General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)
CONFIGURATION ANALYSIS TOOL (CAT)
SYSTEM DESCRIPTION AND USER'S GUIDE (REVISION 1)

DECEMBER 1982
FOREWARD

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration Goddard Space Flight Center (NASA/GSFC) and created for the purpose of investigating the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1977 and has three primary organizational members:

NASA/GSFC (Systems Development and Analysis Branch)
The University of Maryland (Computer Sciences Department)
Computer Sciences Corporation (Flight Systems Operation)

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effect of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document. A version of this document was also issued as Computer Sciences Corporation document CSC/SD-82/6125.

Contributors to this document include

William Decker (Computer Sciences Corporation)
Wayne Taylor (Computer Sciences Corporation)

Other contributors include

Frank McGarry (Goddard Space Flight Center)
Phil Merwarth (Goddard Space Flight Center)

Single copies of this document can be obtained by writing to

Frank E. McGarry
Code 582.1
NASA/GSFC
Greenbelt, Maryland 20771
ABSTRACT

This document presents a system description of, and user's guide for, the Configuration Analysis Tool (CAT). As a configuration management tool, CAT enhances the control of large software systems by providing a repository for information describing the current status of a project. CAT provides an editing capability to update the information and a reporting capability to present the information. CAT is an interactive program available in versions for the PDP-11/70 and VAX-11/780 computers.
TABLE OF CONTENTS

Section 1 - Introduction ...................................... 1-1
Section 2 - CAT System Description ............................... 2-1
   2.1 Introduction ............................................. 2-1
   2.2 CAT Processing ........................................... 2-3
      2.2.1 Executive ........................................... 2-3
      2.2.2 Editor .............................................. 2-3
      2.2.3 Report Generator .................................... 2-3
   2.3 CAT File Structure and Usage ............................... 2-10
      2.3.1 Project File Structure .................. 2-10
      2.3.2 Edit Scratch Files .......................... 2-12
Section 3 - CAT User's Guide .................................... 3-1
   3.1 Introduction ............................................. 3-1
   3.2 CAT Initialization ....................................... 3-4
   3.3 Use of CAT Editor ...................................... 3-6
   3.4 Use of CAT Report Generator ................... 3-10
   3.5 Summary ............................................... 3-13
Appendix A - CAT Subroutine Descriptions
Appendix B - CAT Common Block Information
Appendix C - CAT Project File Record Descriptions
Appendix D - CAT Error Messages
Appendix E - System Generation
Bibliography
References

PRECEDING PAGE BLANK NOT FILMED
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>CAT Data Flow.</td>
<td>2-2</td>
</tr>
<tr>
<td>2-2</td>
<td>CAT Program Structure.</td>
<td>2-4</td>
</tr>
<tr>
<td>2-3</td>
<td>CAT Executive Baseline.</td>
<td>2-5</td>
</tr>
<tr>
<td>2-4</td>
<td>CAT Editor Baseline.</td>
<td>2-7</td>
</tr>
<tr>
<td>2-5</td>
<td>CAT Report Generator Baseline.</td>
<td>2-9</td>
</tr>
<tr>
<td>2-6</td>
<td>Record Linkage in the Project File</td>
<td>2-11</td>
</tr>
<tr>
<td>2-7</td>
<td>Edit Scratch Files After Inserting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a New Block of Records Following the First Block and Then Positioning the File Following the Original Second Block</td>
<td>2-16</td>
</tr>
<tr>
<td>2-8</td>
<td>Edit Scratch File After Backing Up in the File to the End of the First Block.</td>
<td>2-17</td>
</tr>
<tr>
<td>3-1</td>
<td>CAT Menu and Prompt Hierarchy.</td>
<td>3-14</td>
</tr>
<tr>
<td>3-2</td>
<td>Starting CAT and Creating a New Data Base File</td>
<td>3-16</td>
</tr>
<tr>
<td>3-3</td>
<td>Creating a New Discrepancy and Change Subfile</td>
<td>3-17</td>
</tr>
<tr>
<td>3-4</td>
<td>Returning To Edit the Discrepancy and Change Subfile</td>
<td>3-18</td>
</tr>
<tr>
<td>3-5</td>
<td>Creating New Milestone/Deliverable and Test History Subfiles.</td>
<td>3-19</td>
</tr>
<tr>
<td>3-6</td>
<td>Proceeding from the Editor to the Report Generator</td>
<td>3-20</td>
</tr>
<tr>
<td>3-7</td>
<td>CRT Format of the Discrepancy and Change Report When 'ALL' Items are Selected</td>
<td>3-22</td>
</tr>
<tr>
<td>3-8</td>
<td>CRT Format of the Discrepancy and Change Report When 'CHANGE' Items are Selected</td>
<td>3-23</td>
</tr>
<tr>
<td>3-9</td>
<td>CRT Format of the Discrepancy and Change Report When 'DISCREPANCY' Items are Selected</td>
<td>3-24</td>
</tr>
<tr>
<td>3-10</td>
<td>CRT Format of the Discrepancy and Change Report When 'UNFINISHED' Items are Selected</td>
<td>3-25</td>
</tr>
<tr>
<td>3-11</td>
<td>Selecting the Milestone/Deliverable and Test History Reports</td>
<td>3-26</td>
</tr>
<tr>
<td>3-12</td>
<td>CRT Format of the Milestone/Deliverable Report</td>
<td>3-27</td>
</tr>
<tr>
<td>3-13</td>
<td>CRT Format of the Test History Report</td>
<td>3-28</td>
</tr>
<tr>
<td>3-14</td>
<td>Changing the Report Output Device to the Lineprinter</td>
<td>3-29</td>
</tr>
<tr>
<td>3-15</td>
<td>Requesting All Reports While the Output Device is the Lineprinter</td>
<td>3-30</td>
</tr>
</tbody>
</table>
LIST OF ILLUSTRATIONS (Cont'd)

Figure

3-16   Printer Format of the Discrepancy and Change Report When 'ALL' Items are Selected ............. 3-31
3-17   Printer Format of the Milestone/Deliverable Report ................................................. 3-32
3-18   Printer Format of the Test History Report ................................................................. 3-33
3-19   Terminating CAT From the Report Generator ............................................................... 3-34

LIST OF TABLES

Table

2-1   Edit Scratch File Record Type Flags ................. 2-13
3-1   CAT Report Purpose ..................................... 3-2
3-2   CAT Report Data Description ......................... 3-3
3-3   CAT Editor Functions ................................... 3-8
SECTION 1 - INTRODUCTION

The Configuration Analysis Tool (CAT) is an information storage and report generation tool for support of configuration management activities. Configuration management is a discipline composed of many techniques selected to track and direct the evolution of complex systems. Most configuration management activities can be categorized as follows:

- **Information gathering** -- The current state of a system must be known in sufficient detail to support activities in the two categories specified below. The required information must be descriptive, current, and accurate.

- **Configuration analysis** -- The interrelationships among various system components must be described in a manner that permits the impact of configuration changes to be predicted for each system component.

- **Resource allocation** -- In response to the configuration analysis, priorities and schedules must be established that minimize the impact of system modifications and make effective use of all available resources.

CAT provides facilities to aid configuration management activities in the first two categories specified above.

Section 2 contains a detailed system description of CAT. Section 3 provides a user's guide to the program.

Appendixes A through E provide program subroutine descriptions; COMMON block information; file structure information; error messages; and system generation, overlay, and task building information, respectively.

1-1
2.1 INTRODUCTION

CAT is an interactive program designed to accept, organize, and store information describing the status of a project. The information, which is concerned with the design, implementation, testing, and maintenance of a project's software, is resident on a direct-access storage device. The information can be updated with the CAT editor subsystem or reported by the CAT report generator subsystem.

CAT is available in versions to run on either a PDP-11/70 or on a VAX-11/780. There are no source code differences between the PDP and VAX versions. The descriptions of CAT subroutines, COMMON blocks, and file structure, which appear in Appendixes A, B, and C, respectively, apply to both versions of the program. The system generation procedures are different for each version (PDP and VAX) and are presented in Appendix E.

The overall flow of information to and from CAT is shown in Figure 2-1. The user controls all CAT processing via CRT input. Most processing instructions are communicated by selecting an option from a menu display presented at the terminal. CAT reads data from, and writes data to, the project file in response to the user's requests for processing. When the user selects the printer as the output devices for CAT reports, CAT writes the reports to a printer file (CATLST.LST) in the user's directory. This file may be spooled to the system printer via a system utility after the session with CAT is terminated.

The following subsections describe the software that performs the CAT processing and the files used by CAT.
Figure 2-1. CAT Data Flow
2.2 CAT PROCESSING

As shown in Figure 2-2, CAT consists of an executive routine, CAT, and two subsystems: the report generator and the editor. The following subsections discuss the processing performed by these three components of CAT.

2.2.1 EXECUTIVE

The CAT executive routine performs initialization and termination chores, controls the user's selection of a project file, and presents the user with the choice of obtaining reports or editing the project file. The baseline diagram for the executive is shown in Figure 2-3. (The next level of called routines is shown in later figures.)

The executive prompts the user for a project name. If the user enters a null line (carriage return only), the executive closes all files that are open and terminates the session. When a project name is entered, the executive proceeds to prompt the user to select either the report generator or the editor. If the report generator is selected, the executive attempts to open the indicated project file with a status of OLD. If the file does not exist, the user is prompted for another project name. When the editor is selected, the file is opened with a status of OLD if it exists. If the file does not exist, the executive verifies the project name with the user and opens the project file with a status of NEW.

The executive passes control to routine EDTSEL if the editor is selected and to routine REPORT if the report generator is selected.

2.2.2 EDITOR

The CAT editor subsystem controls the user's selection of a subfile from the project file. After the user has selected
Figure 2-2. CAT Program Structure
Figure 2-3. CAT Executive Baseline
When the user selects a subfile that is currently empty, the editor subsystem passes control to routine CATPUT. CATPUT prompts the user for data to be stored in each record to be entered into the initial version of the subfile. As long as the user enters a value for the first field in the record, CATPUT prompts for the succeeding fields. A record is detached from the list of available records (see Section 2.3.1), the user's data is copied into the record, and the record is linked to the end of the current subfile. When the user enters a null line in response to a prompt for the first data field in a record, CATPUT terminates construction of the subfile and returns control to the executive routine for the editor subsystem (EDTSEL).

If the user selects a subfile that contains data the editor subsystem passes control to routine EDITOR. EDITOR controls the disposition of the scratch files used during an edit session with CAT. The scratch files are opened with routine OPNX and the contents of the selected subfile are copied into the primary edit file (see Section 2.3.2) by routine CATFIL. The edit command processing routine, EDIT, is given control until the user uses a "KILL" or "EXIT" command to terminate the edit session. The user's requested modifications to the subfile are made using the scratch files. If the user terminates the edit session with a KILL command, all scratch files are deleted and control is returned to the EDTSEL routine.

When the user terminates an edit session with the EXIT command, routine REDCAT is used to recreate the edited subfile using the data from the scratch files. Any data remaining in either the primary edit file or the backwards edit file...
(see Section 2.1.2) has been placed in the forwards file at the end of the edit session. The forwards file is read from start to finish by REDCAT and each block of sequential records corresponding to a subfile record is examined. When the block of records has been flagged for deletion, the corresponding record in the subfile is detached from the subfile (by adjusting the pointers on the previous and following records in the subfile) and linked at the front of the list of available records. All other blocks of records are returned either to their originating subfile records or to the subfile records assigned and linked in the editor insert operation.

Within the editor command processing routine, each user command is read and parsed by routine GETLIN. Once recognized, each command is executed by the appropriate editor routine (ADD, ADDP, BOT, CATINS, CHG, DEL, LOC, NEX, NEXP, PRI, and TOP). Each editor routine uses the MOVE or MOVLIN utility routine to obtain the target record from the appropriate edit scratch file and to replace any modified record.

2.2.3 REPORT GENERATOR

The CAT report generator subsystem controls the user's selection of report type, report content, and output device. The baseline diagram for the entire report generator subsystem is shown in Figure 2-5. The report generator executive routine, REPORT, obtains the user's choice of output device (terminal or printer) and opens the printer file (CATLST.LST) if it has not been previously used.

The report type selection routine, REPSEL, is called to obtain the user's choice of report type. If the subfile containing the user's requested data type exists, the appropriate report generation routine is called. The routines that
produce the Discrepancy and Change History report (CRTDIS and PRTDIS) use routine PIKONE to identify the subset of Discrepancy and Change History data selected by the user.

2.3 **CAT FILE STRUCTURE AND USAGE**

All data required by CAT for a specified project reside in a single project file. The structure of this file is described in Section 2.3.1 below. When the CAT editor is used to modify an existing subfile in the project file, three edit scratch files are used while manipulating the data. The edit scratch files are described in Section 2.3.2.

2.3.1 **PROJECT FILE STRUCTURE**

CAT project files are direct access files with fixed-length (176 byte) records. Each project file may contain up to five types of records as described in Appendix C. Figure 2-6 shows how the records in a project file are organized.

A newly created project file contains a header record and a list of available records. The header record is record number one in the direct access file and contains pointers to all other linked lists of records in the file. In a new project file, all pointers, except the pointer to the list of available records, are initialized to zero since no subfiles exist.

The list of available records is a singly linked list with forward pointers. A pointer (AVAIL) to the first record in this list (the "front" of the list) is kept in the header record. The last record in the list of available records contains a null forward pointer. The record at the front of the list is detached and added to a subfile whenever a new record is required. A record deleted from a subfile is returned to the front of the list of available records. Subroutine GETMOR (see Appendix A) is used to extend the list of available records whenever it is emptied.
Figure 2-6. Record Linkage in the Project File
The three subfiles are made up of the records that contain data. Subfiles 1, 2, and 3 contain Discrepancy and Chance History data, Milestone/Deliverable History data, and Test History data, respectively. Each subfile is a doubly linked list. The header record contains a pair of pointers for each subfile. If the subfile contains one or more records, the pointers in the header record indicate the first and last records in the subfile. The first record in a subfile contains a null backward pointer and the last record in a subfile contains a null forward pointer as shown in Figure 2-6. Records are added to or deleted from a subfile by adjusting the pointers on the previous and following records.

2.3.2 EDIT SCRATCH FILES

The CAT editor subsystem creates a set of three scratch files for use in manipulating the data in an existing subfile (see Section 2.2.2). The three files are the primary file, the forwards file, and the backwards file. Each file may contain records organized into record blocks.

A record block consists of a "breakpoint record" followed by data records. The breakpoint record contains the record number of the project file record that originated the data contained in the data records. The number of data records in a record block depends upon the number of data fields in the subfile used to create the edit scratch files. A data record is written for each data field in a subfile record. The data record contains a descriptive label (from COMMON block CATCOM) followed by the contents of a data field from the subfile record.

Every record in the edit scratch files contains a record type flag in the first five bytes of the record. Table 2-1 shows the type flags and their meaning.
Table 2-1. Edit Scratch File Record Type Flags

<table>
<thead>
<tr>
<th>Flag Value</th>
<th>Record Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;000-3&quot;</td>
<td>End of file record</td>
</tr>
<tr>
<td>&quot;000-2&quot;</td>
<td>Breakpoint record</td>
</tr>
<tr>
<td>&quot;000-1&quot;</td>
<td>Beginning of file record</td>
</tr>
<tr>
<td>&quot;000000&quot;</td>
<td>Deleted record (all records in a block, including the breakpoint record)</td>
</tr>
<tr>
<td>&quot;00001&quot;</td>
<td>Data record</td>
</tr>
</tbody>
</table>
The three edit scratch files are direct access files and contain formatted records 88 bytes in length. The associated variable for each file is used to point to the next record to be read or written.

The primary file is created at the start of an edit session once the subfile type to be edited is selected. Each record in the subfile is read and a breakpoint record and a group of data records are created and written to the primary file. A record with an end of file record type flag is written to the primary file after the last subfile record is processed. After the primary file is created, its associated variable, LFILEO, is set to point to the first record in the file. From this point on, the records in the primary file are always read in sequential order.

The forwards file and the backwards file are filled with records as the user manipulates what appears to be a single file. The user examines and changes this imaginary file by moving a "current line pointer." As the user moves the current line pointer down the edit file, records are moved from the primary file to the forwards file as the pointer passes them. If the user moves the current line pointer up in the edit file, records are moved in reverse order from the forwards file to the backwards file. If the user again changes direction and moves the current line pointer down the file, records are moved from the backwards file in reverse order to the forwards file until the backwards file is emptied at which point records are again moved from the primary file to the forwards file.

The associated variables for the primary, forwards, and backwards files (LFILEO, LFORWO, and LBKWO, respectively) always point to the next sequential record in each file.
The "current line" pointed to by the current line pointer is the last valid record in the forwards file (record number LFORWO - 1). This record is modified by the change and add commands. Record blocks inserted by the user are added to the end of the forwards file. Records in record blocks that have been deleted are marked with the delete record type flag (Table 2-1) and become unavailable to the user although they continue to be moved between the forwards and backwards file.

Figures 2-7 and 2-8 illustrate a simple example in which the user edits a subfile containing five records. Figure 2-7 shows the primary file containing the original five record blocks and the forwards file containing a new record block inserted by the user following the first block. The current line in this figure is the last record in the original second block. Figure 2-8 shows the contents of these files if the user positions the current line pointer to the last record in the first record block. Note that the records forming the inserted block and the original second block have been moved to the backwards file and are in reverse order.

If the user ends a session with the EXIT command, the current line pointer is moved automatically to the end of the file; that is, any records in the backwards file are moved to the forwards file and then any records remaining in the primary file are moved to the forwards file. The records in the forwards file are then used to reconstruct the subfile.
Figure 2-7. Edit Scratch Files After Inserting a New Block of Records Following the First Block and Then Positioning the File Following the Original Second Block
Figure 2-8. Edit Scratch File After Backing Up in the File to the End of the First Block
SECTION 3 - CAT USER'S GUIDE

3.1 INTRODUCTION

CAT is an interactive software management reporting program implemented on the PDP-11/70 computer under the RSX-11M operating system and on the VAX-11/780 computer under the VMS operating system. The minimum operating configuration is a terminal and a lineprinter. The terminal acts as both the input and the output device when the user is interacting with the program. The processed output is in the form of project reports, which may be directed to either the user terminal or a disk file for listing on the line printer.

CAT stores information in a separate file for each project. Each file contains three subfiles. These subfiles contain the data for the reports generated by CAT (see Tables 3-1 and 3-2).

The following sections provide input formats and user instructions for running CAT. Sections 3.2, 3.3, and 3.4 specify the procedures for program initialization, use of the CAT editor, and use of the CAT report generator, respectively. Section 3.5 provides a summary of CAT user information.

CAT informs the user of abnormal conditions during execution by writing messages to the user's terminal. Appendix D lists each message and an explanation of the probable cause of the message.
Table 3-1. CAT Report Purpose

<table>
<thead>
<tr>
<th>Report Name</th>
<th>Content Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrepancy and Change History</td>
<td>Describes errors (discrepancies) and changes for a project</td>
</tr>
<tr>
<td>Milestone/Deliverable History</td>
<td>Describes milestone event and deliverable schedules</td>
</tr>
<tr>
<td>Test History</td>
<td>Describes tests and test results</td>
</tr>
</tbody>
</table>

9023
### Table 3-2. CAT Report Data Description

<table>
<thead>
<tr>
<th>Discrepancy and Change History</th>
<th>Milestone/Deliverable History</th>
<th>Test History</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date discrepancy or change defined</td>
<td>Date Milestone/Deliverable defined</td>
<td>Date tested</td>
</tr>
<tr>
<td>Date discrepancy or change corrected</td>
<td>Release scheduled date</td>
<td>Programmer assigned</td>
</tr>
<tr>
<td>Actual date released</td>
<td>Actual date released</td>
<td>Description of test performed</td>
</tr>
<tr>
<td>Programmer assigned</td>
<td>Programmer assigned</td>
<td>Test results</td>
</tr>
<tr>
<td>Code denoting discrepancy (D) or change (C)</td>
<td>Type of deliverable</td>
<td></td>
</tr>
<tr>
<td>Description of discrepancy of change</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changes made to the system for correction</td>
<td>Milestone event description</td>
<td></td>
</tr>
</tbody>
</table>

---

9023
1.2 CAT INITIALIZATION

After logging on to the PDP-11.70 computer, the user initiates a CAT session with the following command:

`RUN DB1:(213,2)CAT`

On the VAX-11.780, CAT is started with the following command:

`$ RUN DBB1:([TOOLS]CAT`

There are no prerun allocations to be made since CAT performs these internally as needed. The first prompt issued by CAT requests a project name:

`ENTER PROJECT NAME`

The project name is a unique character string (of up to six characters) that identifies the particular CAT project file. Following the entry of a project name, CAT displays the following menu (OPTION MENU):

```
OPTION MENU
1 EDITOR
2 REPORTS
```

Menus are used throughout CAT to provide a quick and easy way for the user to indicate decisions. Each item on a given menu is preceded by an integer number; to select one of these items, the user types in the appropriate integer and presses the carriage return. If the user enters a null line (carriage return only), CAT displays the menu that preceded the current menu.

Entering anything other than a displayed selection number or a null line (carriage return only) results in the following message:

```
**** ___ IS AN INVALID OPTION
```
where the number at the start of the message is the user's invalid input.

If the user enters a carriage return in response to the ENTER PROJECT NAME prompt, CAT terminates with the following message:

**********CAT TERMINATED**********

and the user is returned to the operating system control.

The correct way to terminate CAT is to return to this prompt and press the carriage return, thus allowing CAT to "clean up" correctly. Terminating in any other way (e.g., CTRL/C) produces unpredictable results.

Menus in CAT are arranged in a hierarchy, or tree. The OPTION MENU is at the top of the tree. In this and the following sections, the level of the menu in the hierarchy is indicated by a circled number to the right of each sample shown.

The following instructions pertain to the OPTION MENU, which is repeated below.

```
OPTION MENU
1  EDITOR
2  REPORTS
```

To edit the project data base, the user enters a 1. Section 3.3 provides a guide for use of the CAT editor. To obtain history reports from the project data base, the user enters a 2. Section 3.4 provides a guide for use of the CAT report generator.
3.3 USE OF CAT EDITOR

The CAT editor is invoked with the following response to the OPTION MENU:

```
OPTION MENU
1  EDITOR
2  REPORTS
CAT>1
```

If the user has entered a new project name (pname), CAT will respond with

```
NO SUCH FILE
PROGRAM CREATING A NEW FILE FOR -- pname.DAT
TO CONTINUE, ENTER Y FOR YES OR N FOR NO>
```

If the answer to this prompt is N (no), CAT prompts again with

```
ENTER PROJECT NAME>
```

If the answer is Y (yes), CAT creates the new project file.

Following the entry of an existing project name or the creation of the new project file, CAT displays the following menu:

```
EDITOR - SELECT A DATA TYPE
1  DISCREPANCY & CHANGE DATA
2  MILESTONE/DELIVERABLE DATA
3  TEST HISTORY
CAT>
```

A carriage return returns the OPTION MENU to the screen. Typing a 1 through 3 requests an edit session for the corresponding data type subfile. Table 3-2 specifies the content of each of the three types of subfiles.
If the user is editing a particular data type subfile for the first time for the selected project, CAT will respond

*** CREATING NEW DATA FOR TYPE n

where n is the integer value given to the EDITOR - SELECT A DATA TYPE menu. CAT then will display each field name on the terminal followed by a caret (>). The caret is preceded by a number in parentheses informing the user of the length of the data field requested by the prompt. After the user enters the data (or enters a carriage return only), CAT prompts for the next field of data.

To exit from the subfile creation mode, the user should respond with a carriage return only to the prompt for DATE DEFINED (Discrepancy and Change subfile or Milestone/Deliverable subfile) or for TEST DATE (Test History subfile). These fields are the first in a sequence of fields (listed in Table 3-2) which make up a "block" of data. After exiting the editor, CAT will redisplay the previous menu, EDITOR - SELECT A DATA TYPE.

If the user edits a subfile that already exists, CAT prompts the user with

CAT> *

The user responds with any of the commands described in Table 3-3. These commands are a subset of the edit commands used by the DEC Line Text Editor (EDI). The syntax and the resulting action for these commands are exactly as described in the EDI manual (Reference 1) with the following exceptions for the insert, delete, locate, and change commands. The insert (I) and delete (D) commands act upon a block of lines rather than on single lines as in the EDI editor.

If an insert command is entered, the editor locates the last line of the block to which the current line belongs and
<table>
<thead>
<tr>
<th>Function</th>
<th>Command Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>A</td>
<td>Adds text following add command to end of current line</td>
</tr>
<tr>
<td>ADD AND PRINT</td>
<td>AP</td>
<td>Adds text following add command to end of current line; prints out current new line</td>
</tr>
<tr>
<td>INSERT</td>
<td>I</td>
<td>Issues prompts for data for new record &quot;block&quot;</td>
</tr>
<tr>
<td>NEXT</td>
<td>N(n)</td>
<td>Establishes a new current line plus or minus n lines from current line</td>
</tr>
<tr>
<td>NEXT AND PRINT</td>
<td>NP(n)</td>
<td>Establishes a new current line plus or minus n lines from current line; prints new current line</td>
</tr>
<tr>
<td>CHANGE</td>
<td>C/STRING1/STRING2</td>
<td>Searches for STRING1 to the right of the '&gt;' and replaces it with text specified in STRING2</td>
</tr>
<tr>
<td>BOTTOM</td>
<td>BO</td>
<td>Sets current line pointer to bottom of file</td>
</tr>
<tr>
<td>TOP</td>
<td>TOP</td>
<td>Sets current line pointer to top of file</td>
</tr>
<tr>
<td>DELETE</td>
<td>D(n)</td>
<td>Deletes current and next n-1 record &quot;blocks&quot; if n is positive; deletes n preceding record &quot;blocks&quot; (but not current record &quot;block&quot;) if n is negative</td>
</tr>
<tr>
<td>EXIT</td>
<td>EX</td>
<td>Exits from CAT editor and saves data in master file</td>
</tr>
<tr>
<td>KILL</td>
<td>KILL</td>
<td>Exits from CAT editor (project file contents are not changed)</td>
</tr>
<tr>
<td>LOCATE</td>
<td>L STRING1</td>
<td>Locates and displays first occurrence of STRING1</td>
</tr>
<tr>
<td>PRINT</td>
<td>P(n)</td>
<td>Prints next n lines at the terminal</td>
</tr>
</tbody>
</table>
begins to prompt the user for data fields. Each line is inserted into the subfile at that location. To terminate the insertion, the user responds with a null line (carriage return only) to the prompt for DATE DEFINED or TEST DATE.

If a delete command is entered, the editor deletes all lines of the block which contains the current line. If a multiple deletion is specified (Dn), the editor deletes the current block and the n-1 following blocks.

The locate (L) and change (C) commands act almost the same as in the EDI editor. The locate command may be used to search for any string displayed to the user, including the prompt text at the start of each line. The change command, however, will affect only text which appears to the right of (and not including) the caret.

The EX or KILL command will result in termination of the edit session and the redisplay of the EDITOR - SELECT A DATA TYPE menu.
3.4 USE OF CAT REPORT GENERATOR

The CAT report generator is invoked by the following response to the OPTION MENU:

OPTION MENU
   1  EDITOR
   2  REPORTS
CAT>2

CAT then requests a selection of the output device for the generated reports:

OUTPUT LISTING DEVICE SELECTOR
   1  CRT
   2  PRINTER
CAT>

A carriage return requests a return to the OPTION MENU. Entering a 1 selects the user terminal as the report output device. If the user enters a 2 in response to the OUTPUT LISTING DEVICE SELECTOR menu, CAT will write the reports to the disk file "CATLST.LST". On termination of the CAT session, this data file may be spooled to the lineprinter by the user via one of the system utilities.

If the user enters a 1 or a 2, CAT requests the selection of a particular report:

REPORT - SELECT DATA TYPE
   1  DISCREPANCY AND CHANGE HISTORY DATA
   2  MILESTONE/DELIVERABLE HISTORY DATA
   3  TEST HISTORY DATA
   4  ALL
CAT>

A carriage return returns the OUTPUT LISTING DEVICE SELECTOR menu to the screen. Entering a 1 through 3 requests the corresponding report. Entering a 4 will display all of the reports.
Tables 3-1 and 3-2 specify the purpose and content of each of the three reports.

If the user terminal is the current output device selected, CAT will display a header line with column labels and 14 lines of data per screen followed by a CAT> prompt. A carriage return in response to the CAT> prompt will display the next screen or redisplay the REPORT menu if there are no more data. The user can terminate a report prematurely by responding to the CAT> prompt with a "?" rather than a carriage return. This response will redisplay the REPORT - SELECT DATA TYPE menu.

If the lineprinter is the current output device, CAT will display on the user terminal the name of the report selected while the CAT program is writing the report to the "CATLST.LST" file.

If the user selects item 1 or 4 from the REPORT - SELECT DATA TYPE menu:

REPORT - SELECT DATA TYPE
1 DISCREPANCY AND CHANGE HISTORY DATA
2 MILESTONE/DELIVERABLE HISTORY DATA
3 TEST HISTORY DATA
4 ALL
CAT>

before the DISCREPANCY AND CHANGE HISTORY DATA report is displayed or written to the print file, the DISCREPANCY - CHANGE SELECTOR menu will be presented:

DISCREPANCY - CHANGE SELECTOR
1 CHANGE
2 DISCREPANCY
3 UNFINISHED
4 ALL
CAT>
If the user responds with a 1, only the data marked with a "C" will be displayed. A response of 2 displays data marked with a "D" only. Selecting item 3 displays only the data (CHANGE and DISCREPANCY) that has a blank DATE CORRECTED field associated with. A response of 4 displays all of the data in the Discrepancy and Change subfile. After the selected reports have been displayed or written to the print file, the REPORT - SELECT DATA TYPE menu is redisplayed.

As noted before, a carriage return can be used to return the previous menu to the screen. Thus, the user may proceed up and down the menu hierarchy, generating reports on different devices as desired.
3.5 **SUMMARY**

This section provides a summary of CAT user information. Sections 3.2, 3.3, and 3.4 should be read for a more detailed discussion of program use.

The essential steps in the use of CAT are as follows:

1. The user initiates a CAT session with the command
   ```
   $RUN DB1: [213,2]CAT (PDP-11/70)
   $RUN DBB1: [TOOLS]CAT (VAX-11/780)
   ```

2. CAT prompts with
   ```
   ENTER THE PROJECT NAME >
   ```
   If the requested project file exists, the user immediately enters the program. If the requested project file does not exist, CAT asks if the user desires to create a new project file.

3. After the user has successfully entered the program, CAT displays a series of menus (see Figure 3-1). The hierarchy of menus allows the user to edit the project files or to generate reports.

   Each item in a menu is preceded by an integer by which the user selects an item. The user selects the item and then presses the carriage return. If the user presses the carriage return without selecting an item, CAT returns the previous menu to the screen.

4. In the edit mode, CAT prompts the user for one of the edit commands in Table 3-3 with
   ```
   CAT>*
   ```
Figure 3-1. CAT Menu and Prompt Hierarchy
The user types the desired command and then presses the carriage return.

5. In the report mode, CAT prompts the user for an output device. If the device selected is the printer, CAT generates a new data file, CATLST.LST, which the user can print after the CAT session.

6. While the output device is the CRT, CAT will display the requested report at the user's terminal.

7. When the Discrepancy/Change report is requested, CAT will display the DISCREPANCY-CHANGE SELECTOR menu. Selecting CHANGE results in a report of only those items flagged as changes. Selecting DISCREPANCY results in a report of only those reports marked as discrepancies. Selecting UNFINISHED will display all items that have not been completed.

8. To exit CAT properly, the user returns to the ENTER PROJECT NAME prompt with a series of carriage returns and enters a carriage return to terminate the session. CAT then displays the following message:

   ******** CAT TERMINATED ********

Figures 3-2 through 3-19 show a session with CAT. The session is described below.

Figure 3-2 shows the start of a session in which the user requests the creation of a new CAT data base file, GESTIT.DAT.

Note that the user only specifies the file name, GESTIT, and does not specify the extension .DAT.

Figure 3-3 shows the editor session in which a new Discrepancy and Change (D/C) subfile is created. Four
RUN CAT
ENTER PROJECT NAME: GESTIT

OPTION MENU
1 EDITOR
2 REPORTS
CAT>1

NO SUCH FILE

PROGRAM CREATING A NEW FILE FOR -- GESTIT.DAT
TO CONTINUE, ENTER Y FOR YES OR N FOR NO<br><br>Y

Figure 3-2. Starting CAT and Creating a New Data Base File
**EDITOR - SELECT A DATA TYPE**

1. DISCREPANCY & CHANGE DATA  
2. MILESTONE/DELIVERABLE DATA  
3. TEST HISTORY

**CAT > 1**

***CREATING NEW DATA FOR TYPE 1***

- **DATE DEFINED (MM/DD/YY) (8)**: 09/01/82
- **DATE CORRECTED (MM/DD/YY) (8)**: 09/15/82
- **DATE RELEASED (MM/DD/YY) (8)**
- **IMPLEMENTOR (8)**: W. TAYLOR
- **CODE (C OR D) (1)**: C
- **DESC (60)**: ENHANCE THE CAT PROGRAM TO BE MORE EFFICIENT
- **CHANGES (22)**: MODIFIED MANY ROUTINES

- **DATE DEFINED (MM/DD/YY) (8)**: 09/02/82
- **DATE CORRECTED (MM/DD/YY) (8)**
- **DATE RELEASED (MM/DD/YY) (8)**
- **IMPLEMENTOR (8)**: W. TAYLOR
- **CODE (C OR D) (1)**: D
- **DESC (60)**: DISCREPANCY WITHIN THE REDCAT ROUTINE
- **CHANGES (22)**: POINTERS INCORRECT FOR DATA FILES

- **DATE DEFINED (MM/DD/YY) (8)**: 09/03/82
- **DATE CORRECTED (MM/DD/YY) (8)**: 09/21/82
- **DATE RELEASED (MM/DD/YY) (8)**
- **IMPLEMENTOR (8)**: W. TAYLOR
- **CODE (C OR D) (1)**: C
- **DESC (60)**: RECONSTRUCTED THE CAT PROGRAM
- **CHANGES (22)**

- **DATE DEFINED (MM/DD/YY) (8)**: 09/02/82
- **DATE CORRECTED (MM/DD/YY) (8)**: 09/21/82
- **DATE RELEASED (MM/DD/YY) (8)**
- **IMPLEMENTOR (8)**: W. TAYLOR
- **CODE (C OR D) (1)**: D
- **DESC (60)**: DELETION OF THE IMPLICIT STATEMENTS
- **CHANGES (22)**

**Figure 3-3. Creating a New Discrepancy and Change Subfile**
Figure 3-4. Returning To Edit the Discrepancy and Change Subfile
EDITOR — SELECT A DATA TYPE

1 DISCREPANCY & CHANGE DATA
2 MILESTONE/DELIVERABLE DATA
3 TEST HISTORY

CAT > 2

### CREATING NEW DATA FOR TYPE 2

DATE DEFINED (MM/DD/YY) (8) > 09/01/82
DATE SCHEDULED (MM/DD/YY) (8) > 09/30/82
DATE ACTUAL (MM/DD/YY) (8) > 09/30/82
IMPLEMENTOR (8) > W. TAYLOR
TYPE (8) > FINAL
MILE (72) > DELIVER THE NEW CAT PROGRAM

DATE DEFINED (MM/DD/YY) (8) > 09/02/82
DATE SCHEDULED (MM/DD/YY) (8) > 09/30/82
DATE ACTUAL (MM/DD/YY) (8) > 09/30/82
IMPLEMENTOR (8) > W. TAYLOR
TYPE (8) > DRAFT
MILE (72) > UPDATE USERS GUIDE FOR THE CAT PROGRAM

DATE DEFINED (MM/DD/YY) (8) >

EDITOR — SELECT A DATA TYPE

1 DISCREPANCY & CHANGE DATA
2 MILESTONE/DELIVERABLE DATA
3 TEST HISTORY

CAT > 3

### CREATING NEW DATA FOR TYPE 3

TEST DATE (MM/DD/YY) (8) > 09/01/82
TEST CONDUCTOR (8) > W. TAYLOR
DESC (72) > TEST OUT THE LATEST CAT PROGRAM
RESULTS (32) > SEEMS TO BE WORKING

TEST DATE (MM/DD/YY) (8) > 09/02/82
TEST CONDUCTOR (8) > W. TAYLOR
DESC (72) > REMOVED ALL IMPLICIT STATEMENTS
RESULTS (32) > CORRECTED LOGIC ERROR

Figure 3-5. Creating New Milestone Deliverable and Test History Subfiles
EDITOR - SELECT A DATA TYPE
1  DISCREPANCY & CHANGE DATA
2  MILESTONE/DELIVERABLE DATA
3  TEST HISTORY
CAT> 

OPTION MENU
1  EDITOR
2  REPORTS
CAT>2

OUTPUT LISTING DEVICE SELECTOR
1  CRT
2  PRINTER
CAT>1

Figure 3-6. Proceeding From the Editor to the Report Generator (1 of 2)
REPORT - SELECT DATA TYPE

1 DISCREPANCY AND CHANGE HISTORY DATA
2 MILESTONE/DELIVERABLE HISTORY DATA
3 TEST HISTORY DATA
4 ALL

CAT>1

DISCREPANCY - CHANGE SELECTOR

1 CHANGE
2 DISCREPANCY
3 UNFINISHED
4 ALL

CAT>4

Figure 3-4. Proceeding From the Editor to the Report Generator (2 of 2)
**PROJECT GESTIT**  
**DISCREPANCY AND CHANGE HISTORY REPORT**  
21-SEP-82  PAGE  1

<table>
<thead>
<tr>
<th>DATE</th>
<th>DESCRIPTION</th>
<th>CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/01/82</td>
<td>C ENHANCE THE CAT PROGRAM TO B</td>
<td>MODIFIED MANY RD POINTERS INCORRE</td>
</tr>
<tr>
<td>09/02/82</td>
<td>D DESCREPANCY WITHIN THE REDCA</td>
<td></td>
</tr>
<tr>
<td>09/03/82</td>
<td>C RECONSTRUCTED THE CAT PROGRA</td>
<td></td>
</tr>
<tr>
<td>09/02/82</td>
<td>D DELETEION OF THE IMPLICIT ST</td>
<td></td>
</tr>
<tr>
<td>CAT&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-7. CRT Format of the Discrepancy and Change Report  
When 'ALL' Items are Selected
<table>
<thead>
<tr>
<th></th>
<th>DATE</th>
<th>CORRECTED</th>
<th>RELEASED</th>
<th>DESCRIPTION</th>
<th>CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>09/01/82</td>
<td>09/15/82</td>
<td></td>
<td>C ENHANCE THE CAT PROGRAM TO B</td>
<td>MODIFIED MANY RO</td>
</tr>
<tr>
<td>2</td>
<td>09/03/82</td>
<td>09/21/82</td>
<td></td>
<td>C RECONSTRUCTED THE CAT PROGRAM</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-8. CRT Format of the Discrepancy and Change Report When 'CHANGE' Items Are Selected
### Figure 1-9. CRT Format of the Discrepancy and Change Report When 'DISCREPANCY' Items are Selected
<table>
<thead>
<tr>
<th>DATE</th>
<th>DEFINED</th>
<th>CORRECTED</th>
<th>RELEASED</th>
<th>DESCRIPTION</th>
<th>CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/01/82</td>
<td></td>
<td>09/15/82</td>
<td></td>
<td>C ENHANCE THE CAT PROGRAM TO B</td>
<td>MODIFIED MANY RO</td>
</tr>
<tr>
<td>09/02/82</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09/03/82</td>
<td></td>
<td>09/21/82</td>
<td></td>
<td>D DESCREPANCY WITHIN THE REDCA</td>
<td>POINTERS INCORRE</td>
</tr>
<tr>
<td>09/02/82</td>
<td></td>
<td>09/21/82</td>
<td></td>
<td>C RECONSTRUCTED THE CAT PROGRA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D DELETEION OF THE IMPLICIT ST</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-10. CRT Format of the Discrepancy and Change Report
When 'UNFINISHED' Items are Selected
REPORT - SELECT DATA TYPE

1 DISCREPANCY AND CHANGE HISTORY DATA
2 MILESTONE/DELIVERABLE HISTORY DATA
3 TEST HISTORY DATA
4 ALL

CAT>2

REPORT - SELECT DATA TYPE

1 DISCREPANCY AND CHANGE HISTORY DATA
2 MILESTONE/DELIVERABLE HISTORY DATA
3 TEST HISTORY DATA
4 ALL

CAT>3

Figure 3-11. Selecting the Milestone Deliverable and Test History Reports
### PROJECT GESTIT MILESTONE/DELIVERABLE HISTORY REPORT  21-SEP-82  PAGE 1

<table>
<thead>
<tr>
<th>DEFINED SCHEDULED</th>
<th>ACTUAL TYPE</th>
<th>MILESTONE/DELIVERABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/01/82 09/30/82 09/30/82</td>
<td>FINAL</td>
<td>DELIVER THE NEW CAT PROGRAM</td>
</tr>
<tr>
<td>09/02/82 09/30/82 09/30/82</td>
<td>DRAFT</td>
<td>UPDATE USERS GUIDE FOR THE CAT PROGRAM</td>
</tr>
</tbody>
</table>

**Figure 3-12.** CRT Format of the Milestone/Deliverable Report
<table>
<thead>
<tr>
<th>TEST DATE</th>
<th>DESCRIPTION</th>
<th>RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/01/82</td>
<td>TEST OUT THE LATEST CAT PROGRAM</td>
<td>SEEMS TO BE WORKING</td>
</tr>
<tr>
<td>09/02/82</td>
<td>REMOVED ALL IMPLICIT STATEMENTS</td>
<td>CORRECTED LOGIC ERROR</td>
</tr>
</tbody>
</table>

Figure 3-13. CRT Format of the Test History Report
REPORT - SELECT DATA TYPE

1 DISCREPANCY AND CHANGE HISTORY DATA
2 MILESTONE/DELIVERABLE HISTORY DATA
3 TEST HISTORY DATA
4 ALL

OUTPUT LISTING DEVICE SELECTOR

1 CRT
2 PRINTER

Figure 3-14. Changing the Report Output Device to the Lineprinter
REPORT - SELECT DATA TYPE

1 DISCREPANCY AND CHANGE HISTORY DATA
2 MILESTONE/DELIVERABLE HISTORY DATA
3 TEST HISTORY DATA
4 ALL

CAT: 4

DISCREPANCY AND CHANGE REPORT SELECTED

DISCREPANCY - CHANGE SELECTOR

1 CHANGE
2 DISCREPANCY
3 UNFINISHED
4 ALL
CAT: 4

MILESTONE/DELIVERABLE REPORT SELECTED
TEST REPORT SELECTED

Figure 3-15. Requesting All Reports While the Output Device Is the Lineprinter
<table>
<thead>
<tr>
<th>DATE</th>
<th>CORRECTED RELEASED</th>
<th>DECREASED</th>
<th>DESCRIPTION</th>
<th>CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/01/82</td>
<td>09/15/82</td>
<td>W.TAYLOR</td>
<td>C ENHANCE THE CAT PROGRAM TO BE MORE EFFICIENT</td>
<td>MODIFIED MANY ROUTINES</td>
</tr>
<tr>
<td>09/02/82</td>
<td></td>
<td>W.TAYLOR</td>
<td>D DECREASED WITHIN THE REDCAT ROUTINE</td>
<td>POINTERS INCORRECT FOR</td>
</tr>
<tr>
<td>09/03/82</td>
<td>09/21/82</td>
<td>W.TAYLOR</td>
<td>C RECONSTRUCTED THE CAT PROGRAM</td>
<td></td>
</tr>
<tr>
<td>09/07/82</td>
<td>09/21/82</td>
<td>W.TAYLOR</td>
<td>D DELETION OF THE IMPLICIT STATEMENTS</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-16. Printer Format of the Discrepancy and Change Report When 'All' Items Are Selected
**PROJECT** GESTIT

**TEST HISTORY REPORT**

**DATE** | **CONDUCTOR** | **DESCRIPTION** | **RESULTS**
--- | --- | --- | ---
09/01/82 | W. TAYLOR | TEST OUT THE LATEST CAT PROGRAM | REFS TO BE WORKING
09/07/82 | W. TAYLOR | REMOVED ALL IMPLICIT STATEMENTS | CORRECTED LOGIC ERROR

**Figure 3-18.** Printer Format of the Test History Report
REPORT - SELECT DATA TYPE

1 DISCREPANCY AND CHANGE HISTORY DATA
2 MILESTONE/DELIVERABLE HISTORY DATA
3 TEST HISTORY DATA
4 ALL

CAT>

OUTPUT LISTING DEVICE SELECTOR

1 CRT
2 PRINTER

CAT>

OPTION MENU
1 EDITOR
2 REPORTS

CAT>

ENTER PROJECT NAME:
**** CAT TERMINATING ****

Figure 3-19. Terminating CAT from the Report Generator
"blocks" of data are entered. Each block starts with the entry of a date for the DATE DEFINED field. Not all fields must be entered; however, if no date is entered into the DATE DEFINED field, data entry is terminated and the EDITOR-SELECT A DATA TYPE menu is displayed (Figure 3-4). The number in parentheses immediately preceding the ' ' at the end of each prompt is the allowed length in characters for the data field to be entered.

Figure 3-4 continues the session by showing how the user returns to the D/C subfile to correct the errors made while creating the subfile. (The sixth editor command fails because the string DES appears to the left of the ' ' only.) The EX command is used to terminate an editor session with an old subfile.

Figure 3-5 demonstrates the creation of the Milestone and Deliverable (M/D) and the Test History (TH) subfiles shown later in the session. The sessions are terminated by entering a carriage return only for the DATE DEFINED (M/D subfile) and TEST DATE (TH subfile) fields.

In this session, the user moves from the editor to the report generator as shown in Figure 3-6. The user selects the terminal as the output device and requests that the D/C report be generated for all data in the D/C subfile.

Figure 3-7 is the resulting report. Figures 3-8 through 3-10 show the D/C reports when the CHANGE, DISCREPANCY, and UNFINISHED options are selected.

Figure 3-11 shows the user selecting the M/D and TH reports shown in Figures 3-12 and 3-13.

The user can change the report output device to the printer as shown in Figure 3-14 by entering a carriage return in response to the REPORT-SELECT DATA TYPE menu and reselecting an item from the OUTPUT LISTING DEVICE SELECTOR menu.
Figure 3-15 continues the session by showing the result of selecting all reports (and a D/C report of all data) while the output device is the line printer. Note that the name of the report is displayed to the user while the report is being written to the printer file. Figures 3-16 through 3-18 shown the printer format of all the reports.

Figure 3-19 demonstrates how multiple carriage returns will terminate CAT.
APPENDIX A - CAT SUBROUTINE DESCRIPTIONS

This appendix presents descriptions of each CAT subroutine. The descriptions provided here appear in alphabetical order by subroutine name. In addition to the routines listed in this appendix, CAT subroutines call the system routines as shown below.

<table>
<thead>
<tr>
<th>System Routine</th>
<th>Called By</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>CRTDIS, CRTMIL, CRTTST, PRTDIS, PRTMIL, PRTTST</td>
</tr>
<tr>
<td>ERRSET</td>
<td>CAT</td>
</tr>
<tr>
<td>ERRSNS</td>
<td>CAT</td>
</tr>
<tr>
<td>EXIT</td>
<td>CAT</td>
</tr>
</tbody>
</table>

These system routines are documented in References 2 and 3.
ROUTINE: ADD

TYPE: Subroutine

PURPOSE: Editor routine - add a character string to the end of the current line in the edit file.

USAGE:
1. Calling Sequence:

   CALL ADD (STRING)

   FORTRAN
   Name   I/O  Type   Dimension  Description
   STRNG1  I  Byte   80  Character string to add to current line

2. COMMON Blocks Used: LUNCOM

3. Subroutines Used: MOVE

4. Subroutines Called by: EDIT

5. External Data Sets Referenced:

   Lun    File Name   Operation(s)
   LFORW  FOR007.DAT  Write
   LOUT   Terminal   Write
ROUTINE: ADDP

TYPE: Subroutine

PURPOSE: Editor routine - add a character string to the end of the current line in the edit file and list the resulting line at the terminal.

USAGE:
1. Calling Sequence:
   
   CALL ADDP(STRNG1)

   FORTRAN Name   I/O   Type     Dimension   Description
   STRNG1         I     Byte     80           Character string to add to current line

2. COMMON Blocks Used: LUNCOM
3. Subroutines Used: MOVE, WRT
4. Subroutines Called by: EDIT
5. External Data Sets Referenced:
   
   Lun    File Name   Operation(s)
   LFORW  FOR007.DAT  Write
   LOUT   Terminal    Write

A-3

9023
ROUTINE: BOT

TYPE: Subroutine

PURPOSE: Editor routine - selects the last line in the edit file as the current line.

USAGE:
1. Calling Sequence:
   
   CALL BOT

2. COMMON Blocks Used: LUNCOM

3. Subroutines Used: MOVE

4. Subroutines Called by: EDIT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOUT</td>
<td>Terminal</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: CAT

TYPE: Main program

PURPOSE: Executive routine - performs initialization and cleanup functions for program execution. Responds to user's selection of report generation or editor functions. Opens and closes the user's selection for a project file.

USAGE:
1. Calling Sequence:
   - VAX: RUN DBB1:[TOOLS]CAT
   - PDP: RUN DB1:[213,2]CAT

2. COMMON Blocks Used: FILCM

3. Subroutines Used: EDTSEL, ERRSET, ERRSNS, EXIT, REPORT

4. Subroutines Called by: None

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE1</td>
<td>User specified</td>
<td>Open, read, close</td>
</tr>
<tr>
<td>LUNA</td>
<td>Terminal</td>
<td>Read, write</td>
</tr>
<tr>
<td>LUNOUT</td>
<td>CATLST.LST</td>
<td>Close</td>
</tr>
</tbody>
</table>
ROUTINE: CATFIL

TYPE: Subroutine

PURPOSE: Editor routine - creates a sequential file from a specified subfile. Each record in the subfile is used to create a block of records in the primary file, each containing a descriptive label and a data field from the subfile record.

USAGE:

1. Calling Sequence:

   CALL CATFIL (ITYPE)

   FORTRAN Name | I/O | Type | Dimension | Description
   ITYPE         | I   | I*2  | 1          | Pointer to the specific subfile from which to create the primary file

2. COMMON Blocks Used: CATCOM, LUNCOM, FILCM

3. Subroutines Used: None

4. Subroutines Called by: EDITOR

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE1</td>
<td>User specified</td>
<td>Read</td>
</tr>
<tr>
<td>LFILE</td>
<td>FOR002.DAT</td>
<td>Write</td>
</tr>
<tr>
<td>LUN5</td>
<td>Terminal</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: CATINS

TYPE: Subroutine

PURPOSE: Editor routine - insert a block of records into the edit file. Locates the next available insertion point in the edit file and prompts the user for data to include in the file. Reserve space for the data in the project file.

USAGE:

1. Calling Sequence:

   CALL CATINS (ITYPE, NEXT)

   **FORTRAN**

<table>
<thead>
<tr>
<th>Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITYPE</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Pointer to the specific subfile in the project file which is contained in the edit file</td>
</tr>
<tr>
<td>NEXT</td>
<td>I/O</td>
<td>I*2</td>
<td>1</td>
<td>Pointer to the next available record in the project file</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: CATCOM, FILCM, LUNCOM

3. Subroutines Used: GETMOR, MOVLIN

4. Subroutines Called by: EDIT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE1</td>
<td>User Specified</td>
<td>Read</td>
</tr>
<tr>
<td>LUN5</td>
<td>Terminal</td>
<td>Read, write</td>
</tr>
<tr>
<td>LOUT</td>
<td>Terminal</td>
<td>Write</td>
</tr>
<tr>
<td>LFORW</td>
<td>FOR007.DAT</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: CATPUT

TYPE: Subroutine

PURPOSE: Editor routine - accepts the initial data for a subfile containing no data. Data is read in response to prompts for each data field. After each block of data is read, a record is added to the subfile in the project file.

USAGE:
1. Calling Sequence:

    CALL CATPUT (ITYPE, STAT)

    FORTRAN Name    I/O  Type     Dimension     Description
    ITYPE            I    I*2     1             Pointer to the specific subfile in the project file which is being created
    STAT             I    L*1     1             Flag indicating presence of header record in project file:
                                                                 = '0', old file, header present
                                                                 = 'N', new file, no header record

2. COMMON Blocks Used: CATCOM, FILCM

3. Subroutines Used: GETMOR

4. Subroutines Called by: EDTSEL

5. External Data Sets Referenced:

    Lun   File Name         Operation(s)
    ------   ---------------   ------------
    FILE1   User Specified   Read, write
    LUN5    Terminal         Read, write
    LUN6    Terminal         Write
ROUTINE: CHG

TYPE: Subroutine

PURPOSE: Editor routine - change a specified character string in the current line in the edit file to a new character string.

USAGE:

1. Calling Sequence:

   CALL CHG (NUM, STRNG1)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Number of occurrences of the character string to be changed</td>
</tr>
<tr>
<td>STRNG1</td>
<td>I</td>
<td>Byte</td>
<td>80</td>
<td>Specification for the change</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: LUNCOM

3. Subroutines Used: COMPL, MOVE, REMVB, WRT

4. Subroutines Called by: EDIT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOUT</td>
<td>Terminal:</td>
<td>Write</td>
</tr>
<tr>
<td>LFORW</td>
<td>FOR007.DAT</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: COMPL

TYPE: Subroutine

PURPOSE: Editor routine - locates a specified character string within another character string.

USAGE:

1. Calling Sequence:

CALL COMPL (XLINE, CSTRNG, NCHR, NPST, NPED)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLINE</td>
<td>I</td>
<td>Byte</td>
<td>80</td>
<td>Character string to search</td>
</tr>
<tr>
<td>CSTRNG</td>
<td>I</td>
<td>Byte</td>
<td>80</td>
<td>Character string to search for in XLINE</td>
</tr>
<tr>
<td>NCHR</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Number of characters in CSTRNG</td>
</tr>
<tr>
<td>NPST</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Position in XLINE at which to start search</td>
</tr>
<tr>
<td>NPED</td>
<td>O</td>
<td>I*2</td>
<td>1</td>
<td>Position in XLINE at which next occurrence of CSTRNG starts. Set to 81 if CSTRNG not found</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: None

3. Subroutines Used: None

4. Subroutines Called by: CHG, LOC

5. External Data Sets Referenced: None
ROUTINE: CRTDIS

TYPE: Subroutine

PURPOSE: Report routine - writes a Discrepancy and Change History report to the terminal.

USAGE:
1. Calling Sequence:

    CALL CRTDIS (PROJ)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJ</td>
<td>I</td>
<td>R*8</td>
<td>1</td>
<td>Project file name to be displayed in report heading</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: FILCM

3. Subroutines Used: DATE, PIKONE

4. Subroutines Called by: REPSEL

5. External Data Sets Reference:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE1</td>
<td>User specified</td>
<td>Read</td>
</tr>
<tr>
<td>LUNOUM</td>
<td>Terminal</td>
<td>Read, write</td>
</tr>
</tbody>
</table>
ROUTINE: CRTMIL

TYPE: Subroutine

PURPOSE: Report routine - writes a Milestone/Deliverable History report to the terminal.

USAGE:

1. Calling Sequence:

   CALL CRTMIL (PROJ)

FORTRAN Name I/O Type Dimension Description

   PROJ   I    R*8    1    Project file name to be displayed in report heading

2. COMMON Blocks Used: FILCM

3. Subroutines Used: DATE

4. Subroutines Called by: REPSEL

5. External Data Sets Referenced:

   Lun      File Name      Operation(s)
   FILE1    User specified  Read
   LUNOUT   Terminal        Read, write
ROUTINE: CRTTST

TYPE: Subroutine

PURPOSE: Report routine - writes a Test History report to the terminal.

USAGE:

1. Calling Sequence:

    CALL CRTTST (PROJ)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJ</td>
<td>I</td>
<td>R*8</td>
<td>1</td>
<td>Project file name to be displayed in report heading</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: FILCM

3. Subroutines Used: DATE

4. Subroutines Called by: REPSEL

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE1</td>
<td>User specified</td>
<td>Read</td>
</tr>
<tr>
<td>LUNOUT</td>
<td>Terminal</td>
<td>Read, write</td>
</tr>
</tbody>
</table>
ROUTINE: DECNUM

TYPE: Subroutine

PURPOSE: Editor routine - locates, and decodes if present, the repetition factor in an editor command.

USAGE:

1. Calling Sequence:

CALL DECNUM (ENTRY, IPT, NUM, IFND, N)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY</td>
<td>I</td>
<td>Byte</td>
<td>80</td>
<td>Editor command to search for repetition factor</td>
</tr>
<tr>
<td>IPT</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Pointer to location in ENTRY at which search starts</td>
</tr>
<tr>
<td>NUM</td>
<td>O</td>
<td>I*2</td>
<td>1</td>
<td>Decoded number if factor present, not altered if factor not present</td>
</tr>
<tr>
<td>IFND</td>
<td>O</td>
<td>I*2</td>
<td>1</td>
<td>= 0, no factor found = 1, factor found and decoded</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Length of ENTRY in bytes</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: None

3. Subroutines Used: None

4. Subroutines Called by: GETLIN

5. External Data Sets Referenced: None
ROUTINE:  DEL
TYPE:  Subroutine
PURPOSE:  Editor routine - deletes a block of records from
the edit file.

USAGE:
1.  Calling Sequence:

    CALL DEL (NUM, NUMLIN)

    FORTRAN Name  I/O  Type  Dimension  Description
    NUM      I  I*2   1          Number of blocks to de-
                        lete from edit file
    NUMLIN   I  I*2   1          Number of records in
                        each block

2.  COMMON Blocks Used:  LUNCOM
3.  Subroutines Used:  MOVLIN
4.  Subroutines Called by:  EDIT
5.  External Data Sets Referenced:

    Lun  File Name  Operation(s)
    LIN  Terminal  Write
    LOUT Terminal  Write
    LFORW FOR007.DAT  Write
ROUTINE: EDIT

TYPE: Subroutine

PURPOSE: Editor routine - controls selection of editor routines in response to editor command entered by user.

USAGE:
1. Calling Sequence:

    CALL EDIT (IOLD, ISKP, ITYPE)

    FORTRAN Name I/O Type Dimension Description
    IOLD      I   I*2    1  Subfile existence on entry flag
               = 1, subfile exists
               = 0, subfile empty (not used)
    ISKP      O   I*2    1  Subfile disposition on exit flag
               = 1, delete subfile
               = 0, save subfile
    ITYPE     I   I*2    1  Subfile content flag
               = 1, Discrepancy/Change data
               = 2, Milestone/Deliverable data
               = 3, Test data

2. COMMON Blocks Used: CATCOM, FILCM, LUNCOM

3. Subroutines Used: ADD, ADDP, BOT, CATINS, CHG, DEL, GETLIN, LOC, MOVE, NEX, NEXP, PRI, TOP

4. Subroutines Called by: EDITOR

5. External Data Sets Referenced:

    Lun       File Name     Operation(s)
    LIN       Terminal      Write
    LFORW     FOR007.DAT    Write
ROUTINE: EDITOR

TYPE: Subroutine

PURPOSE: Editor routine - controls the creation and final disposition of the edit scratch files.

USAGE:

1. Calling Sequence:

CALL EDITOR (ITYPE)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITYPE</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Subfile content flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 1, Discrepancy/Change data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 2, Milestone/Deliverable data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 3, Test data</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: FILCM, LUNCAM, MOVCOM

3. Subroutines Used: CATFIL, EDIT, OPNX, REDCAT

4. Subroutines Called by: EDTSEL

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFILE</td>
<td>FOR002.DAT</td>
<td>Open, close</td>
</tr>
<tr>
<td>LOUT</td>
<td>Terminal</td>
<td>Write</td>
</tr>
<tr>
<td>LFORW</td>
<td>FOR007.DAT</td>
<td>Write, close</td>
</tr>
<tr>
<td>LBWK</td>
<td>FOR008.DAT</td>
<td>Close</td>
</tr>
</tbody>
</table>
ROUTINE: EDTSEL

TYPE: Subroutine

PURPOSE: Editor routine - obtains the user's selection of subfile type to edit. If the selected subfile does not contain data, the initial data entry routine is used, otherwise the full edit routines are used.

USAGE:

1. Calling Sequence:

   CALL EDTSEL (MFLAG, STAT)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFLAG</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Editor authorization flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0, user not authorized for edit (not used)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 1, user may edit</td>
</tr>
<tr>
<td>STAT</td>
<td>I</td>
<td>L*1</td>
<td>1</td>
<td>Flag indicating presence of header record in project file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 'O', old file, header present</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 'N', new file, no header record</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: FILCM

3. Subroutines Used: CATPUT, EDITOR

4. Subroutines Called by: CAT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUNA</td>
<td>Terminal</td>
<td>Read, write</td>
</tr>
<tr>
<td>5</td>
<td>Terminal</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: GETLIN

TYPE: Subroutine

PURPOSE: Editor routine - reads the editor command entered by the user. The type of command is determined.

USAGE:
1. Calling Sequence:

CALL GETLIN (IFUN, NUM, STRNG1, IERROR)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFUN</td>
<td>O</td>
<td>I*2</td>
<td>1</td>
<td>Code for entered command type</td>
</tr>
<tr>
<td>NUM</td>
<td>O</td>
<td>I*2</td>
<td>1</td>
<td>Repetition factor, set to 1 if no factor is found</td>
</tr>
<tr>
<td>STRNG1</td>
<td>O</td>
<td>Byte</td>
<td>80</td>
<td>Target and/or replacement string for string manipulation commands, terminated by a zero byte</td>
</tr>
<tr>
<td>IERROR</td>
<td>O</td>
<td>I*2</td>
<td>1</td>
<td>(Not used, always set to zero)</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: LUNCOM, MOVCOM

3. Subroutines Used: DECNUM, REMVB

4. Subroutines Called by: EDIT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOUT</td>
<td>Terminal</td>
<td>Write</td>
</tr>
<tr>
<td>LIN</td>
<td>Terminal</td>
<td>Read</td>
</tr>
</tbody>
</table>
ROUTINE: GETMOR

TYPE: Subroutine

PURPOSE: Editor routine - adds more records to the linked list of available records in the project file.

USAGE:

1. Calling Sequence:

   CALL GETMOR (FILE, LAST, MORE)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Logical unit number of the project file</td>
</tr>
<tr>
<td>LAST</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Record number of last record in list of available records</td>
</tr>
<tr>
<td>MORE</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Number of new records to add to the end of the list of available records</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: None

3. Subroutines Used: None

4. Subroutines Called by: CATINS, CATPUT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>User specified</td>
<td>Read, write</td>
</tr>
</tbody>
</table>
ROUTINE: LOC

TYPE: Subroutine

PURPOSE: Editor routine - locates the nth occurrence of a specified character string in the following records. If the nth occurrence of the string is found within a record, the record becomes the current line. If the nth occurrence of the string is not found, the current line remains the same as before the command was specified.

USAGE:

1. Calling Sequence:

   CALL LOC (NUM, STRNG1)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Number of occurrence at which to stop search</td>
</tr>
<tr>
<td>STRNG1</td>
<td>I</td>
<td>Byte</td>
<td>80</td>
<td>Character string to search for</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: LUNCOM

3. Subroutines Used: COMPL, MOVE, WRT

4. Subroutines Called by: EDIT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOUT</td>
<td>Terminal</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: MOVE

TYPE: Subroutine

PURPOSE: Editor routine - moves the current line pointer in the edit file. Breakpoint records in the edit file are not counted when they are encountered (see MOVLIN).

USAGE:

1. Calling Sequence:

   CALL MOVE (NX, XLINE, IERROR, IEOFX, ITOFX)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NX</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Number of records to move the current line pointer</td>
</tr>
<tr>
<td>XLINE</td>
<td>0</td>
<td>R*8</td>
<td>11</td>
<td>Content of the record which becomes the current line</td>
</tr>
<tr>
<td>IERROR</td>
<td></td>
<td>L*2</td>
<td>1</td>
<td>(Not used)</td>
</tr>
<tr>
<td>IEOFX</td>
<td>0</td>
<td>I*2</td>
<td>1</td>
<td>End of file (EOF) flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0, no EOF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 1, EOF encountered</td>
</tr>
<tr>
<td>ITOFX</td>
<td>0</td>
<td>I*2</td>
<td>1</td>
<td>Top of file (TOF) flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0, no TOF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 1, TOF encountered</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: LUNCOM, MOVCOM

3. Subroutines Used: MOVECR

4. Subroutines Called by: ADD, ADDP, BOT, CHG, EDIT, LOC, NEX, NEXP, PRI, TOP

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFILE</td>
<td>FOR002.DAT</td>
<td>Read</td>
</tr>
<tr>
<td>LFORW</td>
<td>FOR007.DAT</td>
<td>Read, write</td>
</tr>
<tr>
<td>LBKW</td>
<td>FOR008.DAT</td>
<td>Read, write</td>
</tr>
</tbody>
</table>

A-22
ROUTINE: MOVECR

TYPE: Subroutine

PURPOSE: Editor routine - copy a character array from one location to another for a specified length.

USAGE:
1. Calling Sequence:

   CALL MOVECR (FROM, TO, LEN)

   FORTRAN Name   I/O Type Dimension   Description
     FROM    I   L*1 LEN   Source location
     TO      O   L*1 LEN   Destination location
     LEN     I   I*2 1     Number of characters to copy

2. COMMON Blocks Used: None

3. Subroutines Used: None

4. Subroutines Called by: MOVE, MOVLIN

5. External Data Sets Referenced: None
ROUTINE: MOVLIN

TYPE: Subroutine

PURPOSE: Editor routine - move the current line pointer in the edit file. Breakpoint records in the edit file are counted when they are encountered (see MOVE).

USAGE:

1. Calling Sequence:

CALL MOVLIN (NX, XLINE, IERROR, IEOFX, ITOFX)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NX</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Number of records to move the current line pointer</td>
</tr>
<tr>
<td>XLINE</td>
<td>O</td>
<td>R*8</td>
<td>11</td>
<td>Contents of the record which becomes the current line</td>
</tr>
<tr>
<td>IERROR</td>
<td>-</td>
<td>L*2</td>
<td>1</td>
<td>(Not used)</td>
</tr>
<tr>
<td>IEOFX</td>
<td>O</td>
<td>I*2</td>
<td>1</td>
<td>End of file (EOF) flag = 0, no EOF = 1, EOF encountered</td>
</tr>
<tr>
<td>ITOFX</td>
<td>O</td>
<td>I*2</td>
<td>1</td>
<td>Top of file (TOF) flag = 0, no TOF = 1, TOF encountered</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: LUNCOM, MOVCOM

3. Subroutines Used: MOVECR

4. Subroutines Called by: CATINS, DEL

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFILE</td>
<td>FOR002.DAT</td>
<td>Read</td>
</tr>
<tr>
<td>LFORW</td>
<td>FOR007.DAT</td>
<td>Read, write</td>
</tr>
<tr>
<td>LBKW</td>
<td>FOR008.DAT</td>
<td>Read, write</td>
</tr>
</tbody>
</table>
ROUTINE: NEX

TYPE: Subroutine

PURPOSE: Editor routine - changes the current line pointer in the edit file.

USAGE:

1. Calling Sequence:

   CALL NEX (NUM)

   FORTRAN | I/O | Type  | Dimension | Description
             |     |       |           | Number of records to move the current line pointer

2. COMMON Blocks Used: LUNCOM

3. Subroutines Used: MOVE

4. Subroutines Called by: EDIT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOUT</td>
<td>Terminal</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: NEXP

TYPE: Subroutine

PURPOSE: Editor routine - changes the current line pointer in the edit file. Prints the new current line at the terminal.

USAGE:
1. Calling Sequence:

```
CALL NEXP (NUM)
```

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Number of records to move the current line pointer</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: LUNCOM

3. Subroutines Used: MOVE, WRT

4. Subroutines Called by: EDIT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOUT</td>
<td>Terminal</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: OINSX

TYPE: Subroutine

PURPOSE: Editor routine - initializes the edit scratch files.

USAGE:

1. Calling Sequence:

CALL OINSX (IERR)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IERR</td>
<td>0</td>
<td>I*2</td>
<td>1</td>
<td>Error flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0, No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 1, error reading the sequential file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 2, error opening one of the scratch files</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: LUNCOM

3. Subroutines Used: None

4. Subroutines Called by: EDITOR

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFILE</td>
<td>FOR002.DAT</td>
<td>Read</td>
</tr>
<tr>
<td>LOUT</td>
<td>Terminal</td>
<td>Write</td>
</tr>
<tr>
<td>LFORW</td>
<td>FOR007.DAT</td>
<td>Open, write</td>
</tr>
<tr>
<td>LBKW</td>
<td>FOR008.DAT</td>
<td>Open, write</td>
</tr>
</tbody>
</table>
ROUTINE: PIKONE

TYPE: Subroutine

PURPOSE: Report routine - obtains the user's selection for the type of data to present in a Discrepancy and Change History report when called the first time. Also used to determine whether each Discrepancy and Change data record should be included in the report.

USAGE:

1. Calling Sequence:

CALL PICONE (FIRST, PRTFLG, VALID, CODE, DATA)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST</td>
<td>I/O</td>
<td>L*2</td>
<td>1</td>
<td>= .TRUE., display selection menu</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= .FALSE., no menu display</td>
</tr>
<tr>
<td>PRTFLG</td>
<td>O</td>
<td>L*2</td>
<td>1</td>
<td>= .TRUE., record should be included in report</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= .FALSE., do not include record in report</td>
</tr>
<tr>
<td>VALID</td>
<td>O</td>
<td>L*2</td>
<td>1</td>
<td>= .TRUE., user has selected the data type to include</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= .FALSE., user has selected not to display report</td>
</tr>
<tr>
<td>CODE</td>
<td>I</td>
<td>L*1</td>
<td>1</td>
<td>Data type code for current record</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 'D', discrepancy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 'C', change</td>
</tr>
<tr>
<td>DATA</td>
<td>I</td>
<td>R*8</td>
<td>16</td>
<td>Current record from project file</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: None

3. Subroutines Used: None

4. Subroutines Called by: CRTDIS, PRTDIS

A-28
5. **External Data Sets Referenced:**

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUNA</td>
<td>Terminal</td>
<td>Read, write</td>
</tr>
</tbody>
</table>

A-29

9023
ROUTINE: PRI

TYPE: Subroutine

PURPOSE: Editor routine - move the current line pointer forward in the edit file and print at the terminal each record encountered.

USAGE:

1. Calling Sequence:

   CALL PRI (NUM)

   FORTRAN
   Name   I/O  Type  Dimension  Description
   NUM    I    I*2    1         Number of records to print at the terminal

2. COMMON Blocks Used: LUNCOM

3. Subroutines Used: MOVE, WRT

4. Subroutines Called by: EDIT

5. External Data Sets Referenced:

   Lun   File Name   Operation(s)
   LOUT  Terminal   Write
ROUTINE: PRTDIS

TYPE: Subroutine

PURPOSE: Report routine - writes a Discrepancy and Change History report to the print file.

USAGE:

1. Calling Sequence:

   CALL PRTDIS (PROJ, LUNOUT)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJ</td>
<td>I</td>
<td>R*8</td>
<td>1</td>
<td>Project file name to be displayed in report heading</td>
</tr>
<tr>
<td>LUNOUT</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Logical unit connected to print file</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: FILCM

3. Subroutines Used: DATE, PIKONE

4. Subroutines Called by: REPSEL

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE1</td>
<td>User specified</td>
<td>Read</td>
</tr>
<tr>
<td>LUNA</td>
<td>Terminal</td>
<td>Write</td>
</tr>
<tr>
<td>LUNOUT</td>
<td>CATLST.LST</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: PRTMIL

TYPE: Subroutine

PURPOSE: Report routine - writes a Milestone/Deliverable History report to the print file.

USAGE:
1. Calling Sequence:

   CALL PRTMIL (PROJ, LUNOUT)

   FORTRAN Name    I/O  Type Dimension Description
   -------------------------------------------------------
   PROJ             I     R*8     1  Project file name to be displayed in report heading
   LUNOUT           I     I*2     1  Logical unit connected to print file

2. COMMON Blocks Used: FILCM

3. Subroutines Used: DATE

4. Subroutines Called by: REPSEL

5. External Data Sets Referenced:

   Lun    File Name              Operation(s)
   ------ ------                  --------
   FILE1  User specified         Read
   LUNA   Terminal               Write
   LUNOUT CATLST.LST             Write
ROUTINE: PRTTST

TYPE: Subroutine

PURPOSE: Report routine - writes a Test History report to the print file.

USAGE:

1. Calling Sequence:

```
CALL PRTTST (PROJ, LUNOUT)
```

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJ</td>
<td>I</td>
<td>R*8</td>
<td>1</td>
<td>Project file name to be displayed in report heading</td>
</tr>
<tr>
<td>LUNOUT</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Logical unit connected to print file</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: FILCM

3. Subroutines Used: DATE

4. Subroutines Called by: REPSEL

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE1</td>
<td>User specified</td>
<td>Read</td>
</tr>
<tr>
<td>LUNA</td>
<td>Terminal</td>
<td>Write</td>
</tr>
<tr>
<td>LUNOUT</td>
<td>CATLST.LST</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: REDCAT

TYPE: Subroutine

PURPOSE: Editor routine - creates new version of a subfile from the edit scratch files. Each block of records in the forwards file is used to create a single record in the subfile by concatenating the data fields from the records in a record block.

USAGE:
1. Calling Sequence:
   CALL REDCAT (FILE, FILEO, ITYPE)

   FORTRAN	 Dimension	 Description
   Name	 I/O	 Type	 sion	
   FILE	 I	 I*2	 1	 Logical unit connected to the forwards edit file
   FILEO	 I	 I*2	 1	 Associated variable for logical unit FILE
   ITYPE	 I	 I*2	 1	 Pointer to the specific subfile to receive the data from the forwards file

2. COMMON Blocks Used: CATCOM, FILCM

3. Subroutines Used: None

4. Subroutines Called by: EDITOR

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE1</td>
<td>User specified</td>
<td>Read, write</td>
</tr>
<tr>
<td>LUN5</td>
<td>Terminal</td>
<td>Write</td>
</tr>
<tr>
<td>FILE</td>
<td>FOR007.DAT</td>
<td>Read</td>
</tr>
</tbody>
</table>

A-34
ROUTINE: REMVB

TYPE: Subroutine

PURPOSE: Editor routine - skips blank characters in the editor command line entered by the user.

USAGE:

1. Calling Sequence:

   CALL REMVB (ENTRY, IPT, N)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY</td>
<td>I</td>
<td>Byte</td>
<td>80</td>
<td>Editor command line</td>
</tr>
<tr>
<td>IPT</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Location in command line at which to start search for non-blank character</td>
</tr>
<tr>
<td>N</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Length of command line</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: None

3. Subroutines Used: None

4. Subroutines Called by: CHG, GETLIN

5. External Data Sets Referenced: None
ROUTINE: REPORT

TYPE: Subroutine

PURPOSE: Report routine - obtains the user's selection of output device (terminal or print file).

USAGE:

1. Calling Sequence:

   CALL REPORT (LUNA, LUNOUT, PROJECT)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUNA</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Logical unit connected to the user's terminal</td>
</tr>
<tr>
<td>LUNOUT</td>
<td>O</td>
<td>I*2</td>
<td>1</td>
<td>Logical unit connected to user's selected output device for report display</td>
</tr>
<tr>
<td>PROJECT</td>
<td>I</td>
<td>R*3</td>
<td>1</td>
<td>Project file name to be displayed in report heading</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: FILCM

3. Subroutines Used: REPSEL

4. Subroutines Called by: CAT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUNA</td>
<td>Terminal</td>
<td>Read, write</td>
</tr>
<tr>
<td>LUNOUT</td>
<td>CATLST.LST</td>
<td>Open</td>
</tr>
</tbody>
</table>
ROUTINE: REPSEL

TYPE: Subroutine

PURPOSE: Report routine - obtains the user's selection of report type and passes control to the appropriate routine based upon report type and output device.

USAGE:
1. Calling Sequence:

CALL REPSEL (LUNA, LUNOUT, PROJCT, ICRT)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUNA</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Logical unit connected to the user's terminal</td>
</tr>
<tr>
<td>LUNOUT</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Logical unit connected to user's selected output device for report display</td>
</tr>
<tr>
<td>PROJCT</td>
<td>I</td>
<td>R*8</td>
<td>1</td>
<td>Project file name to be displayed in report heading</td>
</tr>
<tr>
<td>ICRT</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>Report device flag = 1, output device is the terminal = 2, output device is the print file</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: FILCM

3. Subroutines Used: CRTDIS, CRTMIL, CRTTST, PRTDIS, PRTMIL, PRTTST

4. Subroutines Called by: REPORT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUNA</td>
<td>Terminal</td>
<td>Read, write</td>
</tr>
</tbody>
</table>

A-37

9023
ROUTINE: TOP

TYPE: Subroutine

PURPOSE: Editor routine - move the current line pointer to the first data record in the edit file.

USAGE:

1. Calling Sequence:

   CALL TOP

2. COMMON Blocks Used: LUNCOM

3. Subroutines Used: MOVE

4. Subroutines Called by: EDIT

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOUT</td>
<td>Terminal</td>
<td>Write</td>
</tr>
</tbody>
</table>
ROUTINE: WRT

TYPE: Subroutine

PURPOSE: Editor routine - write a record from the edit scratch file to the terminal. Trailing blanks in the record are not written.

USAGE:

1. Calling Sequence:

CALL WRT (OUTPUT, ISX, IEX)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>I/O</th>
<th>Type</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>I</td>
<td>Byte</td>
<td>80</td>
<td>Record to write to the terminal</td>
</tr>
<tr>
<td>ISX</td>
<td>I</td>
<td>I*2</td>
<td>1</td>
<td>First printable character in OUTPUT</td>
</tr>
<tr>
<td>IEX</td>
<td>I</td>
<td>I*2</td>
<td>2</td>
<td>Last printable character in OUTPUT (includes trailing blanks)</td>
</tr>
</tbody>
</table>

2. COMMON Blocks Used: LUNCOM

3. Subroutines Used: None

4. Subroutines Called by: ADDP, CHG, LOC, NEXP, PRI

5. External Data Sets Referenced:

<table>
<thead>
<tr>
<th>Lun</th>
<th>File Name</th>
<th>Operation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOUT</td>
<td>Terminal</td>
<td>Write</td>
</tr>
</tbody>
</table>
APPENDIX B - CAT COMMON BLOCK INFORMATION

Some of the variables used to interface among the CAT executive routine, the editor routines, and the report routines are stored in labeled COMMON blocks. The four CAT COMMON blocks and the subroutines which use them are listed below:

<table>
<thead>
<tr>
<th>COMMON Block</th>
<th>Subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATCOM</td>
<td>CATFIL, CATINS, CATPUT, EDIT, REDCAT</td>
</tr>
<tr>
<td>FILCM</td>
<td>CAT, CATFIL, CATINS, CATPUT, CRTDIS, CRTMIL, CRTTST, EDIT, EDITOR, EDTSEL, PRTDIS, PRTMIL, PRTTST, REDCAT, RFPORT, REPSEL</td>
</tr>
<tr>
<td>LUNCOM</td>
<td>ADD, ADDP, BOT, CATFIL, CATINS, CHG, DEL, EDIT, EDITOR, GETLIN, LOC, MOVE, MOVLIN, NEX, NEXP, OPNX, PRI, TOP, WRT</td>
</tr>
<tr>
<td>MOVCOM</td>
<td>EDITOR, GETLIN, MOVE, MOVLIN</td>
</tr>
</tbody>
</table>

Three of these COMMON blocks; CATCOM, FILCM, and LUNCOM are initialized in block data routines as shown in Figures B-1 through B-3. These three COMMON blocks are always incorporated into CAT source code through the use of INCLUDE statements. The files included in this way are shown in Figures B-4 through B-6. The MOVCOM COMMON block is neither initialized in a block data routine nor incorporated using the INCLUDE statement.

Detailed descriptions of the four COMMON blocks used by CAT are presented on the following pages. The descriptions are presented alphabetically by COMMON block name. The variables for each block are listed in the order in which they are stored.
Figure B-1. Block Data Routine CATBLK
BLOCK DATA FILBLK

INCLUDE 'FILCH.INC'

ARGUMENT TYPE DESCRIPTION

FILE1 | PROJECT FILE
AVAIL | POINT TO NEXT AVAILABLE LOCATION
FILE2 | FORWARD POINTER NODE
LIND | LAST POINTER NODE
FILE2 | GENERAL FILE PER DATA TYPE

DATA FILE 1, FILE 2

END

Figure B-2. Block Data Routine FILBLK
Figure B-3. Block Data Routine LUNBLK
Figure B-4. INCLUDE File for COMMON Block CATCOM
Figure B-5. INCLUDE File for COMMON Block FILCM
Figure B-6. INCLUDE File for COMMON Block LUNCOM
COMMON BLOCK: CATCOM

PURPOSE: Contain information describing the data field lengths of each type of subfile record in the project file. The descriptor labels used in the edit scratch files are also stored in this COMMON block.

VARIABLES:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATEXT(4,7)</td>
<td>R*8</td>
<td>Discrepancy and Change History edit file descriptor labels</td>
</tr>
<tr>
<td>BTEXT(4,7)</td>
<td>R*8</td>
<td>Milestone/Deliverable History edit file descriptor labels</td>
</tr>
<tr>
<td>CTEXT(4,7)</td>
<td>R*8</td>
<td>Text History edit file descriptor labels</td>
</tr>
<tr>
<td>NDATA(7,3)</td>
<td>I*2</td>
<td>Data field lengths in a subfile record or individual edit file record</td>
</tr>
<tr>
<td>CHAR(7,3)</td>
<td>I*2</td>
<td>Length of each descriptor label</td>
</tr>
</tbody>
</table>
COMMON BLOCK: FILCM

PURPOSE: Contain pointers describing the subfiles in the project file.

VARIABLES:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE1</td>
<td>I*2</td>
<td>Logical unit number connected to the project file</td>
</tr>
<tr>
<td>AVAIL</td>
<td>I*2</td>
<td>Pointer to next available record in the project file's list of available records</td>
</tr>
<tr>
<td>FN(7)</td>
<td>I*2</td>
<td>Pointers to the first record in each subfile</td>
</tr>
<tr>
<td>LN(7)</td>
<td>I*2</td>
<td>Pointers to the last record in each subfile</td>
</tr>
<tr>
<td>FILE2</td>
<td>I*2</td>
<td>(Not used)</td>
</tr>
<tr>
<td>NO</td>
<td>I*2</td>
<td>Associated variable for FILE1 (points to the next sequential record in the project file)</td>
</tr>
<tr>
<td>JO</td>
<td>I*2</td>
<td>(Not used)</td>
</tr>
</tbody>
</table>
COMMON BLOCK: LUNCOM

PURPOSE: Contain the logical unit numbers and associated variables for all files except the project file.

VARIABLES:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIN</td>
<td>I*2</td>
<td>Logical unit number used for reading input from the terminal</td>
</tr>
<tr>
<td>LOUT</td>
<td>I*2</td>
<td>Logical unit number used for writing output to the terminal</td>
</tr>
<tr>
<td>LFILE</td>
<td>I*2</td>
<td>Logical unit number connected to the primary edit file</td>
</tr>
<tr>
<td>LFILE0</td>
<td>I*2</td>
<td>Associated variable for LFILE (points to next sequential record in LFILE)</td>
</tr>
<tr>
<td>LFORW</td>
<td>I*2</td>
<td>Logical unit number connected to the forward edit file</td>
</tr>
<tr>
<td>LFORWO</td>
<td>I*2</td>
<td>Associated variable for LFORW</td>
</tr>
<tr>
<td>LBKW</td>
<td>I*2</td>
<td>Logical unit number connected to the backward edit file</td>
</tr>
<tr>
<td>LBKWO</td>
<td>I*2</td>
<td>Associated variable for LBKW</td>
</tr>
</tbody>
</table>
COMMON BLOCK: MOVCOM

PURPOSE: Contain flag indicating which sequential edit file is the source for records to be written to the forward file when the current record pointer is moved downward in the file.

VARIABLES:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPBACK</td>
<td>I*2</td>
<td>Record source indicator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 1, source of records is the primary edit file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 2, source of records is the backwards edit file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 3, at end of file</td>
</tr>
</tbody>
</table>
APPENDIX C - CAT PROJECT FILE RECORD DESCRIPTIONS

Each record within a CAT project file belongs to one of five classes of record: header record, member of the Discrepancy and Change History subfile, member of the Milestone/Deliverable History subfile, member of the Test History subfile, or member of the project file list of available records. The project file is a direct access file containing fixed length (176 byte) unformatted records.

The content of each class of record is given below. All fields described contain either binary integer (I) or alphanumeric (A) data.
**RECORD CLASS:** Header record

**CONTENT:**

<table>
<thead>
<tr>
<th>Byte Position</th>
<th>Length (Bytes)</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>2</td>
<td>1</td>
<td>Pointer to first available record in list of available records</td>
</tr>
<tr>
<td>3-4</td>
<td>2</td>
<td>1</td>
<td>Pointer to first record in the Discrepancy and Change History subfile. If zero, no data in this subfile</td>
</tr>
<tr>
<td>5-6</td>
<td>2</td>
<td>1</td>
<td>Pointer to last record in the Discrepancy and Change History subfile</td>
</tr>
<tr>
<td>7-8</td>
<td>2</td>
<td>1</td>
<td>Pointer to first record in the Milestone Deliverable History subfile. If zero, no data in this subfile</td>
</tr>
<tr>
<td>9-10</td>
<td>2</td>
<td>1</td>
<td>Pointer to last record in the Milestone Deliverable History subfile</td>
</tr>
<tr>
<td>11-12</td>
<td>2</td>
<td>1</td>
<td>Pointer to first record in the Test History subfile. If zero, no data in this subfile</td>
</tr>
<tr>
<td>13-14</td>
<td>2</td>
<td>1</td>
<td>Pointer to last record in the Test History subfile</td>
</tr>
<tr>
<td>15-176</td>
<td>162</td>
<td>-</td>
<td>(Not used)</td>
</tr>
</tbody>
</table>
RECORD CLASS: Member of the Milestone/Deliverable History subfile

CONTENT:

<table>
<thead>
<tr>
<th>Byte Position</th>
<th>Length (Bytes)</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>2</td>
<td>I</td>
<td>Pointer to next record in the subfile. If zero, this record is the last in the subfile</td>
</tr>
<tr>
<td>3-4</td>
<td>2</td>
<td>I</td>
<td>Pointer to previous record in the subfile. If zero, this record is the first in the subfile</td>
</tr>
<tr>
<td>5-6</td>
<td>2</td>
<td>I</td>
<td>Record number of this record</td>
</tr>
<tr>
<td>7-14</td>
<td>8</td>
<td>A</td>
<td>Date defined (MM/DD/YY)</td>
</tr>
<tr>
<td>15-22</td>
<td>8</td>
<td>A</td>
<td>Date scheduled (MM/DD/YY)</td>
</tr>
<tr>
<td>23-30</td>
<td>8</td>
<td>A</td>
<td>Date actual (MM/DD/YY)</td>
</tr>
<tr>
<td>31-38</td>
<td>8</td>
<td>A</td>
<td>Implementor's name</td>
</tr>
<tr>
<td>39-46</td>
<td>8</td>
<td>A</td>
<td>Type of milestone or deliverable</td>
</tr>
<tr>
<td>47-118</td>
<td>72</td>
<td>A</td>
<td>Description of event or deliverable</td>
</tr>
<tr>
<td>119-176</td>
<td>58</td>
<td>-</td>
<td>(Not used)</td>
</tr>
</tbody>
</table>

C-3
RECORD CLASS: Member of the Milestone/Deliverable History subfile

CONTENT:

<table>
<thead>
<tr>
<th>Byte Position</th>
<th>Length (Bytes)</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>2</td>
<td>I</td>
<td>Pointer to next record in the subfile. If zero, this record is last in the subfile</td>
</tr>
<tr>
<td>3-4</td>
<td>2</td>
<td>I</td>
<td>Pointer to previous record in the subfile. If zero, this record is first in the subfile</td>
</tr>
<tr>
<td>5-6</td>
<td>2</td>
<td>I</td>
<td>Record number of this record</td>
</tr>
<tr>
<td>7-14</td>
<td>8</td>
<td>A</td>
<td>Date defined (MM/DD/YY)</td>
</tr>
<tr>
<td>15-22</td>
<td>8</td>
<td>A</td>
<td>Date scheduled (MM/DD/YY)</td>
</tr>
<tr>
<td>23-30</td>
<td>8</td>
<td>A</td>
<td>Date actual (MM/DD/YY)</td>
</tr>
<tr>
<td>31-38</td>
<td>8</td>
<td>A</td>
<td>Implementor's name</td>
</tr>
<tr>
<td>39-46</td>
<td>8</td>
<td>A</td>
<td>Type of milestone or deliverable</td>
</tr>
<tr>
<td>47-118</td>
<td>72</td>
<td>A</td>
<td>Description of event or deliverable</td>
</tr>
<tr>
<td>119-176</td>
<td>58</td>
<td>-</td>
<td>(Not used)</td>
</tr>
</tbody>
</table>
RECORD CLASS: Member of the Test History subfile

CONTENT:

<table>
<thead>
<tr>
<th>Byte Position</th>
<th>Length (Bytes)</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>2</td>
<td>I</td>
<td>Pointer to next record in the subfile. If zero, this record is last in the subfile</td>
</tr>
<tr>
<td>3-4</td>
<td>2</td>
<td>I</td>
<td>Pointer to previous record in the subfile. If zero, this record is first in the subfile</td>
</tr>
<tr>
<td>5-6</td>
<td>2</td>
<td>I</td>
<td>Record number of this record</td>
</tr>
<tr>
<td>7-14</td>
<td>8</td>
<td>A</td>
<td>Test date (MM/DD/YY)</td>
</tr>
<tr>
<td>15-22</td>
<td>8</td>
<td>A</td>
<td>Test conductor's name</td>
</tr>
<tr>
<td>23-94</td>
<td>72</td>
<td>A</td>
<td>Description of test</td>
</tr>
<tr>
<td>95-126</td>
<td>32</td>
<td>A</td>
<td>Test results</td>
</tr>
<tr>
<td>127-176</td>
<td>50</td>
<td>-</td>
<td>(Not used)</td>
</tr>
</tbody>
</table>
RECORD CLASS: Member of the list of available records

CONTENT:

<table>
<thead>
<tr>
<th>Byte Position</th>
<th>Length (Bytes)</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>2</td>
<td>I</td>
<td>Pointer to next record in list of available records. If this pointer is zero, it is the last record in the list</td>
</tr>
<tr>
<td>3-176</td>
<td>174</td>
<td>-</td>
<td>(Not used)</td>
</tr>
</tbody>
</table>
APPENDIX D - CAT ERROR MESSAGES

CAT informs the user of abnormal conditions during execution by writing messages to the user's terminal. The errors detected by CAT are, in general, those encountered while opening or reading an external file, syntax errors in the editor commands entered by the user, or a failure to locate the data requested by the user.

Each message originating from CAT is presented below. The message is presented and followed by an explanation of the probable cause of the error. The messages are arranged according to the alphabetical order of the originating routines.
Message:
I/O ERROR IN CURRENT FILE
Explanation: An error occurred while reading the current line in the edit file.
Originating Subroutines: ADD, ADDP, CHG, LOC

Message:
[ILL CMD]
Explanation: The character string to append or to search for is missing or is not properly separated from the command.
Originating Subroutines: ADD, ADDP, LOC

Message:
ERROR IN READING FILE
Explanation: An I/O error occurred while reading the intervening records while moving the current line pointer in the edit file.
Originating Subroutines: BOT, CATINS, DEL, NEX, NEXP, PRI, TOP

Message:
*** nn IS AN INVALID SELECTION
Explanation: The user's response to a request for a menu selection does not correspond to one of the available choices.
Originating Subroutines: CAT, EDTSEL, PIKONE, REPORT, REPSSEL
Message:
ENCOUNTERED ERROR NUMBER nnnn

Explanation: An error occurred while opening the CAT project file which is not one of the more commonly encountered errors.

Originating Subroutine: CAT

Message:
OPEN FAILURE

Explanation: The CAT project file could not be opened.

Originating Subroutine: CAT

Message:
FILE IN USE

Explanation: Another process currently has exclusive access to the CAT project file.

Originating Subroutine: CAT

Message:
FILE NAME SPECIFICATION ERROR "a...a"

Explanation: The CAT project file name (a...a) contains a syntax error.

Originating Subroutine: CAT

Message:
ERROR IN CHANGE COMMAND

Explanation: The specification for the target string or the replacement string contained an error.

Originating Subroutine: CHG

D-3

9023
**Message:**
NO MATCH IN LINE

**Explanation:** The target string for a change command could not be found in the current line.

**Originating Subroutine:** CHG

---

**Message:**
NO DATA AVAILABLE FOR DISCREPANCY REPORT

**Explanation:** A Discrepancy and Change History report was requested and the Discrepancy and Change History subfile is empty.

**Originating Subroutines:** CRTDIS, PRTDIS

---

**Message:**
NO DATA AVAILABLE FOR MILESTONE/DELIVERABLE REPORT

**Explanation:** A Milestone/Deliverable History report was requested and the Milestone/Deliverable History subfile is empty.

**Originating Subroutines:** CRTMIL, PRTMIL

---

**Message:**
NO DATA AVAILABLE FOR TEST HISTORY REPORT

**Explanation:** A Test History report was requested and the Test History subfile is empty.

**Originating Subroutines:** CRTTST, PRTTST
Message:
UNABLE TO EDIT FILE
Explanation: Errors in opening editor scratch files prevent the editor from accessing the data.
Originating Subroutine: EDITOR

Message:
INPUT LINE NOT RECOGNIZED--TRY AGAIN
Explanation: The specified editor command was not recognized as a valid command.
Originating Subroutine: GETLIN

Message:
ERROR IN READING COMMAND LINE--PLEASE REENTER
Explanation: An I/O error occurred while reading the user's editor command.
Originating Subroutine: GETLIN

Message:
OCCURRENCE OF STRING NOT FOUND
THE CURRENT STRING IS: "a...a"
Explanation: The target string specified in a locate command could not be found in one of the following records in the edit file. The printed string is the current line in the edit file.
Originating Subroutine: LOC
Message:
ERROR IN READING FILE HEADER
Explanation: An I/O error occurred while reading the project file header record.
Originating Subroutine: OPNX

Message:
ERROR IN OPENING WORK FILES
Explanation: An error occurred while opening one of the editor scratch files.
Originating Subroutine: OPNX
The CAT system can be generated from the source code by executing a few commands. The system generation procedure for the PDP-11/70 is described in Section E.1, and for the VAX-11/780 in Section E.2.

E.1 PDP-11/70 SYSTEM GENERATION

To generate the CAT system for the PDP-11/70, only one command procedure needs to be executed. Figure E-1 is a listing of the GENCAT.CMD command procedure. This procedure compiles all CAT source code and builds the CAT task. The GENCAT.CMD procedure uses two other files to build the CAT task; the task builder command file, GENCAT.TKB and the CAT overlay description file, CAT.ODL (Figures E-2 and E-3, respectively). The PDP-11/70 CAT system is generated by executing the following command:

@GENCAT

E.2 VAX-11/780 SYSTEM GENERATION

To generate the CAT system for the VAX-11/780, only one command procedure needs to be executed. Figure E-4 is a listing of the GENCAT.COM command procedure. This procedure compiles all CAT source code, builds an object code library, and links the CAT task. The VAX-11/780 CAT system is generated by executing the following command:

@GENCAT
Figure E-1. CAT PDP-11/70 System Generation Command Procedure (1 of 2)
THE FOLLOWING IS A LISTING OF THE CAT OVERLAY STRUCTURE WHICH HAS TO BE A SEPARATE PROCEDURE

Figure E-1. CAT PDP-11/70 System Generation Command Procedure (2 of 2)
Figure E-2. CAT PDP-11/70 Task Building Command Procedure
Figure E-3. CAT PDP-11/70 Overlay Description
Figure E-4. CAT VAX-11/780 System Generation Command Procedure (1 of 2)
Figure E-4. CAT VAX-11/780 System Generation Command Procedure (2 of 2)
REFERENCES


BIBLIOGRAPHY OF SEL LITERATURE

The technical papers, memorandums, and documents listed in this bibliography are organized into two groups. The first group is composed of documents issued by the Software Engineering Laboratory (SEL) during its research and development activities. The second group includes materials that were published elsewhere but pertain to SEL activities.

SEL-Originated Documents

SEL-76-001, Proceedings From the First Summer Software Engineering Workshop, August 1976


SEL-77-002, Proceedings From the Second Summer Software Engineering Workshop, September 1977

SEL-77-003, Structured FORTRAN Preprocessor (SFORT), B. Chu and D. S. Wilson, September 1977

SEL-77-004, GSFC NAVPAK Design Specifications Languages Study, P. A. Scheffer and C. E. Velez, October 1977


†This document superseded by revised document.

SEL-78-005, Proceedings From the Third Summer Software Engineering Workshop, September 1978


SEL-78-007, Applicability of the Rayleigh Curve to the SEL Environment, T. E. Mapp, December 1978


SEL-79-005, Proceedings From the Fourth Summer Software Engineering Workshop, November 1979


SEL-80-005, A Study of the Musa Reliability Model, A. M. Miller, November 1980
SEL-80-006, *Proceedings From the Fifth Annual Software Engineering Workshop*, November 1980


This document superseded by revised document.


SEL-81-012, The Rayleigh Curve As a Model for Effort Distribution Over the Life of Medium Scale Software Systems, G. O. Picasso, December 1981

SEL-81-013, Proceedings From the Sixth Annual Software Engineering Workshop, December 1981

SEL-81-014, Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL), A. L. Green, W. J. Decker, and F. E. McGarry, September 1981


SEL-82-002, FORTRAN Static Source Code Analyzer Program (SAP) System Description, W. A. Taylor and W. J. Decker, August 1982

SEL-82-003, Software Engineering Laboratory (SEL) Data Base Reporting Software User's Guide and System Description, P. Lo, September 1982

SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982

SEL-82-005, Glossary of Software Engineering Laboratory Terms, M. G. Rohleder, December 1982

SEL-82-006, Annotated Bibliography of Software Engineering Laboratory (SEL) Literature, D. N. Card, November 1982
SEL-Related Literature


Basili, V. R., "SEL Relationships for Programming Measurement and Estimation," University of Maryland, Technical Memorandum, October 1979


‡‡ This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982.


++ This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982.

B-6

9023
Freburger, K., "A Model of the Software Life Cycle" (paper prepared for the University of Maryland, December 1978)

Higher Order Software, Inc., TR-9, A Demonstration of AXES for NAVPAK, M. Hamilton and S. Zeldin, September 1977 (also designated SEL-77-005)

Hislop, G., "Some Tests of Halstead Measures" (paper prepared for the University of Maryland, December 1978)

Lange, S. F., "A Child's Garden of Complexity Measures" (paper prepared for the University of Maryland, December 1978)

Miller, A. M., "A Survey of Several Reliability Models" (paper prepared for the University of Maryland, December 1978)

National Aeronautics and Space Administration (NASA), NASA Software Research Technology Workshop (proceedings), March 1980


Reiter, R. W., "The Nature, Organization, Measurement, and Management of Software Complexity" (paper prepared for the University of Maryland, December 1976)


B-7

9023


++This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982.