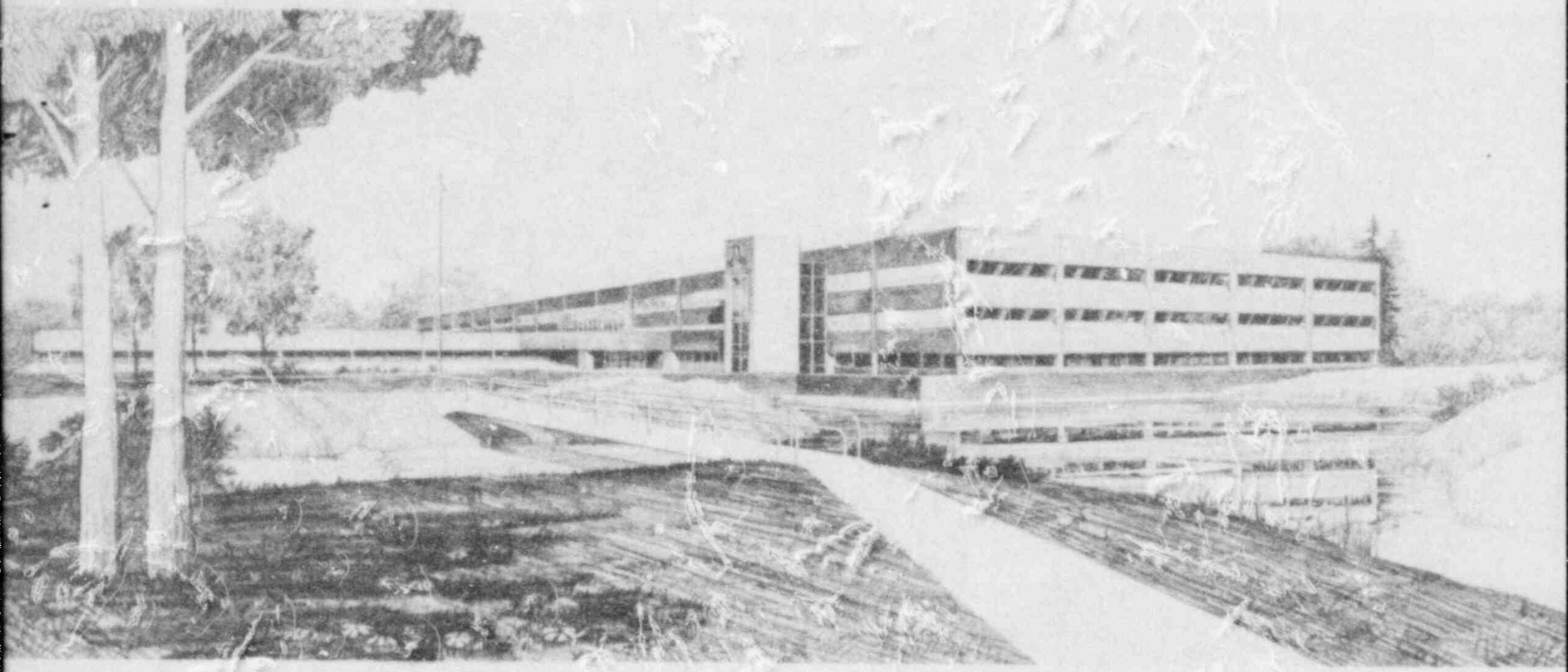
NUREG/CR-3360
EGG-2302



Idaho National Engineering Laboratory

Operated by the U.S. Department of Energy

Computer Program CDCID: An Automated Quality Control Program Using CDC Update

Gilbert L. Singer
Felix Aguilar

April 1984

Prepared for the
U.S. Nuclear Regulatory Commission
Under DOE Contract No. DE-AC07-76IDO1570

EG&G Idaho

NUREG/CR-3360
EGG-2302
Distribution Category: R4

COMPUTER PROGRAM CDCID: AN AUTOMATED QUALITY CONTROL PROGRAM USING CDC UPDATE

Gilbert L. Singer
Felix Aguilar

Published April 1984

EG&G Idaho, Inc.
Idaho Falls, Idaho 83415

Prepared for the
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555
Under DOE Contract No. DE-AC07-76ID01570
FIN No. A6052

ABSTRACT

A computer program, CDCID, has been developed in coordination with a quality control program to provide a highly automated method of documenting changes to computer codes at EG&G Idaho, Inc. The method uses the standard CDC UPDATE program in such a manner that updates and their associated documentation are easily made and retrieved in various formats. The method allows each card image of a source program to point to the document which describes it, who created the card, and when it was created.

The method described is applicable to the quality control of computer programs in general. The computer program described is executable only on CDC computing systems, but the program could be modified and applied to any computing system with an adequate updating program.

FIN No. A6052—Loss-of-Coolant Accident (LOCA) Analysis Development

SUMMARY

An automated quality control program has been implemented to provide a versatile and comprehensive documentation system for computer code development and changes at EG&G Idaho, Inc. This work was sponsored by the United States Nuclear Regulatory Commission and was administered by the United States Department of Energy at the Idaho National Engineering Laboratory (INEL).

This document provides an explanation of a simple yet comprehensive quality control program and it documents the CDCID UPDATE computer program which provides a convenient automated tool for implementing a major part of the quality control program.

The quality control program provides detailed procedures and updated format requirements which aid in the automated development of a high quality computer software product; that is, one which is error free and reliable, well documented, and easy to use and change. The procedures include code development procedures to assure proper design, coding, testing, and documentation of computer code creation or modification. They include code change procedures to assure proper integration of various modifications into a single checked out program. Other elements of quality control are discussed. The update format requirements which allow automation of much of the code documentation are also discussed.

The CDCID computer program provides a versatile and comprehensive automated system to document computer code development and changes. It provides for a one-to-one correspondence between each change to a computer program and a document which describes the change. Furthermore, it provides an easy and automated method for producing a wide variety of selected portions of the total documentation available concerning program changes.

A summary of the input needed to the code in order to insert documentation changes along with a set of changes to the user's program is provided in a separate user's guide section for those who do not need to know all of the program capabilities but only need to know how to prepare update documentation decks which can be processed by CDCID.

CONTENTS

TABLES

COMPUTER PROGRAM CDCID: AN AUTOMATED QUALITY CONTROL PROGRAM USING CDC UPDATE

1. INTRODUCTION

An automated quality control program has been implemented to provide a versatile and comprehensive documentation system for computer code development and changes at EG&G Idaho, Inc. This work reported was sponsored by the United States Nuclear Regulatory Commission and was administered by the United States Department of Energy at the Idaho National Engineering Laboratory (INEL).

The primary goal of the quality control program is to ensure that the software products for the computer codes developed at EG&G Idaho will be of high quality; that is, error free and reliable, well documented, and easy to use and change. A secondary goal is to facilitate the exchange of new models and error corrections to existing computer codes, even for code recipients not using the same method to update the code (as long as they do not destroy the original information contained on earlier code transmittals).

This document provides an explanation of a simple but comprehensive quality control program and documents the CDCID UPDATE computer program which provides a convenient automated tool for implementing a major part of the quality control program.

The quality control program consists of four parts: a configuration control system, code development procedures, code change procedures, and update format requirements. These are explained in detail both because of their intrinsic value and in order to allow the reader to understand the purpose and usage of the CDCID computer program which is documented herein. A reader who is using this document simply as a user's guide may wish to skip the sections on code development and change procedures and begin with the section on update format requirements and then read the CDCID user's guide section.

The code development procedures provide the steps necessary for the efficient development of a computer code model. These steps ensure that:

1. Proper consideration is given to adequate design and review of the model

2. If unexpected problems occur, proper consideration is given to redesign

3. Coding work is not begun prematurely and that time is not wasted in implementing a faulty design

4. Adequate model documentation is provided and adequate testing is performed.

The code change procedures provide steps to ensure that the merging of various models and changes into a computer code is done in a standard manner designed to eliminate errors and the filing of required documents and a final testing of the updates are completed.

The update format requirements provide a standard format for updates and in-line documentation (as part of the update file itself).

The quality control program procedures and the CDCID program were first used with the TRAC-BD1,[1] a large computer code for transient reactor analysis of boiling water reactors. Due to its demonstrated utility, the model became standard for the computer codes of the Code Development Division of EG&G Idaho at the INEL. These procedures have been very effective partly because the system was designed with the intent of making it, while comprehensive, as simple, useful, and convenient as possible for the engineers and programmers who use it. Willing acceptance of this quality control program by the users has been a key element in its success. This acceptance is largely due to the minimizing of paper work and the maximum use of the computer to automate the process.

The following sections of this report describe the code development and change procedures, update format requirements, and CDCID program method, and provide a CDCID user's guide.

The meaning of most of the terms in this report is apparent from the words and the content; however, to ensure that the precise meanings may be determined, a list of definitions of terms used is provided in Appendix A. Appendix B shows sample forms which may be used to implement this quality control program. Appendix C describes a small auxiliary computer program (DFORM) which helps further automate the documentation process. Appendix D suggests what documentation should be filed. Appendix E is a listing of the CDCID source program and contains a structured flow chart as part of the source. Appendix F contains a sample problem output to show the contents of various files associated with a CDCID run.

2. CODE DEVELOPMENT PROCEDURES

Presumably, every computer code development or maintenance effort is part of a project. The framework of this project should include an enumeration of individual responsibilities, a configuration control system, and an overall code development plan. The responsibilities are dependent on existing organizational structure and are not considered here except at the lowest level of interaction, that is, among the code contributors, code librarian, and their immediate supervisors. A method of handling and filing the forms and other documentation for a code development effort is provided in Appendix D.

2.1 Configuration Control System

A configuration control system is needed which, in addition to adhering to the code development and change procedures, will ensure that the following are provided.

1. Complete traceability and reproducibility for all official code versions
2. A naming convention for each official code version which will provide a unique identifier internal to the code and in the code output
3. Appointment of a code librarian to ensure that a single individual is responsible for maintaining the configuration control system, documentation, and computer files
4. The documentation, for a computer code and the distribution of information to the code users (that is, code user's manual and newsletter)
5. Standards to ensure the quality of the programming and for the calculation of physical properties.

2.2 Code Development Plan

The contents of an overall code development plan are generally applicable to all code development efforts, and, therefore, a sample of the contents of such a plan is provided:

1. A statement of overall code development objectives.
2. A description of the planned code capability.
3. A code version sequence description for planned released code versions.
4. A milestone chart for the specific code version under development.
5. Work packages for each element of the effort associated with the specific code version under development or maintenance. Each work package should include a statement of scope, necessary input(s) required, deliverable output(s), the associated milestone(s), allocated resources, and a minor network(s).
6. A major schedule for development of the specific code version under development.
7. A manpower loading chart for each item in the minor network(s).

2.3 Code Development Procedures

The code development procedures are applicable to both development of new computer codes and to all proposed changes of existing codes. These procedures prescribe the responsibilities of the Cognizant Engineer, Cognizant Programmer, Code Librarian, Section Leader, and Branch Manager. They also prescribe documentation and acceptance testing requirements for code changes. Obviously, the amount of documentation and acceptance testing required to support a proposed code change should depend on the complexity of the change.

In addition to new code development, there are two general classes (model and programming) of code changes. The model class includes new physical models, new solution algorithms, improvements to existing models or algorithms, and corrections of formulation errors. The programming class includes changes to optimize code structure, improve data management, improve appearance and legibility, and correct coding errors. These procedures do allow the Section Leader to exercise judgment regarding supporting documentation and acceptance testing.

The first step of the design process is to formulate a set of objectives and requirements for the product. For a new code development effort, the requirements are as specified by the code development objectives. For a code change, requirements for the proposed change may or may not be detailed depending on the extent and type of change. This decision is left to the discretion of the responsible Section Leader. However, all proposed changes should be considered with respect to the overall code development objectives in order to ensure consistency with the code development plan.

2.4 Model Development Procedure

Normally, the development procedure will be initiated by the Cognizant Engineer in the case of a model change and by the Cognizant Programmer for a programming change. The steps in the development procedure are outlined in the following subsections. An asterisk (*) indicates a step which may be optional at the discretion of the responsible Section Leader.

2.4.1 Step 1 (Development Plan—Code Development Form). The cognizant individual completes the Code Development Form, Table B-1 in Appendix B. The form has two functions: it outlines the development plan (that is, it specifies the kinds of documentation required to complete the development task) and it serves as a cover sheet for the development file kept by the Section Leader. The form is largely self-explanatory. The form is submitted to the Section Leader for approval.

2.4.2 Step 2 (*) (Design Report). A design report is written if so indicated on the development form. The purpose of the design report is to ensure that the model formulation or programming change concept is complete before actual coding begins. The design report also serves to inform co-workers of proposed changes that may affect their work.

The suggested content of a design report is provided in Table B-2. Note that an important feature of the design report is the specification of a set of acceptance test cases.

2.4.3 Step 3 (*) (Design Review—Not Optional if Step 2 Undertaken). The design report is submitted to the Section Leader, who arranges a design review. This review normally consists of a review of the design report by other members of the section in tne case of a change to an existing code. In the case of a new code development, the Branch Manager arranges for a formal design review. When the review is complete and the design report is revised accordingly, the design report is inserted into the development file, and continuation of development is approved.

2.4.4 Step 4 (Coding). The actual coding of the new code, model, or programming change is performed. In the case of a change to an existing code, the update set (change cards) is normally developed from the most current official code version.

2.4.5 Step 5 (*) (Acceptance Testing). The update set is exercised with the acceptance test cases defined in the design report, if one exists. If there is no design report, a comprehensive set of test cases is negotiated by the Cognizant Engineer/Programmer and the Section Leader. The test cases should test the entire physical range of applicability, all input options, and diagnostics.

2.4.6 Step 6 (*) (Completion Report). After a proposed set of updates is prepared and tested, a task completion form is written if so indicated on the development form. The completion report provides a detailed description of the model or change and its proposed implementation. It also provides a record of the acceptance testing that has been performed. The contents of a completion report are given in Table B-3.

2.4.7 Step 7 (*) (Completion Report Review). The completion report is submitted to the Section Leader who will arrange a review to ensure that design goals have been achieved. If the completion report is acceptable, it is inserted into the development file. If model performance is judged deficient, the development process reverts to Step 2 or Step 4, depending upon the extent of required design changes.

2.4.8 Step 8 (Update Submitted). The proposed update set is given to the Code Librarian. The update set must conform with the requirements prescribed by the Update Format Requirements, Section 4.

The development process is complete with Step 8. However, update sets given to the Code Librarian are not automatically inserted into an official code version as explained in Section 3.

3. CODE CHANGE PROCEDURE

The Code Librarian has the responsibility of creating official code versions. However, the Cognizant Engineer and/or Programmer responsible for proposed changes participate in the process.

A systematic method of reporting problems in a computer code should be established. The Cognizant Section Leader should maintain a record of all code deficiencies and suggested code improvements. A sample deficiency report is provided in Table B-4. When a code change is proposed to rectify a reported deficiency, a copy of the deficiency report is inserted into the development file.

3.1 Step 1 (Merger of Update Sets)

The Code Librarian attempts to merge all update sets received for insertion into the official code version. If there are no coding conflicts among the update sets, a candidate official version can be created straightforwardly, and Step 6 is the next step in the code change process.

3.2 Step 2 (Resolution of Update Conflicts)

It is the Code Librarian's responsibility to resolve conflicts among update sets. This person determines which update sets are to be revised to achieve compatibility among all sets.

3.3 Step 3 (Completion Report Revision)

It is the Cognizant Engineer's and/or Programmer's responsibility to make revisions to an update set. A revision section explaining any changes is added as an appendix to the completion report. The appendix details any changes to the design and provides revised figures, flow charts, and update listings. The original completion report text is not changed. The completion report cover is changed to note that the report has been revised. The results of acceptance test cases obtained with the revised update set need not be reported in the revised completion report.

3.4 Step 4 (Completion Report Approval)

The revised completion report is submitted to the Section Leader for approval and insertion into the development file.

3.5 Step 5 (Revised Update Submitted)

The revised updates are submitted to the Code Librarian.

3.6 Step 6 (Creation of Candidate Official Version)

The Code Librarian prepares a Code Change Form (see Table B-5), and creates a candidate official code version. The form provides a complete record of the content of the candidate version.

3.7 Step 7 (Acceptance Testing)

The Cognizant Engineer and/or Programmer repeats the acceptance test cases for each update set with the candidate official code version. In some instances, it may not be necessary or even desirable to rerun the entire set of acceptance test cases, and the Section Leader may choose to waive the repetition of certain acceptance test cases. The Section Leader indicates those cases waived by initialing the appropriate space on the Code Change Form.

3.8 Step 8 (Review of Test Results)

The Cognizant Engineer and/or Programmer responsible for each update set reviews the results of the acceptance test cases and compares them with the earlier results reported in the completion report. If the new results are unchanged, the Cognizant Engineer or Programmer initials the appropriate spaces on the Code Change Form. If the results differ from the earlier calculations but are still acceptable, the Cognizant Engineer prepares an acceptance report. The acceptance report compares the two sets of results and explains the observed differences. If the test case results are unacceptable, the particular update set cannot be incorporated into the official code version. An interoffice memo will suffice as the acceptance report.

3.9 Step 9 (Acceptance of Test Results)

The Section Leader reviews all acceptance reports and inserts each into the code development file. The Section Leader initials the appropriate spaces on the Code Change Form to indicate acceptance.

3.10 Step 10 (Creation of Official Code Version)

When all acceptance testing is complete, the Section Leader signs the Code Change Form indicating that the candidate version is judged acceptable. The Code Librarian then catalogs the version under a permanent file name reserved for the official code version.

4. UPDATE FORMAT REQUIREMENTS

This section discusses the use of the standard CDC UPDATE program in conjunction with a CDC UPDATE deck processor program, CDCID, which is the principal computing tool needed to implement the quality control program described herein. The UPDATE deck format requirements, which provide a standard format for the documentation information needed for quality control, are also described.

The quality control program described in this report uses the CDC UPDATE utility program because: (a) the CDC UPDATE program is a commonly used method of making changes to code versions requiring traceability, since it lists changes including both insertions and deletions; (b) it is a standard CDC product which is available at all CDC computing facilities; (c) it is a powerful tool with many updating capabilities even including the reactivation of previously deleted cards; and (d) it allows a relatively large identifier field on each card. This final characteristic will be shown to be very useful in this quality control program.

In addition to the UPDATE card format requirements imposed by the CDC UPDATE program, a few additional requirements are placed on the UPDATE decks by this quality control program. These requirements ensure that a predictable format is provided which allows the CDCID computer program described in this report to perform its documentation and configuration control functions. This format requires inclusion of a documentation section that provides information relevant to the change and prescribes naming conventions for *IDENT, *COMDECK, and *DECK update cards.

Before the formal documentation requirements are explained in detail, a simple example of how a user would create a deck and modify it in a later update is given. While a general knowledge of the use of the CDC update program is assumed, a review of selected items and definitions is provided first.

4.1 Example of UPDATE Deck Creation and Modification

Each card on a CDC UPDATE program library is identified by a card identifier which consists of a name and a sequence number. The name portion of these identifiers may be created either by the use of *DECK, *COMDECK, or *IDENT directives. The deck identifying directives, *DECK and *COMDECK, are used to create new identifier names for consecutive sets of input cards. The *IDENT cards are used to create identifier names for changes to the sets of cards created by the *DECK and *COMDECK cards. Since the sequence numbers are of no concern in the procedures being discussed, the term "card identifier" will henceforth refer only to the name portion of the card identifier.

For the updating procedure of the method discussed, selected character positions of the card identifier are reserved for particular purposes. The first four characters are reserved to provide a constant reference to the deck being changed. The next two characters are used for a sequence number; i.e., the official version number. The last three characters are used for the program change label (PCL), which provides a one-to-one correspondence between each card on the program library and the documentation related to the card.

For example, suppose a model Developer wishes to insert a subroutine named SUBR1 into a code. Since only four characters are reserved to indicate the subroutine being changed or inserted, the name will have to be abbreviated. The Developer should first check the old program library to determine if there are any similar deck names already in use. Normally the subroutine name would simply be truncated to four characters and used for the first four characters of the deck identifier name. However, if a later addition of a subroutine named SUBR2 is expected or if the old program library contains a deck named SUBRA, for instance, the Developer would want to ensure uniqueness by using the characters SBR1 for the first four characters of the identifier name. (The CDCID UPDATE program will force uniqueness of the identifier name if necessary, but not in a manner always acceptable to the Developer.)

With one exception to be discussed later, the Developer does not usually have to be concerned with the contents of the fifth and sixth characters of the card identifier since the CDCID UPDATE program inserts a sequence number here for all

*DECK, *COMDECK, and *IDENT cards. Therefore, these characters are normally filled with the remaining characters of the subroutine name, if not abbreviated, or with minus signs.

The method of choosing a PCL for characters seven through nine will be discussed later. The essential thing about this choice is that the PCL be unique to a particular code change. Suppose the Developer chooses to use SQ1 for the PCL. Then the *DECK card would appear as follows:

*DECK SBR1--SQ1.

In addition to the subroutine source cards to be inserted following the *DECK card, a documentation section is required preceding the *DECK card. These cards provide information relevant to the program change. The minimum cards needed for the documentation section are shown in the example in Table 1.

As seen from the example, each card in the documentation section is a CDC UPDATE comment card (*/Δ) with some of them specially formatted for control purposes. The first such card is a title card for the change. It is identified by the first five characters, */Δ#*, and must contain the PCL (in this case SQ1) in columns 6 through 8. The "N" in column 10 is a sorting index. In this case it indicates that a new model is being inserted. A complete list of sorting indexes is given in Section 5. In the example, the SQ1 in the title card is the same as the PCL for every *DECK, *COMDECK, or *IDENT card pertaining to the change. The PCL also appears on the design and completion reports describing the model as discussed earlier. The actual title on the title card appears in columns 11 through 80, preferably with column 11 blank. The card following the title card of Table 1 indicates that John Doe is acting as both the engineer and programmer responsible for this model.

The next cards in the example are regular CDC update comment cards used to explain the change. In the case of a minor change, this may be the complete documentation for the change. For other changes, the documentation section serves only as a convenient abstract, and details may be retrieved from the design and completion reports.

The "*/Δ##END" card terminates the documentation for the model. The PCL appears again in columns 11 through 13 of the end card. The remaining cards in the example are exactly the same as one would normally use to make CDC UPDATE changes except for the special naming requirements on the *DECK, *COMDECK, and *IDENT identifier names. If the CDCID program were run on the cards in Table 1, the input and output card files would be identical.

There are, however, two good reasons for always running the update cards through the CDCID program prior to the CDC UPDATE program. One reason is that CDCID checks the format of the

Table 1. Sample deck insertion with documentation

```
┌──── Column 1

*/Δ#*SQ1ΔN ENERGY CONVERSION MODEL

*/Δ##ENGPR JOHN DOE

*/ΔTHIS MODEL CONVERTS

*/ΔMASS TO ENERGY

*/Δ##ENDΔΔSQ1

*DECK SBR1—SQ1
```

Note: The Δ symbol is a required blank.

special documentation cards and points out both outright errors and suspicious cards as well as provides an error summary and other diagnostic information. The second reason is that by using the same card that the Code Librarian will use later to automatically insert sequence numbers, this step can be checked out in advance of merging various updates.

The use of the card

*/Δ##IDENTNN

at the beginning of the input cards causes the sequence number NN (given in columns 11 and 12) to be inserted on all *DECK, *COMDECK, and *IDENT cards. For the example in Table 1, the CDCID card output file will contain a modified *DECK card—*DECK SBR123SQ1 with NN = 23.

Suppose now that three updates later, John Doe discovers an error in the model with the PCL of SQ1, and suppose also that for another model an input variable needs to be added. Suppose further that John Smith, a programmer, actually makes the change. Each of these two changes requires a new PCL different from any PCL already used for the program. For the change to the SQ1 model, let the PCL be SQ2 and for the addition of the new input variable, SI1. Table 2 shows the documentation and update cards for these changes.

Several things should be noted about this example. First the documentation for each change precedes the updates for the change. Second, note the keyword identifiers in columns 1 through 10 for the Cognizant Engineer and Programmer. Note after the second title card, the title continuation card which allows additional title information in columns 11 through 80 on as many cards as needed. Observe, also, the use of the pair of specially formatted cards in the second set of changes which delimit the input changes to the code. This will allow easy collection of all input changes for later inclusion into a manual update. If more than one subroutine had been changed for one of the models, each additional subroutine would need an *IDENT card with the same PCL but different characters in columns 1 through 6.

4.2 Identifier Names

The requirements for identifier names are reviewed in the following paragraphs. The identifier name can be described in terms of three fields, namely, the DECK name abbreviation field in columns 1 through 4, the sequence number field in columns 5 and 6, and the PCL field in columns 7 through 9.

The name abbreviation field is normally chosen as the first four characters of the common block or subroutine name when creating a new DECK or COMDECK.

Whenever the developer has good reason to select an abbreviation other than that formed by truncation, he may do so as shown in the example earlier. If the name is already less than four characters, the remaining characters may be filled with minus signs. When correcting any DECK or COMDECK, the name abbreviation field is exactly the same as the name abbreviation field used for the original DECK or COMDECK creation, regardless of how the name was chosen.

The sequence number field normally is set to the fifth and sixth characters of the subroutine or common block name or filled with minus signs if there are less than six characters in the subroutine name. This assumes that the name abbreviation was created by truncation. If it was not, then presumably the name abbreviation was chosen to be unique, and minus signs may be used for the sequence number. Since the CDCID program is going to insert a sequence number in place of the original fifth and sixth characters anyway, the only reason for using characters other than minus signs in this field is to aid in generating a unique name abbreviation field when necessary. The method of doing this is described in the CDCID program description (Section 5).

In order to avoid inserting sequence numbers into a *CALL statement referring to a COMDECK created in earlier updates, the CDCID program does not insert sequence numbers into *CALL statements. However, it automatically inserts the sequence number whenever it finds a $$ in the sequence number field of a *CALL statement. This allows the *CALL cards which refer to COMDECKs being inserted in the current update to have the same sequence number insertion as the new COMDECKs. To avoid confusion, do not use the $$ as a part of any other identifier name.

A further restriction imposed by the CDCID program is that a single blank or comma must be used between the *IDENT, *DECK, or *COMDECK directives and the following identifier.

Table 2. Sample update correction with documentation

```
*/Δ**IDENT26

*/Δ

*/Δ#*SQ2ΔE                ENERGY CONVERSION ARGUMENT SPELLING CORRECTION
*/Δ##ENGRΔ                JOHN DOE
*/Δ##PRGMR                JOHN SMITH

*/Δ

*/Δ                       CORRECT MISSPELLING OF ARGUMENT "N" TO "M"
*/Δ                       IN ENERGY CONVERSION MODEL. ALL PREVIOUS RESULTS
*/Δ                       USING THIS MODEL ARE INVALID. THIS ERROR WAS INTRODUCED
*/Δ                       IN UPDATE 23.

*/Δ

*/Δ##ENDΔΔSQ2

*/Δ

*IDENTΔSBR1—SQ2

*D SBR123SQ1.2            SUBROUTINE SUBR1 (E,C,M)

*/Δ
*/Δ
*/Δ#SI1ΔC                 ALLOW USER CONTROL OF MODEL A

*/Δ##ΔΔΔΔΔ                BY NEW INPUT-VARIABLE, COEFF
*/Δ##ENGR                 JOHN DOE
*/Δ##PRGMR                JOHN SMITH
*/Δ                       SEE REPORT NUMBER 12345 FOR AN EXPLANATION OF THIS
*/Δ                       CHANGE.

*/Δ##INPUT

*/Δ                       ON CARD NUMBER 18 ADD THE INPUT
*/Δ                       VARIANCE COEFF BETWEEN FRIC AND GRAV

*/Δ##END!N
*/Δ##ENDΔΔSI1
*/Δ
*IDENTΔABCDE-SI1
*D ABCDE.5
      * FRIC,COEFF,GRAV
```

The output of CDCID for the above cards will be the same except the *IDENT cards will become the following: *IDENTΔSBR126SQ2 and *IDENTΔABCD26SI1.

The last three characters of the identifier name are the PCL. While the only requirements on the PCL is that it be unique for each program change, it is desirable that the first character of a PCL is reserved to identify individual contributors, so that a person trying to understand a particular card will immediately know whom to question about it.

The second and third characters of the PCL are usually a letter and a number, respectively, and are chosen by the individual responsible for the model. One possible method for selection could be to use the second character to identify a model, and the third character as a model sequence number, which is incremented as the model is revised. However, the method being used most is simply to start with second and third characters A and 1 and increment them sequentially starting with the third until 9 is reached, then increase the second to B and start the third again at 1. The use of both zero and the letter "O" should be avoided.

4.3 Documentation Cards

The formal requirements for the documentation cards are explained in this subsection. Also refer to Table 3 for the order and types of documentation cards needed to explain code changes.

Each special documentation control card is recognized by a keyword in columns 1 through 10,

Table 3. Sample documentation cards

Column 1

*/Δ#*PCLΔI	TITLE IN COLUMNS 1-80
*/Δ##	TITLE CONTINUATION (11-80)
*/Δ##ENGR	NAME OF RESPONSIBLE ENGINEER
*/Δ##PRGMR	NAME OF RESPONSIBLE PROGRAMMER
*/Δ	USE THE ABOVE TWO CARDS OR, IF THE SAME PERSON,
*/Δ	USE THE FOLLOWING CARD
*/Δ##ENGPR	NAME OF ENGINEER-PROGRAMMER
*/Δ	HERE PLACE A DESCRIPTION OF THE MODEL OR A REFERENCE
*/Δ	TO THE DESCRIPTION
*/Δ##INPUT	
*/Δ	IF THE CHANGE REQUIRES AN INPUT CHANGE USE THE ABOVE
*/Δ	CARD AND DESCRIBE HERE OR REFERENCE THE CHANGE WITH
*/Δ	REGULAR CDC UPDATE COMMENT CARDS AND TERMINATE THE
*/Δ	INPUT COMMENT CARDS WITH THE FOLLOWING CARD
*/Δ##ENDIN	
*/Δ	ADDITIONAL MODEL DESCRIPTION CARDS ARE ALLOWED HERE
*/Δ	IT IS RECOMMENDED TO TERMINATE THE DOCUMENTATION
*/Δ	CARDS WITH THE FOLLOWING END CARD, BUT AN *IDENT
*/Δ	CARD WILL ALSO TERMINATE THE DOCUMENTATION SET
*/Δ##END PCL	THE PCL IN COLUMN 11 IS OPTIONAL

except the title card which is recognized by the first five characters only. Characters 6, 7, and 8 of the title card are the PCL.

The tenth character on the title card (I on the sample in Table 3) is the sorting index, which allows the selected printing of categories in addition to the model sorting that may be done using the PCL. These allow, for example, the printing of all important errors (all errors in a version which have been released externally). Sorting indexes are discussed further in Section 5.2.4.

Regular CDC UPDATE comment cards ("*/Δ" in columns 1 through 3) are used between the title card and end card for all documentation aside from that on the specially formatted cards. The use of the title continuation is not recommended since restricting title information to one card allows for a more manageable list of change titles to be produced.

Use of the above documentation card conventions allows the CDCID program, through its sort and edit capability, to provide a wide variety of selected portions of the total documentation available for the program, as well as to provide a concise record of the history of the program and its documentation.

5. CDCID PROGRAM METHOD

The purpose of the CDCID computer program is to provide a versatile and comprehensive automated system to document computer code development and changes. It provides for a one-to-one correspondence between each change to a computer program and a document which describes the change. Furthermore, it provides an easy and automated method for producing a wide variety of selected portions of the total documentation available concerning program changes.

A user's guide is provided in Section 6 to summarize the input needed by the casual users who are interested only in preparing documentation for insertion with their code changes. DFORM, a small auxiliary computer program, is described in Appendix C. It provides further automation to the documentation process by allowing the user to create a development form, complete with an abstract, from the documentation section of a set of updates.

The following description of the program assumes a knowledge of the use of the standard CDC UPDATE program.

5.1 Method Used

The CDC UPDATE program is used in conjunction with specially formatted CDC UPDATE comment cards to describe and delimit documentation sections in line with the regular CDC updates which are used to make program modifications. The one-to-one correspondence between the program changes and the documentation for those changes is maintained by use of a three character PCL which appears on each CDC *IDENT directive and on the documentation describing the changes. For short descriptions of changes or whenever convenient for the developer, the complete documentation for a change appears as specially formatted comment cards in the update input stream. For more lengthy documentation, all that is necessary in the input stream are the standard cards, described below, plus a reference to another document.

The method for obtaining these correspondences is by selecting CDC UPDATE identifiers as follows. The first four characters of the UPDATE identifier are an abbreviation corresponding to the subroutine name. The next two characters are the sequence number of the update being made. The last three characters are the PCL. Since the CDC UPDATE program appends this identifier to each card, every card in the program points to a document with the same three characters designated on it.

The CDCID program has the capability of inserting a common sequence number in the fifth and sixth characters of each card which can cause the creation of a CDC UPDATE identifier. It also has the capability of changing the fourth character of a DECK or COMDECK name to make it unique. The method of selecting the four characters is discussed in Section 5.3.

The PCL which relates the design or change document to the actual source cards is normally two letters followed by a number. Each change to the code should have a unique PCL designator.

5.2 Input Processing

There are 28 different types of CDC UPDATE cards which CDCID recognizes and processes. These are divided into the following major categories: identifier creation cards, regular CDC UPDATE cards, sequence number insertion cards, documentation cards, and sort-edit request cards. A brief description of the function of these types of cards is given below and is followed by a complete card-by-card description of each card in each category. For a complete listing of these different types of cards, see the flow chart for subroutine CTYPE in Appendix E.

The identifier creation cards are the CDC UPDATE cards which cause an identifier to be placed on a card; namely, the *ID, *IDENT, *DK, *DECK, *CD, and *COMDECK cards. The regular CDC UPDATE cards are the cards normally used to change the program, and have no special format. The sequence number insertion cards are instructions to the CDCID program to add a sequence number to certain regular CDC UPDATE cards. The documentation cards provide a description of the program changes. The sort-edit request cards are instructions to the CDCID UPDATE program to print complete or selected documentation according to the PCL or sorting index (which also appears on the documentation title cards) or according to a level of detail specified on the sort-edit request.

Each card used to control the documentation output is identified by use of the first 10 columns on the card. The proper number and location of blank characters in these columns is required by the program for proper card identification.

5.2.1 Identifier Creation Cards. A CDC UPDATE card starting with *ID, *IDENT, *DK, *DECK, *CD, or *COMDECK causes the first parameter on the card to be used as an identifier for each card and will, therefore, be referred to, herein, as an "identifier creation card."

5.2.2 Regular CDC Update Cards. The regular CDC UPDATE cards are all cards which are not specially formatted comment cards and which, therefore, do not fall into any of the other categories below. The regular CDC UPDATE cards are the cards which are normally used to update the program and are not usually part of the documentation. However, CDCID does include the capability of including these cards as part of the documentation. The identifier creation cards are regular CDC UPDATE cards which are subject to modification as described under the description of the sequence number insertion cards, below.

5.2.3 Sequence Number Insertion Cards. In order to conveniently provide an identical sequence number for each card identifier inserted in the same update, CDCID has the capability of automatically inserting a sequence number into the fifth and sixth characters of the first parameter of an *ID, *IDENT, *DK, *DECK, *CD, or *COMDECK CDC UPDATE directive. Cards with the following format in columns 1 through 12 accomplish this, respectively, for these six UPDATE directives taken in pairs:

*/ ##IDENTzz

*/ ##NDECKzz

*/ ##CDECKzz.

The zz represents the sequence number which will be inserted. If zz contains blanks, no insertion will be made. The first occurrence of any one of the above three cards initializes the zz value for the others. No insertion is made for any card that precedes the first occurrence of any sequence insertion card. The insertion of the sequence number continues until the end of the input file, or until another sequence insertion card of the same type overrides the previous value. In addition, the CDECK request controls the insertion of the sequence number on *CA and *CALL CDC UPDATE directives. Only *CA and *CALL statements which contain a $$ in the fifth and sixth characters of the identifier have these characters replaced by the sequence number.

For example, in the normal use of this option, only a single */ ##IDENTzz card is used as the first card in the update input. However, in order to apply the sequence insertion only to *IDENT cards and not to DECK and COMDECK insertions, both the */ ##NDECK and the */ ##CDECK cards with blank values for zz would follow the */ ##IDENTzz card.

To simplify the coding to implement the sequence number insertion CDCID requires the use of the *IDENT or *ID cards in the update input stream as follows. The *IDENT or *ID must be followed by a single blank, then the first six characters of the subroutine name, and then the same three-character PCL used on the documentation of the changes. CDCID takes the sequence number from the IDENT sequence number insertion card and inserts it into the fifth and sixth characters in the *IDENT name. When a new DECK or COMDECK is being added, the DECK name should be formed in the same way as for the *IDENT name.

While the fifth and sixth characters are replaced later, they still should contain the fifth and sixth characters of the DECK name, or dashes if the DECK name is shorter than six characters. Some CDC UPDATE changes such as the *MOVE option do not directly affect a subroutine. In this case, in place of the DECK name on the IDENT card use CDCUPD, or select some other meaningful name.

5.2.4 Documentation Cards. There are eight types of documentation cards as follows:

The title card

*/ #*XXX Y title in columns 11-80

and the end card

*/ ##END

begin and terminate the set of documentation cards, called an edit set. The title card may be continued with an unlimited number of title continuation cards of the form:

*/ ## title continuation in columns 11-80

(Columns 6 through 10 on the title continuation card must be blanks.)

The second and third cards following the title cards should contain the names of the Cognizant Engineer and Programmer, respectively, who are responsible for the changes. The format for these cards is

*/ ##ENGR name in columns 11-80

*/ ##PRGMR name in columns 11-80

If the Engineer is the Programmer, the following card may be used to generate the above two cards.

*/ ##ENGPR name in columns 11-80.

No other cards are allowed between the title, title continuation, or Programmer/Engineer cards to prevent generation of extraneous headings by the edited format of the printed output.

The XXX in the title card above is the PCL. The first character may be assigned to the model Developer. The second character, usually a letter, is chosen by the Developer. The third character, usually a number, may be a sequence number which is 0 for the base document and is incremented by 1 for changes related to the same base document. Alternatively, it may be any character needed to make the PCL unique.

The Y on the title card is the sorting index, which allows the selected printing of categories in addition to the model sorting that may be done using the PCL. These allow, for example, the printing of all important errors (all errors in a version which has been released externally). The indexes that have been reserved are shown in Table 4.

Within an edit set, a set of cards describing input changes should be delimited by the following cards at the start and the end of the changes, respectively.

*/ ##INPUT

and

*/ ##ENDIN.

The use of the */ ##ENDIN card at the end of the input changes is not necessary if the next card is the */ ##END card. For input changes which appear in a document, not in the update input, this */ ##INPUT card should precede a reference to the document.

In order to prevent a missing end card from causing subsequent edit sets or program changes from

Table 4. Sorting indexes

C	=	Model change
E	=	Error correction in a version not released externally
L	=	Error correction in the original base program
N	=	New model insertion
P	=	Programming change (other than Types E, L, R, S and V)
R	=	Error correction in a version released externally
S	=	System-dependent change
T	=	Error correction for which the error occurred only in the development process and not present in any official code version
V	=	Program conversion change which is not system dependent

being incorporated into the previous edit set, an identifier creation card or title card terminates an edit set.

5.2.5 Sort-Edit Request Cards. The sort-edit request cards consist of an option identifier in columns 1 through 10 and option parameters in columns 11 through 20. Two types of sort-edit request cards, the edit and the sort cards, determine which title cards (and associated cards as determined by sort-edit level of detail requests) are to be processed, according to the PCL and sort index, respectively. The remaining seven types of sort-edit requests determine the level of detail of the output for the outstanding sort or edit request (six types) and the format of this output (one type).

The edit request card controls the placing of output on the output file according to the PCL, or portion thereof specified. The card

*/ ##EDIT XYZ

specifies that all documents containing XYZ as the 6th, 7th, and 8th characters on the title card of an edit set are to be placed on the output file for printing. If the Z is omitted, printing is to be done for all PCLs containing XY for the first two characters; similarly, the X may be used alone for the search. If XYZ = 999, then the entire file is printed. The card

*/ ##SORT ABCDEFGHIJL

specifies that each of 10 characters specified in columns 11 through 20 is to be used to compare against the sorting index in column 10 of each edit set title card. Any number of characters up to 10 may be used in the request. Whenever a match is found with any character, the amount of output to be printed is determined by the current sort-edit level of detail request. In order to control the order of the edited output, separate sort and edit cards may be used, since between a sort or edit request the CDCIN file is rewound.

For the following six cards, each subsequent card causes the production of output which includes the output produced by the previous card. The card

*/ ##TITLE

produces a list of the title cards for all documentation requested by the last sort or edit request. The card

*/ ##TOC

includes, in addition, the title continuation cards and Engineer and Programmer names. The card

*/ ##TOCIN

includes, in addition, the input change sections. The card

*/ ##EDSET

includes, in addition, the edit set documentation. The card

*/ ##EDSID

includes, in addition, the *ID, *IDENT, *DK, *DECK, *CD, and *COMDECK cards between the title card to which the sort or edit request applies and the following title card. The card

*/ ##FORM

causes the output to be printed in an edited format, rather than as a straight listing of the cards. The edited format is recommended since it separates the cards according to PCL. Repeated use of the FORM card toggles this option off and on. The default value is off.

5.3 Subroutine Abbreviated Name Generation

Since the CDC UPDATE program requires unique identifiers, CDCID generates a unique fourth character card identifier when necessary to ensure uniqueness, as described below. Nevertheless, the user can improve the clarity of the update by using as many characters as available, up to six, of the original deck name on the identifier creation cards. A suggested convention is to use minus signs to fill out the first six characters if the original DECK name is less than six characters.

CDCID accumulates an array containing the identifiers found in the UPDATE input and if an identifier which is not unique in characters 1 through 4 and 7 through 9 is encountered, then the fourth character is modified to ensure uniqueness. This is done prior to sequence number insertion.

In order to increase the probability that the fifth and sixth characters produce the same substitution characters produce the same substitution character, the choice of the substitution character is dependent on these characters in the following manner. An integer is formed from the fifth and sixth characters; this integer module 7 selects the substitution characters for the following set:

- * / ( ) $ =

These characters are used since they are legal identifier characters, but are rarely used and, therefore, should cause a minimum of conflicts with existing names. If this identifier is still not unique, the fourth character is incremented by one bit (starting over at A if a display code of 54 is exceeded) until uniqueness is obtained.

Note that the automatic name change mechanism is not applied to *CA and *CALL directives.

5.4 Program Change Label Collating

Two new input options are added to allow sorting of title cards alphabetically by PCL and to allow placing selected output on a separate file for further processing.

5.4.1 PCL Collating Input. The following card placed before a sort or edit request toggles a switch (initially false) to place the output card images as modified by the program on the file:

```
*/ ##CLECT.
```

The card

```
*/ ##COLAT
```

placed anywhere in the CDCID input collates the cards on file SELECT alphabetically according to the contents of columns 6, 7, and 8, the position of the PCL on the title cards. Therefore, the only cards which should be placed on the SELECT file prior to use of this option are title cards.

5.4.2 PCL Collating Example. To get PCLs printed alphabetically, use the following cards:

```
*/ ##TITLE

*/ ##CLECT

*/ ##EDIT EVERY PCL or request particular PCLs

*/ ##COLAT.
```

5.5 Program Files

The program card for CDCID contains the information shown in Table 5.

5.6 Program Control Cards

The following sample control cards may be used to execute CDCID and pass a modified file to the CDC UPDATE program.

```
ATTACH,CDCIDE,ID=GLS,MR=1.

COPYBR, INPUT, UPDIN.

REWIND, UPDIN.

CDCIDE.

REWIND, UPDMOD.

UPDATE,I=UPDMOD.
```

5.7 Program Structure and Documentation

CDCID is programmed using structured programming. The program consists of two major sections controlled by subroutines UPDATE and EDITOR. Subroutine UPDATE controls (a) the processing of the input cards and may change a few of them (e.g., by inserting a sequence number into *IDENT cards) and (b) the placing of the modified cards onto file UPDMOD. Subroutine EDITOR is used only if edited documentation output is requested. For example, it may search for and print out a list of all program change title cards or various more detailed amounts of documentation. Table 6 provides a simplified structured flow chart for the program. A more detailed flow chart may be found in Appendix E, which is a source listing of the program.

The program is self-documenting and contains most of the information presented in this section. In addition, it contains information of interest to programmers, such as definitions of the variables used in common blocks and argument lists.

Table 5. Contents of CDCID program card

```
     PROGRAM CDCID(UPDIN,              UPDAUX,              UPDMOD,
1          TAPE5 = UPDIN,      TAPE56 = UPDAUX,     TAPE55 = UPDMOD,
2                OUTPUT,
3        TAPE6 = OUTPUT,        DEBUG = OUTPUT,
4           SPOUT = 256,          SELECT = 256,
5        TAPE76 = SPOUT,      TAPE77 = SELECT)
```

where

UPDIN	=	the input unit which, without the use of CDCID, would be the normal input to the CDC UPDATE program.
UPDAUX	=	an input unit which is processed prior to the UPDIN file. It allows sort-edit type requests to be inserted in front of the regular UPDATE file.
UPDMOD	=	the output unit which contains the same information as UPDIN, except that when the IDENT program option is used, sequence numbers are inserted into the *IDENT or *ID cards. UPDMOD is rewound at the end of execution. Also, the fourth character of the DECK name may change on the identifier creation cards.
SELECT	=	output file for edited card output. Its principal purpose is to hold title cards for collating.
SPOUT	=	temporary output file for sorted output from processed select file. SPOUT is later processed for formatting and its edited contents placed on the output file.

Table 6. Simplified structured flow chart for CDCID

```
**                 CDCID (MAIN PROGRAM)
**   ------------------------------------------------------------------
**   - BINIT (BLOCK DATA)                                             -
**   ------------------------------------------------------------------
**   - UPDATE                                                         -
**   -          -------------------------------------------------------
**   -          - INIT                                                -
**   -          -------------------------------------------------------
**   -          - INITU                                               -
**   -          -------------------------------------------------------
**   -          - // DO WHILE INPUT CARDS EXIST                       -
**   -          - // -------------------------------------------------
**   -          - // - INPUT                                          -
**   -          - // -------------------------------------------------
**   -          - // - CTYPE                                          -
**   -          - // -          --------------------------------------
**   -          - // -          - ASLASH              - ICHECK        -
**   -          - // -------------------------------------------------
**   -          - // - SESTOR                                         -
**   -          - // -------------------------------------------------
**   -          - // - SEQSIN                                         -
**   -          - // -------------------------------------------------
**   -          - // - NAMSUB                                         -
**   -          - // -          --------------------------------------
**   -          - // -          - NAMCHK                              -
**   -          - // -          --------------------------------------
**   -          - // -          - NAMSET                              -
**   -          - // -------------------------------------------------
**   -          - // - SETSEQ                                         -
**   ------------------------------------------------------------------
**   - PINPUT                                                         -
**   ------------------------------------------------------------------
**   - EDITOR                                                         -
**   -          -------------------------------------------------------
**   -          - INITE                                               -
**   -          -------------------------------------------------------
**   -          - // DO WHILE INPUT CARDS EXIST                       -
**   -          - // -------------------------------------------------
**   -          - // - SEDREQ                                         -
**   -          - // -------------------------------------------------
**   -          - // - CTYPE                                          -
**   -          - // -          --------------------------------------
**   -          - // -          - ASLASH              - ICHECK        -
**   -          - // -------------------------------------------------
**   -          - // - LEVDET                                         -
**   -          - // -------------------------------------------------
**   -          - // - EMATCH                                         -
**   -          - // -------------------------------------------------
**   -          - // - CMATCH                                         -
**   -          - // -------------------------------------------------
**   -          - // - COUT                                           -
**   ------------------------------------------------------------------
**   - SUMMAR                                                         -
**   ------------------------------------------------------------------
**   - OUTSET                                                         -
**   ------------------------------------------------------------------
```

6. CDCID USER'S GUIDE

This section summarizes the input needed to the code in order to insert documentation changes along with a set of changes to the user's program. Each special type of card used for program documentation control has a keyword in columns 1 through 10.

To insert a sequence number into the fifth and sixth characters of the *ID cards, the user should use the */ ##IDENT card with a sequence number in columns 11 and 12. Normally, this card precedes all other update cards. For more complicated sequence insertion requests, see the detailed input description in Section 5.2.3.

Prior to each block of CDC UPDATE cards used to modify a user program which have a common purpose, the user should insert the following set of cards to document that purpose. First, insert a title card of the form

*/ #*XXX Y title in columns 11-80

where the XXX is a PCL, typically starting with an initial assigned to the Programmer. The Y is a sorting index. Refer to Table 4, sorting indexes, for a list of possible sorting indexes which may be used. The card

*/ ##

with title continuation information in columns 11 through 80 may be used to extend the title card information. The cards

*/ ##ENGR

*/ ##PRGMR

*/ ##ENGPR

can be used to identify the Engineer and Programmer, or both if the same, respectively, by placing the name of the responsible individuals starting in column 12.

Next, use CDC UPDATE comment cards (*/) to document the changes to the user program or to refer to a document which describes the changes.

If the changes require changes or additions to the program input, bracket the documentation of, or reference to, the input changes (even if the same as the principal reference) with the cards

*/ ##INPUT

and

*/ ##ENDIN.

Terminate the entire set of documentation for the block with card

*/ ##END.

If a specially formatted output edit is desired, place the card

*/ ##FORM

before the first sort-edit request.

The sort-edit requests can then follow to produce complete documentation using the cards

*/ ##EDSAL

and

*/ ##EDIT EVERY PCL.

Examples of various files associated with a CDCID run are provided in Appendix F, which shows the output from a sample problem.

7. CONCLUSIONS

The CDCID computer program, when used in conjunction with the code development and change procedures described in this report, provides a convenient method for automatic documentation of code changes. Its use greatly enhances the documentation of a computer program by providing an identifier on each source card which provides information about the purpose, order of insertion, and originator of the card.

8. REFERENCE

1. J. W. Spore et al., *TRAC-BD1: An Advanced Best Estimate Computer Program for Boiling Water Reactor Loss-of-Coolant Accident Analysis*, NUREG/CR-2178, EGG-2109, October 1981.

APPENDIX A
DEFINITIONS

APPENDIX A
DEFINITIONS

The meaning of most of the terms used in the body of this report are normally apparent from the words and the context; however, to ensure that a precise meaning of each of the significant terms used is available, Table A-1 provides a list of definitions.

Table A-1. Definition of terms

Acceptance Report	Document that explains differences in test case results as reported in a completion report and as generated upon the final acceptance testing required for completion of the Code Change Form. An interoffice memo will suffice as an acceptance report.
Acceptance Test Case Set	A set of test cases that exercises new models or code changes. A good acceptance test case set is one which exercises the additions over, as much as practical, the entire physical range of applicability and the entire range of user options.
Branch Manager	A second level supervisor (the immediate supervisor of a section leader).
Code	A code is an abbreviated way of referring to a computer program.
Code Capability	A description of the ultimate code product.
Code Librarian	The code librarian is responsible for maintaining configuration control of a code. Functions of the code librarian include identifying coding conflicts among merged updates to the code, maintaining needed auxiliary files, and maintaining a test case library.
Code Version	A code version is any set of computer code operations and program data other than user input data. A set of operations and data which differs in any way from another set comprises a separate code version.
Code Version Sequence	The sequence of planned released code versions leading to the ultimate product or code capability.
Cognizant Engineer	Person responsible for designing and/or implementing new models and model changes.
Cognizant Programmer	Person responsible for designing and/or implementing coding changes.
Completion Report	Formal document, see Table B-3 in Appendix B.
Configuration Control	Configuration control is the maintenance of reproducibility and traceability of code versions and supporting data. Good configuration control includes the monitoring of the quality of the changes to the code and documentation of the code.
Deficiency Report	Formal document that identifies a deficiency or problem with an official code version, see Table B-4 in Appendix B.
Design Report	Formal document, see Table B-2 in Appendix B.

Table A-1. (continued)

Development File	A file providing complete traceability of an official code version.
Major Schedule	A description of the schedule for development of a specific code version in terms of the work package start and completion dates.
Manpower Loading Chart	A display of individual commitment by person for each work item in the minor networks.
Milestone Chart	A description of the task steps necessary to produce a given code version.
Minor Network	A description of the detailed work breakdown within each work package. The description should include detailed estimated man-hours and material dollars for each work item and the associated completion schedule for each work item.
Official Code Version	Code version created in conformance with model development and code change procedures.
Released Code Version	An official code version which has been released for public use.
Reproducibility	Reproducibility implies the capability of reproducing code versions by an automated process.
Section Leader	The lowest level supervisor responsible for an entire computer program development effort.
Traceability	Traceability implies that data are preserved which document both the actual updates and the control cards used to make a code version. The quality of the traceability is improved when documentation is added providing the reasons for the updates or when a link between the updates and its documentation is provided.
Work Package	For a given set of related tasks, a work package will specify the scope, the inputs required to complete the tasks, the deliverable outputs, the associated major milestones, the allocated resources, and the minor network.

APPENDIX B
SAMPLE QUALITY CONTROL FORMS

APPENDIX B
SAMPLE QUALITY CONTROL FORMS

The following sample quality control forms are provided to aid the reader interested in establishing a quality control program. Table B-1 provides a Code Development Form, which should be filled out for every update to the computer code.

Tables B-2 and B-3 suggest a format for and contents of a design report and completion report, respectively. Table B-4 suggests the contents needed for a Code Deficiency Report, and Table B-5 provides a Code Change Form.

Table B-1. Code development form

Title of Model or Change: PCL

Work Package Number:
Cognizant Engineer:
Cognizant Programmer:

Abstract:

Manual Change Required (Circle) Yes No

Contents of Development File

	Yes	No	Completion Date
Deficiency Report			
Design Report			
Completion Report			
Acceptance Report			
Update documentation			

Other:

Approval of Development Plan:

Section Leader	Date

Table B-2. Suggested format for design reports

1. Report Cover
2. Acknowledgments
3. Table of Contents
4. Model/Code Change Requirements
   - Background information and motivation for model or change
   - Specific requirements
5. Proposed Model/Code Change Design
   a. Model Description
      - Complete analysis or formulation
      - Complete set of auxiliary data (boundary and initial conditions, supplemental material)

   b. Coding Changes
      - Explanation of how model/change to be implemented
      - List of subroutines and common blocks to be affected
      - General flow charts indicating how subroutines to be added or changed interact with code
      - Input revisions including acceptable ranges of input data and input diagnostic messages
      - Output revisions
6. Proposed Acceptance Test Cases
   - List and describe all test cases
   - Test cases should exercise all input options
7. References

Table B-3. Suggested format for completion reports

1. Report Cover
2. Acknowledgments
3. ˜ ble of Contents
4. Model/Code Change Requirements
   - Background information and motivation for modei
   - Specific requirements
5. Model/Code Change Final Design

   a. Final Model Description
   - Complete analysis or formulation
   - Complete set of auxiliary data

   b. Coding Changes
   - Explanation of how model implemented
   - List of subroutines and common blocks affected
   - Summary flow charts for each new routine
   - General flow charts indicating how added or changed subroutines interact with code
   - Glossary defining FORTRAN names for math variables used in analysis, input data, and important local variables
6. Input and Model Preparation
   - Modeling guidelines if appropriate
7. Output Description
   - Definition of edited quantities
   - Explanation of diagnostic messages and recommended remedial action
8. Control Cards (include only if model/change requires a new set of control cards)
9. Results of Acceptance Test Cases
   - Discuss each test case (refer to microfiche title and run ID)
10. References
11. Microfiche Envelope
    - Test cases
    - Listing of update set
12. Revision Appendix (if required)

Table B-4. Suggested format for code deficiency report

Code/Version ______________________ Date ______________

Deficiency Number ______________________ Deficiency Report Author ________
(to be assigned
by Section Leader)

1. Summary Description of Problem or Deficiency ______________________

____

____

2. Deficiency or Problem Initiator ______________________

Phone ______________ Address ______________

Response Requested? ______________ Yes ____ No ____

3. Deficiency Category

________	Input Error	________	Code Deficiency or Shortcoming
________	Code Error	________	Use Beyond Code Design Limits
________	Machine Error	________	Unknown

4. Plan for Resolution (Use Additional Sheet if Necessary) ______________

____

____

____

____

____

____

____

____

5. Resolution: __________ Yes __________ No __________ N.A.

If yes, Date ______________________________

Table B-5. Code change form

Version Number: PCL:

Title:

ACCEPTANCE TEST CASE RUNS

RESULTS

Test Name	Microfiche Label	Completion Report Version	Last Version Used for Checkout	Comments

APPROVALS

Test Name	Results Code
	I = Identical
	E = Essentially Same
	D = Different
		Version Code
		F = Final
		N = Not Final
			Accepted by Designer or Section Leader	Waved by Section Leader If Codes $\neq$ I,F

Approved-Section Leader

Date

APPENDIX C
DFORM, A COMPUTER PROGRAM FOR AUTOMATIC CREATION OF DEVELOPMENT FORMS

APPENDIX C
DFORM, A COMPUTER PROGRAM FOR AUTOMATIC CREATION OF DEVELOPMENT FORMS

DFORM is a computer program which allows a user to create a development form, complete with an abstract, from the documentation section of a set of updates. The main advantage of using DFORM is so that the abstract which is prepared as part of the required update deck documentation does not have to be retyped onto the development form. DFORM also relieves the code developer of the need to enter onto the form the title of the change, the program change label (PCL), developer's name, and the date of the update documentation. It also helps prevent a mismatch between the information filed on the development form and that included in the actual update.

The documentation card images must conform to the format requirements of the CDCID UPDATE program. However, other cards such as control cards and update cards may also be present on the input file (PCLFILE), since the creation of a development form is activated by the presence of a title card (*/ #* in columns 1 through 5) and terminated by an *ID, *IDENT, or */ ##END starting in column 1.

The DFORM program is written in FORTRAN-77. It requires no input cards since the directives needed to control its options are all accessible as execution card parameters. A call to GETPARM (see Reference C-1) within the program gives the code access to the parameter values. The default name of the file from which the development form information is taken is PCLFILE. The input on this file must be in the proper format since no error checking is done by DFORM. Prior processing of the update through the CDCID program, which performs extensive error checking may be desirable for those inexperienced with the format requirements.

An important requirement on the documentation placed on file PCLFILE is that all */ ##ENGR, */ ##PRGMR, and */ ##ENGPR cards must precede the rest of the documentation, since the first other type of card, except for empty comment cards, signals DFORM to begin processing the abstract printout.

The only other file, besides PCLFILE, used by the program is the OUTPUT file, which contains the development form.

Normally, the user will not need to use any of the user-controlled parameters allowed on the program execution card. The user simply places the update deck(s) on a local file with name PCLFILE and executes the CDC command:

```
DFORM.
```

This produces a development form with abstract, title, etc., on file output.

To place the update deck on a different file, the user places an additional parameter on the DEFORM command as follows.

```
DFORM,PCLFILE=filename.
```

If the changes being made are all minor and require no further documentation of any kind, the user may cause the letter X to mark all of the YES-NO options on the form by using the following.

```
DFORM,NO.
```

To process a file which contains many title cards but only those starting with a particular letter or letters are wanted, the user may use the PCL parameter. If the value used for the PCL is three characters, then only the title cards with this PCL in columns 6, 7, and 8 are processed to produce a development form. If the value for the PCL parameter is one or two characters, only titles with PCLs that start with these letters are used. For example,

```
DFORM,PCL=$A $.
```

creates development forms for all sets of update documentation with a PCL starting with the letter A, that is, processing begins when the following characters are found in columns 1 through 6: */#*A. The above three parameters may also be used together or in any combination.

Since it is desirable to have the development form printed on 8-1/2 by 11 paper, a procedure may be used for this purpose. Table C-1 shows such a procedure and contains its own documentation as

Table C-1. Procedure (PROC) for DFORM

```
GLSFTH9,T37,P2.                 GLSTIO
ACCNT(GLS,CHG=XXXXXXXXXX)
BEGIN,FTITLE,,TITLE1=$PROC/DFORMC4$.
COPY,INPUT,TPROC.
REWIND,TPROC.
RPL,TPROC,TPROCDF,ID=GLS,PW=XXXXXX,SID=GLS,RP=999.      *****
REWIND,TPROC.
COPYSBF,TPROC.
REWIND,TPROC.
COMMENT.CPT.  TEST THE PROCEDURE
ATTACH,PCLFILE,ID=GLS,MR=1.
BEGIN,DFORMX,TPROC,UID=GLS.
BEGIN,DFORMX,TPROC,GLS,PCL=SF.
BEGIN,DFORMX,TPROC,GLS,HELP.


.PROC,DFORMX,UID,HELP=NULL/HELP,PCLFILE=PCLFILE,NO=NULL/NO,
PCL=$   $,REP=$O$.
IFE,$#HELP$.NE.$HELP$,NOHELP.
ATT,DFORM,ID=GLS,SID=GLS.
REQUEST,ROUTDFO,*Q.
IFE,$#NO$.EQ.$NO$,DONONO.
DFORM,ROUTDFO,#PCLFILE=PCLFILE,#PCL=$PCL$,#NO=NO.
ELSE,DONONO.
DFORM,ROUTDFO,#PCLFILE=PCLFILE,#PCL=$PCL$.
ENDIF,DONONO.
RETURN,DFORM.
ROUTE,ROUTDFO,SC=2,DC=PR,FID=UID_DF,FC=41,TID=C,#REP=REP.
RETURN,ROUTDFO.
ELSE,NOHELP.
SET,DSC=1.
COMMENT..* -- HELP OUTPUT --
COMMENT..* DOCMENTATION FOR DFORMX PROC.
COMMENT..*
COMMENT..* THE USF OF DFORM REQUIRES:
COMMENT..* ATT,TPROC,ID=GLS,SID=GLS.
COMMENT..* PLUS A BEGIN CARD LIKE:
COMMENT..* (AS A MINIMUM:)
COMMENT..* BEGIN,DFORMX,TPROC,YOUR ID.
COMMENT..* (AS A MAXIMUM:)
COMMENT..* BEGIN,DFORMX,TPROC,YOUR ID,
COMMENT..* HELP,#PCL=XXX,
COMMENT..* #PCLFILE=XXXXXXX,#NO,#REP=X.
COMMENT..*
COMMENT..* MEANING OF THESE PARAMETERS.
COMMENT..*
COMMENT..* USER ID -- REQUIRED.
COMMENT..*   THE USER ID IS NORMALLY USED
COMMENT..*   AS A POSITIONAL PARAMETER,
COMMENT..*   BUT CAN BE USED WITH
COMMENT..*   #UID=YOUR ID.
COMMENT..*
COMMENT..* #PCL = 1, 2, OR 3 CHARACTERS --
COMMENT..* IF 3 CHARACTERS, ONLY THIS #PCL
COMMENT..* IS PROCESSED.  IF 1 OR 2
COMMENT..* CHARACTERS, A #PCL'S THAT START
COMMENT..* WITH THESE CHARACTERS ARE
COMMENT..* PROCESSED.
COMMENT..*
COMMENT..* #NO -- OMITTED, DOES NOTHING.
COMMENT..* #NO -- PRESENT, PUT X IN ALL NO COLUMNS.
COMMENT..*
COMMENT..* HELP -- PRINTS THIS DOCUMENATION.
COMMENT..*   IF USED, DFORMX DOES NOTHING ELSE.
COMMENT..*   HELP IS NORMALLY USED AS A
COMMENT..*   POSITIONAL PARAMETER, BUT CAN BE
COMMENT..*   USED WITH
COMMENT..*   HELP=HELP.
COMMENT..*
COMMENT..* #PCLFILE -- #PCLFILE IS DEFAULT INPUT
COMMENT..* UNIT FOR UPDATE DECK.  IT CAN BE SET
COMMENT..* EQUAL TO THE NAME OF AN ALTERNATE FILE.
ENDIF,NOHELP.
```

comment cards which the reader may refer to in order to understand all of its options.

The simplest use of this procedure is as follows. Attach a local file, PCLFILE, containing the cards for which development forms are to be generated with the following card:

ATTACH, PCLFILE, PFNAME, ID = UID.

Here, PFNAME is the name of your permanent file and UID is your user ID.

Next, attach the file containing the PROC, as with the following control card.

ATT, TPROC, ID = GLS, SID = GLS.

INEL users, note that the above card assumes the PROC is on a file, TPROC, belonging to user GLS, and that the file uses an ATT, not an ATTACH, since it is a file contained on a Lehigh University Small Permanent File Manager managed file.

Next, execute the procedure with the following card:

BEGIN, DFORMX, TPROC, UID.

Here, again, UID is your user ID.

Table C-2 provides a listing of the DEFORM program set up to produce development forms for the TRAC-BWR program developed at the Idaho National Engineering Laboratory by EG&G Idaho, Inc. Table C-3 provides a sample development form for this particular program as generated by DFORM.

REFERENCE

1. Control Data Corporation, *FORTRAN, Version 5, Reference Manual*, Rev. F, 60481300, 1982.

Table C-2. Source listing for program DFORM

```
GLSFTH9,T37,P2.                    GLSTIO FTN5 CDCID/DFORX
ACCNT(GLS,CHG=[redacted],PW=[redacted])
BEGIN,FTITLE,,TITLE1=$CDCID/DFORM6$.
COMMENT.CMT.ATTACH,FILESET,ID=GLS,MR=1.
COMMENT.CMT.GF,TRC/A/DFORX.
COMMENT.CMT.ATTACH,PCLFILE,ID=GLS,MR=1.
COMMENT.CMT.ATTACH,PCLFILE,ID=GLS,MR=1.
EXIT,U.
REQUEST,LGO,*PF.
FTN5,LO=S/A/R/M,DB.
RPL,LGO,DFORM,ID=GLS,PW=[redacted],XR=[redacted],SID=GLS,RP=999.
COPYBR,INPUT,PCLFILE.
REWIND,PCLFILE.
REQUEST,ROUT,*Q.
MAP(PART)
COMMENT.CMT.LGO,ROUT,PCL=A,NO.
LGO,ROUT,NO.
REWIND,ROUT.
COPY,ROUT,OUTPUT.
REWIND,ROUT.
COMMENT.CMT.ROUTE,ROUT,SC=2,DC=PR,FID=GLSCU,FC=41,TID=C.***REP=1.
      PROGRAM DFORM(OUTPUT)
*
* * * * * * * * * * PROLOGUE
*
*     TITLE:  DFORM -- CREATE A TRAC-BWR DEVELOPMENT FORM
*
*     PURPOSE:  THE PURPOSE OF THIS PROGRAM IS TO
*               CREATE A COPY OF THE TRAC-BWR DEVELOPMENT FORM
*               FROM A LISTING OF THE STANDARD DOCUMENTATION
*               ON A COMPUTER FILE.
*
*     AUTHOR-- G. L. SINGER, EG&G IDAHO, BOX 1625, IDAHO FALLS, ID. 83401
*
*     IMPLEMENTATION NOTES:
*                            ATT,DFORM,ID=GLS,SID=GLS,MR=1.
*                            FTN5,I=DFORMX,OPT=2,R=3,B=DFORM.
*                            DFORM.
*-----------------------------------------------------------------
*** INPUT DESCRIPTION ***
*
* INPUT ON FILE PCLFILE MUST BE STANDARD TRAC-BWR DOCUMENTATION
* IN GOOD FORM.
* NO ERROR CHECKING IS DONE ON THE FORM OF THE INPUT.
*
*------------------------------------------------------------------
*                     *** PROGRAM DOCUMENTATION ***
*
*       DFORM IS A COMPUTER PROGRAM WHICH ALLOWS A USER TO CREATE A
* DEVELOPMENT FORM, COMPLETE WITH AN ABSTRACT, FROM THE DOCUMENTATION
* SECTION OF A SET OF UPDATES.  THE MAIN ADVANTAGE OF USING DFORM IS SO
* THAT THE ABSTRACT WHICH IS PREPARED AS PART OF THE REQUIRED UPDATE
* DECK DOCUMENTATION DOES NOT HAVE TO BE RETYPED ON TO THE DEVELOPMENT
* FORM.  DFORM ALSO RELIEVES THE CODE DEVELOPER OF THE NEED TO ENTER
* ONTO THE FORM THE TITLE OF THE CHANGE , THE PROGRAM CHANGE LABEL,
* HIS NAME, AND THE DATE OF THE UPDATE DOCUMENTATION.  IT ALSO HELPS
```

Table C-2. (continued)

```
* PREVENT A MISMATCH BETWEEN THE INFOMATION FILED ON THE DEVELOPMENT
* FORM AND THAT INCLUDEDIN THE ACTUAL UPDATE.
*       THE DOCUMENTATION CARD IMAGES MUST CONFORM TO THE FORMAT
* REQUIREMENTS OF THE COCID UPDATE PROGRAM.  HOWEVER, OTHER CARDS SUCH
* AS CONTROL CARDS AND UPDATE CARDS MAY ALSO BE PRESENT ON THE INPUT
* FILE (PCLFILE), SINCE THE CREATION OF A DEVELOPMENT FORM IS ACTIVATED
* BY THE PRESENCE OF A TITLE CARD ('*/ **' IN COLUMNS 1-5) AND
* TERMINATED BY AN '*ID', '*IDENT', OR '*/ **END  ' STARTING IN
* COLUMN 1.
*       THE DFORM PROGRAM  IS WRITTEN IN FORTRAN-77.  IT REQUIRES
* NO INPUT CARDS SINCE THE DIRECTIVES NEEDED TO CONTROL ITS OPTIONS
* ARE ALL ACCESSABLE AS EXECUTION CARD PARAMETERS.  A CALL TO
* GETPARM WITHIN THE PROGRAM GIVES THE CODE ACCESS TO THE PARAMETER
* VALUES.  THE DEFAULT NAME OF THE FILE FROM WHICH THE DEVELOPMENT
* FORM INFORMATION IS TAKEN IS PCLFILE.  THE INPUT ON THIS FILE MUST
* BE IN THE PROPER FORMAT SINCE NO ERROR CHECKING IS DONE BY DFORM.
* PRIOR PROCESSING OF THE UPDATE THROUGH  THE COCID PROGRAM, WHICH
* PERFORMS EXTENSIVE ERROR CHECKING MAY BE DESIRABLE FOR THOSE
* INEXPERIENCED WITH THE FORMAT REQUIREMENTS.
*       AN IMPORTANT REQUIREMENT ON THE DOCUMENTATION PLACED ON
* FILE PCLFILE IS THAT ALL '*/ **ENGR', '*/ **PRGMR', AND
* '*/ **ENGPR' CARDS MUST PRECEDE THE REST OF THE DOCUMENTATION,
* SINCE THE FIRST OTHER TYPE OF CARD, EXCEPT FOR EMPTY COMMENT
* CARDS, SIGNALS DFORM TO BEGIN PROCESSING THE ABSTRACT PRINT OUT.
*       THE ONLY OTHER FILE, BESIDES PCLFILE, USED BY THE PROGRAM
* IS THE OUTPUT FILE, WHICH CONTAINS THE DEVELOPMENT FORM.
*       NORMALLY, THE USER WILL NOT NEED TO USE ANY OF THE USER
* CONTROLLED PARAMETERS ALLOWED ON THE PROGRAM EXECUTION CARD.  HE
* SIMPLY PLACES HIS UPDATE DECK(S) ON A LOCAL FILE WITH NAME PCLFILE
* AND EXECUTES THE CDC COMMAND:
*                                DFORM.
*
* THIS PRODUCES A DEVELOPMENT FORM WITH ABSTRACT, TITLE, ETC. ON FILE
* OUTPUT.
*       IF THE USER WISHES TO PLACE HIS UPDATE DECK ON A DIFFERENT FILE,
* HE MAY PLACE AN ADDITIONAL PARAMETER ON THE DEFORM COMMAND AS
* FOLLOWS.
*                                DFORM,PCLFILE=FILENAME.
*
*       IF THE CHANGES BEING MADE ARE ALL MINOR AND REQUIRE NO FURTHER
* DOCUMENTATION OF ANY KIND, THE USER MAY CAUSE THE LETTER 'X' TO
* MARK ALL OF THE YES-NO OPTIONS ON THE FORM BY USING THE FOLLOWING.
*
*                                DFORM,NO.
*
*       IF THE USER WISHES TO PROCESS A FILE WHICH CONTAINS MANY TITLE
* CARDS BUT ONLY WANTS ONES STARTING WITH A PARTICULAR LETTER OR
* LETTERS, HE MAY USE THE PCL PARAMETER.  IF THE VALUE USED FOR THE PCL
* IS THREE CHARACTERS, THEN ONLY THE TITLE CARDS WITH THIS PCL IN
* COLUMNS 6, 7, AND 8 WILL BE PROCESSED TO PRODUCE A DEVELOPMENT FORM.
* IF THE VALUE FOR THE PCL PARAMETER IS ONE OR TWO CHARACTERS, ONLY
* TITLES WITH PCL'S THAT START WITH THESE LETTERS WILL BE USED.
*       FOR EXAMPLE,
*                                DFORM,PCL=$A  $.
*
* WILL CREATE DEVELOPMENT FORMS FOR ALL SETS OF UPDATE DOCUMENTATION
* WITH A PCL STARTING WITH THE LETTER 'A', THAT IS, PROCESSING WILL
```

Table C-2. (continued)

```
* BEGIN WHEN THE FOLLOWING CHARACTERS ARE FOUND IN COLUMNS 1-6:
* '*/ **A'.  THE ABOVE THREE PARAMETERS MAY ALSO BE USED TOGETHER OR IN
* ANY COMBINATION.
*
*                       *** END OF PROGRAM DOCUMENTATION ***
*
      IMPLICIT LOGICAL (A-Z)
*
*** CONSTANTS...
*
*             (FILE UNIT NUMBERS)
      INTEGER F5
      PARAMETER (F5 = 5)
*
*
*** VARIABLES...
*
       INTEGER IOS, ABSTR, J, LN, LIN, I, NPARM
       INTEGER IPARM(10)
       CHARACTER PCLFILE*16,PCLSAV*3
       CHARACTER CHOLD*80,STAT*10,BL*10,CONLY*80
       CHARACTER ENGR*40,PRGMR*40,BLANK4*40,PDAT*10
       CHARACTER DDAT*10,DLINE*80,PAREN*35,CDATE*10,DATE*10
       CHARACTER PARNX*35,PCLADD*3,CARDAD*3,PCL*7,AND*7
       CHARACTER PARM1(10)*10,PARM2(10)*10
*
*** DATA
*
      DATA STAT /'OLD'/
      DATA PCLFILE/'PCLFILE'/
      DATA BL /'          '/
      DATA ENGR /' '/, PRGMR /' '/
      DATA PDAT /'##########'/,DDAT /'----------'/
      DATA PAREN /'(    )        (  )      (               )'/
      DATA PARNX /'(    )        ( X)      (               )'/
      DATA PCL /'       '/, AND /'       '/
*
* * * * * * * * * EXECUTION
*
      CDATE = DATE()
C
C     SET UP CHARACTER TEST VARIABLES
C
      CONLY = '*/              '//BL//BL//BL//BL//BL//BL//BL
      BLANK4 = BL//BL//BL//BL
      DLINE = DDAT//DDAT//DDAT//DDAT//DDAT//DDAT//DDAT//DDAT
C
C     GET JOB CARD PARAMETERS
C
         AND = 'XX'
         PCL = '  '
      DO 188 I = 1,10
      CALL GETPARM(PARM1(I),PARM2(I),IPARM(I) )
      IF (IPARM(I) .LT. 0) GO TO 199
  188 CONTINUE
  199 CONTINUE
      NPARM = I-1
```

Table C-2. (continued)

```
C
      PCL = '    '
      ANO = 'XX'
      DO 248 I = 1,NPARM
      IF (PARM1(I) .EQ. 'PCL') THEN
            IF (IPARM(I) .EQ. 1 .OR. PARM2(I) .EQ. '    ') THEN
               PCL = '    '
            ELSE
               PCL = PARM2(I)
            ENDIF
         ELSE IF (PARM1(I) .EQ. 'NO') THEN
             ANO = 'NO'
         ELSE IF (PARM1(I) .EQ. 'PCLFILE') THEN
            IF (IPARM(I) .EQ. 0) PCLFILE = PARM2(I)
         ELSE
            PRINT *,'UNRECOGNIZED PARAMETER:',PARM1(I)
      ENDIF
  248 CONTINUE
C
C     OPEN THE FILES
C
C*
      STAT = 'OLD'
      OPEN(F5,FILE=PCLFILE,STATUS=STAT,IOSTAT=IOS)
C*
      IF (IOS .GT. 0) THEN
              PRINT *,'OPENING OF ',STAT, 'FILE ' ,PCLFILE,
     :        ' NOT SUCCESSFUL.  $*-*$ FATAL ERROR'
                STOP
      ENDIF
C*
C
C     SET FOR 6 LINES PER INCH
      PRINT '(''1'',/''S'')'
C
C     LOOP OVER PCLFILE CARDS
C
      ACTIVE = .FALSE.
      ABSTR = 0
      DO 288 J = 1,99999
         READ(F5,'(A)',IOSTAT=IOS,END=212) CHOLD
C*
  212    IF (IOS .NE. 0) THEN
               GO TO 299
         ENDIF
C
C     SET UP TO CHECK WHETHER THERE ARE RESTRICTIONS ON WHICH PCL'S
C     ARE TO BE PROCESSED, AS SPECIFIED ON THE DFORM EXECUTION CARD
C     AND BROUGHT IN ON THE GETPARM CARD.
C
      LN = INDEX(PCL,' ') - 1
      IF (LN .LE. 0) THEN
         LN = 0
         PCLADD = '   '
         CARDAD = '    '
      ELSE IF (LN .EQ. 1) THEN
         PCLADD = PCL(1:1)//'  '
```

```
            CARDAD = CHOLD(6:6)//'  '
         ELSE IF (LN .EQ. 2) THEN
            PCLADD = PCL(1:2)//' '
            CARDAD = CHOLD(6:7)//' '
         ELSE IF (LN .EQ. 3) THEN
            PCLADD = PCL(1:3)
            CARDAD = CHOLD(6:8)
         ELSE IF (LN .GT. 3) THEN
            PRINT *,' FATAL ERROR IN DFORM -- PCL ON EXEC CARD TOO LONG'
            STOP ' FATAL IN DFORM -- PCL ON EXEC'
         ENDIF
C*
C
C           CHECK FOR A TITLE CARD OR TERMINATOR CARD.
C
            IF (CHOLD(1:5)//CARDAD.EQ.'*/ **'//PCLADD) THEN
               ACTIVE = .TRUE.
               ABSTR = 0
               ENGR = BLANK4
               PRGMR = BLANK4
               PRINT '(''1'', 23X,''*** TRAC-BWR DEVELOPMENT FORM ***'')'
               PRINT '(''0'',T10,A,T66,A)',CHOLD(10:10),CHOLD(6:8)
               PRINT '(T5,A,T58,A)',DLINE(1:10),DLINE(1:20)
               PRINT '(T5,''SORT INDEX'',/T58,''PROGRAM CHANGE LABEL'')'
               PCLSAV = CHOLD(6:8)
               PRINT '(''0'',T5,A,/T5,A,/T5,''TITLE OF MODEL OR CHANGE'')',
         :          CHOLD(11:80),
         :            DLINE(1:73)
            ELSE
               IF (CHOLD(1:4).EQ.'*ID ' .OR.
         :         CHOLD(1:7).EQ.'*IDENT ' .OR.
         :         CHOLD(1:9).EQ.'*/ **END ') THEN
                  ACTIVE = .FALSE.
                  ABSTR = 0
               ENDIF
            ENDIF
C
         IF (ACTIVE) THEN
            EORP = .FALSE.
C                 THIS SECTION STORES THE ENGINEER AND PROGRAMMER NAMES
C              FOR PRINT OUT IN THE ABSTR = 1 SECTION.
C
               IF (ABSTR .EQ. 0 .OR. ABSTR .EQ. 1) THEN
               IF (CHOLD(1:5) .EQ. '*/ **') THEN
                  IF (CHOLD(6:9) .EQ. 'ENGR') THEN
                     ENGR = CHOLD(11:50)
                     ELSE IF (CHOLD(6:10) .EQ. 'PRGMR') THEN
                     PRGMR = CHOLD(11:50)
                     ELSE IF (CHOLD(6:10) .EQ. 'ENGPR') THEN
                     ENGR = CHOLD(11:50)
                     PRGMR = CHOLD(11:50)
                  ENDIF
               ENDIF
               ENDIF
C
C        CURRENTLY IN DOCUMENTATION PART OF UPDATES
C
```

Table C-2. (continued)

```
            IF (ABSTR .EQ. 0) THEN
C
C              MAIN PURPOSE OF THE ABSTR = 0 SECTION IS TO DETERMINE
C              WHEN THE ABSTRACT SECTION BEGINS, IN WHICH CASE
C              ABSTR WILL BE SET TO 1.
C
C                    THE FOLLOWING LOGIC SETS THE ABSTRACT FLAG ON
C                    (ABSTR = 1) WHEN BOTH ENGR AND PRGMR HAVE
C                    BEEN SET OR WHEN EITHER HAS BEEN SET AND
C                    A NON BLANK */ CARD IS ENCOUNTERED.
C
                     IF (ENGR .NE. BLANK4 .AND. PRGMR .NE. BLANK4) THEN
                        ABSTR = 1
                        EORP = .TRUE.
                     ELSE IF ((ENGR.NE.BLANK4 .OR. PRGMR.NE.BLANK4)
     :                        .AND. ((CHOLD .NE. CONLY) .AND.
     :                               (CHOLD(6:9).NE.'ENGR') .AND.
     :                               (CHOLD(6:10).NE.'PRGMR') .AND.
     :                               (CHOLD(6:10).NE.'ENGPR')))THEN
                        ABSTR = 1
                        EORP = .TRUE.
                     ENDIF
            ENDIF
                 IF (ABSTR .EQ. 1) THEN
                     PRINT '(''0'',T5,A,A,/T5,A31,T45,A32,/
     :                      T5,''RESPONSIBLE ENGINEER'', T45,
     :                      ''RESPONSIBLE PROGRAMMER'')',
     :                      ENGR,PRGMR,DLINE(1:31),DLINE(1:32)
                     PRINT '(''0'',T55,''YES     NO''/T5,A24,T45,A32/,
     :                     T5,A,T45,A)',
     :                     DLINE(1:30),DLINE(1:32),
     :                     'DEVELOPMENT BASE VERSION',
     :                     'MANUAL CHANGE REQUIRED  (CIRCLE)'
                     PRINT '(''0'',/T25,''** CONTENTS OF DEVELOPMENT'',
     :                     '' FILE **'')'
                     PRINT '(/,T35,''YES (X)   NO (X)'',T55,
     :                     ''COMPLETION DATE'')'
                     IF (AND .NE. 'NO') THEN
                     PRINT '(T5,''DEFICIENCY REPORT'',T35,A,
     :                      /T5,''REQUIREMENTS DOCUMENT'',T35,A,
     :                      /T5,''DESIGN REPORT'',T35,A,
     :                      /T5,''COMPLETION REPORT'',T35,A,
     :                      /T5,''ACCEPTANCE REPORT'',T35,A,
     :                      /T5,''UPDATE DOCUMENTATION'',
     :                       T35,''( X)'',T55,''(              )'',
     :                             T57,A,/T5,''OTHER:''/)',
     :                             PAREN,PAREN,PAREN,PAREN,PAREN,CDATE
                     ELSE
                     PRINT '(T5,''DEFICIENCY REPORT'',T35,A,
     :                      /T5,''REQUIREMENTS DOCUMENT'',T35,A,
     :                      /T5,''DESIGN REPORT'',T35,A,
     :                      /T5,''COMPLETION REPORT'',T35,A,
     :                      /T5,''ACCEPTANCE REPORT'',T35,A,
     :                      /T5,''UPDATE DOCUMENTATION'',
     :                       T35,''( X)'',T55,''(              )'',
     :                             T57,A,/T5,''OTHER:''/)',
     :                             PARNX,PARNX,PARNX,PARNX,PARNX,CDATE
```

Table C-2. (continued)

```
                  ENDIF
                  PRINT '(''0'',T5,''APPROVAL OF DEVELOPMENT PLAN:    '',
     :                  A,A,A,/
     :                  T37,''TRAC-BWR SECTION LEADER'',
     :                  T71,''DATE'')',
     :                  DLINE(1:23),BL(1:9),DLINE(1:10)
                  PRINT '(''0'',/32X,'' * ABSTRACT *''/)'
                  LIN = 33
               ENDIF
               IF (ABSTR .EQ. 1 .AND. EORP) THEN
                  ABSTR = 2
                  ELSE IF (ABSTR .GT. 0 .AND. .NOT. EORP) THEN
                     LIN = LIN + 1
                     IF (LIN .GE. 57) THEN
                        LIN = 1
                        PRINT '(''1'',15X,''** TRAC-BWR DEVELOPMENT'',
     :                        '' FORM -- CONTINUED **'',T65,
     :                        ''PCL = '',A //)', PCLSAV
                     ENDIF
                     PRINT '(T5,A)',CHOLD(3:80)
                     ABSTR = 2
               ENDIF
         ENDIF
  288 CONTINUE
  299 CONTINUE
      PRINT '(''1''/''T'')'
      STOP
      END
*/ ##ABC N TITLE 1 FOR PCL=ABC
*/ ##ENGPR G. L. SINGER
*/      THE ABSTRACT FOR PROGRAM CHANGE ABC SHOULD DESCRIBE
*/ THE REASON FOR THE UPDATE.
*/ ##INPUT
*/       PLACE THE DESCRIPTION OF ANY INPUT CHANGES
*/ NEEDED BY THIS UPDATE HERE.
*/ ##ENDIN
*/       OTHER DOCUMENTATION IS ALLOWED AFTER THE INPUT
*/ DESCRIPTION WHEN THE */ ##ENDIN CARD IS USED.
*/ ##END
*ID SUB1SCABC
*/ THE CARDS TO UPDATE THE PROGRAM FOR CHANGE ABC GO HERE.
*/
*/
*/ ##ACE T TITLE 2 FOR PCL=ACE
*/ ##ENGR G. L. SINGER
*/ ##PRGMR M. A. STONE
*/      THE ABSTRACT FOR PROGRAM CHANGE ACE SHOULD DESCRIBE
*/ THE REASON FOR THE UPDATE.
*ID SUB2--ACE
*/ THE CARDS TO UPDATE THE SUBROUTINE SUB2 FOR CHANGE ACE GO HERE.
*ID SUB3--ACE
*/ THE CARDS TO UPDATE THE SUBROUTINE SUB3 FOR CHANGE ACE GO HERE.
*/
*/
*/ ##DEF T TITLE 3 FOR PCL=DEF
*/ ##PRGMR G. L. SINGER
*/ ##ENGR  G.L.SINGER
```

Table C-2. (continued)

```
*/      THE ABSTRACT FOR PROGRAM CHANGE DEF SHOULD DESCRIBE
*/ THE REASON FOR THE UPDATE.
*ID SUB4--DEF
*/ LAST CARD
```

Table C-3. Sample DFORM output for a TRAC-BWR development form

```
                    *** TRAC-BWR DEVELOPMENT FORM ***

     P                                                        SH7
----------                                         --------------------
SORT INDEX
                                                   PROGRAM CHANGE LABEL

$MAO$ ELIMINATE UNUSED ARGUMENT IN CALL TO DPOWR FROM DMPIT
---------------------------------------------------------------------------
TITLE OF MODEL OR CHANGE

 G. L. SINGER                          G. L. SINGER
--------------------------------      ---------------------------------
RESPONSIBLE ENGINEER                  RESPONSIBLE PROGRAMMER

                                                YES    NO
------------------------              ---------------------------------
DEVELOPMENT BASE VERSION              MANUAL CHANGE REQUIRED  (CIRCLE)

                    ** CONTENTS OF DEVELOPMENT FILE **

                            YES (X)    NO (X)     COMPLETION DATE
DEFICIENCY REPORT            (  )      (  )       (               )
REQUIREMENTS DOCUMENT        (  )      (  )       (               )
DESIGN REPORT                (  )      (  )       (               )
COMPLETION REPORT            (  )      (  )       (               )
ACCEPTANCE REPORT            (  )      (  )       (               )
UPDATE DOCUMENTATION         ( X)                 (   02/23/83    )
OTHER:

APPROVAL OF DEVELOPMENT PLAN:   ----------------------        ----------
                                TRAC-BWR SECTION LEADER          DATE

                              * ABSTRACT *

       PROGRAM DMPIT CALLS SUBROUTINE DPOWR WITH AN ARGUMENT, BUT
 SUBROUTINE DPOWR DOES NOT HAVE AN ARGUMENT LIST.  THIS SHOULD BE
 A HARMLESS ERROR, BUT IS CORRECTED HERE FOR CLEAN UP PURPOSES.
```

APPENDIX D
SUGGESTED FILING PROCEDURES

APPENDIX D SUGGESTED FILING PROCEDURES

This appendix provides a method of handling and filing the forms and other documentation for a code development effort.

The development forms should be prepared by the individual contributors to the code and given to the Section Leader for his information, review, and approval. The Section Leader should pass these forms (or copies thereof) on to the Code Librarian as they are approved. The Code Librarian accumulates these forms, assures that the corresponding update cards are incorporated into a candidate version of the code, when appropriate, and returns a complete set of forms and internal documentation for an apparently acceptable version to the Section Leader for approval.

The acceptance reports should be addressed to the Section Leader but given first to the Code Librarian. The librarian accumulates and files them for presentation to the Section Leader prior to the acceptance of an official version.

The following files should be maintained to back up and document a computer code:

1. Computer Files:

   In addition to permanent files needed for current use of the computer code, the following items should be maintained on tape (or equivalent) backup: the source program as a CDC Update Program Library, object code, absolute code (optional), and an accumulation of the update cards used to create each previous version of the code.

   This method allows complete reconstruction of any version of the code by simply retaining an original and final code tape. Reconstruction may be accomplished from the original version by performing the successive updates contained on the final tape or by yanking correction sets from the final program library. In addition, the accumulated updates may be processed by CDCID in order to provide a variety of types of selected documentation.

   Another desirable item to accumulate on the code tape is a list of title cards. This allows a complete set of titles of the changes to all previous versions to be listed both chronologically and alphabetically each time a new version is created.

2. Microfiche or Computer Paper Output:

   The hard copy output of all computer runs used to generate new code versions should be saved in order to show card insertions and deletions, code compilations, and load maps. The output of computer runs for all test problem cases should also be saved. Questions concerning the insertion of previous update and test case results frequently occur during subsequent code development. Rapid access to this information proves invaluable in code debugging and development.

3. Notebook for Each Code Version:

   A notebook should be maintained for each official code version and should consist of the following items as applicable. Items a through f should be filed by program change label (PCL). Items g through j should be filed by update.

   a. Code development forms

   b. Formatted documentation of code update listings down to the *IDENT cards in level of detail

   c. Acceptance reports

   d. Deficiency reports

   e. Design reports

   f. Completion reports

   g. List of changes for update merge

   h. List of notes on the update

   i. Code change forms

   j. Plot output.

APPENDIX E
SOURCE LISTING OF PROGRAM CDCID

APPENDIX E
SOURCE LISTING OF PROGRAM CDCID

A source listing in the form of the COMPILE file output from a CDC UPDATE run is provided in Table E-1, which is included on microfiche on the inside of this report's back cover. The source listing contains a structured flow chart of the CDCID program.

APPENDIX F
CDCID SAMPLE PROBLEM

APPENDIX F
CDCID SAMPLE PROBLEM

The following sample problem output shows the contents of various files associated with a CDCID run. Notice that the input file, UPDIN, and the modified input file, UPDMOD, are identical except for the insertion of the sequence number specified on the */Δ##IDENT12 card into the *ID, *CALL, and *COMDECK cards. Note that the sequence number was not added to the *DK cards due to the blanks in the sequence number position on the */Δ##NDECK card. Also note that the sequence number insertion took place for the *CALL card with $$ in columns 5 and 6 of the identifier, but not for the other *CALL card.

The OUTPUT file should be checked for the occurrence of any nonzero values following SUMMARY ERROR COUNT, which would indicate errors or the existence of cards that appear to be intended as CDCID program cards but are not properly formatted. The change count column opposite the old and new identifier columns simply tells how many modifications were made to the old card identifier.

The SELECT file contains a list of the title cards. Examples of the control cards needed to execute CDCID for two different purposes are also included.

```
****************************************
*                                      *
* LISTING OF FILE UPDIN BEFORE CDCID RUN *
*                                      *
****************************************
```

```
*/
*/ NOTE 1: THE FOLLOWING CARDS DETERMINE THE TYPE OF OUTPUT
*/          PRODUCED.  THE TYPICAL USER MIGHT USE ONLY THE */ ##IDENT
*/          CARD TO INSERT THE SEQUENCE NUMBER.  OR, HE MIGHT WISH
*/          TO USE ALL OF THESE INPUT REQUESTS IN ORDER TO PRODUCE
*/          A DOCUMENTATION FILE FOR THE CHANGES.
*/ ##FORM
*/ ##EDIT EVERY PCL
*/ ##EDSAL
*/ ##IDENT12
*/ ##NDECK
*/ ##SA1 P   SAMPLE TITLE CARD FOR CHANGE WITH PCL=SA1
*/ ##ENGPR G. L. SINGER
*/ */
*/ DOCUMENTATION FOR CHANGE SA1 GOES HERE.  THE 'P' IN COLUMN 10
*/ OF THE TITLE CARD IS A SORT INDEX.
*/ ##INPUT
*/          DESCRIBE ANY INPUT CHANGES CAUSED BY UPDATE SA1 HERE.
*/ ##ENDIN
*/ OTHER COMMENTS MAY BE CONTINUED HERE.
*/
*/ NOTE 2:  THE FOLLOWING CARDS SHOW HOW SOME OF THE CDC UPDATE CARDS
*/           WHICH MAY BE MODIFIED BY CDCID ARE PROCESSED.
*/
*ID SUBX--SA1
*I SUBX.5
        INSERT CARDS AFTER EXISTING CARD.
*IDENT,SUBA--SA1
*D SUBA.8
        INSERT CARDS AFTER DELETING CARD.
*AF ,PREVDEK
*COMDECK CDEKXXSA1
        INSERT COMMON DECK CARDS.
*/
*/ ##SB3 T   SECOND SAMPLE TITLE CARD
*/ ##ENGR   G. L. SINGER
*/ ##PRGMR G. L. SINGER
*/          INSERT ABSTRACT HERE
*/ ##END   SB3
*AF ,PREVDEK
*DK NEWDECSB3
       SUBROUTINE NEWDEC
       NOTE THAT DUE TO THE '*/ NDECK*4' CARD, NO SEQUENCE NUMBER
       IS INSERTED IN THE DECK IDENTIFIER.
*CALL,CDEK$$SB3
        NOTE THAT ONLY THE CALL TO THE COMDECK WHICH HAS
        $$ FOR CHARACTERS 5 AND 6 GETS A SEQUENCE NUMBER INSERTED.
*CALL,CDKOLDSA1
*/
*/ ##SA5 P FINAL SAMPLE TITLE CARD
*/ ##ENGPR G. L. SINGER
*/          INSERT ABSTRACT HERE
*/ ##END  SA5
*ID DEND--SA5
        THIS IS THE END OF THE SAMPLE PROBLEM INPUT DECK
```

```
*****************************************
*                                       *
* LISTING OF FILE OUTPUT AFTER CDCID RUN *
*                                       *
*****************************************
```

```
     CDCID/MOD 1, UPDATE 12, 03/21/83       G.L.SINGER
A COMPUTER PROGRAM FOR DOCUMENTATION OF CDC UPDATE CHANGES.  SEC. 1

     NEW IDENTIFIER CARD                         OLD IDENTIFIER CARD  CHANGE COUNT

     *ID SUBX12SA1                               *ID SUBX--SA1            1
     *IDENT SUBA12SA1                            *IDENT SUBA--SA1         1
     *COMDECK CDEK12SA1                          *COMDECK CDEKXXSA1       1
     *CALL CDEK12SB3                             *CALL CDEK$$SB3          1
     *ID DEND12SA5                               *ID DEND--SA5            1
```

```
     CDCID/MOD 1, UPDATE 12, 03/21/83       G.L.SINGER
A COMPUTER PROGRAM FOR DOCUMENTATION OF CDC UPDATE CHANGES.  SEC. 2

                                     PROGRAM CHANGE LABEL TITLE INDEX

     */ **SA1 P  SAMPLE TITLE CARD FOR CHANGE WITH PCL=SA1
     */ **SB3 T  SECOND SAMPLE TITLE CARD
     */ **SA5 P FINAL SAMPLE TITLE CARD
```

```
     CDCID/MOD 1, UPDATE 12, 03/21/83       G.L.SINGER
A COMPUTER PROGRAM FOR DOCUMENTATION OF CDC UPDATE CHANGES.  SEC. 3

     *** LIST OF CDCID INPUT DIRECTIVES WHICH AFFECT PROGRAM CONTROL

     INDEX           CDCID INPUT DIRECTIVES/
         1            */ **FORM
         2            */ **EDIT EVERY PCL
         3            */ **SDSAL
         4            */ **IDENT12
         5            */ **NDECK
     CDCID/MOD 1, UPDATE 12, 03/21/83         G.L.SINGER
A COMPUTER PROGRAM FOR DOCUMENTATION OF CDC UPDATE CHANGES.  SEC. 4

     EDIT REQUEST:   EVERY PCL             REQUEST LEVEL=  TITLE


     PCL: *** SA1 ***   SORT INDEX: P   UPDATE NUMBER: 12

TITLE:
          SAMPLE TITLE CARD FOR CHANGE WITH PCL=SA1

     PCL: *** SB3 ***   SORT INDEX: T   UPDATE NUMBER: 12

TITLE:
          SECOND SAMPLE TITLE CARD

     PCL: *** SA5 ***   SORT INDEX: P   UPDATE NUMBER: 12

TITLE:
         FINAL SAMPLE TITLE CARD




     CDCID/MOD 1, UPDATE 12, 03/21/83         G.L.SINGER
A COMPUTER PROGRAM FOR DOCUMENTATION OF CDC UPDATE CHANGES.  SEC. 5

                                    *** SUMMARY CARD COUNT ***

INPUT CARDS       IDENTIFIER CREATION       SORT-EDIT
                        CARDS               REQUESTS

        55                    5                    5



                                    ***SUMMARY ERROR COUNT ***

FATAL ERROR        WARNING         CARDS NEEDING REVIEW
MESSAGES           MESSAGES        FOR POSSIBLE ERRORS

          0              0                        0
```

```
*****************************************
*                                       *
* LISTING OF FILE UPDMOD AFTER COCID RUN *
*                                       *
*****************************************
```

```
*/
*/ NOTE 1: THE FOLLOWING CARDS DETERMINE THE TYPE OF OUTPUT
*/          PRODUCED.  THE TYPICAL USER MIGHT USE ONLY THE */ **IDENT
*/          CARD TO INSERT THE SEQUENCE NUMBER.  OR, HE MIGHT WISH
*/          TO USE ALL OF THESE INPUT REQUESTS IN ORDER TO PRODUCE
*/          A DOCUMENTATION FILE FOR THE CHANGES.
*/ **FORM
*/ **EDIT EVERY PCL
*/ **EDSAL
*/ **IDENT12
*/ **NDECK
*/ **SA1 P  SAMPLE TITLE CARD FOR CHANGE WITH PCL=SA1
*/ **ENGPR G. L. SINGER
*/ */
*/ DOCUMENTATION FOR CHANGE SA1 GOES HERE.  THE 'P' IN COLUMN 10
*/ OF THE TITLE CARD IS A SORT INDEX.
*/ **INPUT
*/         DESCRIBE ANY INPUT CHANGES CAUSED BY UPDATE SA1 HERE.
*/ **ENDIN
*/ OTHER COMMENTS MAY BE CONTINUED HERE.
*/
*/ NOTE 2:  THE FOLLOWING CARDS SHOW HOW SOME OF THE CDC UPDATE CARDS
*/           WHICH MAY BE MODIFIED BY COCID ARE PROCESSED.
*/
*ID SUBX12SA1
*I SUBX.5
        INSERT CARDS AFTER EXISTING CARD.
*IDENT SUBA12SA1
*D SUBA.3
        INSERT CARDS AFTER DELETING CARD.
*AF ,PREVDEK
*COMDECK CDEK12SA1
        INSERT COMMON DECK CARDS.
*/
*/ **SB3 T  SECOND SAMPLE TITLE CARD
*/ **ENGR  G. L. SINGER
*/ **PRGMR G. L. SINGER
*/         INSERT ABSTRACT HERE
*/ **END  SB3
*AF ,PREVDEK
*DK NEWDECSB3
        SUBROUTINE NEWDEC
       NOTE THAT DUE TO THE '*/ NDECK**' CARD, NO SEQUENCE NUMBER
       IS INSERTED IN THE DECK IDENTIFIER.
*CALL CDEK12SB3
        NOTE THAT ONLY THE CALL TO THE COMDECK WHICH HAS
        $$ FOR CHARACTERS 5 AND 6 GETS A SEQUENCE NUMBER INSERTED.
*CALL CDKOLDSA1
*/
*/ **SA5 P FINAL SAMPLE TITLE CARD
*/ **ENGPR G. L. SINGER
*/         INSERT ABSTRACT HERE
*/ **END  SA5
*ID DEND12SA5
        THIS IS THE END OF THE SAMPLE PROBLEM INPUT DECK
```

```
*******************************************
*                                         *
* LISTING OF FILE SELECT AFTER CDC10 RUN *
*                                         *
*******************************************
```

```
*/ **SA1 P  SAMPLE TITLE CARD FOR CHANGE WITH PCL*SA1
*/ **SB3 T  SECOND SAMPLE TITLE CARD
*/ **SA5 P FINAL SAMPLE TITLE CARD
```