located at **https://www.my.af.mil/afrims/afrims/afrims/rims.cfm**. Send recommendations for improvements to Headquarters Air Force Safety Center (AFSEC/SEWN), 9700 G Avenue SE, Kirtland AFB, NM 87117-5670, or email **HQAFSCSEWN@kirtland.af.mil**

## *SUMMARY OF CHANGES*

**This document is substantially revised and shall be completely reviewed.** This revision includes substantive changes. It provides nuclear safety design certification and evaluation criteria for software systems, including facilities, used to support, maintain, handle or store nuclear weapons. In addition, organization names were changed to reflect changes since the last

publication of this Manual. Changes include certification of Field Programmable Gate Arrays (FPGAs), Application Specific Integrated Circuit (ASIC) designs, Complex Programmable Logic Devices (CPLD) and software that is unique and developed specifically to control essential facility systems outlined in AFMAN 91-118. In addition, organization names were changed to reflect changes since the last publication of this Manual.

# Chapter 1

## GENERAL STANDARDS AND CONTROL

### *Section 1A—Responsibilities and Scope*

**1.1. Philosophy.** The goal of this manual is to provide integration of nuclear safety into nuclear weapon system software development and sustainment processes and disciplines. It is not intended to encourage an isolated view of software surety independent of hardware development (systems view is essential). This manual addresses software consisting of sequences of machine instructions or code interpreted by such instructions, whether the memory it resides in is volatile or nonvolatile (e.g., firmware). This manual addresses nuclear weapon system test software and Test Program Sets used in nuclear weapon system testing. This manual addresses certification of Field Programmable Gate Arrays (FPGAs), Application Specific Integrated Circuit (ASIC) designs and Complex Programmable Logic Devices (CPLD). Although these are hardware technologies, they share many of the same development techniques and entail the same threats traditionally associated with software. The certification requirements of this document apply to the end product derived from any technology involving automatic translation from a design language to an operational format.

**1.2. Department of Defense (DOD) Safety Standards.** The DOD Nuclear Weapon System Safety Standards form the basis for the safety design and evaluation criteria for nuclear weapon systems. The DOD Nuclear Weapon System Safety Standards state that:

1.2.1. There shall be positive measures to prevent nuclear weapons involved in accidents or incidents or jettisoned weapons from producing a nuclear yield.

1.2.2. There shall be positive measures to prevent DELIBERATE prearming, arming, launching or releasing of nuclear weapons, except upon execution of emergency war orders or when directed by competent authority.

1.2.3. There shall be positive measures to prevent INADVERTENT prearming, arming, launching or releasing of nuclear weapons in all normal and credible abnormal environments.

1.2.4. There shall be positive measures to ensure adequate security of nuclear weapons, under DOD Directive O-5210.41, *Security Policy for Protecting Nuclear Weapons*.

**1.3. Development Standards.** Nuclear systems shall comply with a software development standard to ensure a systematic process is used that will increase confidence for successful nuclear certification.

1.3.1. New systems shall comply with IEEE/EIA 12207, *Systems and Software Engineering-Software Life Cycle Processes*.

1.3.2. Modification of existing systems shall comply with at a minimum with the standard it was developed under (e.g. DOD-STD-2167A, *Defense System Software Development*, MIL-STD-498, *Software Development and Documentation,* EIA/IEEE J-STD-016, *Standard for Information Technology Software Life Cycle Processes, Software Development Acquirer-Supplier Agreement,* IEEE/EIA 12207)

**1.4. National Security Agency (NSA) Certification.** Codes, coding procedures, software encryption and decryption material with codes and encryption and decryption programs produced by the NSA receive certification equivalent to the Air Force (AF) process. Therefore, no additional testing is required for design certification of software or firmware certified by NSA.

**1.5. Department of Energy (DOE) Certification.** DOE certified software for use in DOE provided equipment shall not require AF design certification, providing the operating environments are identical to which it was certified.

**1.6. Air Force Criteria.** To comply with the DOD safety standards, the Air Force has implemented a set of minimum design and evaluation criteria for Air Force nuclear weapon systems software. Since the criteria in this manual are not design solutions and are not intended to restrict the designer in the methods and techniques used to meet operational design requirements, they are not all-inclusive. Air Force nuclear weapon system software designers may add feasible and reasonable safety features as needed. These nuclear surety software criteria apply equally to software, ASICs, and firmware associated with FPGA designs.

**1.7. Nuclear Safety Software Categories.** Nuclear certified software and other complex mechanisms (e.g. FPGAs) with nuclear safety implications are designated by the Nuclear Weapons System Safety Group (NWSSG) or HQ AFSEC/SEWN into one of four categories: Category I, Category II, Category III and Category IV. Software in Categories I and II shall be separately safety-certified and listed in the Master Nuclear Certification List (MNCL). Software in Category III may or may not be separately safety certified; and may or may not be subjected to Independent Verification & Validation (IV&V). Software in category IV shall be separately safety certified and shall be included with the facility listing in the MNCL. HQ AFSEC/SEWN provides guidance for the scope of the certification process per AFI 63-125, *Nuclear Certification Program*. Additionally, Discrepancy Reports (DRs) shall be generated during Nuclear Safety Cross-Check Analysis (NSCCA) or IV&V for any errors or discrepancies detected in program design, code or documentation. Each nuclear safety DR shall be recorded and tracked in a discrepancy log that identifies the DR and gives its status. The priority scheme is shown in **Attachment 3**. Nuclear Safety Objectives (NSOs) represent the overall objectives a nuclear weapon system must satisfy in order to obtain nuclear safety design certification. A list of NSOs for each nuclear weapon system or modification program shall be prepared. A generic NSO list is shown in **Attachment 2**. All appropriate NSOs shall be satisfied in the design and verified by the IV&V or NSCCA.

1.7.1. Category I software is software that controls or implements critical function(s) and has been designated by the NWSSG or HQ AFSEC/SEWN as a critical component. Software shall be separately safety-certified and listed in the MNCL. An NSCCA is required. (Note: Software may be designated as Category I for other reasons; e.g., the software is responsible for the primary security of a nuclear weapon at a remote launch point, the software processes clear text command and control data, etc.)

1.7.2. Category II software is software that controls critical function(s), but is not designated as a critical component. Software shall be separately safety-certified and listed in the MNCL. An IV&V is required.

1.7.3. Category III software is software that does not control critical function(s), but interfaces with hardware/software that does control critical function(s). Category III

software may or may not be subjected to an IV&V, as determined by HQ AFSEC/SEWN. Category III software includes, but is not limited to, mission planning system software and software used to monitor, test or maintain weapon system interfaces when the warhead or bomb is not mated to or loaded on the delivery system. Category III software also includes software designed for data handling/transfer actions where no recognition, interpretation or decisions are required based on the contents of the data. Examples are software for operation of MIL-STD-1553 bus controller and remote terminal interfaces. If IV&V is required, NSOs must be prepared and satisfied.

1.7.4. Category IV software is software that is unique and developed specifically to control essential facility systems outlined in AFMAN 91-118, *Safety Design and Evaluation Criteria for Nuclear Weapon Systems*, paragraph 2.44 (e.g. Blast Containment Management System (BCMS) software at Kirtland Underground Munitions Maintenance Storage Complex). Category IV software shall be subjected to an IV&V.

**1.8. Development Process.** The designer of nuclear weapon system software shall have a process for both management and engineering activities that is documented, standardized and integrated into a standard software process for the organization.

**1.9. Requests for Deviations.** If the design of Air Force nuclear weapon system software does not meet the requirements contained in this manual, a deviation shall be obtained according to the requirements of AFI 91-107, *Design, Evaluation, Troubleshooting and Maintenance Criteria for Nuclear Weapon Systems,* Section B. Previously granted Waivers or Deviations may be included in the Nuclear Certification Impact Statement (NCIS) for continued approval.

**1.10. Software Advisory Group (SAG) or equivalent.** For any development of or modification to nuclear surety certified software, there shall be a committee chaired by the single manager or designated representative. This group should consist of major stakeholders (i.e., developers, test organizations such as NSCCA contractor, and users). The group shall meet as necessary and disposition software discrepancies.

**1.11 Use of Media Involving Complex Formatting and Data Reconstruction.** Commercial media offer an attractive way to distribute software and data. When these systems involve complex manipulations to render reasonable reliability from an underlying unreliable physical media (such as with *Compact Disks* (CD)), or have multiple compliance levels in the defining standard, the following requirements apply.

1.11.1. Data not visible during bit-for-bit compare demonstrations and system testing shall not be visible in the operational environment. For example an NSCCA compare demonstration should be configured the same as is used operationally (This may, for example, be achieved by using the same model peripherals with the same revision of any internal firmware in tests and demonstrations as is used operationally.)

1.11.2. Software shall not use non-standard protocols to access media with critical software or critical data.

1.11.3. Category III software with access to critical software or data residing on complex media shall undergo sufficient IV&V to determine that restrictions regarding that media are complied with.

1.11.4. Application software shall not access low-level data for which all standard error correcting operations have not been accomplished.

1.11.5. Specific restrictions regarding the use of CDs are found in Attachment 4.

**1.12 Electronic Transmission of Certified Software and Critical Data.** Electronic transmission of Certified Software and Critical data are permissible under the following circumstances. The data are encrypted or digitally signed by NSA approved protocols. Verification must either be through independent channels or itself comply with NSA approved protocols.

## Chapter 2

## DESIGN CRITERIA FOR NUCLEAR WEAPON SYSTEMS SOFTWARE

### *Section 2A—General Philosophy and Criteria*

**2.1. Application.** These design safety criteria apply to:

2.1.1. Software that receive, store, process or transmit data to monitor, target, prearm, arm, launch, release or authorize the use of a nuclear weapon. Software shall be designed to provide the greatest extent of protection against accidental or deliberate unauthorized operation of nuclear critical functions as defined in AFI 91-107.

2.1.2. Software that is unique and developed specifically to control essential facility systems outlined in AFMAN 91-118, paragraph 2.44 (e.g. BCMS software at Kirtland Underground Munitions Maintenance Storage Complex (KUMMSC)).

2.1.3. FPGA designs, ASIC designs and CPLDs which store, process, release or control nuclear critical functions.

**2.2. Software Specifications.** Nuclear surety design requirements shall be incorporated into software specifications (e.g., requirements, design and product specifications).

### *Section 2B—Combat Delivery Aircraft, Ground Launched and Air Launched Missiles*

**2.3. Higher-Order Language (HOL).** Software development shall be based on an internationally recognized standard that emphasizes application and verification of safety and security. Code shall be developed in Assembly or Machine Language only when the HOL does not provide adequate capability for time-critical or hardware-interfacing functions. Use of Assembly or Machine Language shall be justified in the NCIS in accordance with AFI 63-125, paragraph 3.2.2. Otherwise, a request for deviation shall be submitted. The original language shall be used when modifying critical software.

2.3.1. Programming Language-specific Safety Restrictions. The American National Standard for Programming Languages (ANSPL) defines specific prohibitions for software languages that exhibit unacceptable safety behavior. The software developer shall comply with applicable safety prohibitions and consider industry best practices (C, and C++ are examples of languages with restrictions. C and C++ restrictions are addressed in the applicable industry standards referenced in Attachment 1.)

2.3.2. Run time dependant code restrictions. Code which has run-time dependent configuration (such as just-in-time compilers for Java) is prohibited.

**2.4. Hierarchical Design.** Software shall be designed in a hierarchical structure to reduce complexity and avoid mistakes. Computer Software Units (CSUs) performing critical functions shall be "single purpose" (i.e., one critical function per CSU). The intent of "single purpose" is to reduce complexity of code implementing specific critical functions. Each critical CSU shall have a single unique entry point and a structured (must return to where it was called from) and complete exit path (all logical paths in the CSU must lead to an exit).

**2.5. Fault Detection and Tolerance.** Software shall be designed to provide self-check, confidence or test routines to verify the integrity and proper state of hardware devices that affect or execute critical functions. Software shall be designed to detect critical function failure modes during power-up and operation. Transitory faults (such as corrupted message packets) that do not indicate degraded processing capability shall be detected and dealt with, but do not necessarily need to be reported to the operator. The system specification shall specify acceptable transitory fault rates. Troubleshooting and maintenance operations must prohibit using any nuclear weapon as a troubleshooting tool.

**2.6. Fault Response.** Critical function failure modes or attempted illegal entry into a critical function (refer to paragraph **2.17**, Critical Function Initiation and Entry Checks) shall result in operator notification with the status of any automated actions taken. Software shall be designed to revert to a known safe software state when a critical function system fault is detected. The software shall stop transmitting critical commands upon detecting faults and recycle, self-test or perform automatic shutdown. Display associated crew indications, if practical (e.g. prior to launch or release).

**2.7. Real-Time Software.** Real-time software interfacing with critical signals (e.g., Enable) shall include specification of fixed deadlines for service of events associated with nuclear critical signals and functions. The software shall implement scheduling protocols, task priorities or other techniques to ensure that such deadlines will be met under all system load and interrupt conditions (e.g., rate monotonic analysis).

**2.8. Responsiveness.** Techniques shall be provided for deadlock (waiting for a particular event that will not occur) prevention, detection or resolution.

**2.9. Interrupts.** Specific priorities and responses shall be defined for events that interrupt program execution or disable interrupts. The software shall be designed with interrupt handlers to ensure the software either continues to run or shuts down in accordance with paragraph **2.12**.

**2.10. Idle Operations.** Software shall not use a STOP or HALT instruction or cause a central processing unit (CPU) WAIT state. The unit shall always be executing as designed, whether idle or actively processing.

**2.11. Instruction Alterations.** Developed software shall be prevented from modifying its own instructions or the program instructions of other programs (i.e., no self-modifying code).

**2.12. Initialization and Shutdown.** Design measures shall be taken to ensure critical function hardware, which is controlled or monitored by software and memory containing nuclear critical information, is initialized or verified to be in a known safe state. Upon a controlled system shutdown or program termination, the software shall attempt to set all settable, nonvolatile devices and relays are set to a known safe state.

**2.13. Memory Characteristics.** These requirements can be allocated between hardware and software.

2.13.1. Program Loading and Initialization. The system shall be designed to prevent normal program execution or continuation until all program instructions or data (or both) are loaded and verified. Results of program load verification shall be displayed to system operators. All non-volatile memory shall be loaded with executable code, data or a non-use pattern. If executed as an instruction, this non-use pattern shall be detected and handled safely.

Initialization for memory shall be done at system startup or immediately upon completion of program and data loading.

2.13.2. Memory Protection. Memory protection systems shall be designed to ensure operational use does not alter or degrade memory content over time. Storage of critical function programs and data in nonvolatile and read-only memory is preferred in order to enforce the requirement that the software cannot modify its own code or data. The system shall have provisions to ensure erroneous data resulting from power interrupt, uncorrected memory system error or other phenomena, do not affect any critical function or component in a manner that causes it to go to an unknown safe state condition.

2.13.3. Memory Accessibility. Critical function (as defined in AFI 91-107, paragraph 4) routines and data elements shall be partitioned (i.e. functionally and/or physically) so that access to these areas is strictly controlled. These areas shall be controlled such that, if access is attempted by elements or routines that should not have access, execution of the critical process shall be treated as a fault (refer to paragraph **2.6**).

2.13.4. Memory Declassification. Methods to erase or obliterate, as appropriate for the memory technology, any clear-text secure codes from memory shall be provided. Use NSA approved design criteria found in DODI S-5200.16, *Objectives and Minimum Standards for Communications Security (COMSEC) Measures Used in Nuclear Command and Control (NC2) Communications*.

**2.13.5. FPGAs and CPLDs.** FPGAs and CPLDs shall not themselves be construed as memory devices for the purposes of paragraph **2.13**. In the event that a processor is instantiated within an FPGA or CPLD, the provisions of **2.13** shall be applied to any associated memory.

**2.14. Validity Checks and Special Code Restrictions.** Initiation of a critical function (as defined in AFI 91-107) shall originate with manual operator inputs (refer to paragraph **2.16**). Before performing a critical function the system shall verify the state of all applicable preconditions and inhibits. The system shall not have the information that defines the prearm unique signal pattern within the software that contains the routine(s) for generating the unique signal.

**2.14.1. Unique Signal Restrictions.** The prearm unique signal shall not be stored in a directly useable form such as a numerical sequence stored on a portable thumb drive. The prearm unique signal shall only be assembled as a result of a crew member action, such as entering a series of digits.

**2.14.2. Authorization and Enable Code Restrictions.** The system shall not attempt to repair authorization codes (eg. PAL codes) or enable codes that fail.

**2.15. Operating System (OS) and Run-Time-Executive (RTE) Development and Procurement.**

2.15.1. The OS or RTE shall be developed in accordance with a published standard for operating systems (e.g., POSIX, ISO) or procured from a commercial source.

**2.15.2. The design of the OS shall have a measureable metric to justify confidence in the design.**

2.15.3. Commercial off-the-shelf (COTS) Procurement Criteria. The procured OS or RTE shall reliably perform its necessary functions. Indicators of reliability include maturity (it has

been commercially available for sufficient time to allow thorough identification and correction of design errors), maintainability (a user group exists and/or the manufacturer has demonstrated adequate response to comments from its user community) and stability (based on metrics or an independent evaluation).

2.15.4. OS and RTE Operating Constraints. The OS or RTE shall not continue real-time operation after experiencing either a stack overflow or a hard frame overrun. In either case, manual or automatic restart to a specified safe condition shall be required; faults shall be reported in accordance with paragraph **2.6**. Design certification requires that adequate margins are designed to handle worst case conditions.

2.15.5. Operating System Access Restrictions. Unnecessary functionality (i.e., functions not needed for correct operation of the critical software or of the OS or RTE themselves) shall be disabled or removed from the system. The OS or RTE software shall prohibit or severely restrict Operator access to Administrative/Root privileges, including login to privileged accounts. Any erroneous/unintentional entry into or compromise of the OS or RTE shall be detected and reported in accordance with paragraph **2.6**. These requirements shall apply during all phases of operation, including booting the system and loading and executing the critical software.

**2.16. Operator Interface.** The design shall ensure that all nuclear critical functions can be easily terminated by the operator when required. The design shall ensure that nuclear critical functions cannot be initiated with a single action by an operator (two or more independent human actions, are required). The design shall minimize the number of points within the system where human actions could degrade nuclear safety or security. The software shall provide detection and notification of improper operator entries in accordance with paragraph **2.6**. For authorization and unique signal information, software shall take no action on the contents of this information and transmit it unaltered.

**2.17. Critical Function Initiation and Entry Checks.** Ensure that only the nuclear weapon control software components initiate critical functions (as defined in AFI 91-107) routines and modules. All entries into nuclear critical functions shall have checks to ensure such entries are both authorized and approved.

**2.18. Command and ID Word Format.** Select decision logic data values shall consist of specific binary data pattern of "ones" and "zeros" (not all "ones" or all "zeros") to reduce the likelihood of hardware or software malfunctions that satisfy the decision logic for critical function initiation or propagation. Such data patterns shall be selected such that a single bit flip will not result in another valid command or data word. Data patterns for critical function initiation shall not be bitwise complementary. Selection of critical commands and message ids shall maximize the hamming distance from other commands and message ids.

**2.19. Software Configuration.** Each software configuration submitted for certification shall comply with nuclear surety requirements**.**

**2.20. Hardware/software Interactions.** The design agent shall present all known failure modes of interfacing hardware. The agent shall design or implement measures to ensure that each failure mode is recognized and handled in a safe manner in accordance with paragraph **2.6**.

**2.21. Unused Code.** Software systems shall contain no unused code (as defined in the Glossary). Certified code that becomes unused incidental to standing down a weapon system

variant shall not be regarded as unused code for the purposes of this requirement unless nuclear safety impact can be demonstrated. Non-use patterns used to fill unallocated memory in compliance with paragraph 2.13.1 shall not be considered unused code unless nuclear safety impact can be demonstrated.

**2.22. Global Variables.** Software systems that have access to critical functions shall not contain nor use any global variables, except where necessary to meet other nuclear safety objectives.

**2.23. Complex Mechanisms.** Software, firmware and automata such as FPGA and ASICs designs shall be provided with a means of determining that the correct code and logic are present before items containing them are placed in operational service if the code or logic requires NSCCA or IV&V per paragraph **1.7**. Logic comprising ASIC or FPGA designs that cannot be verified to be present shall not be used to perform or control critical functions as defined by AFI 91-107.

**2.24. Unique Aircraft Requirements.**

2.24.1. Command Verification Protocol. For aircraft and air-launched nuclear weapons, it shall be ensured that the verification of critical commands transferred to remote units for processing or execution is accomplished. If a non-transitory communication error occurs, the command sequence shall be reset to its initial state.

2.24.2. In-flight Reversible Lock. For in-flight reversible locks under software control, the software controlling release or launch of a nuclear weapon shall have a unique control or control setting for locking and unlocking the in-flight reversible lock. This control shall be separate from the release and launch controls and the release consent.

2.24.3. Prearm Consent. The design may allow implementation of prearm consent through software inhibits and controls. However, the consent signal shall originate only through crew action. Removal of prearm consent shall result in terminating the prearm functions in process and shall inhibit prearm until consent is reestablished. Any change in consent status shall also be sent to the weapon, which shall then inhibit any critical function processing under weapon system control.

***Section 2C—Maintenance, Handling and Storage Facilities***

**2.25. Fault Detection.** Software that controls essential facility systems shall be designed to provide self-check, confidence or test routines to verify the integrity and proper state of hardware devices. Software shall be designed to detect hardware failures during power-up, operation and shut-down.

**2.26. Fault Response.** Failure modes or attempted tampering with essential facility system hardware shall provide detection and notification of improper operator entries in accordance with paragraph **2.6**.

**2.27. Alarms.** Essential facility system software shall be designed to provide both audible and visual alarms.

**2.28. Responsiveness.** Techniques shall be provided for deadlock (waiting for an event that will not occur due to error) prevention, detection or resolution.

**2.29. Initialization and Shutdown.** Upon system initialization, the essential facility system software shall confirm that hardware is in a known safe state. Upon system shutdown or program termination, the software shall attempt to set all settable, nonvolatile devices and relays are set to a known safe state.

**2.30. Operator Interface.** Ensure essential facility system software cannot be bypassed with a single action by an operator (two or more human independent actions). The software shall provide detection and notification of improper operator entries in accordance with paragraph **2.6**.

**2.31. Hardware/software Interactions.** The design agent shall present all known failure modes of interfacing hardware. The agent shall design or implement measures to ensure that each failure mode is recognized and dealt with in a safe manner in accordance with paragraph **2.6**.

***Section 2D—Test Equipment Software***

**2.32. Automated Test Equipment (ATE).** For ATE and the associated software and firmware used to verify the proper operation, safe state and control of critical nuclear functions (as defined by AFI 91-107), the ATE shall properly assess the operation of the critical functionality of the unit under test and shall verify the safe state of the unit upon test completion. The nuclear certification plan shall include at least one test of a production ATE. These objectives shall be fulfilled for all environmental conditions within which the ATE is designed to operate.

## Chapter 3

## EVALUATION CRITERIA FOR NUCLEAR WEAPON SYSTEMS SOFTWARE

### *Section 3A—General Philosophy and Criteria*

**3.1. Evaluation Program.** The NCIS will define applicability of design criteria to each software component.

**3.2. Software Standards Compliance.** For new systems, evaluate software design documentation for compliance with IEEE/EIA 12207. For modification to existing systems, evaluate documentation for compliance with the standard they were developed to (e.g. MIL-STD-498, DOD-STD-2167, EIA/IEEE J-STD-016 or IEEE/EIA 12207). Ensure incorporation of applicable nuclear surety design requirements for new systems or modification to existing systems.

### *Section 3B—Combat Delivery Aircraft, Ground Launched and Air Launched Missiles*

**3.3. Critical Data Element Analysis.** Identify the CSUs that change or evaluate critical function data elements (such as constants, variables and flags). Identify the chronological sequence of data element changes and evaluations. Identify the minimum conditions required for critical function execution (sensitivity analysis). Use this information to evaluate the safety features and estimate a probability for hardware induced software problems (inadvertent activation of critical functions) according to AFI 91-107, Table 2.

3.3.1. Programming Language-specific Safety Restrictions. Verify applicable prohibitions as listed by ANSPL are adhered to.

3.3.2. Run-time code restrictions. Verify code is not being generated at run-time, but is generated so it remains consistent from run to run.

**3.4. Hierarchical Design.** Verify presence of hierarchical structure. Evaluate critical function CSUs for single purpose.

**3.5. Fault Detection and Tolerance.** Validate the self-check, confidence or test routines. Verify the fault detection capability of the system during power-up, operation and shut-down. This may be accomplished with an analysis if demonstration could activate a critical function. The design documentation shall enumerate the fault conditions detected. All reasonable error conditions shall be included. Fault conditions shall be designated as transitory or system degrading and handled appropriately.

**3.6. Fault Response.** Validate the software response to detected system faults. Fault detection shall show:

3.6.1. Reversion to known safe software state

3.6.2. Transmission of critical commands halted

3.6.3. Recycle, self-test or perform automatic shutdown

3.6.4. Operator notification

3.6.5. Status of automated actions

**3.7. Real-Time Software.** Verify that documentation for real-time software includes specification of fixed deadlines for service of events associated with nuclear critical signals and functions. Ensure that documentation also includes sufficient information about scheduling protocols, task priorities and other such information as may be necessary to rigorously prove that such deadlines will be met under all system load and interrupt conditions (e.g., rate monotonic analysis).

**3.8. Responsiveness.** Verify that techniques for deadlock prevention, detection or resolution are implemented.

**3.9. Interrupts.** Verify that there are specific priorities and responses for events that interrupt program execution or disable interrupts. Ensure that the software either continues to run or shuts down in accordance with paragraph **2.12**.

**3.10. Idle Operations.** Verify that **s**oftware does not cause a WAIT, STOP or HALT state. Verify that the system is always executing, whether idle or actively processing.

**3.11. Alterations Detection.** Verify that **s**oftware contains no self-modification capability.

**3.12. Safe State Verification.** Verify that critical function hardware is in a known safe state at software initialization and shutdown.

**3.13. Memory Characteristics.** These requirements can be allocated between hardware and software.

3.13.1. Program Loading and Initialization. Verify that errors are detected and operators notified. Verify that reset and synchronization functions are operable. Verify that automatic operation will not start until all valid and correct data are loaded and verified. Verify that all non-volatile memory is filled with executable code, program data, or a non-use pattern.

3.13.2. Memory Protection. Verify that developed software will not perform error correction on nuclear permission or enable codes.

3.13.3. Memory Accessibility. Verify that only appropriate elements/routines of the software can access the critical function areas.

3.13.4. Memory Declassification. Evaluate the method used to erase or obliterate clear-text secure codes from memory.

3.13.5. FPGAs and CPLDs. Constraints on memory do not normally apply to FPGA designs. However, if an FPGA design instantiates memory that memory is subject to the evaluation criteria of 3.13.1 – 3.13.4

**3.14. Validity Checks.** Evaluate all critical command transmissions and their preconditions. Evaluate the process for preventing transmission if preconditions are not satisfied. Verify that no transmission occurs if inhibited. Verify that the prearm unique signal is not stored in a directly usable form. Verify that the usable prearm unique signal can only be assembled as a result of a crew member action. Identify software that contains the routine(s) for generating the unique signal. Verify that the routine(s) is/are initiated through crewmember action. Verify that the unique signal generator software does not contain the prearm unique signal pattern.

**3.15. Operating System and Run-Time-Executive.** Verify that operation remains stable in a variety of worst-case operating conditions (refer to paragraph **2.15.3**). Verify that all escape sequences leave the software in a safe state. Verify that the priority promotion or demotion scheme does not cause a failure of hard or soft real-time scheduling. Verify that data skew for degraded soft real-time scheduling does not affect software performing critical functions. Verify that unnecessary functionality/access is disabled or removed from the system to the maximum extent possible (refer to paragraph **2.15.4**).

**3.16. Operator Interface.** Verify that operator input errors are detected and handled appropriately. Verify that the software will notify the operator and will not enable any critical function if an error occurs. Verify that the operator can, with a single action, cancel nuclear critical functions. Verify that a single operator action cannot initiate nuclear critical functions. Verify that authorization and unique signal information are transmitted unaltered.

**3.17. Critical Function Initiation and Entry Checks.** Verify the presence of a process that identifies unauthorized entries prior to performing a nuclear critical function. Verify that termination and notification occurs following unauthorized entry.

**3.18. Command and ID Word Format.** Verify that the Computer Software Configuration Item (CSCI) contains only bit patterns allowed by paragraph **2.18**.

**3.19. Software Configuration.** Evaluation of software for compliance with nuclear surety requirements should be done on as high a level as possible subject to the condition that correspondence with executed code is unambiguous. Compiled high level languages such as Ada generally meet this requirement. In situations where code is automatically generated from graphical models or other higher level constructs (such as those associated with Object Oriented Analysis), such models may be evaluated if they can be presented in a human readable format and if the translation rules are sufficiently well explained that correspondence with executable code is unambiguous. When design information exists from programs that are developed by manual processes, inclusion of this information for reference purposes is encouraged, but cannot form the sole basis of evaluation. In no case shall software be certified based on evaluation of out-of-date code; an update to the executable requires a corresponding update to the code evaluated.

**3.20. Hardware/software Interactions.** Verify that all known failure modes of the interfacing hardware are recognized and that there are handlers for each failure mode.

**3.21. Unused Code.** Verify that there is no unused code. If there is unused code, verify that a safety analysis has been accomplished. Unused code may be allowed to remain in the software, at the discretion of the SAG or equivalent authority, based upon the safety impact of execution of the unused code and the discrepancy priority (Refer to paragraph 2.21). See attachment 3 for further guidance.

**3.22. Global Variables.** Verify that no global variables (as defined in attachment 1) are used, except where necessary to meet other nuclear safety objectives.

**3.23. Complex Mechanisms.** Verify that NSCCAed or IV&Ved software, firmware and automata such as FPGA designs and ASICs are provided a means of determining that the correct code or logic is actually present before items containing them are placed in operational service. In the case of firmware and FPGA designs which can be “read out,” verification can ordinarily be accomplished by comparing the output code with a known-to-be-good reference or

performing a bit-for-bit compare with a known-good reference. In case of volatile designs and software, the operational code is loaded from a known-to-be-good source. In a case where an ASIC or FPGA design that cannot be "read out" must be used to implement a critical function, the chip must be subject to birth-to-death Two Person Concept (TPC) Control if the presence of the correct design cannot otherwise be verified. Tamper protection per AFI 91-104 can be used instead of TPC control in certain stages of the life-cycle.

**3.24. Unique Aircraft Requirements.**

3.24.1. Command Verification Protocol. Identify critical commands transferred to remote units for processing or execution. Validate communication error detection method (refer to paragraph **3.5**) and reset process following error detection (refer to paragraph **3.6**).

3.24.2. In-flight Reversible Lock. Identify any software controlling release or launch of a nuclear weapon. Verify that the programming scheme provides a unique control or control setting for locking and unlocking the in-flight reversible lock. Verify that the control is separate from the release and launch controls and the release consent.

3.24.3. Prearm Consent. If prearm consent is used, verify that the software accomplishes the following:

3.24.3.1. Crew action origination

3.24.3.2. Termination following consent removal

3.24.3.3. Prearm inhibited until reestablished

3.24.3.4. Critical function inhibited following change in consent status

***Section 3C—Maintenance, Handling and Storage Facilities***

**3.25. Fault Detection.** Verify that the software that controls essential facility systems provides self-check, confidence, or test routines that verify the integrity and proper state of hardware devices. Verify that the software detects hardware failures during power-up, operation and shut-down.

**3.26. Fault Response.** Verify that essential facility system software failure modes, attempted tampering with essential facility system hardware and improper operator entries are detected and reported in accordance with paragraph **2.6**.

**3.27. Alarms.** Verify that essential facility system software provides both audible and visual alarms.

**3.28. Responsiveness.** Verify that techniques for deadlock (waiting for an event that will not occur due to error), prevention, detection and resolution are adequate.

**3.29. Initialization and Shutdown.** Verify that upon system initialization, the essential facility system software confirms that hardware is in a known safe state. Verify that upon system shutdown or program termination, the software ensures all settable, nonvolatile devices, and relays are set to a known safe state.

**3.30. Operator Interface.** Verify that essential facility system software cannot be bypassed with a single action by an operator (two or more human independent actions). Verify that the software provides detection and notification of improper operator entries per paragraph **2.6**.

**3.31. Hardware/Software Interactions.** Verify that the software recognizes all known failure modes of interfacing hardware and that it deals with them in a safe manner per paragraph **2.6**.

***Section 3D—Test Equipment Software***

**3.32. Automated Test Equipment.** Verify that the ATE and the associated software and firmware properly assess the operation of the critical functionality of the unit under test and verify the safe state of the unit upon test completion. Verify that objectives for all environmental conditions within which the ATE is designed to operate were tested on at least one production or production-like ATE.

GREGORY A. FEEST, Maj Gen, USAF
Chief of Safety

## Attachment 1

## GLOSSARY OF REFERENCES AND SUPPORTING INFORMATION

***References***

AFI 91-107, *Design, Evaluation, Troubleshooting and Maintenance Criteria for Nuclear Weapon Systems,* 26 Jan 2011

AFMAN 91-118, *Safety Design and Evaluation Criteria for Nuclear Weapon Systems,* 4 Aug 2010

AFI 63-125, *Nuclear Certification Program,* 15 Mar 2004

AFPD 13-5, *Air Force Nuclear Enterprise,* 6 July 2011

DOD Instruction S-5200.16, *Objectives and Minimum Standards for Communications Security (COMSEC) Measures Used in Nuclear Command and Control (NC2) Communications (U),* 14 Nov 2007

DOD Directive O-5210.41, *Security Policy for Protecting Nuclear Weapons,* 1 Nov 2004

DOD-STD-2167A, *Defense System Software Development,* 29 Feb 1988

MIL-STD-498, *Software Development and Documentation,* 5 Dec 1994

EIA/IEEE J-STD-016, *Standard for Information Technology Software Life Cycle Processes, Software Development Acquirer-Supplier Agreement,* 30 Sep 1995

IEEE/EIA 12207, *Systems and Software Engineering-Software Life Cycle Processes,* 27 May 1998

ISO 13490, *Compact Disk File System,* 1995

ISO 9660, *Compact Disk File System,* 1999

ISO 9001, *Quality Management Systems Requirements,* 17 Apr 2001

INCITS/ISO/IEC 9899-1999, *Programming Languages – C,* 2005

ISO/IEC 14882:1998(E), C++

*MISRA-C, Guidelines for the USE of the C Language in Critical Systems,* ISMN 0 9524156 2 3 (paperback, ISBN 0 9524156 4 X (PDF), October 2004

MISRA-C++, Guidelines for the Use of the C++ Language in Critical Systems, ISBN 978-906400-03-3 (paperback), ISBN 978-906400-04-0 (PDF), June 2008

***Abbreviations and Acronyms***

**AF**—Air Force

**AFB**—Air Force Base

**AFDPO**—Air Force Departmental Publishing Office

**AFPD**—Air Force Policy Directive

**AFI**—Air Force Instruction

**AFMAN**—Air Force Manual

**ANSI**—American National Standards Institute

**ANSPL**—American National Standard for Programming Languages

**ASIC**—Application Specific Integrated Circuit

**ASCII**—American Standard Code for Information Interchange

**ATE**—Automated Test Equipment

**BCMS**—Blast Containment Management System

**CD**—Compact Disc

**COTS**—Commercial off-the-shelf

**CPLD**—Complex Programmable Logic Devices

**CPU**—Central Processing Unit

**CSCI**—Computer Software Configuration Item

**CSU**—Computer Software Unit (separately testable element of a CSC)

**DOD**—Department of Defense

**DODD**—Department of Defense Directive

**DOE**—Department of Energy

**DR**—Discrepancy Report

**EIA**—Electronic Industries Association

**FPGA**—Field Programmable Gate Array

**HQ AFSEC**—Headquarters Air Force Safety Center

**HOL**—Higher Order Language

**IEC**—International Electro Technical Commission

**IEEE**—Institute of Electrical and Electronics Engineers

**INCITS**—InterNational Committee for Information Technology Standards

**ISO**—International Organization of Standardization

**IV&V**—Independent Validation and Verification

**KUMMSC**—Kirtland Underground Munitions Maintenance Storage Complex

**MIL**—Military

**MNCL**—Master Nuclear Certification List

**NCIS**—Nuclear Certification Impact Statement

**NSA**—National Security Agency

**NSCCA**—Nuclear Safety Cross Check Analysis

**NSO**—Nuclear Safety Objective

**NWSSG**—Nuclear Weapons System Safety Group

**OS**—Operating System

**PAL**—Permissive Action Link

**POSIX**—Portable Operating System Interface

**RDS**—Records Disposition Schedule

**RTE**—Run-Time-Executive

**SAG**—Software Advisory Group

**SEWN**—Nuclear Weapons Safety Branch

**STD**—Standard

**STD**—Standard

**T.O.**—Technical Order

**TPC**—Two Person Concept

**WWW**—World Wide Web

***Terms***

**Independent Verification and Validation**—is an analysis and test of computer software by an organization that is independent of the development contractor or organization to ensure that the software complies with established nuclear safety design criteria.

**Nuclear Safety Cross-Check Analysis**—is an analysis by an organization that is independent of the software developer to ensure critical software does not contain improper design, programming, fabrication, or application that could contribute to:

- Unauthorized or inadvertent authorization, prearming, arming, or launching or releasing of a nuclear weapon or nuclear weapon system.

- Premature or unsafe operation of a nuclear weapon system.

- Delivery of a nuclear weapon outside the specified boundary of the planned target.

- Unauthorized, improper, or erroneous display of status or classified information that could degrade nuclear surety.

- Improper handling of classified cryptographic codes, invalid verification or the retrieval of such codes by unauthorized persons in a manner that could degrade nuclear surety.

**Global Variable**—is an object of information that is defined in the top-level software system design that is accessible from all parts of the software system. The following is a partial list of what an attribute can be: Integer, Real number, Byte, Bit, Boolean, String, Char, Record, Array and Linked List. Presence of global attributes in software systems is cause for concern because lack of control of such an attribute can very likely set the attribute to an unknown value or unknown state. This is due to the fact that a global attribute can be accessed from any part of the software system, such as from within a procedure, function or any such object. A global attribute can even be accessed by a function within a function.

**Unused code**—is a function, procedure, method, object, variable, or other section of code that is not accessed or used by the software system or other software systems.

## Attachment 2

## GENERIC NUCLEAR SAFETY OBJECTIVES (NSOS) WITH AFMAN 91-118 OR AFMAN 91-119 CROSS-REFERENCE

**A2.1.** The **authorization** function will (AFMAN 91-118, paragraph 2.2.2.1):

A2.1.1. Authorize use of the weapon system

A2.1.2. Prevent prearming, arming or launching without prior authorization (examples of designs include Permissive Action Link and the Minuteman enable device).

A2.1.3. Be implemented using the information control concept.

A2.1.4. Not prevent safing regardless of the state of authorization device.

A2.1.5. Be reversible.

**A2.2.** The **prearming** function will (AFMAN 91-118, paragraph 2.2.2.2):

A2.2.1. Permit arming.

A2.2.2. Preclude prearming in the absence of a prearm command signal.

A2.2.3. Prevent arming if the prearming device is bypassed.

A2.2.4. Isolate the prearming signal from the authorization function.

A2.2.5. Be derived from some part of the weapon system that is under direct human control.

A2.2.6. Be reversible up to the time of launch.

A2.2.7. Use a uniquely coded signal.

A2.2.8. Be physically unavailable until its use is required.

**A2.3.** The **launching** function will (AFMAN 91-118, paragraph 2.2.2.3):

A2.3.1. Permit operation of the propulsion system.

A2.3.2. Be controlled with two independent functions (e.g., booster arm/safe and ignition commands).

A2.3.3. Prevent ignition when device has been "safed."

A2.3.4. Preclude unauthorized/accidental transmission of booster arm and ignition commands.

A2.3.5. Have a reversible propulsion system ignition coding device.

**A2.4.** The **releasing** function will (AFMAN 91-118, paragraph 2.2.2.4):

A2.4.1. Permit release of aircraft-carried weapons only through two independent functions.

A2.4.2. Preclude accidental transmission of unlock and release commands.

A2.4.3. Preclude any failure from allowing bypass of the lock device that would permit release of the weapon when the device is locked.

**A2.5.** The **arming** function will (AFMAN 91-118, paragraph 2.2.2.5):

A2.5.1. Permit warhead detonation through a selected fuse signal (e.g., radar, contact and timer).

A2.5.2. Prevent detonation, if the arming function has been bypassed.

A2.5.3. Preclude arming prior to measurement of the proper operational environment; this measurement shall include a "good guidance" signal.

A2.5.4. Prevent erroneous transmission of the proper operational environment signal.

**A2.6.** The **targeting** function will (AFMAN 91-118, paragraph 2.2.2.6):

A2.6.1. Prevent accidental/unauthorized changes to targeting.

A2.6.2. Display to the launch control point operators any changes to targeting data/functions that occur prior to launch.

A2.6.3. Prevent nuclear detonation except within boundaries of the designated target area.

**A2.7. Single component failures** will (AFMAN 91-118, paragraph 2.10):

A2.7.1. Not cause the system to be in an authorized, prearmed or armed state.

A2.7.2. Not cause the inadvertent transmission/operation of the critical functions.

**A2.8. Human engineering** will (AFMAN 91-118, paragraph 2.11):

A2.8.1. Be used to add features that either eliminate human error and unauthorized acts or limit their consequences.

A2.8.2. Prevent any two independent human errors from causing the authorization, prearming, arming, releasing or launching of the weapon system.

A2.8.3. Incorporate positive measures to prevent deliberate, unauthorized or accidental operations of the weapon system that could degrade nuclear safety.

A2.8.4. Eliminate/minimize dependency of the safety and security of the weapon system on administrative procedures.

**A2.9.** The **launch control system** will (AFMAN 91-118, paragraph 2.26):

A2.9.1. Permit launching only through the intentional operation of the authorization and launch control function/ devices.

A2.9.2. Be the only system that will be able to authorize/start a launch sequence, launch a missile or operate the propulsion system.

A2.9.3. Remain in or return to a safe state when component failure occurs.

A2.9.4. Not implement the prearming and safing functions as complementary functions (i.e., the absence of prearming will not be construed as safe and vice versa).

**A2.10.** The **monitor system** will (AFMAN 91-118, paragraph 2.25):

A2.10.1. Provide the operator continuous or on-demand monitoring of the safety status of the missile’s propulsion system, warhead (if the warhead has monitoring circuits), reentry vehicle/system and launch control system.

A2.10.2. Provide the operator with positive indications of any changes to the safety status of the monitored systems.

A2.10.3. Automatically remove power if an unsafe condition is detected.

**A2.11.** The **command, control and communications system** will (AFMAN 91-118, paragraph 2.26):

A2.11.1. Use secure codes to authorize the launch of or to launch a missile.

A2.11.2. Not allow the authorization or launching of a missile by a single person; at least two people shall actively cooperate to command authorization and launch, even after secure codes are available.

A2.11.3. Prevent bypass or unauthorized readout of the secure code devices.

A2.11.4. Control access to the code storage devices to prevent unauthorized code changes.

A2.11.5. Implement the prearming and launching commands as unique signals.

A2.11.6. Not store the prearming and launching signals at the launch point in a directly usable form.

A2.11.7. Prevent unauthorized use of the prearming and launching signals (implementation examples: derive signals from secure codes; store signals in permuted form; store parts of the signals in separate locations).

A2.11.8. Allow one or more missiles to be launched without revealing or compromising the codes for other missiles.

A2.11.9. Allow one or more launch control points to monitor and take compensatory action if an unauthorized critical command message or status is detected.

**A2.12. FPGA and ASIC designs and software** will:

A2.12.1. Not be made up of memory whose contents alter or degrade over a period of time (RAM-based FPGAs which lose their configuration when powered off are acceptable if properly reconfigured when powered on).

A2.12.2. Have provisions to prevent unexpected changes in volatile memory from adversely impacting a critical function.

A2.12.3. Prevent a single hardware failure from causing a memory change that could initiate a critical function.

A2.12.4. Verify and validate the correct loading of all data and programs.

A2.12.5. Prevent unauthorized changes to memory.

A2.12.6. Notify the launch control point operator or maintenance/coding personnel of errors detected during authorized memory loads/changes.

A2.12.7. Detect, inhibit and report (to the operator) any unauthorized attempts at loading/changing critical functions in memory.

A2.12.8. Ensure that each section of memory is filled by the block of proper program data or instructions to be transferred. This exact-fill requirement can be met by tailoring data for

exact fill or using filler bits. If this requirement cannot be met, the memory will be overwritten or cleared before writing the transferred data or program instructions.

A2.12.9. Declassify all clear-text secure codes through erasure/obliteration after their authorized use.

A2.12.10. Verify/validate all critical functions upon initiation.

A2.12.11. Inhibit transmission of any critical function found to be in error and notify the operator of the error.

A2.12.12. Not be able to bypass operator control of critical functions.

A2.12.13. Detect erroneous entries into critical routines/functions and immediately recycle to the proper sequence, self-test mode or automatic shutdown.

A2.12.14. Adhere to top-down design decomposition and structured programming methodology.

A2.12.15. Provide self-check, confidence or test routines of all critical hardware /circuits.

A2.12.16. Prevent non-critical functions or data from accessing critical memory

A2.12.17. Not contain any unused code

A2.12.18. Not contain any global variables on any systems that have access to critical functions

A2.12.19. Restrict unnecessary access to OS or RTE

## Attachment 3

## SUMMARY PRIORITY SCHEME FOR NUCLEAR SAFETY DISCREPANCIES

**A3.1. Critical (Priority 1).** An error that could lead to a violation of one or more of the DOD Nuclear Weapon System Safety Standards or compromise one or more of the critical functions. Priority 1 discrepancies should be directly traceable to certain and/or positive violations of at least one of the DOD safety standards or to inadvertent or unauthorized execution of any one of the critical functions (as specified in AFMAN 91-107). These discrepancies shall be corrected prior to granting certification. Workarounds are not acceptable.

**A3.2. Urgent (Priority 2).** An error that could lead to a violation of one of the DOD Nuclear Weapon System Safety Standards, with a small probability of occurrence (within a credible but abnormal environment, including single-point component failures). Priority 2 discrepancies have a serious nuclear safety impact, but are low probability events. Note that a credible probability shall be no smaller than the probability of a single component failure. Code with Priority 2 discrepancies may be certified for limited periods if there is a reliable workaround. For example, if the discrepancy is discovered at the end of a development cycle and another development cycle is scheduled to immediately follow, the code may be certified under the understanding that the issue will be corrected in the next cycle. Priority 2 discrepancies discovered at the beginning of a development cycle should be resolved before certification is granted. Workarounds shall be incorporated into Technical Order (T.O.) procedures.

**A3.3. Degraded (Priority 3).** An error that degrades a nuclear surety related safeguard, but does not compromise one of the DOD Nuclear Weapon System Safety Standards. An example would be a test incorrectly performed on a critical component by ATE, but no damage results and a manual work-around detects the condition. Priority 3 errors are "high" candidates for correction but may be tolerated (in the system) while an update is pending (i.e., this error may be deferred but not ignored). Workarounds shall be incorporated into T.O. procedures.

**A3.4. Noncritical (Priority 4**). A technical noncompliance violation of AFMAN 91-119 that does not lead to a compromise of the DOD Nuclear Weapon System Safety Standards or critical functions. Priority 4 discrepancies are technical violations of AFMAN 91-119, with minimal or unquantifiable impact (for example, a coding error resulting in an erroneous display but not leading to operator error). These problems are recommended for correction, but they may exist in the system indefinitely. Note, however, that too many errors in this category may require that some corrections be made, particularly when the cumulative effects are likely to impair operator capability (i.e., "many" Priority 4 errors may equal the effects of "one" Priority 2 error).

**A3.5. Minor (Priority 5). Unused code.** Unused code (i.e. “dead code”) shall be accorded Priority 5 unless nuclear surety impact warranting a more serious priority can be demonstrated. Priority 5 discrepancies are recommended for correction, but would not normally be grounds for withholding Nuclear Design Certification.

**A3.6. Documentation Errors.** Requirement and design document errors shall be accorded the priority they would have if implemented in the code. (Note that a documentation error not reflected in the code would not normally be grounds for withholding Nuclear Safety Design Certification of the executable.) Other documentation errors shall be accorded a Priority 5 rating unless nuclear surety impact warranting a more serious priority can be demonstrated.

## Attachment 4

## DISTRIBUTION OF NUCLEAR CERTIFIED SOFTWARE ON COMMERCIAL COMPACT DISK MEDIA

**A4.1.** Applicable to recording on CD, a master image shall be prepared in accordance with ISO 9660:1999, *Compact Disk File System* which is by design a read-only, pre-mastered, file system and saved as a master image file.

A4.1.1. Software shall only be recorded and distributed on factory fresh Write-Once media, i.e., once written there is no provision for altering the stored content.

A4.1.2. All CDs used to record and distribute software shall be single session only; no multi-session disks as defined by ISO 13490, *Compact Disk File System* shall be allowed for this purpose.

A4.1.3. All optical media disks used to record and distribute software shall be finalized (that is, the session shall be closed) upon completion of recording of software onto the disk.

A4.1.4. Upon completion of a CD recording session, the content of the newly recorded disk shall be verified for correctness and accuracy. Verification shall be accomplished by comparing the data on the CD with the original data that the CD recording system used to record onto the CD. An example of verifying accuracy is to ensure that the boot block has not been changed.