ARKANSAS NUCLEAR ONE - UNIT 2

DOCKET 50-368

CEN-39(A)-NP

SUPPLEMENT 1-NP
REVISION 01

CPC PROTECTION ALGORITHM SOFTWARE CHANGE PROCEDURE SUPPLEMENT

JANUARY 5, 1979

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

Revision 01

7907120606

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 information, appartus, method or process disclosed in this report.

TABLE OF CONTENTS

Section	Title		Page No.
1.0	IMPLEMENTATION OF SOFTWARE CHANGES		6
	1.1	PURPOSE	6
	1.2	REFERENCES	6
	1.3	IMPLEMENTATION OF AN SCR	7
	1.3.1	Preparation of Software Change Package	7
	1.3.2	Design of Software Changes	8
	1.3.3	Coding of Software Changes	9
	1.3.4	Implementation and Assembly of Software Changes	10
	1.3.5	Debug of Software Changes	11
	1.4	QUALITY ASSURED OBJECT DISC GENERATION PROCEDURE	20
	1.4.1	Input	20
	1.4.2	Output	20
	1.4.3	Procedures	21
	1.4.4	Generation of Master Test Disc	23
	1.4.5	Generation of Discs for On-Line Operation	23
	1.5	CONTROL OF QUALITY RECORDS	47
2.0	PROJECT DISC GENERATION PROCEDURE		89
	2.1	PURPOSE	89
	2.2	REFERENCES	89
	2.3	PROJECT DISC GENERATION	90

LIST OF APPENDICES

Appendix	Title	Page No.
1A	Software Organization and Design Considerations	48
1B	CSS Files	58
1C	Source Updater Commands	60
1D	Automated CPC Integration	71
1E	Periodic Test Disc Generation	81
2A	Project Disc Generation	95

LIST OF FIGURES

Figure	Title	Page No.
1.3-1	Change Applicability Form	14
1.4-1	Data Flow for Disc Generation	25
1.4-2	CPC Disc Generation Log	26
1.4-3	CPC System	38
1.4-4	CEAC System	40
1.4-5	CPC Test System	42
1.4-6	CPC Files	44
1A-1	CPC Program Title Block	57

LIST OF TABLES

Table	Title	Page No.
1.3-1	DNBR/LPD Calculator System	15

LIST OF ATTACHMENTS

Attachment	Title	Page No.
1.3-1	CPC Source Update Checklist for Batch Processing	12
1.4-1	CPC Object Tape Generation	29
1.4-2	CPC Disc Generation	31
1.4-3	CPC Disc Duplication	37
2.3-1	CPC Project Disc Generation Checklist	91

ABSTRACT

This document presents procedures to be followed during the process of implementation of modifications to the CPC/CEAC software. Additionally, this document presents procedures to be followed during the process of generating CPC/CEAC software for application to a new project. Section 1.0 of this document contains procedures describing the process of implementing changes to the CPC System software and data base. Section 2.0 contains procedures applicable to the process of generating new project software.

1.0 IMPLEMENTATION OF SOFTWARE CHANGES

1.1 PURPOSE

The purpose of Section 1.0 is to present the procedures required to implement a design modification to the CPC/CEAC software system. Adherence to this procedure is intended to avoid the introduction of errors during the revision of a quality assured software system.

1.2 REFERENCES

1.2.1 Software Change Procedure for CPC/CEAC Protection Algorithms, CEN-39(A)-P, Revision 02.

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

1.2.3 Series 40 Disc Drive Maintenance Manual, Diablo Systems Incorporated, Publication No. 81601

1.2.4 CPC Disc Utility Program User's Manual

1.2.5 OS/32 MT Operator's Reference Manual, Interdata Publication No. 29-574, Section 7.5

1.2.6 ANSI Standard Flowcharting Symbols and Their Use in Information Processing, X.35.

1.3 IMPLEMENTATION OF AN SCR

1.3.1 Preparation of Software Change Package

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 1.3-1) for each plant to which the SCR pertains. Table 1.3-1 lists all elements of the software system. The implementor 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 implementor.

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. Upon completion of his review, the originator will transmit the package to the cognizant engineer who will then file the Software Change Package in the CPC design file.

1.3.2 Design of Software Changes

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. To ensure that the listings being used are the latest version, the date and time on the listing will be checked against the latest date and time posted for that listing in the automated CPC log CPCLOG.DAT. Also, when a new listing is created, the previous version of the listing will be marked "superseded."

3) The Phase I Test Cases will be modified as required to meet the requirements of Reference 1.2.1.

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

The software change will be designed to be consistent with Appendix 1A of this document entitled "Software Organization and Design Considerations."

The software change will then be incorporated in a specification revision. The revised software design will then be independently reviewed in accordance with Reference 1.2.2.

1.3.3 Coding of Software Changes

The CPC Source Update Checklist (Attachment 1.3-1) outlines the procedure to be used in the coding, implementation, and assembly of software changes. The required coding changes, as marked on the assembly listings, will be converted into a series of Source Updater Utility program commands (Appendix 1-C) and punched on cards in order of ascending sequence numbers. The last cards of the deck must be the "ENDUP" card followed by a "/&" card. In the course of coding the changes, the following documentation will be performed.

1) A revision to CPC software consists of one or more SCR's.

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 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.

1.3.4 Implementation and Assembly of Software Changes

The software change will be implemented in the appropriate source file using the UPDATE command (Appendix 1B). The appropriate project disc is mounted (Reference 1.2.3) and marked on. The Source Update command deck is placed in the card reader and the UPDATE command is entered on the console. After successful termination of the procedure, the Source Update command listing will be attached to the software change package.

Should a new source file be required as a result of the implementation of an SCR, the INITIAL command (Appendix 1B) will be used. The procedure for using the INITIAL command is the same as the Source Update procedure with the following exceptions. "INITIAL" is entered instead of "UPDATE"; a source deck replaces the Source Updater command deck; and "/*" is the last card in the source deck.

1.3.5 Debug of Software Change

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 originator and will initial and date the SCR Log. The originator will review the change package and transmit the package to the cognizant engineer. The cognizant engineer will check the change package for completeness and file the completed package in the system design file.

Attachment 1.3-1

CPC Source Update Checklist for Batch Processing

1) Punch the Source Update command deck
   a) Increment revision level on "PROG" card
   b) Add "SCR Implementation Record" line
   c) Place commands in order of ascending sequence numbers
   d) "ENDUP" card followed by a "/&" card are last cards in command deck

2) Mount the appropriate project disc (Reference 1.2.3)

3) Mark disc:,ON
   where disc is the device the project disc was mounted in.

4) Place the Source Update command deck in the card reader
   a) Depress RESET on the card reader.

5) Enter UPDATE (project volume name), (file name), (rev. #), (initial)
   a) If error occurs in execution of Source Updater commands, enter ..BG; $CLEAR; CANCEL. The run is aborted. Correct the deck and continue at step 4.

6) Debug the software change

7) Attach the Source Updater command listings to the software change package

8) Initial and date the appropriate columns in the Change Applicability Form (Figure 1.3-1)

9) Initial and date the SCR Log

Sheet ___ of ____

Figure 1.3-1

SCR#: _____

Date: _____

Plant: _____

Initials: _____

CHANGE APPLICABILITY FORM
DNBR/LPD CALCULATOR SYSTEM

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

Table 1.3-1

DNBR/LPD CALCULATOR SYSTEM

No.	Software Item
1	[
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	]

Table 1.3-1 (Cont.)

DNBR/LPD CALCULATOR SYSTEM

No.	Software Item
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	

Table 1.3-1 (Cont.)

DNBR/LPD CALCULATOR SYSTEM

No.	Software Item
29	[redacted]
30	[redacted]
31	[redacted]
32	[redacted]
33	[redacted]
34	[redacted]
35	[redacted]
36	[redacted]
37	[redacted]
38	[redacted]

Table 1.3-1 (Cont.)

DNBR/LPD CALCULATOR SYSTEM

No.	Software Item
39	[
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	]

Table 1.3-1 (Cont.)

DNBR/LPD CALCULATOR SYSTEM

No.	Software Item
50	
51	
52	
53	
54	
55	
56	
57	
58	

1.4 QUALITY ASSURED OBJECT DISC GENERATION PROCEDURE

This procedure applies to the integration of object modules and the generation of load modules on flexible discs. Portions of this procedure have been automated as described below. Details of the automated aspects are outlined in Appendix 1D.

1.4.1 Input

The input to this procedure shall be from two sources:

1) Revised (i.e. newly assembled) object code.

2) Quality Assured object code from a previously generated and tested reference disc.

1.4.2 Output

The output of this procedure shall be a core image load module on mass storage media. The load module shall be tested as required by Reference 1.2.1.

1.4.3 Procedures

The core image load module will be written to disc in 4 KB blocks. Each block will occupy one disc track. A disc track may be generated entirely from revised object code, copied entirely from the reference disc, or copied from the reference disc with a revised object overlay.

The CPC Object Tape Generation procedure (Attachment 1.4.1) is used to load the proper object modules, in correct sequence, onto a magnetic tape. It also provides a guide to creating the absolute load module on the CPC Support System.

The CPC Disc Generation procedure (Attachment 1.4.2) is used to create an absolute load module, which is then written to disc to obtain a new reference disc.

Figure 1.4-1 shows the data flow for disc generation. The following paragraphs describe the processes involved in this figure. All tracks of a new disc will be initialized to all zeroes before any programs are written to the disc.

Source files shall be modified and assembled on the Software Development System (Inte data 7/32) according to the procedures described in Section 1.3 of this document to produce one object module per source file on a project disc. The object modules will be copied onto magnetic tape in the order which they will be

loaded in core. Magnetic Tape is used as it is the only medium common to the Software Development System and the CPC Support System. The object module of any program containing external references or entry points (i.e., containing CAL EXTRN or ENTRY statements) must be linked to the object modules of all programs resolving these references using the Interdata Library Loader. To accomplish this, these object modules must be contiguous on the tape. This is assured by use of the CPC Object Tape Generation Procedure.

Using the Library Loader on the CPC Support System, an absolute load module will be generated on a second tape. This process is described in the CPC Disc Generation Procedure. (Attachment 1.4.2).

The load module on magnetic tape and the core image on reference disc shall then be merged onto a target disc using the CPC Disc Generation Procedure.

All disc generations must be logged on the Disc Generation Log (Figure 1.4-2) associated with each disc.

All programs and overlays must undergo Phase I and Phase II testing in accordance with Reference 1.2.1 of this document after being written to disc. A disc which has successfully undergone all testing required by the above reference shall become the new reference disc for the software system.

1.4.4 Generation of Master Test Disc

Once a reference disc has been established for the software system a master test disc may be generated. This is accomplished by restoring the project backup to disc and re-assembling the source files for the system to generate new object modules. The details of this procedure are outlined in Appendix 1E. An entire system load module will be generated from these object modules and written to disc using the Periodic Test Disc Generation Procedure (Appendix 1-E).

When all of the tracks of the test disc have been generated, the master test disc will be compared to the reference disc using the verify feature of CPCDUP.

If differences are encountered the error in the test disc will be determined by examining the contents of the dissimilar tracks and matching the contents to the object code on the final listing of the program contained on that track. The error will be corrected and the faulty disc will be regenerated in accordance with all applicable procedures for test disc generation and testing.

1.4.5 Generation of Discs for On-Line Operation

Discs for shipment to customer sites for on-line operation will be generated using the duplicate and verify features of the CPC Disc Utility Program (Reference 1.2.4). These discs will be

duplicated from the system reference disc which will be maintained by the Software Design group. The test discs will be similarly generated by duplicating the master test disc generated above. The master test disc will be maintained by the Software Design group. The procedure for duplicating a disc is found in Attachment 1.4.3.

Figure 1.4-1

DATA FLOW FOR DISC GENERATION

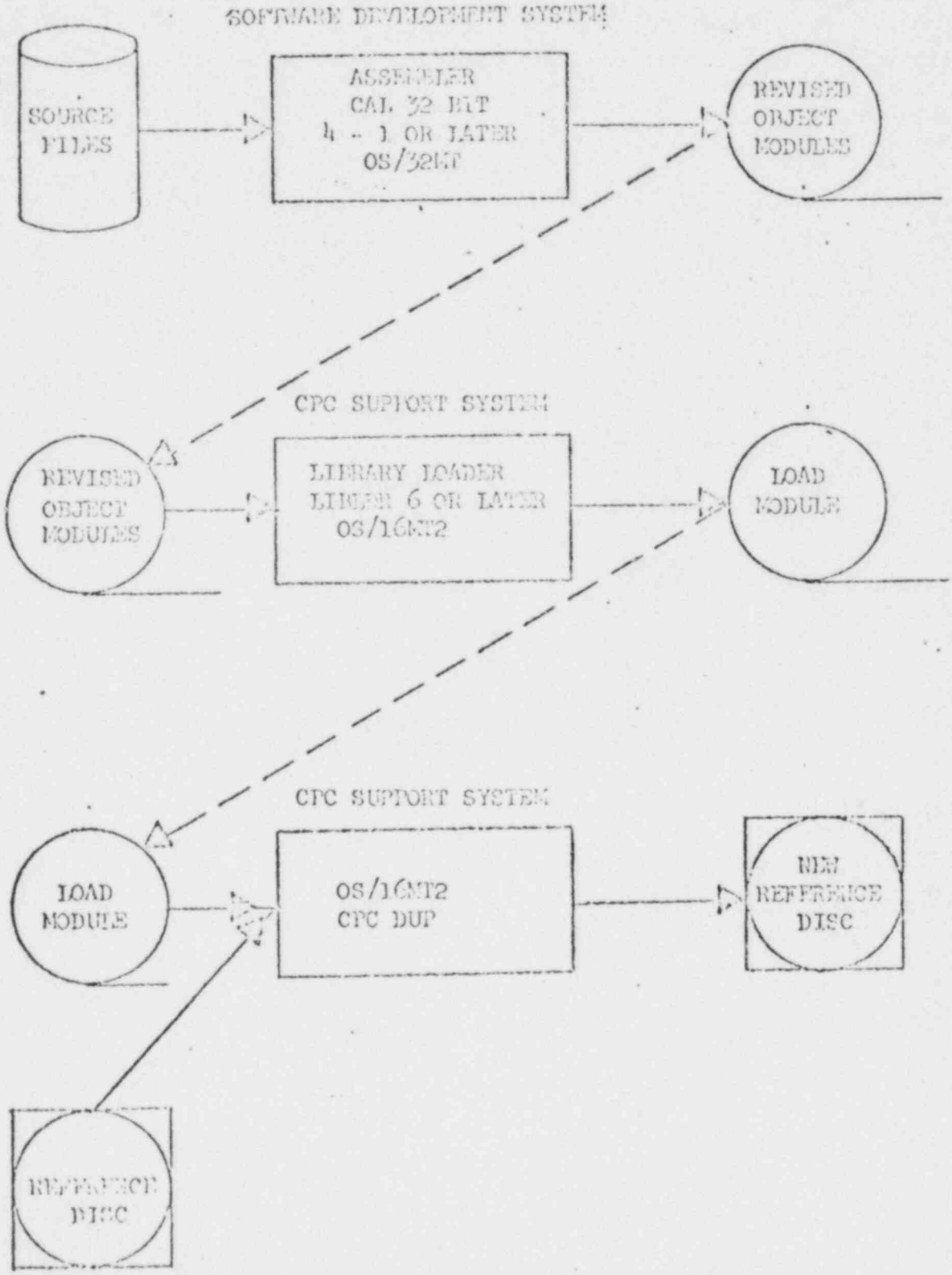


Figure 1.4-2

CPC Disc Generation Log -1

Track	Software	Revision	Date	Engineer
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

Figure 1.4-2 (Cont.)

CPC Disc Generation Log -2

Track	Software	Revision	Date	Engineer
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40				
41				
42				
43				

Figure 1.4-2 (Cont.)

CPC Disc Generation Log -3

Track	Software	Revision	Date	Engineer
44				
45				
46				
47				
48				
49				
50				
51				
52				
53				
54				
55				
56				
57				
58				
59				
60				
61				
62				
63				

Attachment 1.4-1

CPC Object Tape Generation

Purpose: To generate a magnetic tape containing the correct number and sequence of object modules in order to generate a new reference disc.

System Configuration:

[

]

Procedure:

[

]

CPC Object Tape Generation

Attachment 1.4-2

CPC Disc Generation

Purpose: To generate a new reference disc, given an existing reference disc and a tape containing revised object modules.

System Configuration:

[

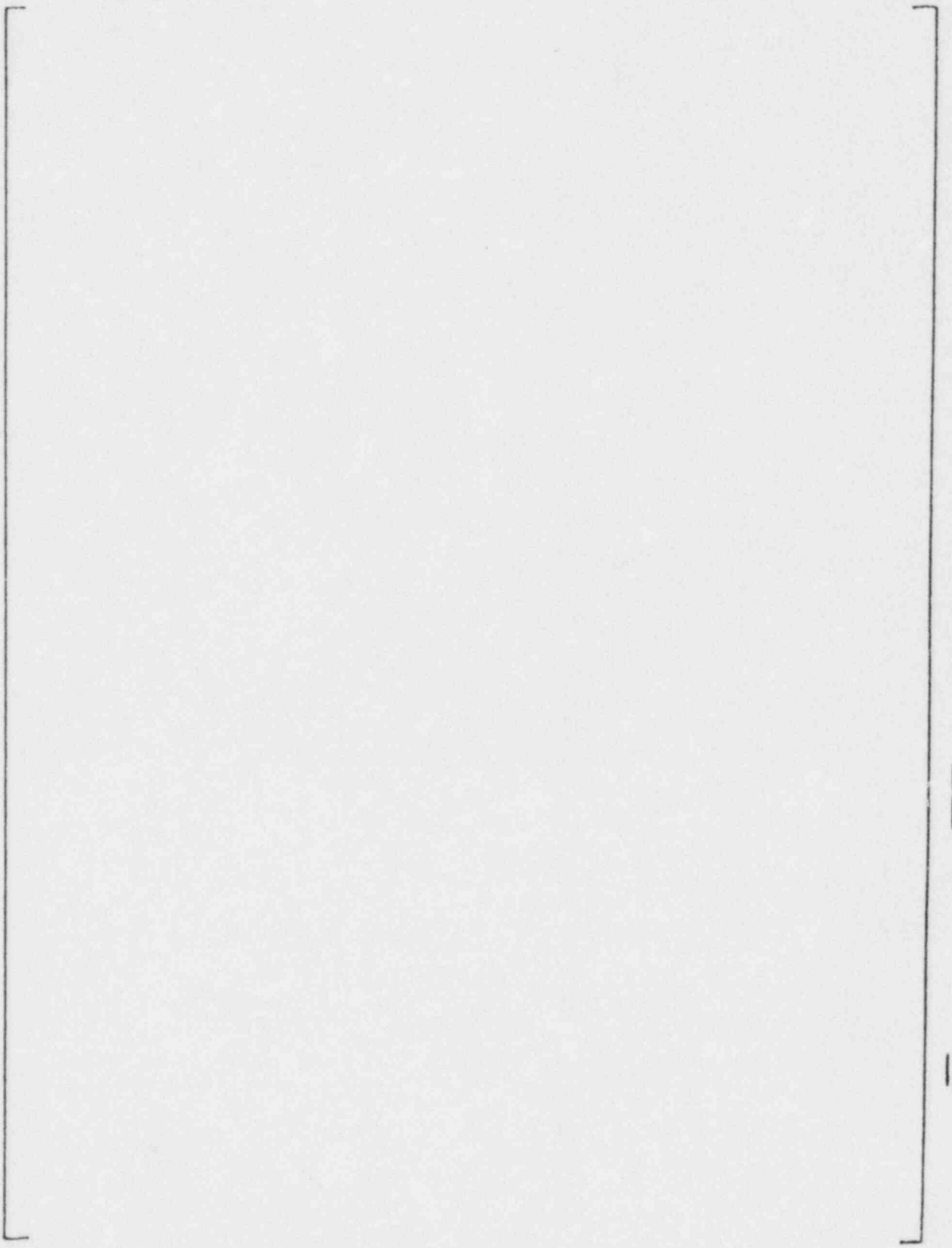
]

Procedure:

[

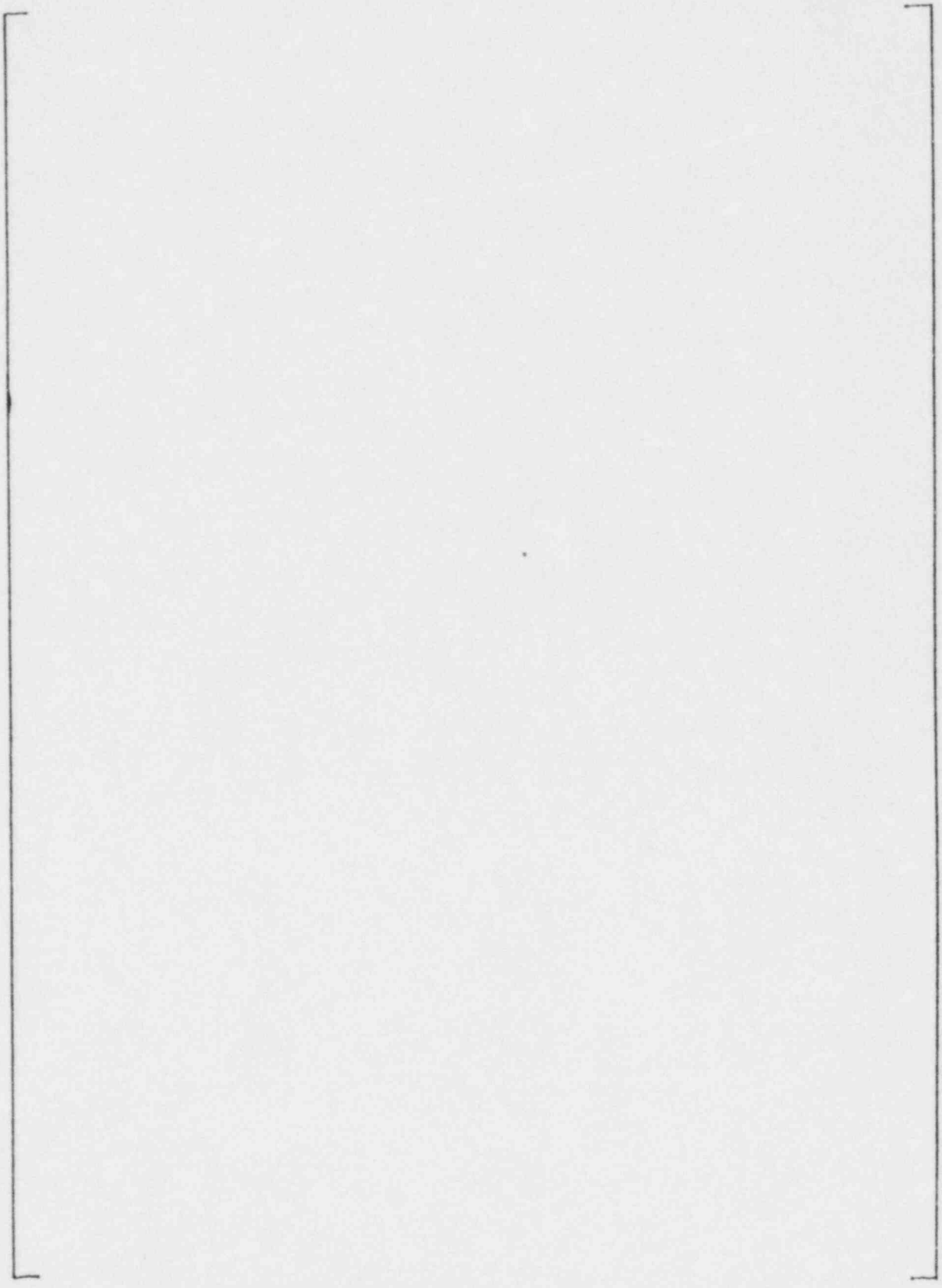
]

CPC Disc Generation



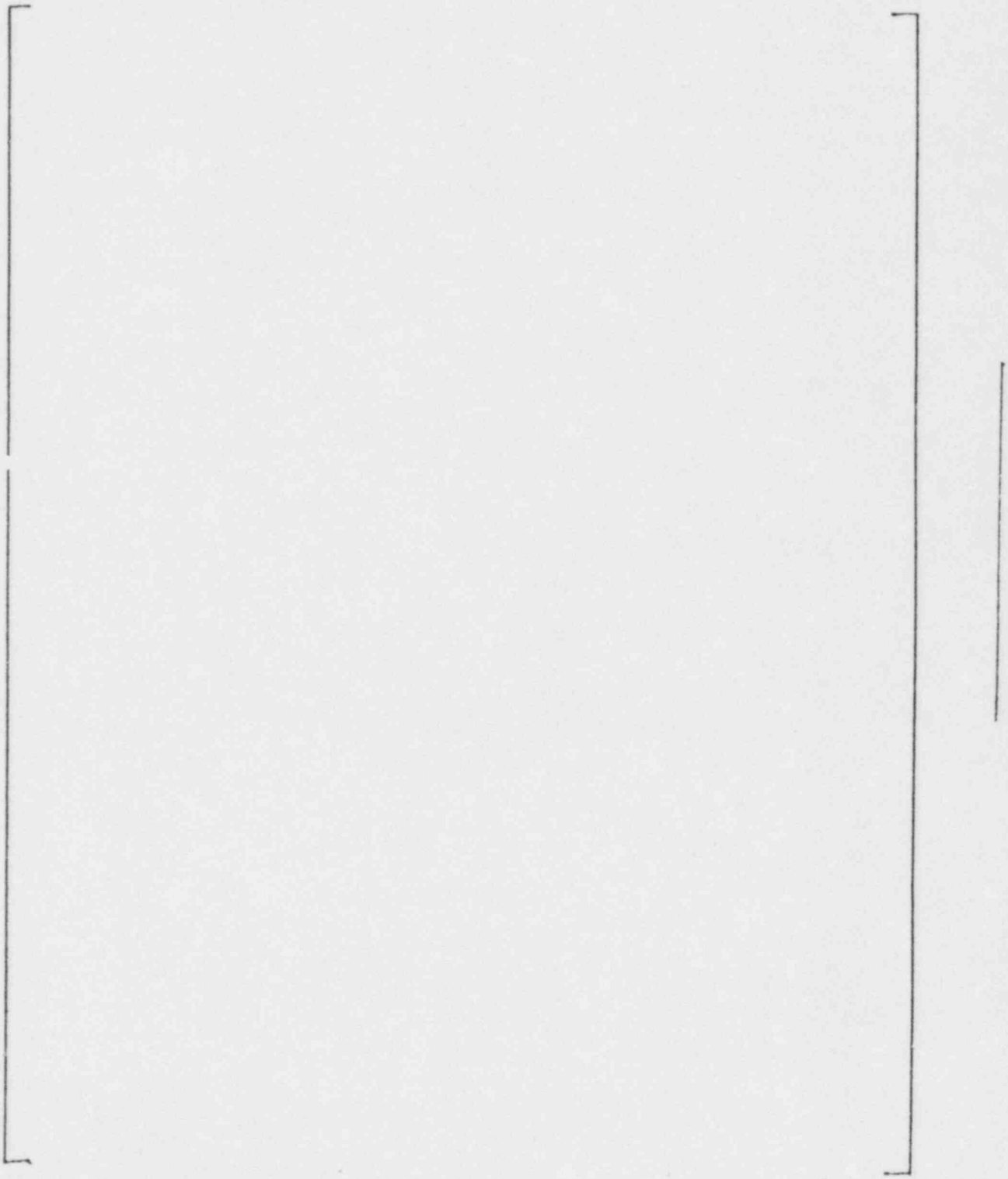
Attachment 1.4-2 (Cont.)

CPC Disc Generation



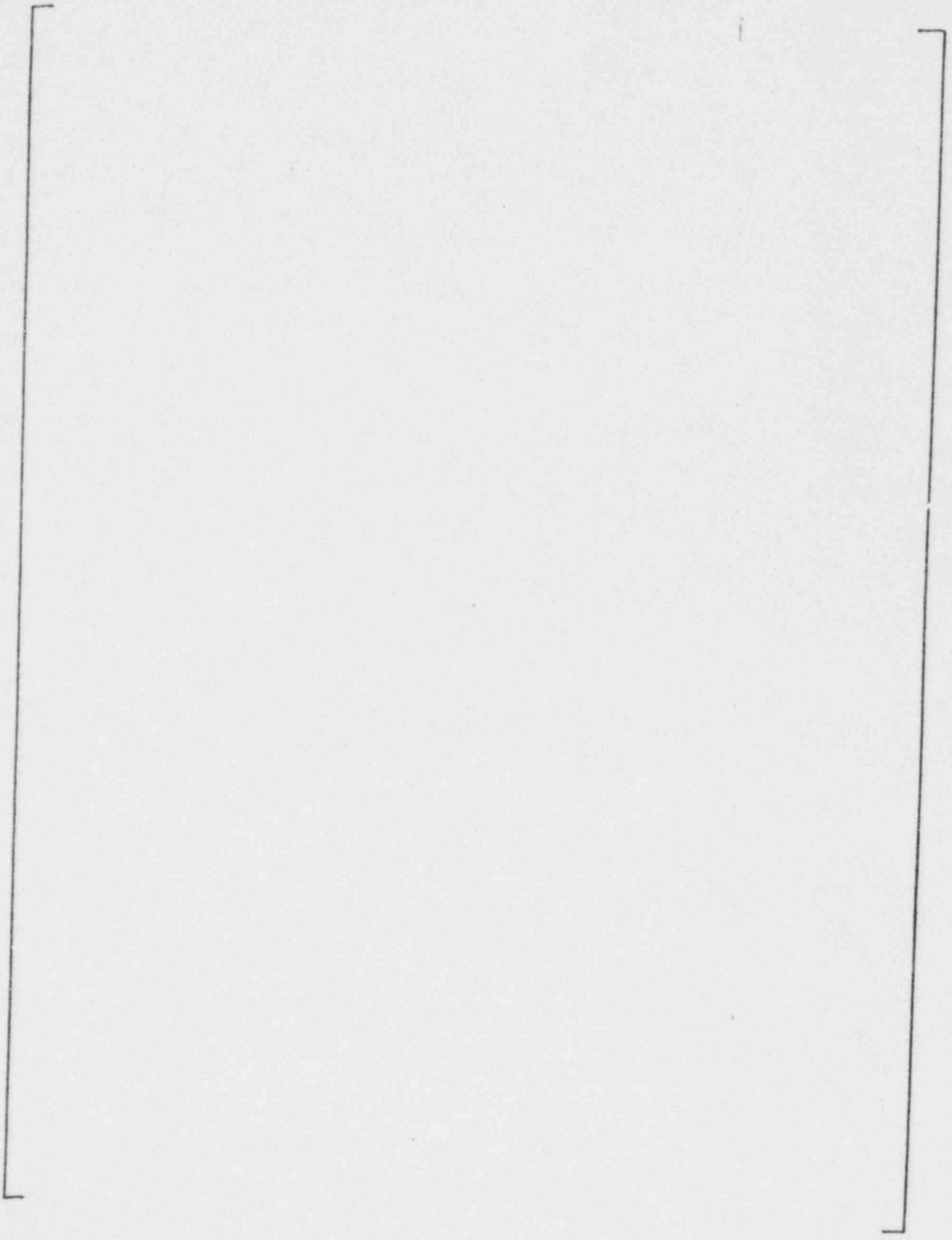
Attachment 1.4-2 (Cont.)

CPC Disc Generation



Attachment 1.4-2 (Cont.)

CPC Disc Generation



CPC Disc Generation

Attachment 1.4-3

CPC Disc Duplication

Purpose: To duplicate a disc using CPCDUP.

Figure 1.4-3

CPC System -1

Contents	Core Locations	Disc Track	Memory Block

Figure 1.4-3 (Cont.)

CPC System -2

Contents	Core Locations	Disc Track	Memory Block

Figure 1.4-4

CEAC System

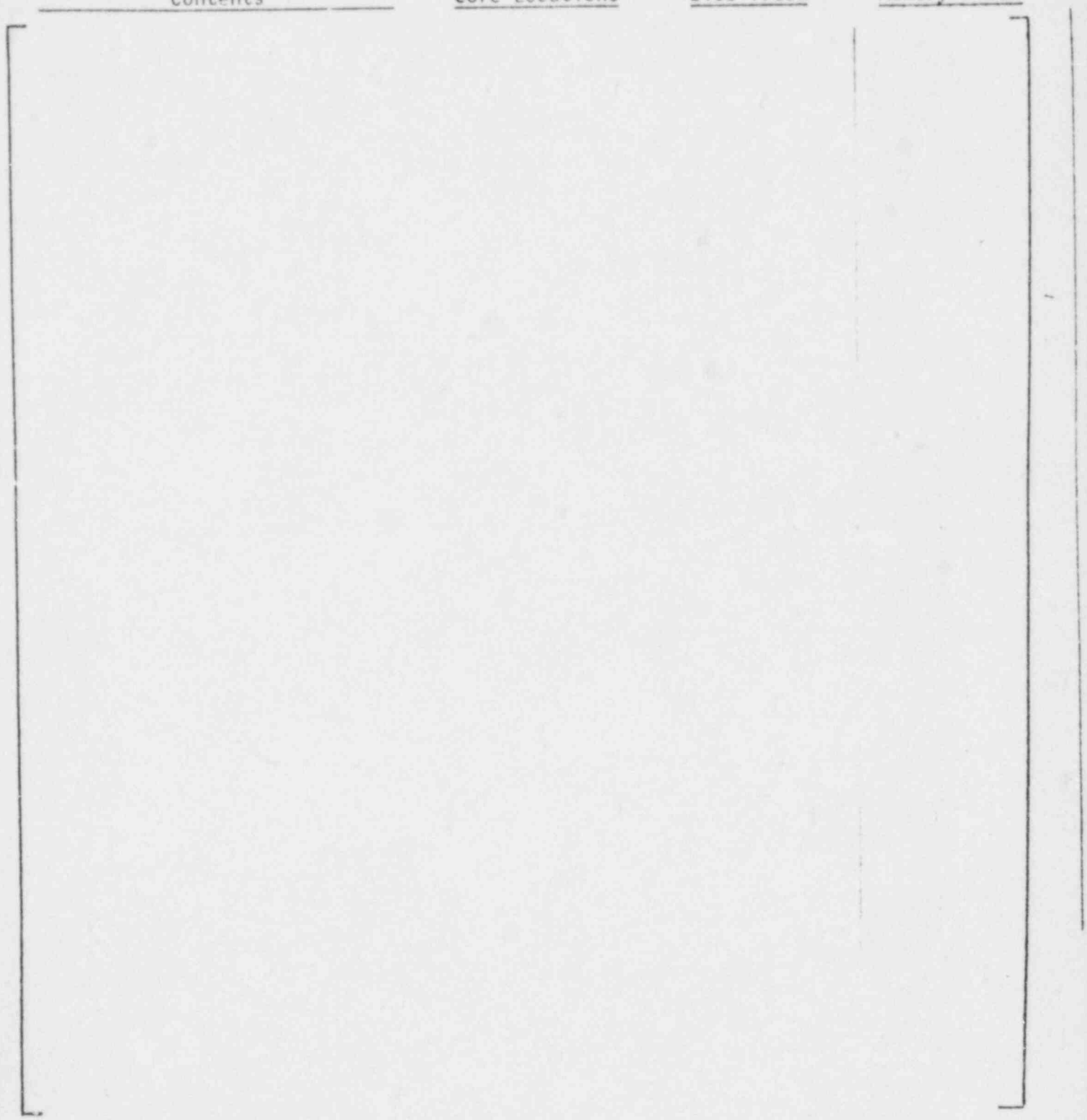


Figure 1.4-4 (Cont.)

CEAC System

Contents	Core Locations	Disc Track	Memory Block
[redacted]			

Figure 1.4-5

CPC Test System

Contents

Core Block

Disc Track

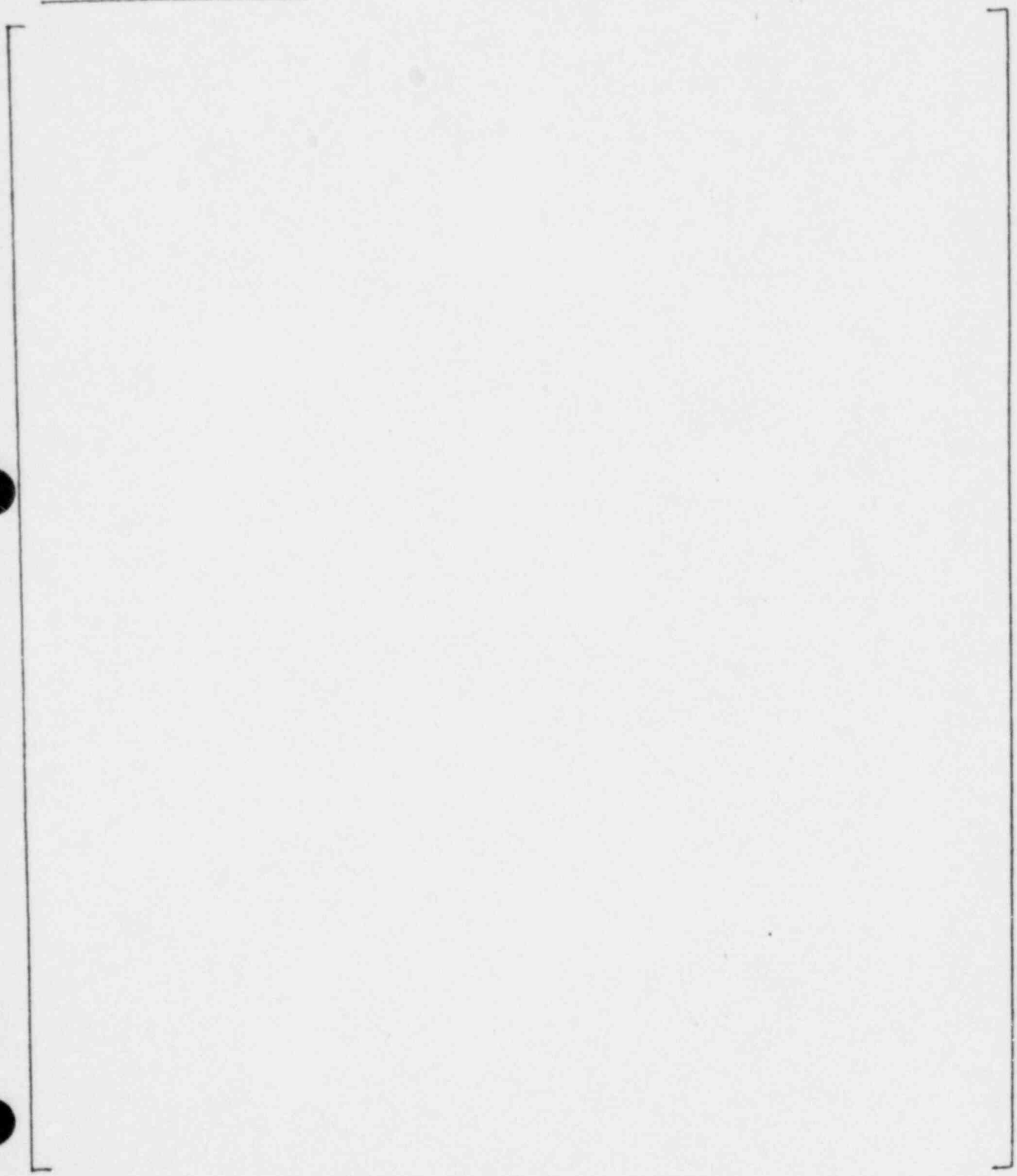


Figure 1.4-5 (Cont.)

CPC Test System

Contents	Core Block	Disc Track
[		]

Figure 1.4-6

CPC Files -1

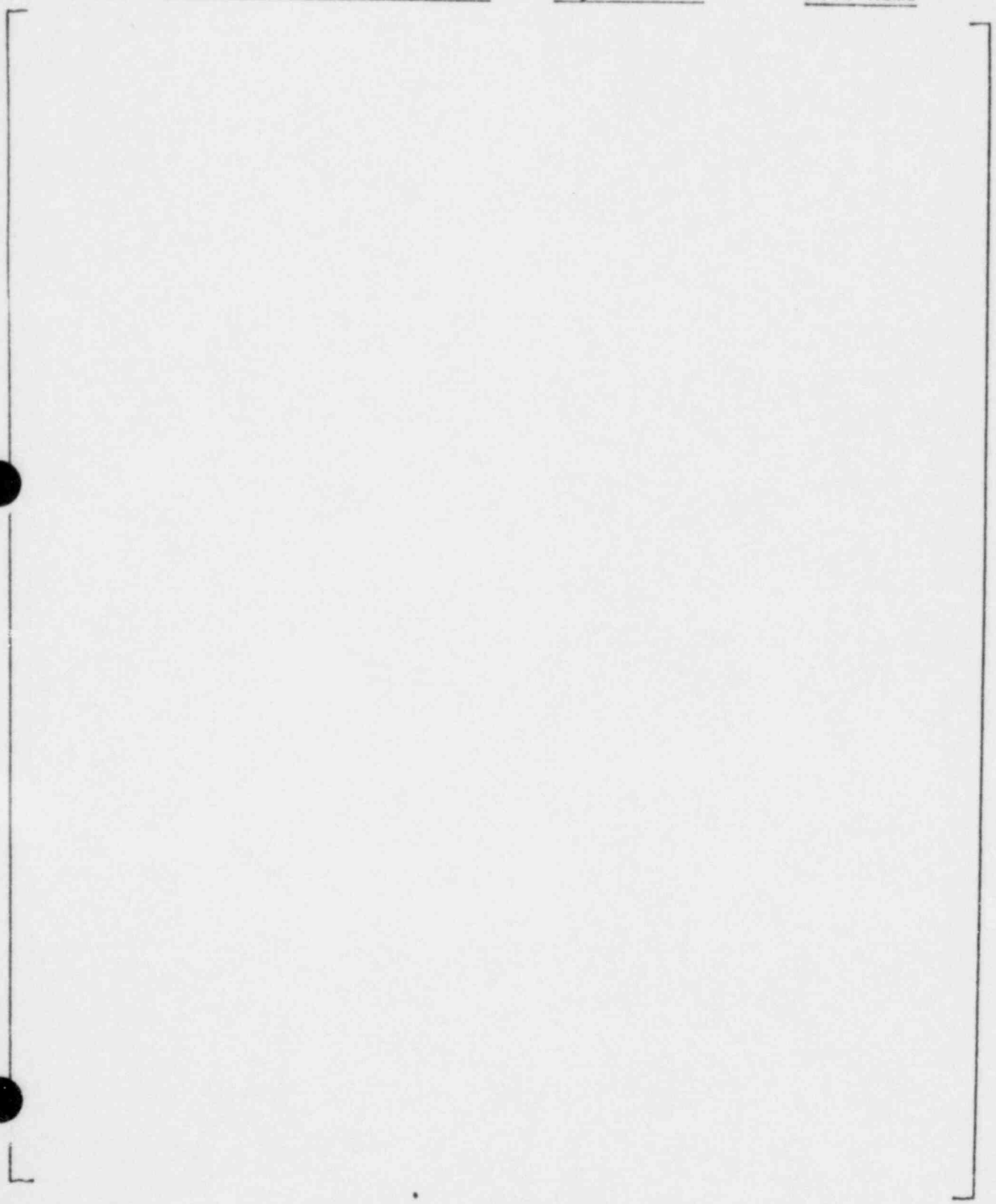


Figure 1.4-6 (Cont.)

CPC Files -2

Name	Object Label	File Name

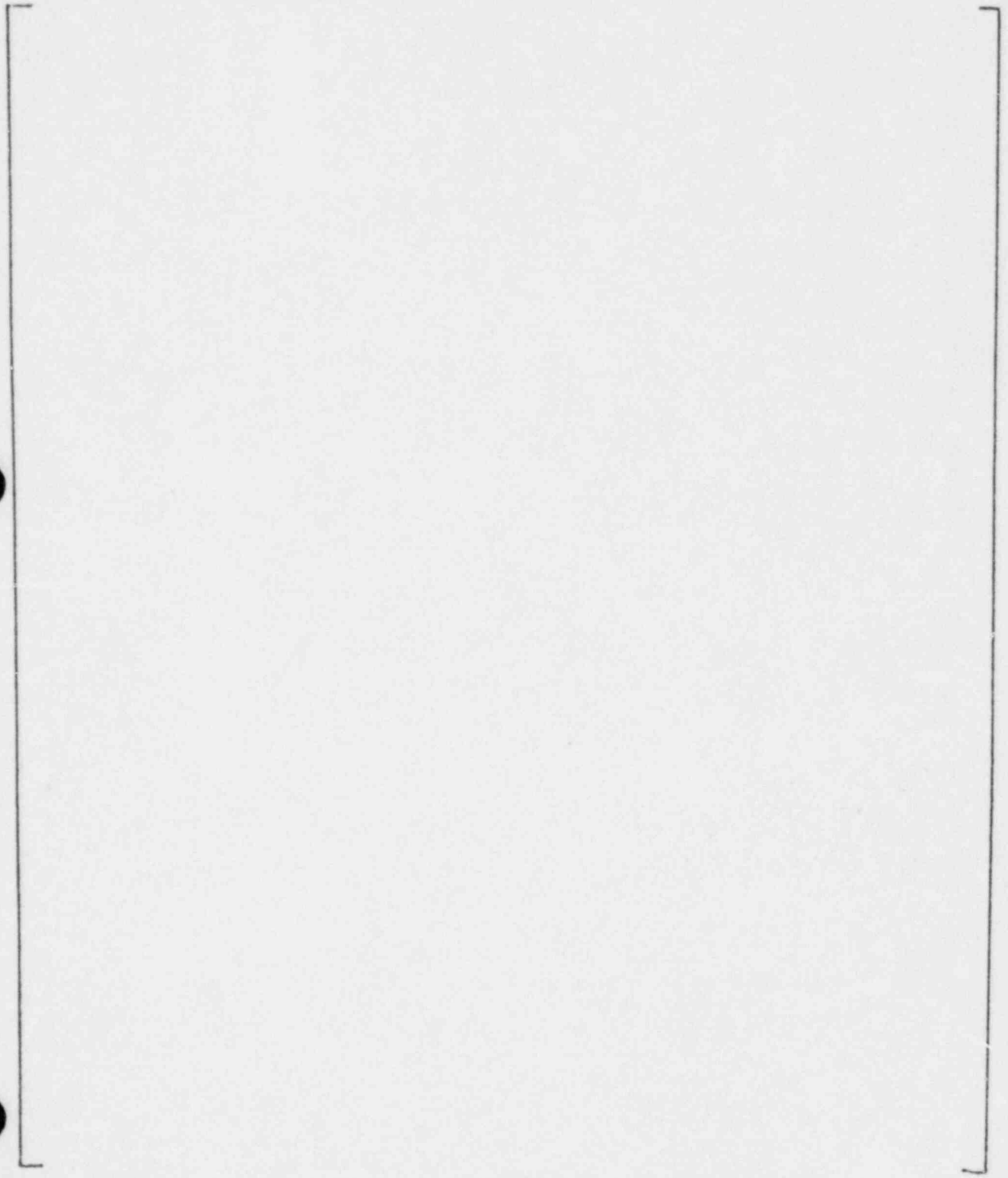


Figure 1.4-6 (Cont.)

CPC Files -3

Name	Object Label	File Name
[		]

1.5 CONTROL OF QUALITY RECORDS

Generic and Project documentation of software systems will be maintained in accordance with Reference 1.2.2.

In addition, a duplicate of the revised master source files shall be maintained in a separate location from the master source files under controlled access and with suitable environmental provisions.

The duplicate will be generated whenever a new reference disc is sent to a project. After the duplicate is generated, its source files will be assembled without listings. The resulting object code will be used to generate the periodic test discs to be sent to the project. This is accomplished using the commands described in Appendix 1-E.

Project discs will be part of the normal backup program. Two magnetic tapes will be assigned to each disc. The BACKUP utility (Reference 1.2.5) will be used to perform the backup, with alternate tapes used each backup. The date of backup and the contents of the disc will be obtained in accordance with established operating procedures.

APPENDIX 1A

Software Organization and Design Considerations

APPENDIX 1A

1A.0 SOFTWARE ORGANIZATION AND DESIGN CONSIDERATIONS

To achieve modular, structured, and standardized software, several constraints shall be adhered to in the design. These constraints are intended to allow modification to be performed with minimum impact and to simplify the testing of the software.

1A.1 SOFTWARE ORGANIZATION

1A.1.1 Modular Structure

[

]

1A.1.2 Program-to-Program Interface

Strict control of program-to-program interface allows each program to be developed and tested in a stand-alone mode and to be integrated into the system at a later time. [

]

[

[

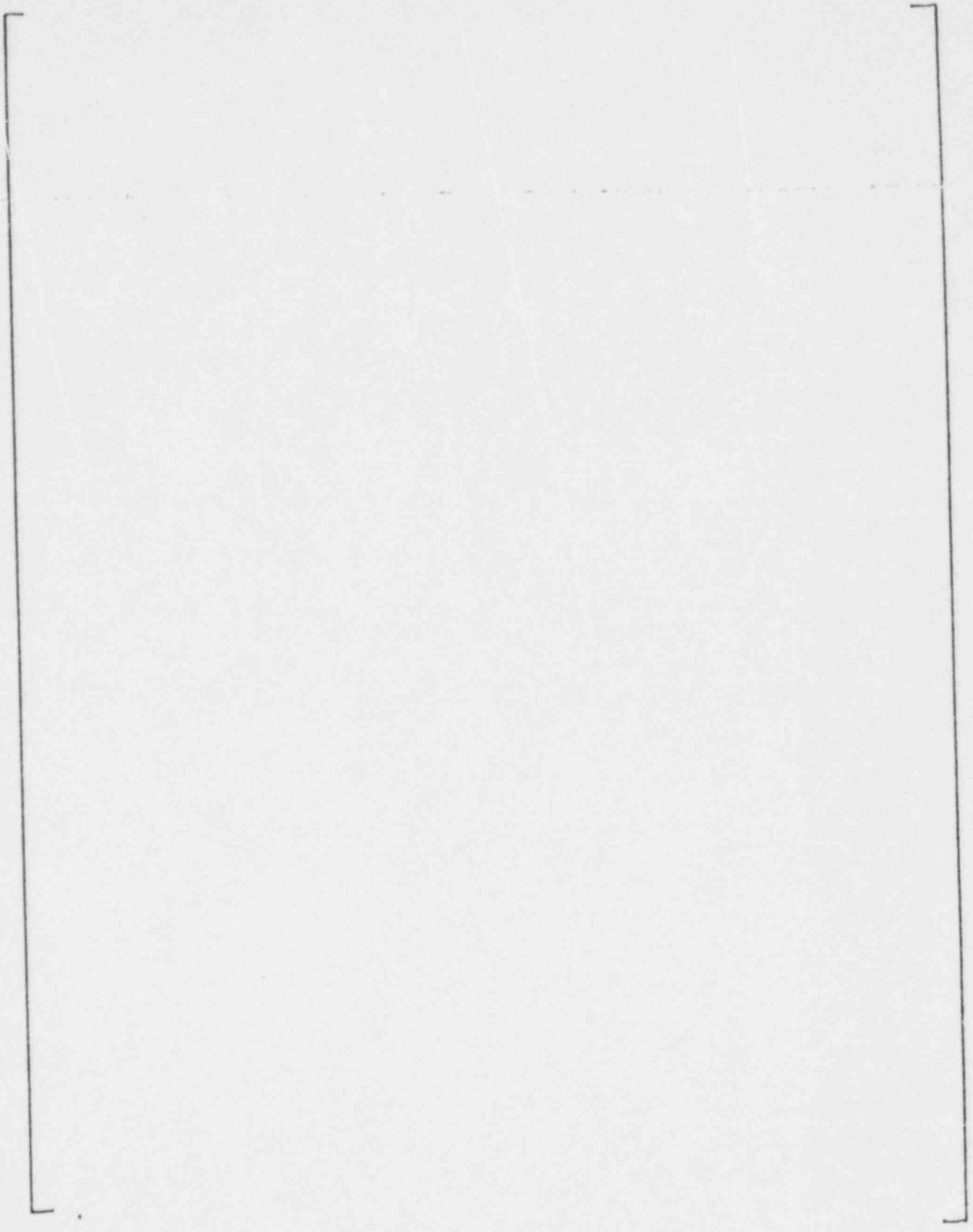
]

1A.1.3 Data Base Structure

To allow plant dependent data to be changed without impacting the program code, application programs will be divided into two separate assemblies. [

[

]



1A.1.4 Timing and Processor Loading

The design goal for the processor loading for all applications programs in the on-line software system is less than or equal to 60 percent. [

[

]

]

1A.2 DESIGN CONSIDERATIONS AND DOCUMENTATION

1A.2.1 Flow Charting Conventions

ANSI standard flow charting symbols are to be used for all program documentation flow charts. These symbols and guidelines are defined in Reference 1.2.6.

1A.2.2 Program Identification

The following guidelines shall be followed to uniquely identify each program and the statements within the program:

1. The program must adhere to Interdata CAL assembler conventions so that the program may be assembled with the CAL assembler at the current revision level.

2. Every line in the source file shall be identified by a sequence number in positions 73 through 80. The sequence numbers shall be of the form:

   cccnnnnn

   where:

   -ccc is a three letter code assigned to the software system under consideration.

   -nnnnn is a 5 digit decimal number which should initially be assigned in increments of 10 to allow for later insertions.

All programs shall be assembled with the sequence check option enabled.

3. Each program shall be organized into the following sections, in the order listed below:

-Title Block (See Figure 1A-1)
-Entry Point Definition or External References
-Global Data Base References (EQUs)
-Main Program
-Program Modules
-Initialized Storage Definition
-Scratch Storage Definition

Note that not all of the sections listed above will be required in all files in the system (e.g. constant files). The title block, however, is mandatory.

4. A prototype of the Title Block is shown in Figure 1A-1. The lower case information will be replaced with information specific to each program. The upper case data shall be entered as given. The complete program name shall include the plant or utility name or the word "GENERIC" if the program is common to all plants which contain a specific software system.

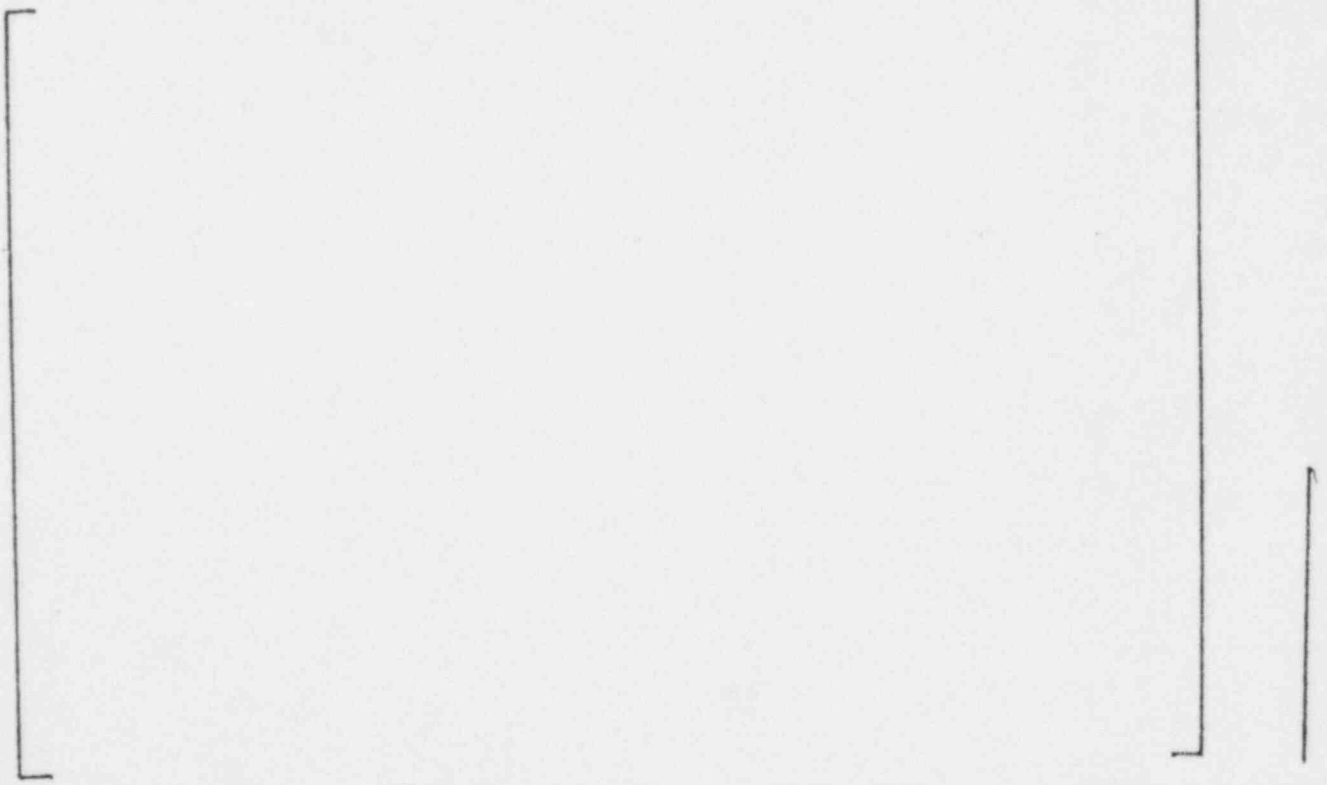
1A.2.3 Comments

Every line of code shall contain a comment where such a comment is of benefit in explaining the operation. Such comments should be functional in nature and shall not elaborate upon the machine instruction as it is assumed that the program is reviewed by qualified personnel.

Full line comments should be inserted freely to delineate and explain sections of the code and data base.

1A.2.4 Arithmetic Fault Detection

Every arithmetic operation other than simple array index calculations shall be checked for arithmetic fault either by program logic or by hardware trap features. [



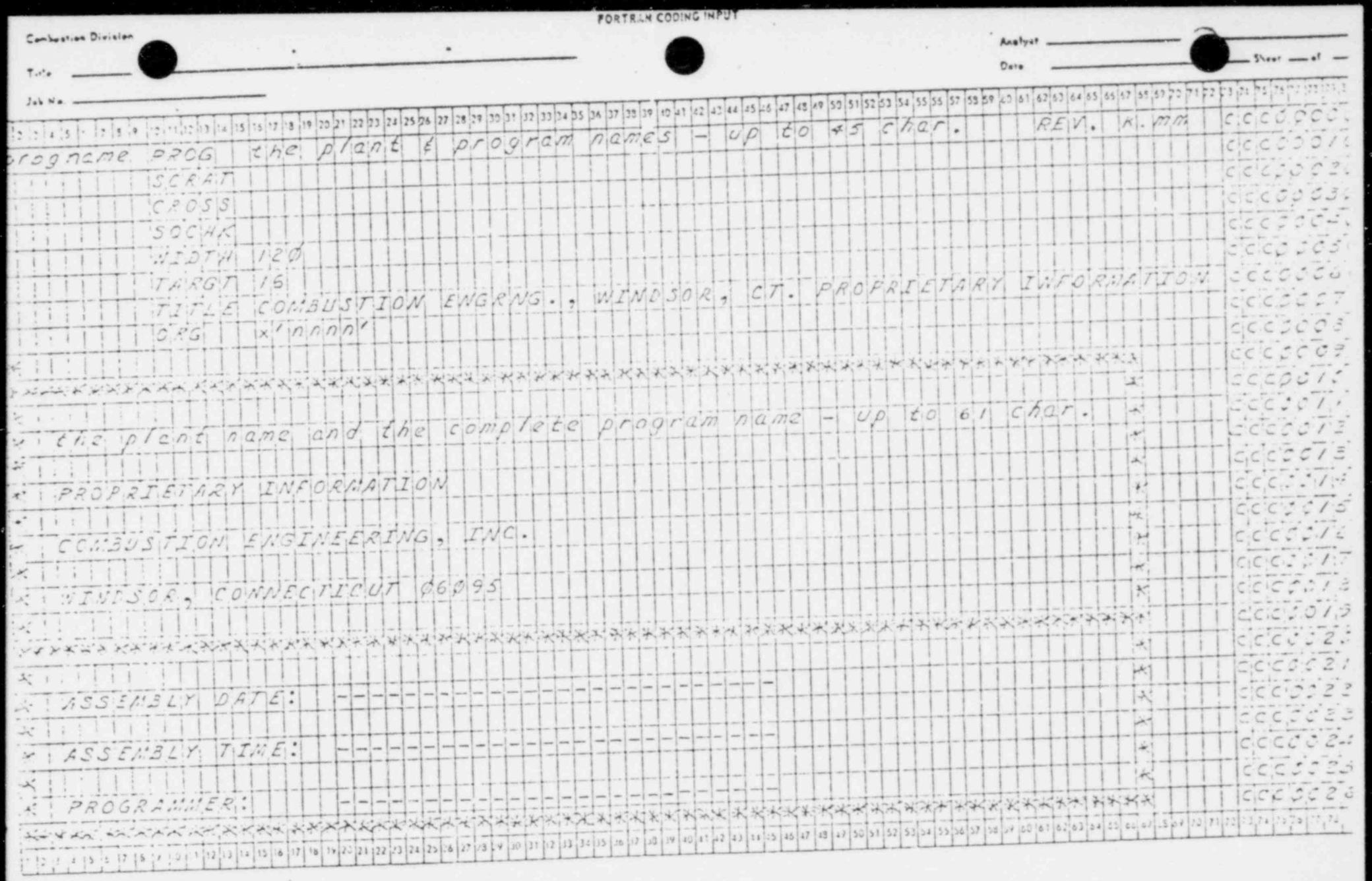


Figure 1A-1

CPC PROGRAM TITLE BLOCK

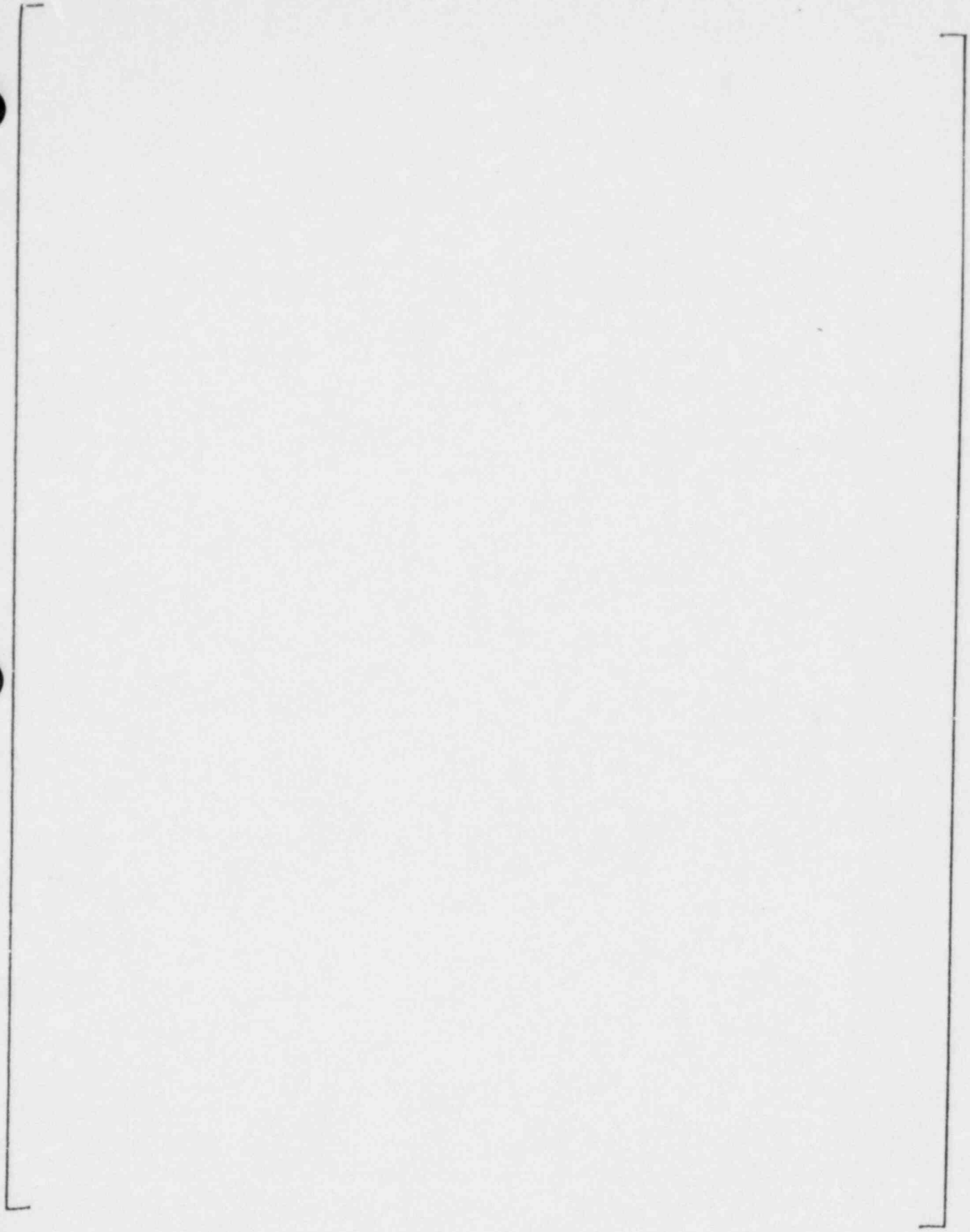
APPENDIX 1B

CSS Files

UPDATE A SOURCE FILE

Name: UPDATE

Purpose: Update a CPC source file and perform a CAL assembly



Example:

[

]

UPDATE THE REVISION LOG

Name: UPDLOG

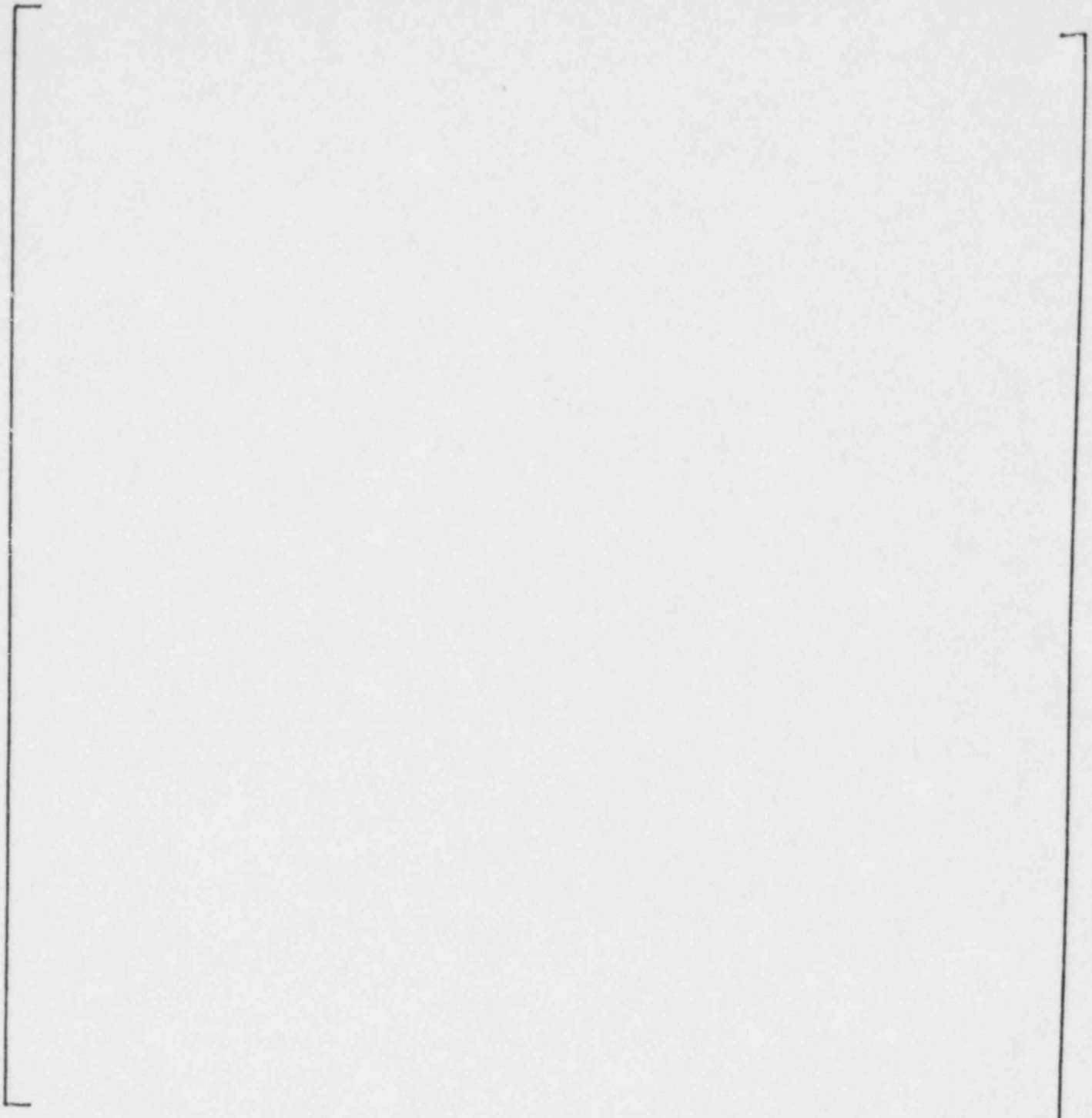
Purpose: Add a record of any revision to CPC software onto the revision log.

This CSS is not called by the user, but by UPDATE and INITIAL.

CREATE A SOURCE FILE

Name: INITIAL

Purpose: Create a CPC source file and perform a CPYA and CAL assembly



APPENDIX 1C

SOURCE UPDATER COMMANDS

The following is a description of the six update commands which can be used in the Source Updater command file to update a source file. Note that all updates must be made in order of ascending sequence numbers.

1. INSERT Command

The INSERT Command is used to insert one or more new statements into an existing source program. The records following the INSERT Command are inserted in the new master until a record with a '/*' in positions 1 and 2 is read from the source update file. The format of this command is:

INSERT seqnbr

where 'seqnbr' is the sequence number after which one or more new statements are to be inserted. If, before 'seqnbr' is found a sequence number is detected on the old master that is larger than 'seqnbr' or an EOF/EOM or the ending string is detected, the error message:

SEQUENCE NUMBER NOT FOUND

is printed on the line printer and the operation is aborted.

To insert two statements between XXX00150 and XXX00160, the following statements must be placed in the Source Updater command file:

```
INSERT XXX00150
- statement #1 -
- statement #2 -
/*
```

2. DELETE Command

The DELETE Command provides the capability to delete a single statement or multiple statements from the input source file. The format of the command is:

```
DELETE seqnr1, seqnr2
```

where 'seqnr1' specifies the sequence number of the first statement to be deleted. If 'seqnr2' is omitted, only the statement specified by 'seqnr1' is deleted. If 'seqnr2' is specified, statements between 'seqnr1' and seqnr2' inclusively are deleted.

If, before 'seqnr1' is found, a sequence number is detected on the old master that is larger than 'seqnr1', or, before 'seqnr2' is found, a sequence number is detected on the old master that is larger than 'seqnr2', the error message:

```
SEQ NBR NOT FOUND
```

for 'seqnr1' or the error message:

```
SECOND SEQNBR NOT FOUND
```

for 'seqnr2' is printed on the line printer. The operation is aborted when a sequence number is not found.

To delete statements AAA01010 through AAA01500, the following command is placed in the Source Updater command file

```
DELETE AAA01010, AAA01500
```

3. MODIFY Command

The MODIFY Command is used to delete a single statement and insert a new statement in its place. The format of the command is:

MODIFY seqnbr

where 'seqnbr' indicates the sequence number of the statement to be modified. If the sequence number cannot be found the error message:

SEQ NBR NOT FOUND

is printed on the line printer.

To use the MODIFY command to delete statement AAA01000 and replace it with a new statement, the following sequence must be placed in the Source Updater command file:

MODIFY AAA01000
- new statement -

4. REPLACE Command

The REPLACE Command is used to both modify a specified statement and insert statements thereafter. The format of the command is:

REPLACE seqnbr

where seqnbr indicates the sequence number of the statement where the insert procedure is to begin. If the sequence number cannot be found an error message is generated. Replace is terminated by a statement with a /* in Columns 1 and 2.

5. SELECT Command

This command is used to modify or insert source lines at selected locations specified by sequence numbers. The format of the command is:

```
SELECT
Line 1    seqnumb1
Line 2    seqnumb2
Line 3    seqnumb3
:
:
/*
```

where seqnumb1, seqnumb2, etc. start at position 73 in an 80 byte record. When a sequence number is found in the old master that matches the sequence number specified, then a 'modify' is in effect; if a sequence number is found that is larger than the one searched for, then an 'insert' of one line is done before the larger sequence number. Select is terminated with a record with /* in positions 1 and 2.

6. ENDUP Command

This command terminates the UPDATE mode. If the old master file has not reached the 'ending string' specified in the Update command or EOF/EOM, this command copies the rest of the old master until the 'ending string' or EOF/EOM onto the new master.

APPENDIX 1D

Automated CPC Integration

APPENDIX 1D

Automated CPC Integration

A) Portions of the procedures needed to perform an integration of a CPC system on disc have been automated. While the Attachments describe the application of these automated procedures, this Appendix outlines the operation of specific modules.

B) The documentation covers three functional procedures. The CPC Log Print procedure prints the CPC Program Update Log in a formatted manner. The output is grouped by the name of the program that was changed. The CPC Integration procedure produces a magnetic tape with the necessary object files, in the correct order, for subsequent final integration at the CPC Support System. It also prints the steps to be performed at the Support System in order to create an absolute load module. The CPC Label procedure will scan a magnetic tape, such as that produced by the Integration procedure, and list the object file labels contained on it. This is useful for identifying the contents of magnetic tapes when the complete integration process does not occur at one time. A list of program, file name, and object file label is provided in Figure 1.4-6.

Purpose: Produce an organized formatted report based on the CPC Change Log.

Purpose: Read the log file on LU 5 and produce a report on LU 6.

[

]

Purpose: Output the object files needed to integrate a given CPC program, and print further instructions.

[

]

Purpose: Read LU 10 for the name of the program to be integrated. Find this name in the directory file on LU 12, and obtain the names of other files to be included in the integration, and the name of the instruction file. Create a CSS file to copy the needed object files and print the instructions.

[

]

CPCCOPYB.CSS

Purpose: Copy an object file of record length 108.

Purpose: Print a list of 16 bit object labels from a magnetic tape containing 16 bit object modules.

APPENDIX 1E

Periodic Test Disc Generation

APPENDIX 1E

Periodic Test Disc Generation - Tape Creation

Purpose: To create a magnetic tape backup of a project disc, for purposes of generating a Periodic Test disc.

[

]

Periodic Test Disc Generation - Tape Restoration

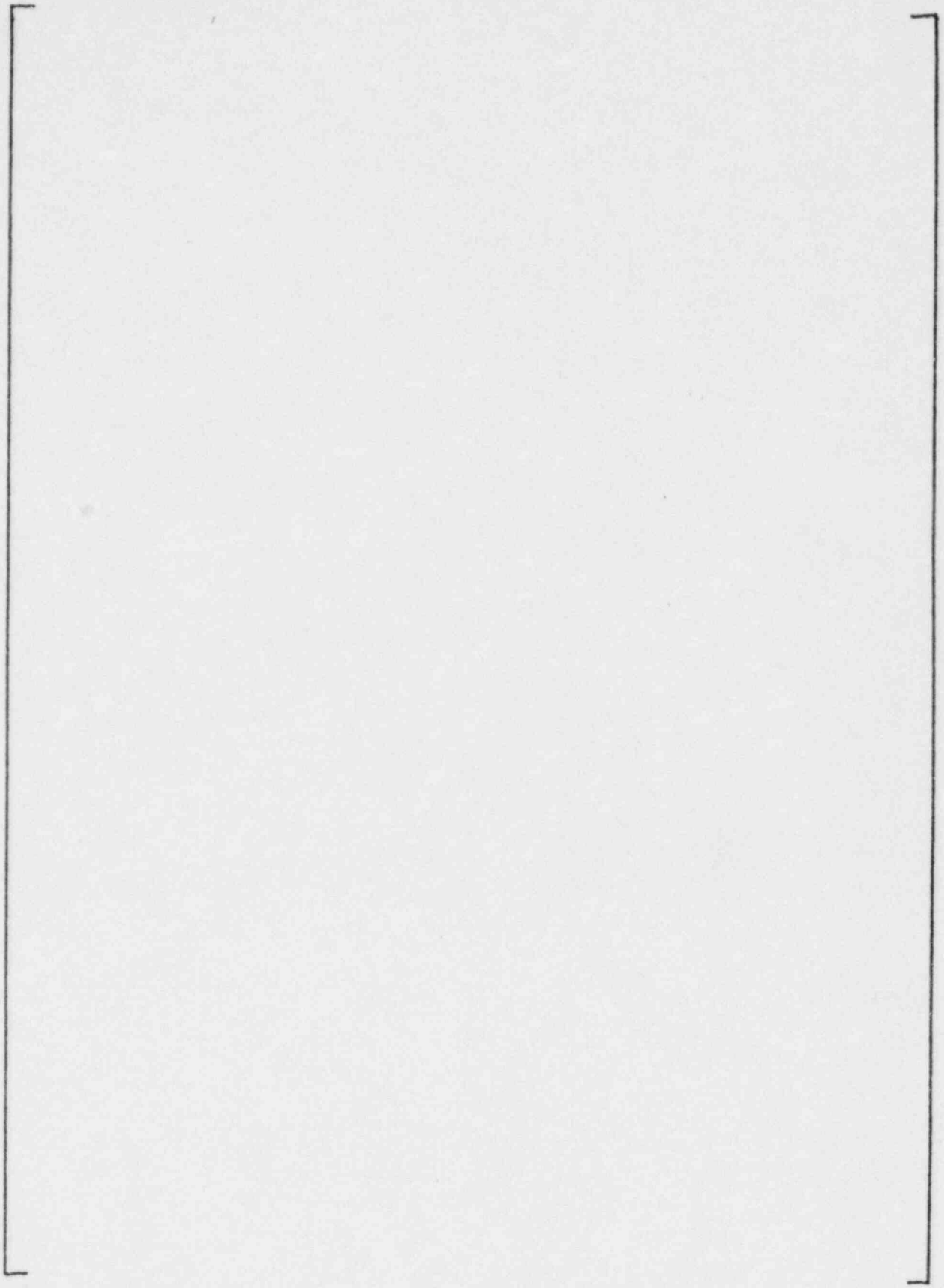
Purpose: To create a magnetic tape containing the CPC system object modules by reassembling from a backup tape.

[

]

Periodic Test Disc Generation - Disc Integration

Purpose: To generate a Periodic Test disc from a reference disc and object modules produced by the Tape Restoration process.



2.0 PROJECT DISC GENERATION PROCEDURE

2.1 PURPOSE

The purpose of Section 2.0 is to present the procedures required to generate a new project disc from the generic source disc.

2.2 REFERENCES

2.2.1 Series 40 Disc Drive Maintenance Manual Diablo Systems Inc., Publication No. 81601

2.2.2 OS/32 MT Operator's Reference Manual, Interdata Inc., Publication Number 29-574, Section 7.2.

2.2.3 Common Disc Test/Formatter Program, Interdata Inc., Publication Number 06-173.

2.2.4 OS/32 MT Operator's Reference Manual, Interdata Inc., Publication Number 29-574, Section 7.3.

2.2.5 OS/32 MT Operator's Reference Manual, Interdata Inc., Publication Number 29-574, Section 7.5.5.

2.3 PROJECT DISC GENERATION

The CPC Project Disc Generation Checklist (Attachment 2.3-1) outlines the procedure to be used in the generation of a new project disc. A new disc is formatted and initialized using the Common Disc Formatter program (Appendix 2-A), the Disc Initializer (Reference 2.2.2), and the Disc Integrity Check (Reference 2.2.4). The source files on the generic disc are then moved to source files on the new project disc. This is done using the command "PROJECT" (Appendix 2-A).

A reference disc is generated for the project by copying the generic reference disc using the CPC Disc Duplication Procedure (Attachment 1.4-3).

Attachment 2.3-1

CPC PROJECT DISC GENERATION CHECKLIST

1 Mount the new disc onto USR2 (Reference 2.2.1).

2 Format the disc (if it hasn't already been formatted) (Reference 2.2.3)

a) Mount the Multimedia Diagnostic Tape on the magnetic tape drive.

b) Load the 50 sequence through the hexadecimal panel:

Address	Value	Enter on Hexadecimal Panel					
30	0000	DTA	30	ADD	DTA	0	WRT
32	0000						WRT
34	0000						WRT
36	0050				DTA	50	WRT
50	D500	DTA	50	ADD	DTA	D500	WRT
52	00CF				DTA	CF	WRT
54	4300				DTA	4300	WRT
56	0080				DTA	80	WRT
78	85A1	DTA	78	ADD	DTA	85A1	WRT
7A	0000				DTA	0	WRT
7C	0000						WRT
7E	0001				DTA	1	WRT

Verify that the 50 sequence was loaded correctly by reading these same locations. This is done by entering the following on the hexadecimal panel:

DTA	30	ADD	RD	RD	RD	RD
DTA	50	ADD	RD	RD	RD	RD
DTA	78	ADD	RD	RD	RD	RD

c) Start the loader:

DTA 30 ADD RUN

FFFF should be displayed on the hex panel.

d) Load the Formatter program sequence number:

DTA 412 RUN

e) Place the format switch on the disc controller in the "FMT" position.

f) Make sure the protect light for USR2 is off

g) Start the Disc Formatter Program:

DTA A00 ADD RUN

h) Enter the following Formatter commands:

```
LOCYL 0
HICYL 197
PACTYP 1
DISCON B6
TFILE 1
FILE 2
TRKDEN 1
DEFSEC 1
FMREAD 1
TIMCON 140
RUN
```

3 Initialize the disc (Reference 2.2.2)

a) Mark SYS1 on:

```
MARK 'SYS1: ON,PROTECT
```

b) Load the Disc Initializer Utility:

```
LOAD  .BG,DISCINIT, 12
TASK  .BG
```

c) Mark all discs off:

```
MARK  USR1: ,OFF
MARK  USR2: ,OFF
MARK  SYS1: ,OFF
MARK  SYS2: ,OFF
```

d) Start the program:

```
START, DISC=USR2:, VOLUME=volume name, CLEAR, BLOCK=64/203
```

Volume name is a four character mnemonic for the project.

4 Check the integrity of the disc (Reference 2.2.4).

a) Load the disc check utility;

```
MARK  SYS1:, ON,PROTECT
LOAD  .BG,DISCHECK, 12
TASK.BG
```

b) Mark the disc off:

```
MARK  USR2:, OFF
```

c) Start the program:

```
START, USR2:, PR:
```

5 Move all necessary files to the project disc by entering:

PROJECT (project name)

This procedure is detailed in Appendix 2-A.

APPENDIX 2A

This procedure describes the generation of a project disc from a generic disc. The project mnemonic and disc volume share the same name for identification purposes. The generic software is assumed to reside on volume CPC1.

To create a project disc, all applicable files are read from the generic disc and written to the project disc. This process is invoked as follows:

1)	MARK USR1:,ON,PROTECT	Put the generic disc on-line in device USR1:
2)	MARK USR2:,ON	Put the project disc on-line in device USR2:
3)	Enter the command PROJECT proj	Where proj is the four character project mnemonic

Some errors which may occur are:

** VOLUME proj IS NOT ON-LINE **

The disc volume whose name is the project mnemonic is not on-line.

** GENERIC DISC VOLUME CPC1 IS NOT ON-LINE **

The disc volume containing the generic files is not on-line.

ASSIGN ERROR FILE filename

The file called filename is already on the project disc. This disc should have been empty.

Other errors are possible; refer to Reference 2.2.5.

4) When the process is completed, the message ** CPC PROJECT DISC proj CREATED ** is displayed.