ARKANSAS NUCLEAR ONE - UNIT 2

DOCKET 50-368

CEN-39(A)-NP

CPC PROTECTION ALGORITHM
SOFTWARE CHANGE PROCEDURE

SEPTEMBER 22, 1978

Combustion Engineering, Inc.
Nuclear Power Systems
Power Systems Group
Windsor, Connecticut

Rev. 01

LEGAL NOTICE

This report was prepared as an account of work sponsored by Combustion Engineering, Inc. Neither Combustion Engineering nor any person acting on its behalf:

a. Makes any warranty or representation, express or implied including the warranties of fitness for a particular purpose or merchantability, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

b. Assumes any liabilities with respect to the use of, or for damages resulting from the use of, any infomation, appartus, method or process disclosed in this report.

ABSTRACT

This document presents procedures to be followed when specifying and implementing modifications to quality assured CPC/CEAC software and documentation. Section 1.0 of this document contains procedures applicable to specifying modifications to the CPC/CEAC protection algorithm functional design and data base. Section 2.0 of this document contains procedures applicable to implementing and testing changes to the CPC/CEAC system software design and data base. Section 3.0 contains procedures applicable to the creation of project specific software and documentation from the approved generic design.

TABLE OF CONTENTS

TABLE OF CONTENT (Cont.)

LIST OF FIGURES

LIST OF TABLES

1.0 SPECIFICATION OF SOFTWARE FUNCTIONAL DESIGN AND DATA BASE CHANGES

1.1 PURPOSE, SCOPE AND APPLICABILITY

1.1.1 Purpose

The purpose of Section 1.0 is to present the relationships between various steps in the process of specifying modifications to the CPC/CEAC protection algorithm software functional design or data base constants. A series of steps resulting in modification of design documents prepared in accordance with Reference 1.2.1 are described to demonstrate that the process of specifying modifications is complete and is accomplished in a structured manner. As a result, the integrity and consistency of the software functional design documents are maintained and generation of errors during the modification process is avoided.

1.1.2 Scope

The scope of Section 1.0 covers requirements on the process of specification of a change to the CPC/CEAC protection algorithm software functional design and data base, including generation of test results from the CPC/CEAC FORTRAN code for comparison to the implemented CPC/CEAC software. The process covered in this section begins when the specific details of a change to the protection algorithm software functional design or to the data

base have been established. The process covered in this section ends when the following documents have been revised in accordance with the procedures given in Reference 1.2.1.

1. Recorded calculations or other design analysis documents establishing specifics of the CPC/CEAC protection algorithm functional design or values of data base constants.

2. The CPC Functional Description; the CEAC Functional Description; and the CPC/CEAC Data Base Document

3. The CPC/CEAC FORTRAN code.

4. The Phase I Test Report

5. The Phase II Test Report

1.1.3 Applicability

These requirements are to be followed for all modifications to the CPC/CEAC protection algorithm software functional design and non-addressable constants that are made subsequent to completion of design qualification of the CPC/CEAC software and data base constants for a given plant. A prerequisite for use of these requirements is that all applicable CPC/CEAC design documents have previously been completed in accordance with procedures

given in Reference 1.2.1, and together reflect the complete QA'd status of software specification and implementation for a given plant. The applicable CPC/CEAC design quality assurance documentation for a given plant and the relationship between the various design documents are shown in Figure 1.1-1.

1.2 REFERENCES FOR SECTION 1.0

1.2.1 Quality Assurance of Design Manual for C-E Nuclear Power Systems.

1.2.2 Core Protection Calculator Phase II Test Report, CEN-73(A)-P.

1.3 DESIGN INTERFACES AND MODIFICATION REQUIREMENTS

The interfaces and activities required for modification of CPC/CEAC design documents and Phase I and Phase II design qualification tests are shown in Figure 1.3-1. During modification to the CPC/CEAC software design and/or data base constants, these documents are revised as required in accordance with procedures given in Reference 1.2.1 to consistently reflect the status of the modified design. Section 1.3-1 describes the organization and interfaces between the various blocks shown in Figure 1.3-1. Section 1.3.2 describes the details of each block of Figure 1.3-1 that are associated with specification of the modification after the requirements of the modification have been established.

1.3.1 Design Interfaces

The modification process is initiated when details of the modification have been established by the responsible C-E engineering group. For these requirements to be incorporated and referenced by the CPC/CEAC Functional Descriptions or Data Base Document, they must be completed as recorded calculations or other design documents in accordance with Reference 1.2.1. After establishment of the details of the modification, a Software Change Request (SCR) (Figure 1.3-2) may be issued to the C-E engineering group responsible for software implementation. As explained in Section 1.3.2.2, the SCR is designed to allow software implementation to be planned or to proceed in parallel with the completion of the

functional design QA process. Although individual SCR's are not required by the Reference 1.2.1 QA process they are included herein for the following reasons:

1. SCR's are referenced in revisions to the CPC/CEAC Functional Description and Data Base Document to provide complementary information in the form of traceability to the record of revisions listed in these documents. The references to the SCR's are contained in the reference sections of these documents and are of the form:
   Revision i of this document implements the following Software Change Requests: k, l, m . . .
   where i = the document revision number
   k, l, m = SCR numbers implemented.

2. Individual SCR's provide traceability, since originals are retained in the design files of the originating engineering group and copies are disseminated to the engineering group using the SCR's to begin software modifications.

3. Details of implementation of SCR requirements including a Change Applicability Form are transmitted back to the originating design group so that confirmation that the SCR was correctly understood is provided. A copy of the SCR form used for CPC/CEAC software modifications is shown in Figure 1.3-2 and the Change Applicability Form is shown in Figure 1.3-3.

Upon completion of detailed modification requirements in accordance with Reference 1.2.1, these requirements are referenced by the associated revision of the CPC/CEAC Functional Description and/or Data Base Document. The revisions of these documents are discussed in Sections 1.3.2.3 and 1.3.2.4, respectively. The CPC/CEAC Functional Description and Data Base Documents are revised in accordance with Reference 1.2.1 and are transmitted to the C-E engineering group responsible for implementation of the revision in the CPC/CEAC software.

For those modifications which impact the CPC/CEAC FORTRAN code, the CPC/CEAC FORTRAN code certification document is revised. All revisions to the FORTRAN code are done in accordance with the requirements of Reference 1.2.1. The code certification document is referenced to the appropriate revision of the CPC/CEAC Functional Description. Revisions to values of existing data constants in the Data Base Document do not require alteration of the FORTRAN code, since the values of data constants are not specified within the code, but are inputs to the FORTRAN code.

For those modifications which impact the conclusions of Phase I and Phase II Test results, the CPC/CEAC FORTRAN code is used to generate expected results for Phase I and Phase II qualification tests. The results of the FORTRAN cases are used in the revisions of Phase I and Phase II Test Reports. These Test Reports will reference the appropriate revision of the CPC/CEAC FORTRAN code.

As shown in Figure 1.3-1, the Software Change Request (Figure 1.3-2), the CPC/CEAC functional descriptions, the CPC/CEAC data base document, and the CPC/CEAC FORTRAN code results all interface with the implementation portion of the software modification process. The implementation of software modifications is discussed in Section 2.0.

1.3.2 CPC/CEAC Protection Software Functional Design Changes

This section describes the actions to be taken by the CPC/CEAC functional design group to specify changes to the CPC/CEAC protection algorithm software functional design and data base. The sequence of events is described from determination of the requirement for a change to completion of QA documentation of a revision to the CPC/CEAC functional design or data base.

The following major steps in the design change process are detailed within this section:

1) Specification and transmittal of a functional change and/or a data base change

2) Revision of Functional Description

3) Revision of Data Base Document

4) Recertification of CPC/CEAC FORTRAN Simulation

5) Change Qualification

1.3.2.1 Specification of CPC/CEAC Protection Algorithm Software Functional Change

When a CPC/CEAC protection algorithm software functional design change has been identified and fully defined, it shall be transmitted to a CPC/CEAC Design Engineer responsible for execution of the CPC/CEAC functional design change procedure. If the change does not affect CPC/CEAC FORTRAN Simulation Code, the functional design change shall be sent directly to the software implementation group according to Section 1.3.2.2. An example of a functional design change not requiring a FORTRAN code change is addition or deletion of a point ID assignment to a CPC calculated variable for accessing the value of the variable at the CPC operator's module. If the change does affect aspects of the CPC/CEAC System simulated in the FORTRAN code, the change will be implemented in the CPC/CEAC FORTRAN Simulation Code according to Section 1.3.2.5.

When the CPC/CEAC FORTRAN Simulation Code has been modified to reflect the functional design change, a series of debug cases shall be run to assure that no FORTRAN errors are present in the update to the FORTRAN code. If FORTRAN errors are detected, they shall be corrected and the test cases rerun before the change is transmitted to the software implementation group. The test cases will normally be Phase II static or dynamic test cases, Reference 1.2.2, selected to exercise the modified portions of the FORTRAN code.

1.3.2.2 Transmittal of the Change

The medium for transmittal of individual software functional design changes or data base changes from the software functional group to the software implementation group is the Software Change Request (SCR) form shown as Figure 1.3-2. The SCR supplements the formal quality assured functional description and data base documents. It is a means of providing orderly communication between groups with design specification responsibility and groups with design implementation responsibility. Quality assurance of a software design change is established by revision of the CPC/CEAC Functional Descriptions and/or the CPC/CEAC Data Base Document. Revisions of the CPC/CEAC Functional Descriptions and Data Base will have references to applicable SCRs as a supplement to the record of revisions required for these documents.

Completed SCR forms must be approved by both the functional design group and the software implementation group. At the top of the form are blanks for entering the following information:

1) Change #: a unique sequence number for the SCR obtained by entering a line in the SCR Log (see Section 2.3.1.1).

2) Date: date on which the SCR is filled out.

3) Plant(s): plant or plants for which the SCR applies.

The remainder of the SCR consists of six parts. Guidelines for completion of each part are provided below:

I. Originator: Section and name of the person filling out the SCR.

II. References: These are documents giving the basis for the change or are other SCRs affected by this SCR.

III. Summary of Change: If the change is minor, it may be described in the space provided. For more extensive changes a descriptive phrase is entered in the space provided, and a detailed description of the change is provided on attached pages. The change description should adhere to the following guidelines for each plant affected by the change:

1) All parameters are referred to by their functional description variable names.

2) Arithmetic operations are described in standard mathematical notation.

3) Logical operations, branching, and loops are described in concise prose using relational operators such as >, $\leq$, $\neq$, etc.

4) If necessary to avoid ambiguity in location of inserted or deleted operations, copies of flow charts from the current revision of the plant software specifications should be annotated and included in the change description. Copies of the old and new CPC FORTRAN Simulation coding may also be attached to avoid ambiguity.

5) Constants and variables deleted from a given program by the change must be identified.

6) Values for new constants must be given, and any new addressable constants must be identified.

7) Any existing constants that have changed from protected to addressable status or from addressable to protected status must be identified.

8) Changes in the values of existing constants must be stated.

IV. Program(s) Affected: List of CPC/CEAC programs affected by this SCR.

V. Approvals:

1) Prior to transmittal to the software implementation group the cognizant functional design supervisor or designate must review and approve the SCR.

2) Upon receipt of the SCR, the software implementation supervisor or designate must indicate his acceptance of the SCR by signing the indicated blank.

3) The software implementation engineer implementing the change annotates the SCR to indicate any clarification obtained verbally and signs the "Implemented by" line when implementation of the SCR has been completed.

4) Copies of the annotated SCR, revised flow charts, Change Applicability Form, and details of the implementation necessary for clarification are returned to the functional design group for review of the implementation to verify correct interpretation of the SCR and signature of the "Impl. Reviewed by" line is obtained.

VI. Design Documents Affected: The SCR originator must determine whether the change described in the SCR requires revision of each item listed for each plant affected by the change. When revision of a given document is completed the person responsible for completion of the respective document enters the date of completion and the "Rev" number of the revised document in the appropriate columns on the original of the SCR.

1.3.2.3 Revision of CPC/CEAC Functional Descriptions

As stated above, the SCR is not a formal quality assured document. The SCR allows software implementation design to be planned or to begin as soon as the details of a required functional design change are known. In this way revision and review of quality assured functional design documentation can proceed in parallel with software design activities that require substantial lead time.

Revisions to the CPC/CEAC Functional Descriptions required by an SCR shall be incorporated in accordance with Reference 1.2.1. Upon final approval of the revised CPC/CEAC Functional Descriptions, lines VI A and/or VI B (CPC Functional Description and CEAC Functional Description) of all SCRs encompassed by the revision shall be completed. In addition, for all SCRs encompassed by the revision, any recorded calculation revisions or new recorded calculations referenced by the revision to the functional description(s) shall be listed on line VI F (Recorded Calculations) and the revision number and date of completion of those calculations shall be filled in. The approved revisions of the CPC/CEAC Functional Descriptions shall be transmitted to the software implementation group to provide quality assured input for the revision of the CPC/CEAC software specifications.

1.3.2.4 Revision of Plant Data Base Document

Revisions to a Plant Data Base Document shall be incorporated in accordance with Reference 1.2.1. Upon final approval of the revised Plant Data Base Document, line VI C (Plant Data Base Document) of all SCRs encompassed by the revision shall be completed. In addition, for all SCRs encompassed by the revision, any recorded calculation revisions or new recorded calculations referenced by the revision to the data base shall be listed on line VI F (Recorded Calculations) and the revision number and date of completion of those calculations shall be filled in. The approved revision of the Plant Data Base Document shall be transmitted to the software implementation group as quality assured input to the next revision of the CPC/CEAC software specifications.

1.3.2.5 Recertification of CPC/CEAC FORTRAN Simulation Program

For CPC/CEAC functional design changes requiring modification of the CPC/CEAC FORTRAN Simulation Program, recertification of the code is required. CPC/CEAC FORTRAN Simulation Program recertification must be accomplished in accordance with Reference 1.2.1. When recertification has been completed, the "Rev" number of the new certified version of the code is entered in the appropriate column of line VI D in SCRs encompassed by the revision.

1.4 PHASE I DESIGN CHANGE QUALIFICATION TESTING

Phase I test cases to be rerun shall be determined by the software implementation group and a definition of the test cases shall be provided to the functional design group. The functional design group shall execute the test cases using the CPC/CEAC FORTRAN Simulation Code and return the results to the implementation group for analysis and incorporation into the Phase I Test documentation. Further discussion on Phase I testing is provided in Section 2.0.

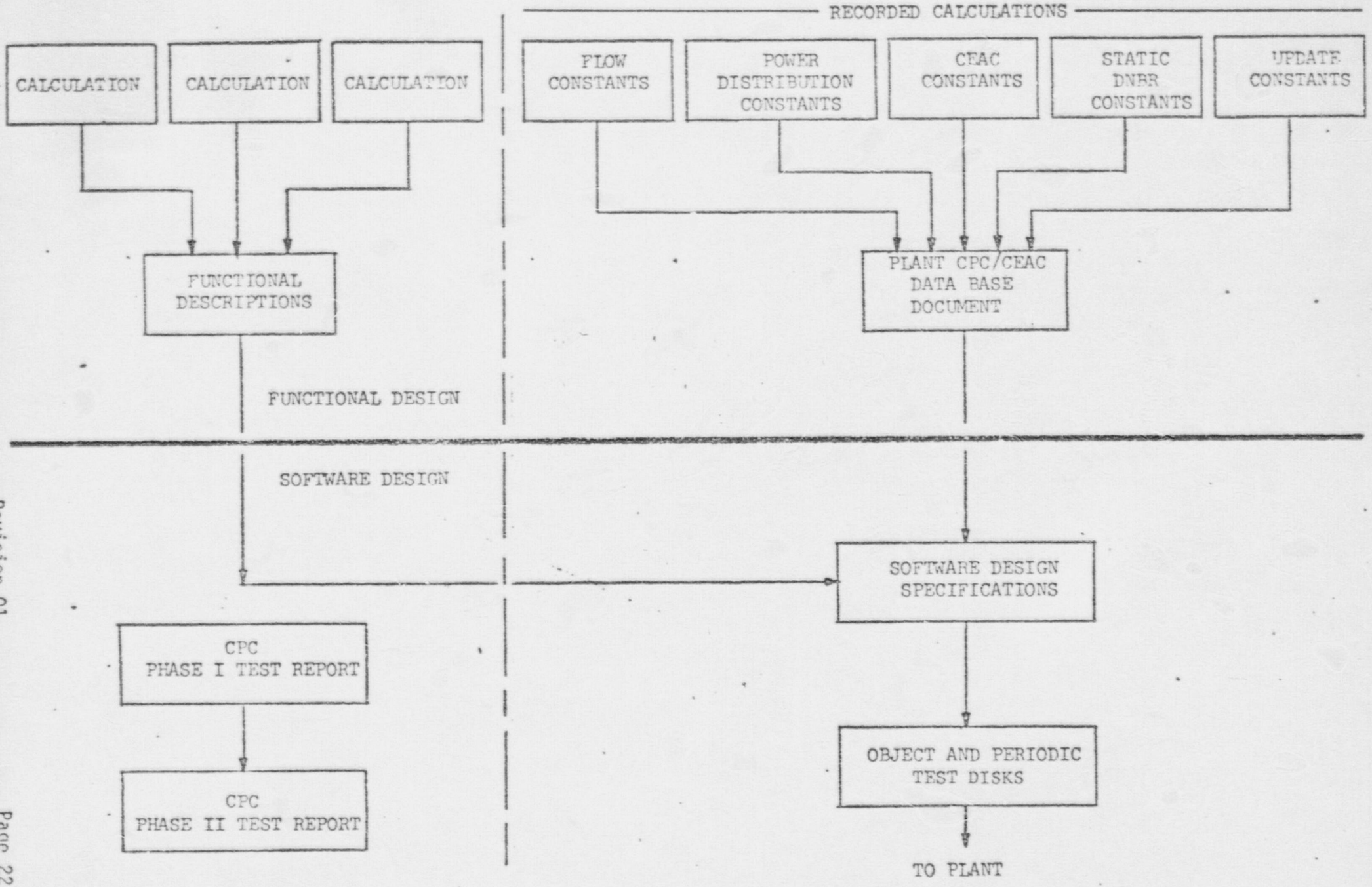


FIGURE 1.1-1

CPC SOFTWARE DESIGN QA DOCUMENTS

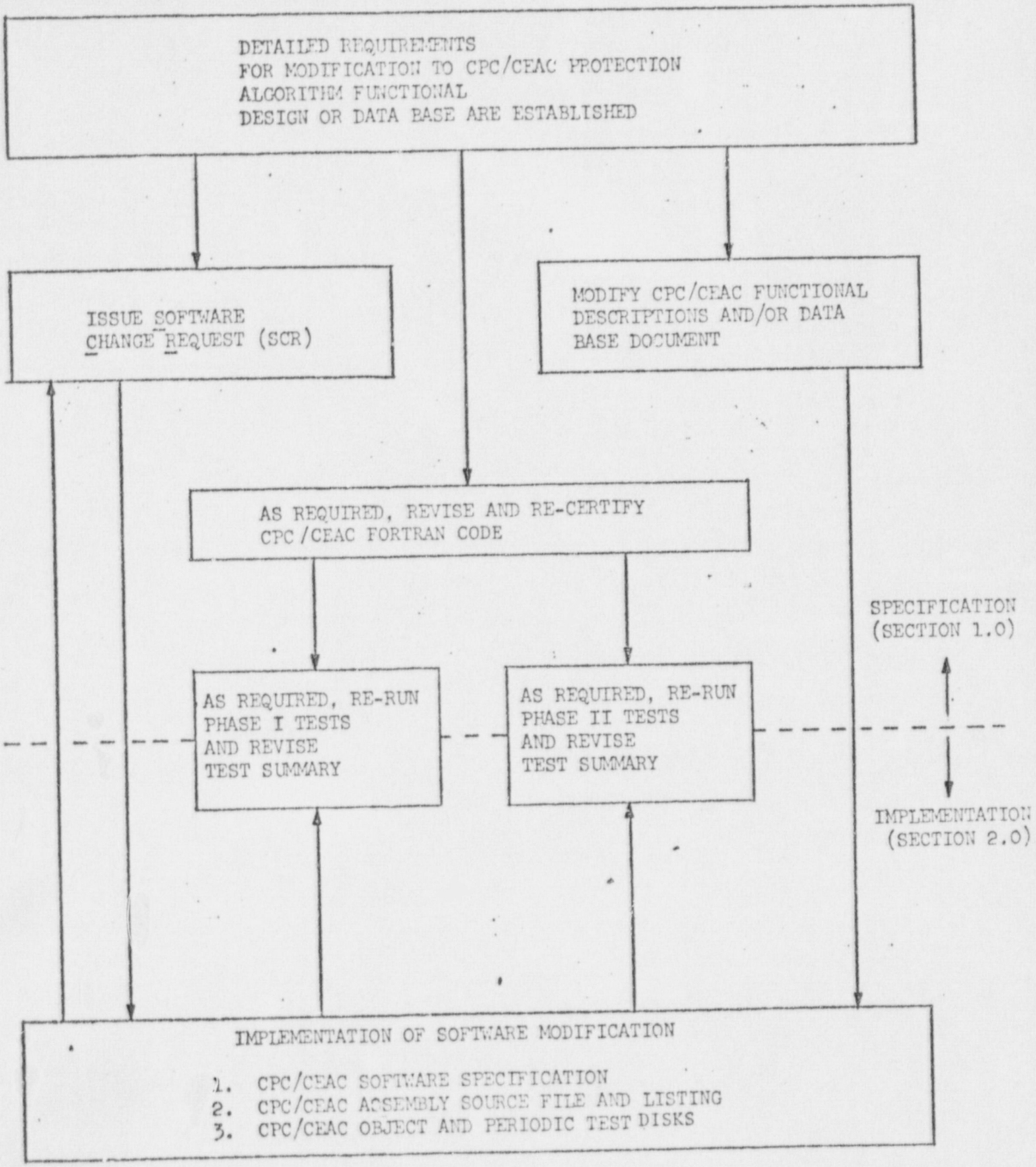


FIGURE 1.3-1

CPC/CEAC PROTECTION ALGORITHM FUNCTIONAL DESIGN
AND DATA BASE MODIFICATION BLOCK DIAGRAM

SHEET___OF___

Change #:

Date:

Plant(s):

CPC SOFTWARE CHANGE REQUEST

I. Originator:

II. References:

III. Summary of Change: (use additional sheets if necessary)

IV. Program(s) Affected:

V. _____ _____

Functional Design Group Approval | Software Design Group Approval

_____ _____

Implemented by | Implementation Reviewed by

Figure 1.3-2

SHEET___OF___ SCR#:

Date:

VI. FUNCTIONAL DESIGN DOCUMENTS AFFECTED

Plant: _____	Rev	Yes	No	Date Completed
A CPC Functional Description				
B CEAC Functional Description				
C Plant Data Base Document				
D FORTRAN Simulation Code				
E Phase II Test Report				
F Recorded Calculations:				

Figure 1.3-2. (Cont.)

Sheet ___ of ___

Figure 1.3-3

Date: _____

Plant: _____

Initials: _____

CHANGE APPLICABILITY FORM
DNBR/LPD CALCULATOR SYSTEM

Applicable Software Item	Change Completion Date and Initials			
	Listing	Source	Object	Specification

2.0 IMPLEMENTATION, DOCUMENTATION AND TESTING OF SOFTWARE CHANGES

2.1 PURPOSE, SCOPE AND APPLICABILITY

2.1.1 Purpose

The purpose of this section is to specify the steps required to test and document a design modification to the CPC/CEAC system software. Adherence to this procedure is intended to avoid the introduction of errors during the revision of a quality assured software system and its related documentation.

2.1.2 Scope

This section covers the implementation of a software design change. The specific activities include modification of the system software specifications including program equations, data base, flowcharts, the Phase I Test Report, and the Phase II Test Report. Requirements are included for documentation and testing of software changes.

2.1.3 Applicability

Application of the procedures covered in this section is mandatory for modifications to CPC/CEAC software systems subsequent to the initial design qualification of the software system. A prerequisite

for application of this procedure is that all applicable system design documents have been completed in accordance with Reference 2.2.1.

2.2 REFERENCES FOR SECTION 2.0

2.2.1 Quality Assurance of Design Manual for C-E Nuclear Power Systems.

2.2.2 Core Protection Calculator System Phase I Design Qualification Test Report, CEN-72(A)-P, October, 1977.

2.2.3 Core Protection Calculator Single Channel Qualification Test Report, CEN-71(A)-P, Supplement 1-P, September, 1978.

2.2.4 Core Protection Calculator Software Change Procedure Supplement CEN-39(A)-P Rev. 01, Supplement 1-P. September, 1978.

2.3 SOFTWARE CHANGE PROCEDURE

2.3.1 Origin of Software Changes

2.3.1.1 SCR Log

A software change originates when either the responsible functional design group or software implementation group identifies the need for a software change and obtains a unique software change request (SCR) number by entering a line in the SCR Log. The SCR Log is a formal notebook containing for each SCR:

1) The change number assigned to the SCR. SCR numbers shall be assigned in increasing consecutive numerical order.

2) The date on which the SCR is entered in the log.

3) The originator of the SCR.

4) The subject of the SCR. A descriptive phrase defining the change to be requested.

5) The initials of the implementor of the SCR and the date completed (i.e. programmed and debugged).

2.3.1.2 Software Change Request

After obtaining an SCR number, the cognizant engineer in the originating group generates a Software Change Request, the contents of which are discussed in Section 1.3.2.2 of this document.

2.3.2 Implementation of Software Changes

2.3.2.1 SCR Signoff

All SCRs will be signed off by the group supervisor or designate responsible for the software system requiring modifications. This signoff is to acknowledge receipt of the SCR.

2.3.2.2 SCR Review

A signed-off SCR will be forwarded to a system cognizant engineer who evaluates the change for overall system impact and reports confirmed or potential problems to the responsible supervisor who will seek their resolution. If the cognizant engineer receives additional information verbally, it will be recorded on the SCR and initialed by the cognizant engineer. The verbal information will be highlighted by underlining or bracketing to facilitate validation by the originating group when the annotated SCR is returned.

2.3.2.3 SCR Implementation

The implementor will prepare, design, code, and implement the Software Change Request using the following procedures the details of which are contained in Reference 2.2.4.

When an SCR is ready for implementation, the implementor will prepare the software change package by filling out and attaching a Change Applicability form (Figure 1.2-3) for each plant to which the SCR pertains. The engineer must list those items which are affected by the SCR on the Change Applicability form and must date and initial each item when the required changes are completed.

If the implementor determines that additional information is required, he will contact the originator to obtain the required clarification. All pertinent information received verbally will be recorded on the SCR and initialed by the engineer.

The software change package will be maintained by the implementor until the change has been implemented and tested. The package will then be returned to the originator for review. The implementation group will file the package in the CPC design file.

After preparing the software change package, the engineer will design and implement the required change as follows:

1) The most recent working copy of the affected calculation descriptions, flowcharts, input/output lists, variable lists, EQU lists, and constant lists of the System Software Specification will be marked up to reflect the change and initialed by the engineer.

2) The required coding changes will be marked on the most recent working copy of the assembly listings. When a new listing is created, the previous version of the listing will be marked "superseded."

3) The Phase I Test Cases will be reevaluated as detailed in Section 2.4 of this document.

4) If the change affects scaled-fixed-point coding, the appropriate scaling recorded calculations will be revised accordingly.

The software change will then be incorporated in a specification revision. The revised software design will then be independently reviewed in accordance with Reference 2.2.1.

All CPC/CEAC source files affected by the change will be updated. In the course of coding the changes, the following documentation will be performed:

1) A revision to CPC software consists of one or more SCRs.

In the initial updating of a source file during a revision, the revision number of the source file is incremented by 1.00 and the decimal part of the number is reset to .00. In the implementation of subsequent SCR's within this revision, which result in another updating of this source file, the revision number of the source file is incremented by 0.01.

2) An "SCR Implementation Record" line will be inserted into the source file. This line is a comment of the form:

*REV n.nn, SCRs: i, j, k, ...

where: n.nn is the new revision number

i, j, k are the SCR numbers implemented in this revision.

These lines will never be removed from the file and will thus provide a running account of the SCRs implemented in a particular program.

The updated CPC/CEAC source files are then assembled to create object modules.

To complete a software change package and the implementation of an SCR, the implementor will fully debug the generated object module(s). If a program error is detected during debug it will be corrected and all affected documentation, including the software

change package, will be modified as required. The revision level, however, will not be incremented for changes to correct errors in the SCR implementation uncovered during debug testing.

When the object module has been fully debugged, the implementor will complete the change package by attaching the debug test cases, filling in the applicable columns in the Change Applicability Form and initialing and dating the SCR. The implementor will then return the change package to the cognizant engineer and will initial and date the SCR Log. The cognizant engineer will check the change package for completeness and file the completed package in the system design file.

2.3.2.4 Review of SCR Implementation

An annotated copy of all SCRs will be returned to the originator with copies of the marked up calculation descriptions and flow charts. The originator will then review the implementation and sign off the SCR to complete his design change package.

2.3.2.5 Disk Generation

The implementor will generate new reference and test disks using the following procedure the details of which are contained in Reference 2.2.4.

To generate a new reference disk, load modules will be created from object code. These load modules will be written to disk in 4KB blocks. Each block will occupy one disk track. A disk track may be generated entirely from revised object code, copied entirely from the reference disk or copied from the reference disk with a revised object overlay. All tracks of a new disk will be initialized to all zeroes before any programs are written to disk.

Phase I and Phase II testing shall be performed on the new disk where required in accordance with Sections 2.4 and 2.5 of this document. A disk which has satisfactorily undergone all testing required by the above sections shall become the new reference disk for the software system.

Once a reference disk has been established for the software system a test disk may be generated. This is accomplished by re-assembling all source files for the system to generate new object modules. An entire system load module will be generated from these object modules and written to disk.

When all of the tracks of the test disk have been generated, the test disk will be compared to the reference disk. If differences are encountered, it will be determined whether the error is in the reference disk (i.e. an error missed by required testing) or the test disk. The error will be corrected and the faulty disk will be regenerated in accordance with all applicable procedures for reference or test disk generation and testing. If no errors

are encountered, the listings generated from the assembly of the source files will become the final listings for the current revision of the software system.

Disks for shipment to customer sites for on-line operation will be duplicated from the system reference disk which will be maintained by the Software Design group. The test disks will be similarly generated by duplicating the master test disk generated above. The master test disk will be maintained by the Software Design group.

2.3.3 Quality Assurance of Software Changes

The revised system functional descriptions and/or data base document are the design input to the CPC/CEAC software modifications. Any revisions to CPC or CEAC software are documented by revising the system software specifications. Quality assurance of the final software implementation is established by independent review of the design in accordance with Reference 2.2.1 (QADM) and by Phase I and Phase II design qualification testing described in Sections 2.4 and 2.5 respectively. A flow chart of design and QA activities is shown in Figure 2.3-1.

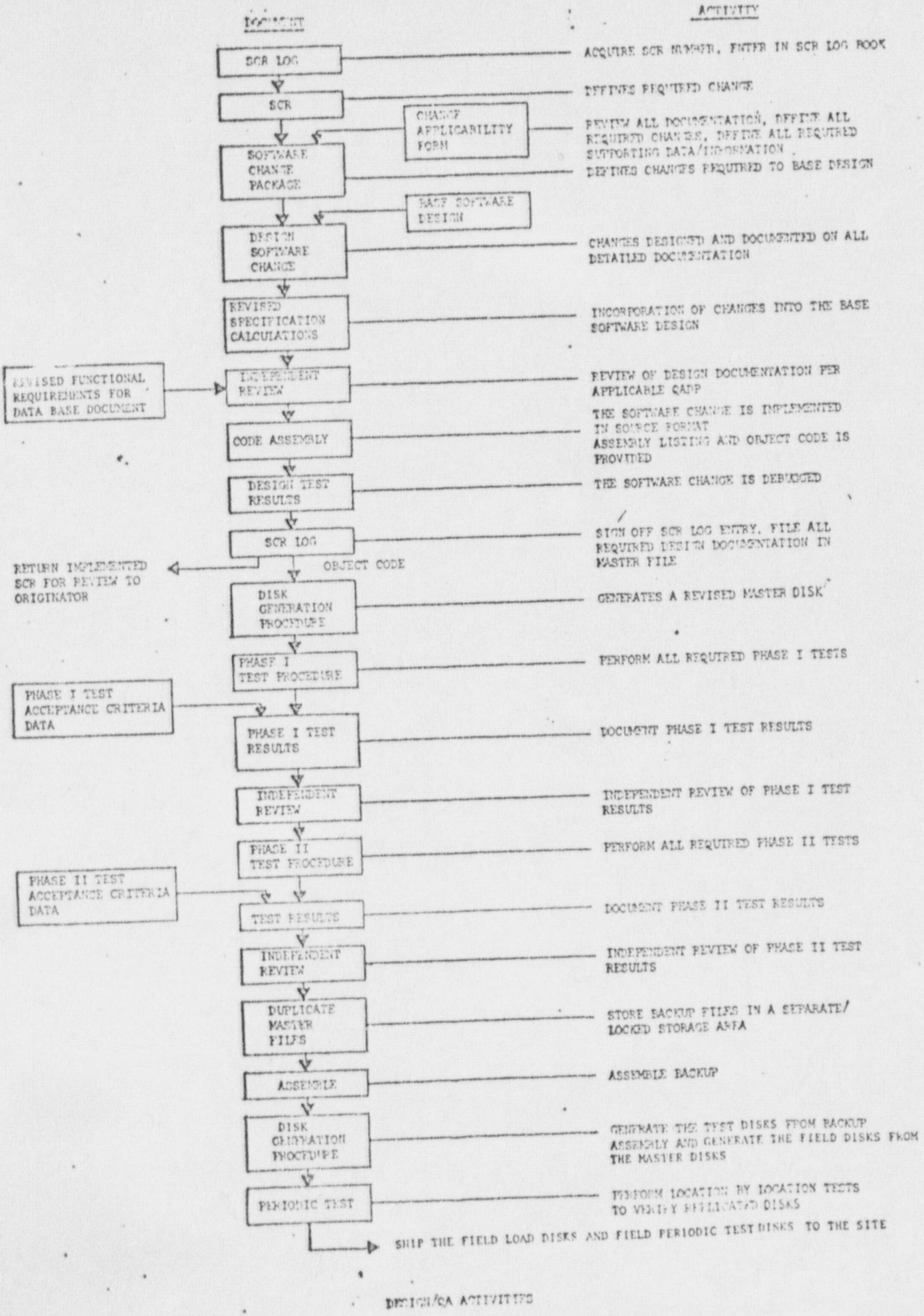


Figure No. 2.3-1

Revision 01

2.4 PHASE I TESTING

Phase I testing will be performed to verify the implementation of modifications to the CPC/CEAC Software. Implementation is defined as the translation of the system functional requirements into modules of machine executable code, and the integration of these modules into a real-time software system. Phase I testing is performed on relatively small, single-entry/single-exit segments of code called modules.

Any modifications to a program's code will require the regeneration of the entire program. All of the modules that were modified will undergo a complete rerun of Phase I testing. Test case inputs will undergo reevaluation such that each functional branch and each instruction in the affected modules is exercised. Program wide test cases will then be run to assure correct generation of the program.

Any modifications to a program data base will require regeneration of the program constants. All of the modules of a program which access the affected constants will undergo a complete rerun of Phase I testing. The program containing these modules will then be Phase I tested with the program wide test case inputs.

All Phase I test documentation will be revised. This includes the test case selection calculations and the Phase I Test Report.

All Phase I Test documentation will be quality assured including independent review in accordance with Reference 2.2.1. Details of Phase I testing are provided in a Phase I Test Procedure.

2.5 PHASE II TESTING

2.5.1 Description of Phase II Testing

Phase II testing consists of the following tests:

(1) Input Sweep Test,
(2) Dynamic Software Verification Test, and
(3) Live Input Single Parameter Test.

These tests are performed on a single channel CPC/CEAC system with integrated software that has undergone successful Phase I testing. The single channel CPC/CEAC system is described in Reference 2.2.3. The objectives of Phase II testing are to:

1. verify that the CPC and CEAC software modifications have been properly integrated with the CPC and CEAC software and the system hardware, and

2. confirm that the static and dynamic operation of the integrated system as modified is consistent with that predicted by design analyses.

The Phase II testing uses the CPC/CEAC FORTRAN simulation code as a basis for comparison. The results of Phase II testing including test case selection and acceptance criteria are included in the Phase II Test Report.

2.5.1.1 Input Sweep Test

The Input Sweep Test is a real-time exercise of the CPC application software and executive software with steady-state CPC input values read from a storage device. The Input Sweep Test has the following objectives:

(1) To determine the processing uncertainties (of the CPC algorithms as used in the CPC/CEAC system hardware) that are inherent in the CPC design. Processing uncertainties are defined as those resulting from differences in machine precision between the system hardware and the CDC 7600 (on which the FORTRAN code is executed). The processing uncertainties will be factored into acceptance criteria for the Dynamic Software Verification Test, and Live Input Single Variable Test, and into CPC data base constants affected by these uncertainties.

(2) To verify the ability of the CPC algorithms used in the system hardware to initialize to a steady state after an Auto-Restart for each of a large number of input combinations within the CPC/CEAC operating space.

(3) To complement Phase I module testing by identifying any abnormalities in the CPC algorithms used in the system hardware which were not uncovered previously.

This test is performed using the CPC software overlayed with Input Sweep software, using unused portions of memory and a portion of the Executive Initialization Task for the overlay; the CPC application programs to be tested remain unaltered. The Input Sweep software allows test case input values to be read from a storage device. A large number of static test cases are selected which cover the range of CPC operation, including the upper and lower limits of all CPC inputs. These test cases are allowed to initialize, after which calculated parameter values are compared off-line with expected values generated with the CPC FORTRAN code.

2.5.1.2 Dynamic Software Verification Test (DSVT)

The Dynamic Software Verification Test is a real-time exercise of the CPC application software and executive software with transient CPC input values read from a storage device. The DSVT has the following objectives:

1. To verify the dynamic response of the integrated CPC software is consistent with that predicted by design analyses, and

2. To supplement design documentation quality assurance, Phase I module tests, and input sweep tests in assuring correct impementation of software modifications.

The Dynamic Software Verification Test (DSVT) is similar to the Input Sweep Test in that special software is overlayed on portions of the executive software and on unused portions of memory, leaving the CPC application programs unaltered. Like Input Sweep, CPC input values are read from a storage device. However, the Input Sweep Test uses CPC input values that do not vary, such that each test case initializes to a steady-state condition. DSVT (after initialization) uses time-variant test case input values to more thoroughly exercise "dynamic" portions (recursive or time-derivative equations) of the CPC software.

2.5.1.3 Live Input Single-Parameter (LISP) Test

The LISP test is a real-time exercise of the CPC/CEAC application and executive software, with transient CPC/CEAC input values generated from an external source and read through the CPC/CEAC input hardware. The objectives of this test are:

1. To verify that the dynamic response of the integrated CPC/CEAC software and hardware is consistent with that predicted by design analyses.

2. To supplement design documentation quality assurance, Phase I module tests, input sweep tests, and DSVT testing in assuring correct implementation of software modifications.

3. Evaluate the integrated hardware/software system during operational modes approximating plant conditions.

Unlike Input Sweep and DSVT tests, the LISP test employs no special test software for generating test case input in the CPC/CEAC system. Dynamic test case inputs are generated by external test equipment to produce "live" analog and digital signals that are input to the CPC/CEAC input processing hardware.

Since multi-variable transients are already performed in the DSVT test, each LISP test case varies only one input parameter, holding other inputs at constant values. Limiting live input transients to single variables is also based upon minimizing test uncertainties due to input signal generation and transmission, enabling a higher degree of precision in test acceptance criteria. All dynamic portions of the CPC/CEAC algorithms will be exercised with LISP tests.

2.5.2 Phase II Test Case Selection

2.5.2.1 Input Sweep Test Case Selection

The objective of the Input Sweep Test case selection is to define a minimum of 500 cases which cover the region of CPC operation, including upper and lower limits on CPC inputs. Variations in values of selected addressable constants are included. The selection is designed to ensure a high confidence level in determination of processor uncertainty, anomaly detection, and initialization capability. The spectrum of selected cases must therefore be based upon a sufficiently wide spectrum of cases covering the range of CPC inputs, allowing for mechanistic constraints associated with plant operation (i.e., the 3 ex-core detector inputs must be a matched set, hot leg temperature must be greater than cold leg temperature, etc). Within these constraints, test case selection may be random or systematic. A typical test case selection process is described in Section 4.2 of Reference 2.2.2.

2.5.2.2 Dynamic Software Verification Test Case Selection

The objective of the selection process is to employ cases that adequately exercise dynamic portions of the application software, with emphasis on thoroughly testing those portions of software that have been modified.

Table 2.5-1

Dynamic Software Verification Test Cases

The DSVT test cases for a given software modification will, as a minimum, consist of cases 1, 7, 12, 16 and 25 from Table 2.5-1, which includes design basis events. Depending on the nature and complexity of the modification, additional cases from Table 2.5-1 will be selected as necessary to exercise the particular portions of the CPC software that have undergone modification.

2.5.2.3 Live Input Single-Parameter Test Case Selection

The objective of LISP test case selection is to demonstrate that the integrated CPC/CEAC hardware/software system functions as designed in response to externally-generated transient input signals in a real time environment. Test cases will be variations of single variables encompassing, as a minimum, cases 17 through 21 of Table 2.5-1. In addition, major aspects of the operator's module operation, particularly those accessing portions of modified applications software, will be exercised.

2.5.3 Generation of Acceptance Criteria

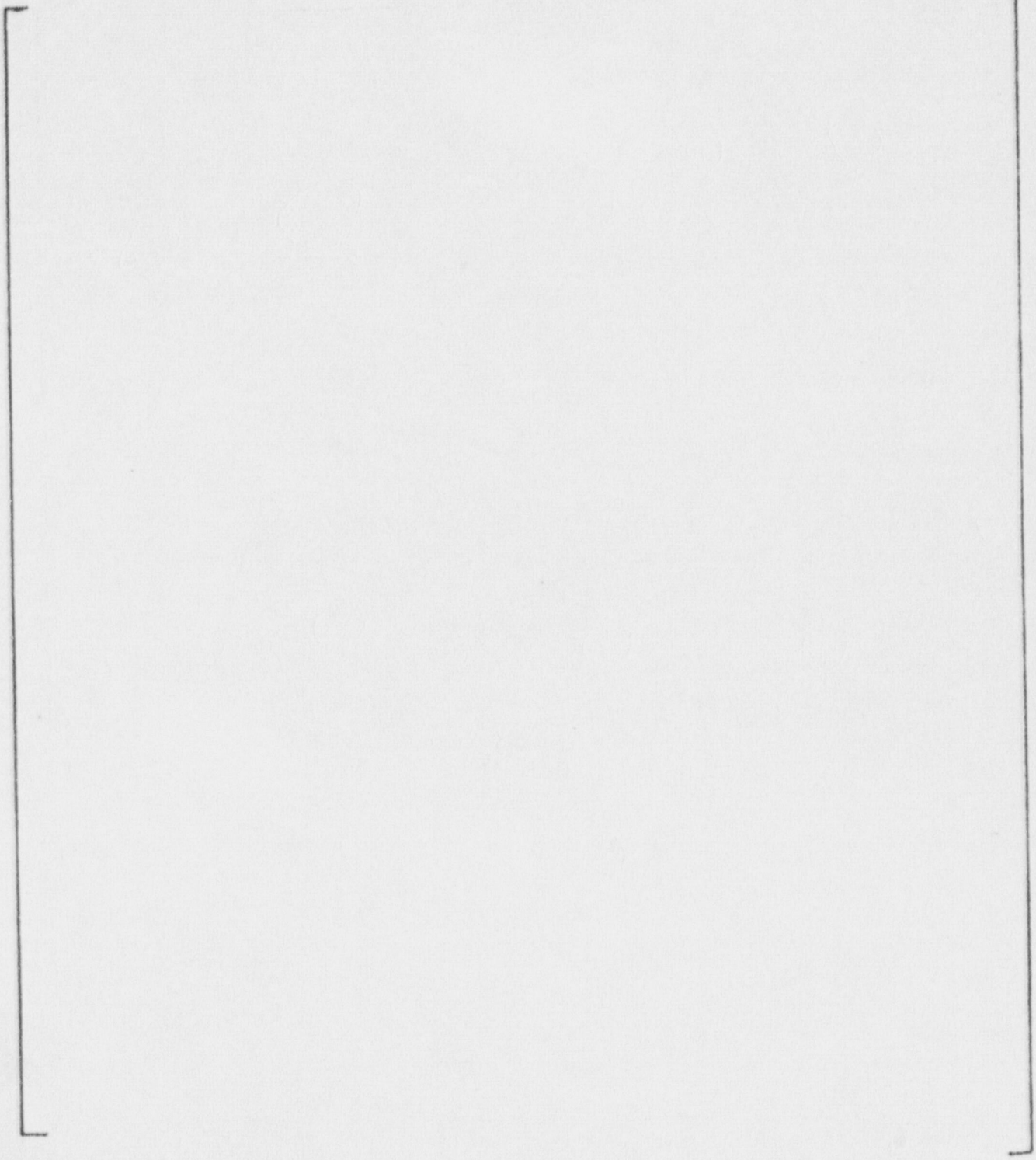
2.5.3.1 Input Sweep Acceptance Criteria

As stated in Section 2.5.1.1, the Input Sweep Test has three objectives:

(1) Determination of processing uncertainties.

(2) Verification of initialization.

(3) Identification of abnormalities.



2.5.3.2 Acceptance Criteria for Dynamic Software Verification Test

[

]

2.5.3.3 Acceptance Criteria for Live Input Single Parameter Testing

[

]

2.6 PROCESS NOISE EVALUATION

Software changes will be evaluated for their potential to significantly alter (with respect to the previously qualified software) the CPC/CEAC System response to plant process noise. The evaluation will be performed during the functional development of the software modification, and will be supplemented by observations of system behavior during Phase II testing. Particular attention during such evaluations will be given to changes in data or equations involving (1) recursive or time derivative calculations, (2) deadbands, nodes, range limits, or discontinuities, and (3) the static sensitivity of DNBR, LPD, or quality margin with respect to process inputs. If the evaluation indicates that the potential for significant alteration of the noise response exists, the modified CPC/CEAC software will be evaluated by testing to verify that the altered noise response is acceptable.

In the event that noise testing is necessary, the tests will be performed by adding noise signals to the nominal (static or dynamic) values of selected simulated process inputs to the Single Channel System. These inputs include pumpspeeds, temperatures, pressure, and excore detector signals. To the extent practical, the noise signals that are used must be the best available representation of actual plant noise, with the preferred source being FM tape recordings of in-plant noise on CPC/CEAC process inputs. Recorded noise from in-plant and CPC/CEAC process inputs may be supplemented with synthesized noise

(having frequency and amplitude characteristics similar to acutal plant noise), or recorded noise from non-CPC/CEAC plants.

The noise generation capability of the Single Channel Test Facility includes a 16-channel FM tape recorder and appropriate amplification equipment to enable the CPC/CEAC single channel system to use recorded in-plant noise as part of the noise evaluation. Random noise synthesis capability will be available with a broadband noise generator capable of well defined frequency spectra, frequency cutoffs, and power output. In addition, systematic noise synthesis capability will be available with two sweep generators that cover a broad range of frequencies and waveforms.

Acceptance of the noise response test results is based upon (1) determination that any non-conservative (relative to the noise-free condition) DNBR or LPD values are accommodated by measurement uncertainty allowances in the protection algorithms, and (2) determination that the plant availability will not be unacceptably impacted by the altered noise response.

3.0 GENERATION OF PROJECT SOFTWARE AND SOFTWARE DESIGN DOCUMENTATION

3.1 PURPOSE, SCOPE, AND APPLICABILITY

3.1.1 Purpose

The purpose of this procedure is to specify the steps required to generate CPC/CEAC project software and software design documentation. Adherence to this procedure is intended to avoid errors in the reproduction of a quality assured software system and its related software design documentation.

3.1.2 Scope

This procedure covers the generation of CPC/CEAC project software and software design documentation. The specific activities include generation of project software and design documentation through reproduction and modification of generic software.

3.1.3 Applicability

These requirements are to be followed for the generation of new CPC/CEAC project software and documentation. A prerequisite for application of this procedure is that all applicable generic software design documents and software have been completed in accordance with Reference 3.2.1.

3.2 REFERENCES FOR SECTION 3.0

3.2.1 Quality Assurance of Design Manual for C-E Nuclear Power Systems.

3.2.2 Software Implementation Procedure for CPC/CEAC Protection Algorithm, CEN-39(A)-P, Supplement 1-P, September, 1978.

3.3 SOFTWARE DESIGN DOCUMENTATION GENERATION

The following is a list of software design documentation which will be generated for each project.

CPC Software Specification

CEAC Software Specification

Executive Software Specification

CPC Functional Design Specification

Data Base Document

CEAC Functional Design Specification

Phase II Test Procedure

Phase I Test Procedure

Executive Phase I Test Procedure

Flow Test Cases

Update Test Cases

Power Test Cases

Static Test Cases

Trip Sequence Test Cases

Common Subroutines Test Cases

Penalty Factor Test Cases

Position Display Test Cases

Phase I Test Report

Phase II Test Report

Test & Certification Analysis of
the Core Protection Calculator
System Software for Channels A & B

3.4 PROJECT DISK GENERATION

A project disk will be generated using the following procedure the details of which are contained in Reference 3.2.2.

A new disk is formatted and initialized. The source and object files on the generic disk are then copied to the project disk.