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.

SOFTWARE ENGINEERING LABORATORY (SEL) DATA BASE MAINTENANCE SYSTEM (DBAM) USER'S GUIDE AND SYSTEM DESCRIPTION

APRIL 1983

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771
AC 301 982-4955

SOFTWARE ENGINEERING LABORATORY (SEL) DATA BASE MAINTENANCE SYSTEM (DBAM) USER'S GUIDE AND SYSTEM DESCRIPTION

APRIL 1983

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

FOREWORD

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-83/6016.

The primary contributors to this document include

Pei-Shen Lo (Computer Sciences Corporation)
David Card (Computer Sciences Corporation)

Other contributors include

Frank McGarry (Goddard Space Flight Center)
Victor Church (Computer Sciences Corporation)

Single copies of this document can be obtained by writing to

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

ABSTRACT

This document provides the information necessary to understand the Software Engineering Laboratory (SEL) Data Base Maintenance System (DBAM). It describes the various software facilities of the SEL, DBAM operating procedures, and DBAM system information. Appendixes provide the relationships among DBAM components (baseline diagrams), component descriptions, overlay descriptions, indirect command file listings, file definitions, and sample data collection forms.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont'd)

Figure

LIST OF TABLES

SECTION 1 - INTRODUCTION

1.1 DOCUMENT ORGANIZATION

This document is intended to serve as a reference for operators and programmers involved in Software Engineering Laboratory (SEL) data base maintenance activities. Section 1 provides an overview of the data base and the various software facilities of the SEL. Section 2 describes in detail the operation of the SEL Data Base Maintenance System (DBAM), and Section 3 discusses the programming and implementation considerations of DBAM. Appendix A contains baseline diagrams; Appendix B, component descriptions; Appendix C, overlay description files; Appendix D, indirect command file listings; Appendix E, file definitions; and Appendix F, sample data collection forms. The reader is assumed to be generally familiar with the Digital Equipment Corporation PDP-11/70 computer and the RSX-11M operating system, the environment in which DBAM operates.

1.2 SOFTWARE ENGINEERING LABORATORY OVERVIEW

The primary objective of the SEL is to collect and maintain a data base for the use of managers and researchers. The data are used to analyze the software development process, to monitor the progress of ongoing projects, and to provide information to development project members. A significant body of software has been developed to facilitate this task. The SEL DBAM, which provides interactive facilities for the management of collected SEL data, is the principal subject of this document. Two related systems are discussed in the following subsection. They are explained in detail in References 1 and 2 and more briefly in Section 1.3.

1.3 RELATED SOFTWARE

Two DBAM-related software packages are described in this subsection. They are the Profile Reporting System and the Source Analyzer Program. Both systems are operational on a

PDP-11/70 computer under the RSX-11M operating system, although some preprocessing is required on other machines. The SEL data base is stored online to the PDP-11/70. The relationships of all software are shown in Figure 1-1.

The Profile Reporting System (PRS) provides formatted listings and summaries of the data base contents. Plots are also available for some data. PRS reports are distributed for review by managers, project leaders, and researchers. These reports facilitate the quality assurance of data collected from completed projects and generate information for personnel of active projects.

The Source Analyzer Program (SAP) enumerates occurrences of specific features in FORTRAN source code (e.g., statement types). Input is usually in the form of a tape prepared on the machine used by the development team. SAP produces a sequential file of the data generated; this file is later processed by a DBAM function (see Section 2.4.4)

1.4 DBAM/DATA BASE OVERVIEW

DBAM is organized into five functions: create, archive, restore, update, and compress. The update function is further subdivided into 13 subfunctions, each of which allows the operator to add to or modify data in a specific file type interactively. This updating may be done as part of the regular data collection procedure or in response to requests for changes by reviewers. Subfunctions may operate in any of three modes: add, change, or delete (except as noted in Section 2.3). The create, archive, restore, and compress functions are utilities affecting the entire range of data base files. They are discussed in Sections 2.5, 2.6, 2.7, and 2.8, respectively. Two frequently used special-purpose utilities, UPDSFG and UPDMFN, are included in DBAM. They are described in Sections 2.9 and 2.10, respectively.

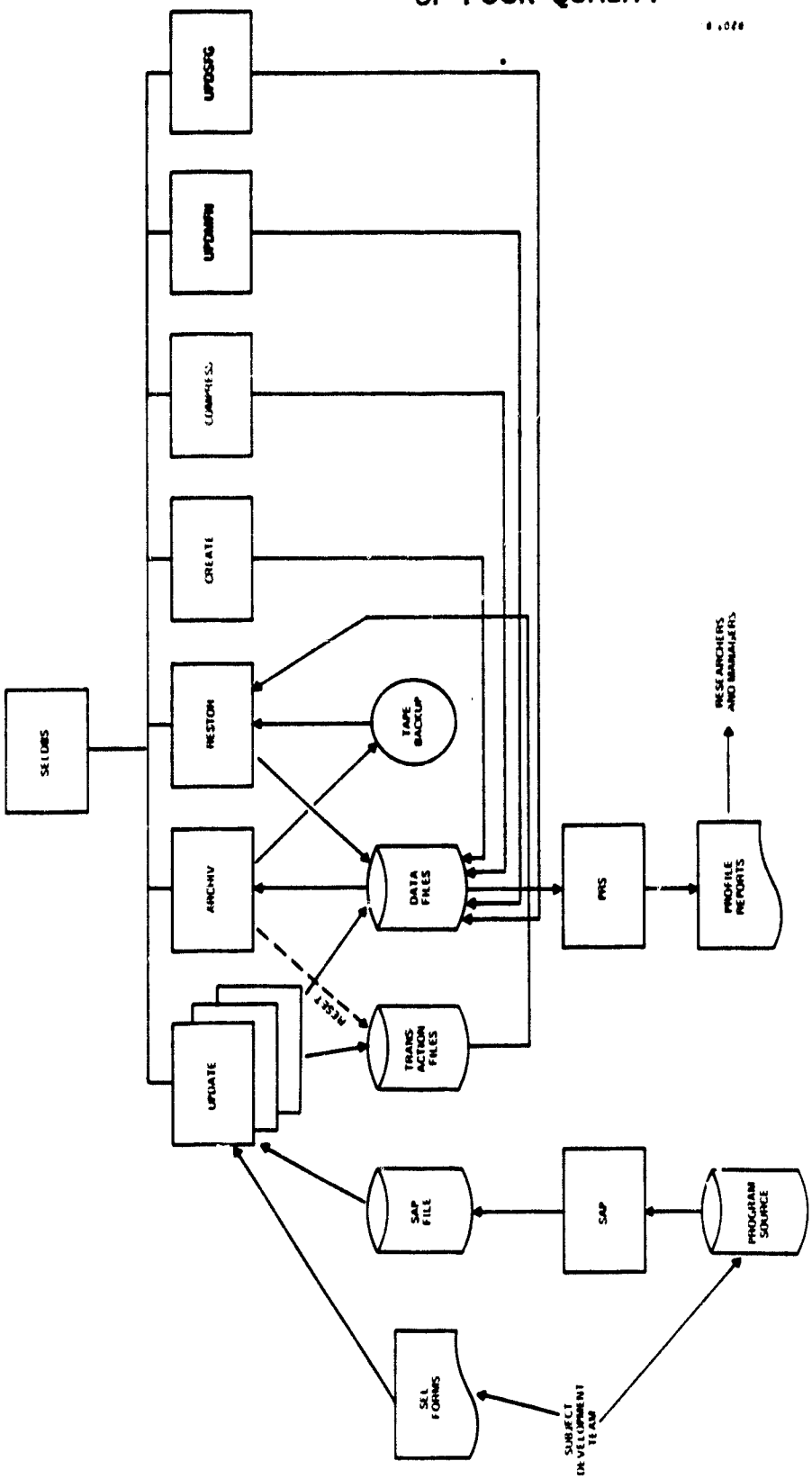


Figure 1-1. Organization of DBAM and Related Systems

Seven types of data are collected and maintained in the data base:

- Project. Project information is collected on a by-project basis. It is usually provided by the project managers.
- Form. The project leader and the developers fill out forms describing their activities; this information is referred to as "form" data.
- SAP. SAP data are generated by the Source Analyzer Program.
- Code. Many nonnumeric data fields are encoded (i.e., a numeric code is substituted for character values). The equivalence between numeric codes and character values is code information.
- Status. Status data include file names, record counts, and access dates for all data base files.
- Comments. Comments can be associated with some forms but are stored separately.
- Computer. Computer data include computer usage accounting information and source code usage information from the development machine.

The data are distributed among several files. Each file type corresponds to a record format. There may be multiple occurrences of files of a particular file type, as outlined in Section 3.2. The data collection forms and data base files are described in detail in Reference 3. Software and data base files are cataloged under separate User Identification Codes (UICs).

DBAM includes an interactive update program (editor) for most file types. The relationship between files and DBAM

programs managing those files is defined in Table 1-1. The interactive update programs are described in detail in Section 2.

Table 1-1. Relationship of DBAM Programs to Data Base Files

Type of File	Abbreviation	Type of Data	Update Program
Accounting Information	ACC	Computer	UPDACC UPDSFG
Component Information	CIF	SAP/code	UPDCIF UPDSFG UPDMFN
Comments	CMT	Comments	a
Change Report Form	CRF	Form	UPDCRF UPDSFG
Component Summary Form	CSF	Form	UPDCSF UPDSFG
Component Status Report	CSR	Form	UPDCSR UPDSFG
Subjective Evaluations Directory	DIR	Code	UPDDIR
Encoding Dictionary	ENC	Code	UPDENC
Estimated Statistics	EST	Project	UPDEST
Growth History	HIS	Computer	UPDHIS UPDSFG
Phase Dates (Header)	HDR	Project	UPDHDR
Run Analysis Form	RAF	Form	UPDRAF UPDSFG
Resource Summary Form	RSF	Form	UPDRSF UPDSFG
SAP Output	SAP	SAP	UPDCIF
Subjective Evaluations	SEF	Project	UPDSEF
File Name and Status	STS	Status	UPDSTS

[a]See Section 3.1.2.

SECTION 2 - OPERATING PROCEDURES

2.1 INTRODUCTION

The following subsections contain instructions and notes on the operation of the SEL interactive data maintenance programs. Section 2.2 lists operational restrictions, and Section 2.3 describes types of error messages. Section 2.4 describes interactive updating of the data base. The maintenance utilities are described in Sections 2.5 through 2.10. The compiter is assumed to be available and the data base already installed in these instructions.

The DBAM system is entered by logging on the PDP-11/70 with UIC [204,3] and entering "@SELDBS". The operator is then prompted for a function and responds with one of the following function codes:

Function Code	Description
UPDATE	Allows interactive update of data base files. The operator will be prompted for the file type. The file types and subsequent operating instructions are described in Section 2.4.
CREATE	Initializes all files and required header records for a new project (Section 2.5).
ARCHIV	Makes a tape copy of data base files (Section 2.6).
RESTOR	Recovers the data base from a backup tape and the transaction files (Section 2.7).
COMPRESS	Compresses data bases files (Section 2.8).
UPDSFG	Updates all status fields in a selected data base file to a particular value (Section 2.9).
UPDMFN	Updates the FORTRAN module function value for all records of a given Component Information File (Section 2.10).
STOP	Terminates execution of DBAM.

Alternatively, any program can be invoked directly by an experienced operator, as indicated in the specific program (function) descriptions.

2.2 OPERATING RESTRICTIONS

The following restrictions apply to concurrent use of DBAM file and program resources:

- No other interactive updates can be made while an update of the Encoding Dictionary is in progress.
- Information (last access date and record count) stored in the File Name and Status File may become inaccurate if other files are updated while the File Name and Status File is being updated.
- Component names cannot be validated for a project whose Component Information File is in the process of being updated.
- No other DBAM programs should be operated while a create, archive, restore, or compress operation is in progress.
- No more than one copy of each DBAM program may be in use at any time.

2.3 MESSAGES

The following four classes of messages may be displayed during execution:

- Prompt. These messages terminate with "(X)=>", where "X" is a FORTRAN format specification, or a list of allowable responses separated by "/". Data or a decision is required of the operator.

- Informational. Messages beginning with "+++" advise that some action has occurred (e.g., record-added) or that some condition exists (e.g., operation in progress). No immediate operator response is required.

- Error. Messages beginning with "***" advise that some intended action has not occurred or that an undesirable condition exits. Examples include "file already in use," "validation error," and "record not found." The operator should reexamine previous input and assumptions about the data. This message is usually followed by a prompt for a corrective action.

- Severe Errors. Messages starting with "*** RMS" indicate that a hardware, software, or operation error has occurred that may affect the program execution. These errors should be reported to the Data Base Administrator. He or she will correct the problem or refer it to the appropriate support personnel. These errors are numbered and may be interpreted by examining Table A-2 of RMS-11 MACRO-11 Programmer's Reference (Reference 4). The Data Base Administrator will determine when processing may be resumed.

2.4 INTERACTIVE UPDATE PROGRAMS

2.4.1 INTRODUCTION

The interactive update programs can be classed into five groups based on the type of files accessed and the organization of DBAM. The characteristics of these program groups are as follows:

- Update Data Base Header and Summary Files
  - Data do not come from the forms shown in Appendix F, but from other paper records
  - Only one file of each kind exists
  - Programs: UPDSTS, UPDENC, UPDEST, UPDHDR, UPDSEF, and UPDDIR
- Update Complex Data Files
  - Data come from forms shown in Appendix F
  - One file of each type exists for each project
  - May have multiple comments per record
  - Records must be displayed in several segments
  - ADD mode prompts field by field
  - Changes require two responses (field identification and new value)
  - Programs: UPDCRF and UPDCSF
- Update Simple Data Files
  - Data come from forms shown in Appendix F (except the Growth History, which is on other paper records)
  - One file of each type exists for each project
  - One or no comments per record
  - Records are small enough to be displayed in one segment

- ADD mode prompts for data in one group (on one line)
- Changes may be made with one response (field-id=value)
- Programs: UPDHIS, UPDCSR, UPDRAF, and UPDRSF

- Update Component Information File (CIF)
  - Optionally reads SAP file to update CIF records
  - Must be used to assign component names to component codes
  - Other capabilities are the same as simple data files
  - Program: UPDCIF and UPDMFN
- Update Record Status Flags
  - Updates status flag fields of entire data file
  - Changes require two responses (file identification and new status value)
  - Program: UPDSFG only

Program activities are divided into three modes: ADD new records, CHANGE existing records,[1] and DELETE existing records. Program activity proceeds in three hierarchical steps or levels: select mode, identify record, and execute mode activity. Mode selection is accomplished by specifying the first letter of the mode (e.g., A) in response to the

[1]Changes to an existing record are made by first identifying the field to be changed. Each field is tagged with a two-character code. This code is indicated in the field title in the data display by "+" symbols next to the characters comprising the code. Codes are read top to bottom, then left to right. Programs vary in whether or not the new value may be entered on the same line as the field code (see the following subsections).

prompt for mode. The programs automatically cycle between the last two steps until the operator intervenes.[1]

The last access date and record count on the record for the accessed file (in the File Name and Status File) are automatically updated. A sequential transaction file is maintained for each form type. The afterimage of each updated record is copied to it with a transaction code (A = ADD, C = CHANGE, D = DELETE), and the current date.

The individual programs are described in detail in Sections 2.4.2 through 2.4.14. Figure 2-1 shows the general outline of activity in the interactive update programs.

---

[1] The control-Z character may be entered in response to any prompt to terminate the program. Entering "/*" will cause a return to the program driver (which prompts for mode).

Figure 2-1. Generalized Flow Diagram of Interactive Update Program Operation (Some activities are omitted in some programs.) (1 of 2)

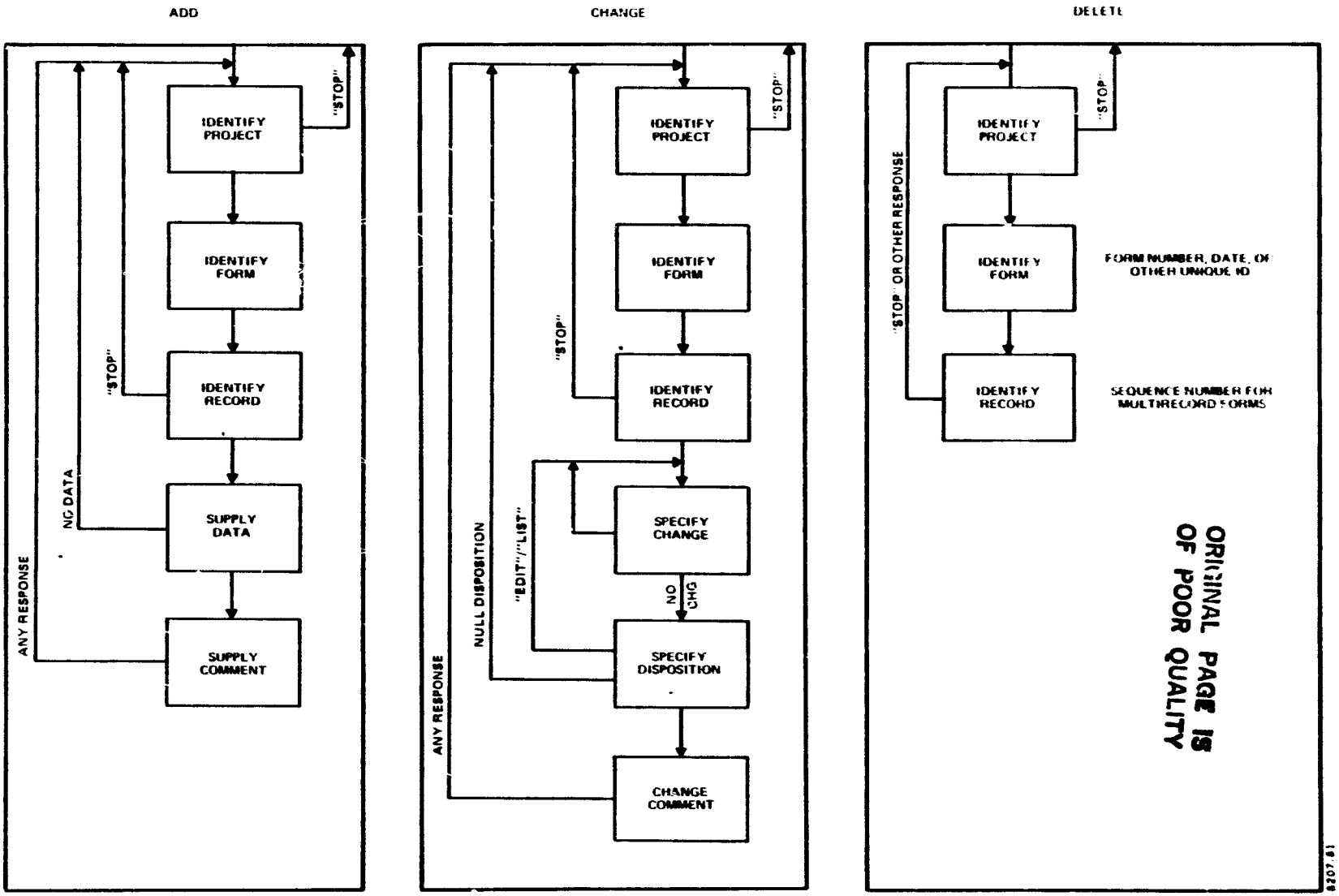


Figure 2-1. Generalized Flow Diagram of Interactive Update Program Operation (Some activities are omitted in some programs.) (2 of 2)

2.4.2 UPDATE COMPONENT SUMMARY FORM (UPDCSF) PROGRAM DESCRIPTION

Program Function. UPDCSF supports the interactive addition, deletion, and editing of records in the Component Summary Form (CSF) File.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
CSF File (by project)	Read, write, update, delete
CIF File (by project)	Read only
Comments (CMT) File (by project)	Read, write, delete
CSF Transaction File	Write (append) only
File Name and Status File	Read, update

Program Invocation. Execution of UPDCSF may be initiated by either logging on with UIC [204,3] and entering "@UPDCSF" or logging on with another UIC and entering "@[204,3]UPDCSF".

Program Operation. Program activities are divided into three modes: ADD, CHANGE, and DELETE. Within these functions, other activities are divided into hierarchical levels. Program levels are summarized as follows:

Level	Description
1	Select mode (ADD/CHANGE/DELETE)
2	Identify CSF record (project, form number)
3	Perform activity (mode)

These activities are described below.

- Select Mode (Level 1). One of three modes may be selected: ADD new records, CHANGE existing records, or DELETE existing records. Mode selection can only be made at this level.

- Identify-Record (Level 2). A valid project and form number must be supplied in response to the prompts. Entering "S" (stop) returns the program to the "select mode" prompt. A carriage return, in response to the "identify project" prompt, retains the previous response to that prompt.

- ADD Mode (Level 3). The message "duplicate record" appears if the form number supplied is already defined for this project. Otherwise, the operator is prompted for each field in turn. Prompts are also made for corrections to any input errors that have been detected. A carriage return may be entered for any field for which a value is not available. The completed record is then displayed. The data display should be carefully examined, and any discrepancies from what was intended should be noted. The record can be corrected later in the CHANGE mode. The prompt "enter x" appears next. The operator should respond with an "X" if the record is acceptable. If the data are marked acceptable, the operator is prompted for comments, and the record and comments are added to the data base. The program then returns to the "identify record" prompts (level 2).

- CHANGE Mode (Level 3). The message "record not found" is displayed if the specified record is not defined for this project. Otherwise, the record is displayed in five segments. The values in the current display may be changed by entering the code corresponding to the field in response to the prompt "enter code for field." The operator is then prompted for a new value for that field. Entering a carriage return in response to the "enter code" prompt causes the next record segment to be displayed, or, in the

case of the last segment, causes a prompt for "disposition" to appear. Four responses are possible:

- LIST--List the current record; prompt again for disposition
- EDIT--Resume editing (as in paragraph above)
- EXEC--Update the record on the data base; exit
- [Carriage return only]--Make no changes to the original record; exit

The last two responses return the program to the CSF "identify record" prompts (level 2) after completing the specified action.

DELETE Mode (Level 3). The message "not found" is displayed if the record is not defined for this project. Otherwise, the record is deleted and an appropriate message is displayed. The program returns to the "identify record" prompts (level 2).

End of Session (Level 1). Specifying "S" (stop) in response to the prompt for function terminates the UPDCSF session. A summary of transactions is displayed.

2.4.3 UPDATE CHANGE REPORT FORM (UPDCRF) PROGRAM DESCRIPTION

Program Function. UPDCRF supports the interactive addition, deletion, and editing of records in the Change Report Form (CRF) Files.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
CRF File (by project)	Read, write, update, delete
CIF File (by project)	Read only
CMT File (by project)	Read, write, delete
CRF Transaction File	Write (append) only
File Name and Status File	Read, update

Program Invocation. Execution of UPDCRF may be initiated by either logging on with UIC [204,3] and entering "@UPDCRF" or logging on with another UIC and entering "@[204,3]UPDCRF".

Program Operation. Proceeds as in UPDCSF (Section 2.4.2).

2.4.4 UPDATE COMPONENT INFORMATION FILE (UPDCIF) PROGRAM DESCRIPTION

Program Function. UPDCIF supports the interactive addition, deletion, and editing of the CIFs. Optionally, CIF records can be updated from the SAP output.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
CIF File (by project)	Read, write, update, delete
SAP Output File	Read only
CIF Transaction File	Write (append) only
File Name and Status File	Read, update
Error message data set	Write only

Program Invocation. Execution of UPDCIF may be initiated by either logging on with UIC [204,3] and entering "@UPDCIF" or logging on with another UIC and entering "@[204,3]UPDCIF".

CIF Data Structure. CIF records may be divided into two parts: an area that is filled with values supplied by SAP and an area that is filled with values collected elsewhere (PANVALET level, function, origin, etc.). The latter data are displayed first, on a single line (segment). The SAP data are displayed on two lines (subsequent segments).

Program Operation. Program activities are divided into four modes: ADD, CHANGE, and DELETE and READ from the SAP Output File. The READ mode is fully automatic. Records from the SAP Output File are matched (by component name) with CIF records; then the SAP data are copied into the CIF record. The record on the SAP Output File is then deleted. No checks are made for duplicate names or records in the SAP Output File or for already filled fields in the CIF records. The last SAP record read (for a component) supersedes all

previous data in SAP-supplied fields. Unmatched SAP records remain in the SAP Output File for later processing. Component names in SAP records that were not matched with a CIF record are listed on an external data set as well as displayed on the terminal. The operator is offered an opportunity to list this data set at the end of the session. Program activities in the other modes are divided into recurring layers:

Level	Description
1	Select mode (ADD/CHANGE/DELETE/READ) (There are no further levels for read mode)
2	Identify project
3	Identify CIF record (component name)
4	Perform mode activity (supply values)

These activities are described below.

- Select Mode (Level 1). Four modes are available: ADD new records, CHANGE existing records, DELETE existing records, and READ (and update existing data) from SAP Output File. Mode selection can only be made at this level.

- Identify Project (Level 2). In response to a prompt, the operator must specify the project name to identify and open the proper CIF. The operator may respond to the "identify project" prompt with a project name, an "S," or a carriage return. "S" terminates the current mode and returns the program to the "select mode" prompt (level 1). Entering a carriage return causes the last response (project name) to this prompt to be reused. Any other response is verified as a project name by comparison with the Encoding Dictionary.

- Identify Record (Level 3). The operator must explicitly specify the component name in response to the "identify record" prompt (level 2). Furthermore, a CIF

record must already exist for that component if the mode is CHANGE or DELETE; a CIF record for that component cannot already exist if the mode is ADD. An error returns the program to the "identify project" prompt (level 2).

- ADD Mode (Level 4). Three different displays of data field titles are presented. The operator may supply values for some, none, or all the fields by entering a value directly below the field to be filled. The data fields identified in the last two displays can be filled from the SAP Output File by the READ function. The operator usually skips these fields when adding new records (by entering a carriage return only). Input values are validated immediately, and the operator is offered the opportunity to correct any errors detected in this line. The record is then added to the data base, and the program activity transfers to the "identify project" prompt (level 2).

- CHANGE Mode (Level 4). Each of the three segments of data is displayed in turn. After each display, the operator is prompted for a field to change and a new value. The field and value are validated immediately, and the operator is offered an opportunity to correct any errors detected. Entering a carriage return only advances the program to the next segment. After the last segment, the operator is prompted for the disposition of the revised record. Entering "X" (execute) causes the corresponding data base record to be replaced with the revised record. Entering "L" (list) transfers program activity to the beginning of the change sequence (including displays) for this record. A null response (carriage return) aborts the transaction; the data base is not changed. Following an "X" or null response, program activity transfers to the "identify project" prompt (level 2).

- DELETE Mode (Level 4). Identification of a valid component name results in the deletion of the related record. Program activity returns to the "identify project" prompt (level 2).

- End of Session (Level 1). Specifying "S" (stop) in response to the "select mode" prompt (level 1) terminates program execution. A summary of transactions is displayed.

2.4.5 UPDATE COMPONENT STATUS REPORT (UPDCSR) PROGRAM DESCRIPTION

Program Function. UPDCSR supports the interactive addition, deletion, and editing of the Component Status Report (CSR) File.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
CSR File (by project)	Read, write, update, delete
CIF File (by project)	Read only
CSR Transaction File	Write (append) only
File Name and Status File	Read, update

Program Invocation. Execution of UPDCSR may be initiated by either logging on with UIC [204,3] and entering "@UPDCSR" or logging on with another UIC and entering "@[204,3]UPDCSR".

CSR Data Structure. Any given CSR form may generate several CSR records. Each record corresponds to a horizontal line of data from the CSR form. The fields of a CSR record are divided into the following three areas for access and display purposes:

- CSR form-identification area--Project, form, programmer, phase, and date are fields common to all CSR records derived from the same form. All fields are required.
- CSR component-status area--Sequence number; component; and design, code, and test hours are unique within the group of records making up a form.
- CSR other-activity area--Other-activity name and hours may be filled instead of the component-status area. The component-status and other-activity areas cannot both be filled.

Program Operation. Program activities are divided into three modes: ADD, CHANGE, and DELETE. Within these modes, other activities are organized into hierarchical levels. Activity in a given level is terminated by entering "S" (stop) at the entry point for that level, or by providing the program with the necessary information to proceed. (An exception is the lowest level, which may be exited only by entering carriage returns in response to all prompts.) "S" always returns the program to the next higher level. The program levels are summarized as follows:

Level	Description
1	Select mode (ADD/CHANGE/DELETE)
2	Identify CSR form (project, programmer, date, form number)
3	Identify CSR record (sequence number)
4	Perform mode activity

These activities are described below.

- Select Mode (Level 1). Three modes are available: ADD new records, CHANGE existing records, and DELETE existing records. Mode selection can only be made at this level.

- Identify Form (All Modes, Level 2). All fields must be supplied in the ADD mode. Only project and form number need be supplied for the CHANGE and DELETE modes. Entering "S" returns the program to the "select mode" prompt (level 1). A carriage return may be entered to retain the last project entered as the current project.

- ADD Mode (Levels 3 and 4). The user may enter a specific sequence number or enter a carriage return to use automatic sequencing option. Automatic sequencing increments the sequence number by 1 or sets it to 1 if no sequence number has previously been entered. If the sequence number has already been used with this form number, a "duplicate record" message is displayed. "S" may be entered

to return to the "form identification" prompts. Responding "A" will bypass subsequent number prompts and data titles (for fast data entry).

Identification of a valid sequence number produces a display of the component-status area field titles. The operator enters values (where available) for the fields below the dashed line. The TAB key may be used to skip blank fields. Entering a carriage return instead of a component name at this point produces a prompt for the other-activity area. The operator responds to this by entering the other-activity name (up to eight characters) and the hours separated by an equal sign (e.g., "TRAVEL=123.4"). Entering a carriage return at this point aborts the transaction and returns the program to the "identify record" (sequence number) prompt. The operator is prompted for corrections to any input errors detected.

- CHANGE Mode (Levels 3 and 4). The operator may enter a specific sequence number or "S" to stop. A carriage return results in the default sequence number "Ø1" being used. If the indicated sequence number is not found for the specified form number, the message "record not found" is displayed. Responding "A" causes all records of this form to be displayed. (The program returns to the "identify form" prompts (level 2).)

Once an existing record is identified and retrieved, it is displayed. The operator is then prompted for the field to be changed and the value to be used. These should be entered in the following format: "code=value," where code is the two-letter code for the field (indicated by "+" in the display titles) and value is the component name (if appropriate) or a decimal number (for hours). The other-activity area may be changed by specifying "OA=name=value", where name is the activity name (up to eight characters) and value

is a decimal number. Note that inserting data in the other-activity area of a record that contains information in the component-status area results in the loss of the component-status data. The operator is prompted for corrections to any input errors detected. Entering a carriage return in response to the prompt for field and value terminates editing of that record.

The operator is then prompted for the disposition of the record. Four responses are possible:

- LIST--List the current record: prompt for disposition appears again
- EDIT--Reinitiate editing (as in paragraph above)
- EXEC--Update the record on the data base; exit
- [Carriage return only]--Make no changes to the original record; exit

The first two responses return the program to the "disposition" prompt. The last two responses return the program to the CSR "identify form" prompts (level 2) after completing the specified action.

- DELETE Mode (Level 3). The operator may enter a specific sequence number or "S" for stop. A carriage return results in the default sequence number "Ø1" being used. If the indicated sequence number is not found for the specified form number, the message "record not found" is displayed. If the record is located, the message "deleted" appears (the record has just been deleted). Responding "A" causes all contiguous records of this form to be deleted. The program returns to the "identify form" prompts (level 2).

- End of Session (Level 1). Specifying "S" in response to the prompt for mode terminates the UPDCSR session. A summary of transactions is displayed.

2.4.6 UPDATE RUN ANALYSIS FORM (UPDRAF) PROGRAM DESCRIPTION

Program Function. UPDRAF supports the interactive addition, deletion, and editing of the Run Analysis Form (RAF) Files.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
RAF File (by project)	Read, write, update, delete
CIF File (by project)	Read only
CMT File (by project)	Read, write, delete
RAF Transaction File	Write (append) only
File Name and Status File	Read, update

Program Invocation. Execution of UPDRAF may be initiated by either logging on with UIC [204,3] and entering "@UPDRAF" or logging on with another UIC and entering "@[204,3]UPDRAF".

RAF Data Structure. Any given RAF may generate several RAF records. Each record corresponds to one horizontal line of data from the form. The fields of an RAF record are divided into three areas for access and display purposes:

- RAF identification area--Project, form number, programmer, and computer are fields common to all records of the same form. All fields are required.
- RAF data area--Other fields define the specific run.
- Comment area--Maintained in a separate file.

Program Operation. Proceeds as in UPDCSR (Section 2.4.5).

2.4.7 UPDATE RESOURCE SUMMARY FORM (UPDRSF) PROGRAM DESCRIPTION

Program Function. UPDRSF supports the interactive addition, deletion, and editing of records in the Resource Summary Form (RSF) Files.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
RSF File (by project)	Read, write, update, delete
RSF Transaction File	Write (append) only
File Name and Status File	Read, update

Program Invocation. Execution of UPDRSF may be initiated by either logging on with UIC [204,3] and entering "@UPDRSF" or logging on with another UIC and entering "@[204,3]UPDRSF".

RSF Data Structure. Any given RSF form may generate several RSF records. Each record corresponds to one horizontal line of data from the form. The fields of an RSF record are divided into two areas for access and display purposes:

- RSF identification area--Form number and project are common to all records of a form. Both fields are required.
- RSF data area--Other fields define a resource and its utilization.

Program Operation. Proceeds as in UPDCSR (Section 2.4.5).

2.4.8 UPDATE GROWTH HISTORY FILE (UPDHIS) PROGRAM DESCRIPTION

Program Function. UPDHIS supports the interactive addition, deletion, and editing of the Growth History (HIS) Files.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
HIS File (by project)	Read, write, update, delete
HIS Transaction File	Write (append) only
File Name and Status File	Read, update

Program Invocation. Execution of UPDHIS may be initiated by either logging on with UIC [204,3] and entering "@UPDHIS" or logging on with another UIC and entering "@[204,3]UPDHIS".

Program Operation. Program activities are divided into three modes: ADD, CHANGE, and DELETE. Activities within these modes are divided into hierarchical levels. Activity on a level is terminated by entering "S" in response to the first prompt at that level. Program levels are as follows:

Level	Description
1	Select mode (ADD/CHANGE/DELETE)
2	Identify project
3	Identify date of data
4	Perform mode activity

These activities are described below.

- Select Mode (Level 1). Three modes are available: ADD new records, CHANGE existing records, and DELETE existing records. Mode selection can only be made at this level.

- Identify Project (Level 2). The operator must supply a project name or enter "S" (stop). A null response causes the previous project name to be reused. "S" returns program activity to the "select mode" prompt.

- Identify Date of Data (Level 3). The date must be supplied. The add mode accepts the date from the same line as the rest of the data. DELETE and CHANGE modes require that the date be entered in response to the specific prompt for that field.

- ADD Mode (Level 4). A line of data titles is displayed. Values for these fields should be entered in a single line below the titles. Errors are then reported, and the operator is provided an opportunity to correct the error. No record is added to the data base without a complete and correct date. The operator continues to be prompted for data in this manner until a null date or line is entered. The program then transfers to the "identify project" prompt (level 2).

- CHANGE Mode (Level 4). The data for the specified date are displayed, if found; otherwise, a message is displayed and the program transfers to level 2. The operator is prompted for the field to be changed and the replacement value. Responses should take the form "code=value," where code is the two-character code for the field and value is an acceptable value for that field. Errors are reported immediately, and the operator is given an opportunity to correct any error. Entering a null response results in a prompt for record "disposition." Entering "X" (execute) causes the changed record to replace the old record in the data base. "L" (list) transfers activity back to the display of data. A null or blank response causes the transaction to be ignored (no change will be made to the data base). After null or "X", the program transfers to the "identify project" prompt (level 2).

- DELETE Mode (Level 4). The record for the specified date is deleted if found; otherwise, a message is displayed. The program returns to the "identify project" prompt (level 2).

- End of Session (Level 1). Specifying "S" in response to the prompt for mode terminates the UPDHIS session. A summary of transactions is displayed.

2.4.9 UPDATE FILE NAME AND STATUS FILE (UPDSTS) PROGRAM DESCRIPTION

Program Function. UPDSTS supports interactive deletion and editing of existing records in the File Name and Status (STS) File. (Records may only be added to the File Name and Status File through the CREATE utility.)

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
File Name and Status File	Read, update, delete

Program Invocation. Execution of UPDSTS may be initiated by either logging on with UIC [204,3] and entering "@UPDSTS" or logging on with another UIC and entering "@[204,3]UPDSTS".

Program Operation. The operator is prompted for the project name and file type (qualifier). The record for this file is retrieved from the data base and displayed. Then the operator is prompted for the field to be changed and the value to use. Responses should take the form "code=value," where code is the two-character code for the field and value is an acceptable value for the field. Errors are reported, and the operator is given an opportunity to provide corrections. Entering a null or blank response results in a prompt for disposition. Allowed responses are "L", list and resume editing; "X", complete the transaction by updating the data base; "D", complete the transaction by deleting the record from the data base; and "S", discard this transaction. Exercising any of the last three options (X, D, S) returns the program to the "project name" prompt. Entering a null response for the project name causes the next sequential record in the file to be retrieved. Entering "S" terminates the program.

2.4.10 UPDATE ENCODING DICTIONARY (UPDENC) PROGRAM DESCRIPTION

Program Function. UPDENC supports the interactive addition, deletion, and editing of records on the Encoding Dictionary (ENC).

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read, write, update, delete
File Name and Status File	Read, update

Program Invocation. Execution of UPDENC may be initiated by either logging on with UIC [204,3] and entering "@UPDENC" or logging on with another UIC and entering "@[204,3]UPDENC".

Program Operation. The operator must first identify the mode/function in response to that prompt. "A" for add, "C" for change, and "S" for stop (terminate program) are allowed responses. The operator is prompted for the code type (category of codes) and item name (eight-character abbreviation). The operator is prompted for a description of the entry in the ADD mode. The previous description is displayed, and the operator is prompted for a replacement in the CHANGE mode. Entering "D" in response to this prompt causes the record to be deleted. A null or blank response leaves the record unchanged. Any other value is accepted as the new description. (Because of the nature of the Encoding Dictionary, only the description is a changeable field.) The program then returns to the "code type" prompt.

2.4.11 UPDATE ESTIMATED STATISTICS FILE (UPDEST) PROGRAM DESCRIPTION

Program Function. UPDEST supports the interactive addition, deletion, and editing of the Estimated Statistics (EST) File.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
Estimated Statistics File	Read, write, update, delete

Program Invocation. Execution of UPDEST may be initiated by either logging on with UIC [204,3] and entering "@UPDEST" or logging on with another UIC and entering "@[204,3]UPDEST".

Program Operation. Program activities are divided into three modes: ADD, CHANGE, and DELETE. Activities within these modes are divided into three hierarchical levels. Program levels are summarized below.

Level	Description
1	Select mode (ADD/CHANGE/DELETE)
2	Identify project
3	Perform mode activity

- Select Mode (Level 1). Three modes are available: ADD new records, CHANGE existing records, and DELETE existing records. Mode selection can only be made at this level. "S" (stop) displays a summary of transactions, then terminates the program.

- Identify Project (Level 2, All Modes). The operator must supply a project name or enter "S". "S" returns program activity to the "select mode" prompt (level 1).

- ADD Mode (Level 3). Several sets of data titles are displayed consecutively. Each is followed by a prompt for data. Values should be entered in a single line immediately below the field to which they are to be applied.

The operator is notified of any errors detected and is given an opportunity to correct them. The newly constructed record is displayed after all data have been entered. The program then returns to the "identify project" prompt (level 2).

- CHANGE Mode (Level 3). The data for the specified project are displayed, if found; otherwise, a message appears and the program returns to the "identify project" prompt (level 2). The operator is prompted for the field to be changed and the value to use. Responses should take the form "code=value," where code is the two-character field code and value is an acceptable value for this field. Entering a null or blank response results in a prompt for 'disposition." Allowed responses are "L", list and resume editing; "X", complete the transaction by updating the data base; and null (blank) response, discard this transaction. Exercising either of the last two options returns the program to the "identify project" prompt (level 2).

- DELETE Mode (Level 3). The record for the specified project is deleted, if found; otherwise, a message is displayed. The program returns to the "identify project" prompt (level 2).

2.4.12 UPDATE SUBJECTIVE EVALUATIONS FILE (UPDSEF) PROGRAM DESCRIPTION

Program Function. UPDSEF supports the interactive addition, deletion, and editing of records on the Subjective Evaluations File (SEF).

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
Subjective Evaluations Directory File	Ready only
Subjective Evaluations File	Read, write, update, delete

Program Invocation. Execution of UPDSEF may be initiated by either logging on with UIC [204,3] and entering "@UPDSEF" or logging on with another UIC and entering "@[204,3]UPDSEF".

Program Operation. Program activities are divided into three modes: ADD, CHANGE, and DELETE. Activities within these modes are divided into three hierarchical levels. Program levels are summarized below.

Level	Description
1	Select mode (ADD CHANGE DELETE)
2	Identify project
3	Perform mode activity

- Select mode (Level 1). Three modes are available: ADD new records, CHANGE existing records, and DELETE existing records. Mode selection can only be made at this level. "S" (stop) displays a summary of transactions, then terminates the program.

- Identify Project (Level 2, All Modes). The operator must supply a project name or enter "S". "S" returns program activity to the "select mode" prompt (level 1).

- ADD Mode (Level 3). Titles of measures of different categories are displayed consecutively. Each category may contain two or more lines. Each title line display is followed by a prompt for data. Values should be entered in a single line immediately below the field to which they are to be applied. The operator is notified of any errors detected and is given an opportunity to correct them. After each record is entered into the SEF, the operator is notified by the message "+++ DATA RECORD XX ADDED," where XX is the data record sequence number. After all seven records of a given project have been entered, the program returns to the "identify project" prompt (level 2).

- CHANGE Mode (Level 3). After entering the CHANGE mode, the program first prompts for the name of the category of measure to be changed. The operator must supply a category name that is valid (see Appendix A of Reference 3). Otherwise, the message "RECORD NOT FOUND" is displayed and the program returns to the prompt for the name of the category of measure to be changed. If the category is valid, the data for the specified category of a given project are displayed. The operator is then prompted for the name of the field to be changed. After responding with a code name, the operator is prompted for a new value. The operator should respond with an acceptable new value for this field. Allowed responses for the field to be changed, besides the actual field name, are "L", list and resume editing; "X", complete the transaction by updating the SEF; and "S", discard this transaction and return to the prompt for the name of the category of measure to be changed.

- DELETE Mode (Level 3). All seven records of a specified project are deleted; otherwise, a message "***DATA RECORD XX NOT FOUND" (where XX is the data record sequence number) is displayed. The program returns to the "identify project" prompt (level 2).

2.4.13 UPDATE SUBJECTIVE EVALUATIONS DIRECTORY (UPDDIR) FILE PROGRAM DESCRIPTION

Program Function. UPDDIR supports the interactive addition, deletion, and editing of records on the Subjective Evaluations Directory (DIR) File.

Data Sets Accessed. The following data sets are accessed:

Date Set	Operation
Subjective Evaluations Directory File	Read, write, update, delete
File Name and Status File	Read, update

Program Invocation. Execution of UPDDIR may be initiated by either logging on with UIC [204,3] and entering "@UPDDIR" or logging on with another UIC and entering "@[204,3]UPDDIR".

Program Operation. The operator must first identify the mode/function in response to that prompt. "A" for add, "C" for change, "D" for delete, and "S" for stop (terminate program) are allowed responses. In all modes (except STOP), the operator is prompted for the code-type. (With an entry of "S", the program returns to the "mode/function" prompt.) In the ADD mode, the operator is prompted for a description of the entry. In the CHANGE mode, the existing record is displayed and the operator is prompted for replacement values. In the DELETE mode, the "code-type" entered by the operator is deleted. In all modes, completion of the function is noted, and the program returns to that function's "code-type" prompt. With an entry of "S" to the "mode/function" prompt, the STATUS.HDR is updated accordingly and a count of the functions performed is displayed prior to the program ending.

2.4.14 UPDATE PHASE DATES FILE (UPDHDR) PROGRAM DESCRIPTION

Program Function. UPDHDR supports the interactive addition, deletion, and editing of records on the Phase Dates (HDR) File.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
Phase Dates (Header) File	Read, write, update, delete

Program Invocation. Execution of UPDHDR may be initiated by either logging on with UIC [204,3] and entering "@UPDHDR" or logging on with another UIC and entering "@[204,3]UPDHDR".

Program Operation. Proceeds as in UPDEST (Section 2.4.11).

2.5 CREATE NEW PROJECT FILES (CREATE) UTILITY DESCRIPTION

Program Function. CREATE initializes required records in the Encoding Dictionary and File Name and Status File and creates data base files for a new project.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Indirect command files	Read only
Temporary data sets	Read, write, delete
Encoding Dictionary	Read, write
File Name and Status File	Read, write
CIF File (by project)	Define only (by DEF utility)
CMT File (by project)	Define only (by DEF utility)
CRF File (by project)	Define only (by DEF utility)
CSF File (by project)	Define only (by DEF utility)
CSR File (by project)	Define only (by DEF utility)
HIS File (by project)	Define only (by DEF utility)
RAF File (by project)	Define only (by DEF utility)
RSF File (by project)	Define only (by DEF utility)

Program Invocation. Execution of CREATE may be initiated by either logging on with UIC [204,3] and entering "@CREATE" or logging on with another UIC and entering "@[204,3]CREATE".

Program Operation. The program first prompts for the project name. It then attempts to add the project name to the Encoding Dictionary and to create records in the File Name and Status File for the project files that are to be created. Any errors are reported to the operator. The operator must decide, on the basis of the messages received, whether or not to proceed with the actual file creation. Any reported error is sufficient cause to terminate CREATE. If the operator chooses to terminate CREATE, care should be taken to delete the records that have just been added to the Encoding Dictionary and the File Name and Status File using UPDENC and UPDSTS, respectively.

2.6 ARCHIVE DATA BASE FILES (ARCHIV) UTILITY DESCRIPTION

Program Function. ARCHIV makes a tape copy of the entire data base or of a list of files specified by the user.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
All or part of data base	Read only (by BCK utility)
File Name and Status File	Read, update, write
Encoding Dictionary	Read only
Magnetic tape	Write only (by BCK utility)

Program Invocation. Execution of ARCHIV may be initiated by either logging on with UIC [204,3] and entering "@ARCHIV" or logging on with another UIC and entering "@[204,3]ARCHIV".

Program Operation. The operator is first reminded to ready the tape from the console and specify the necessary command format. When this has been done, the operator should respond to the prompt for files to be saved by entering "ALL" or "LIST". The tape identification is then requested. This should be the same alphanumeric string used in the INI/MOU commands.[1] If "ALL" was specified, all data sets under UIC [204,1] are copied to the tape. Otherwise, the program prompts for file names until a null response is entered. These files are verified against the File Name and Status File for accuracy, the last access date is reset, and the files are copied to tape. The File Name and Status File is updated before any data sets are copied under the "ALL" option.

[1] See RSX-11 Command Language Manual (Reference 5) for details about the INI and MOU commands.

2.7 RESTORE DATA BASE FILES (RESTOR) UTILITY DESCRIPTION

Program Function. RESTOR recovers all or a list of data sets stored on a tape and reestablishes them under UIC [204,1].

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Magnetic tape	Read only (by RST utility)
Transaction Files	Read only
Recovered data base	Read, write, update, delete, write (by RST utility)
Encoding Dictionary	Read only
File Names and Status File	Write only

Program Invocation. Execution of RESTOR may be initiated by either logging on with UIC [204,3] and entering "@RESTOR" or logging on with another UIC and entering "@[204,3]RESTOR".

Program Operation. The operator is first reminded to ready the tape from the console and specify the necessary command format. When this has been done, the operator should respond to the prompt for "files to recover" by entering "ALL" or "LIST". Specifying "LIST" causes the program to prompt for and recover files by name. Specifying "ALL" recovers the entire contents of the tape. The operator is then prompted for the versions of the Transaction Files to be merged with the data base. The operator should determine these files before beginning RESTOR by selecting the last versions in existence before the date of the archive (backup) tape copy from which the recovery (restore) is being made.

2.8 COMPRESS DATA BASE FILES (COMPRESS) UTILITY DESCRIPTION

Program Function. COMPRESS compresses user-specified data base files. It first defines new versions of the files and then copies all records from the old versions to the new versions of the files.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Desired data base files	Read, write
File Name and Status File	Read only

Program Invocation. Execution of COMPRESS may be initiated by logging on with UIC [204,3] and entering "@COMPRESS" or logging on with another UIC and entering "@[204,3]COMPRESS".

Program Operation. After "@COMPRESS" is entered, the user is given the prompt F* COMPRESSF[S]: to which the user can enter the name of the data base file to be compressed. (The "[S]" indicates that a character string is expected.) Wildcards (*) may be used to specify all projects, all file types, or both. For example, F* COMPRESSF[S]:*.RAF would cause all Run Analysis Form Files to be compressed.

Some file characteristics, such as the file-extension quantity, vary depending on whether the file is considered "active" or "inactive." This is indicated by the current data base file version number. Any file with a version number less than octal 100 is considered active; all other files are inactive. (Most data base files are inactive.) Version numbers may be changed if necessary by using the PDP PIP command.

HELP may be entered in response to the COMPRESS prompt for user help information.

2.9 UPDATE STATUS FLAG (UPDSFG) UTILITY DESCRIPTION

Program Function. UPDSFG updates the status flag field of every record in a selected data base file to a particular value. UPDSFG cannot be used to update the status flags in header files.

Data Sets Accessed: The following data sets are accessed:

Data Set	Operation
Desired data base file	Read, update
Encoding Dictionary	Read only
Transaction Files	Write (append) only

Program Invocation. Execution of UPDSFG may be initiated by either logging on with UIC [204,3] and entering "@UPDSFG" or logging on with another UIC and entering "@[204,3]UPDSFG".

Program Operation. After the UPDSFG utility is entered, the operator is prompted first for the project name. The operator must supply a valid project name or enter "S" (stop). A null response causes the previous project name to be reused. "S" stops program execution. If an invalid project name is given, the program displays an error message on the terminal and prompts for the project name again. If a valid project name is entered, the program then prompts for the file type. The operator must enter a valid file type or "S" (stop). "S" or a null response returns program activity to the "project name" prompt. "ACC", "CIF", "CMT", "CRF", "CSF", "CSR", "HIS", "RAF", and "RSF" are valid file types. If an invalid file type is entered, an error message is displayed on the terminal and the program prompts the operator for the file type again.

If both project name and file type are valid, help information on the allowed status values is displayed, and the operator is prompted for the new status value. The operator must respond with one of the allowed values; otherwise, an

error message is displayed and the prompt for the status value is given again. The following status values are allowed:

Status Value	Description
1	Data unchecked
2	Data hand-checked
3	Data verified by usage

After the program updates the status flag for all records in the selected file to the new value, a summary report is printed on the terminal. The information displayed contains the file name, total number of records processed, and the new status value. In addition, the old status values in the file and the number of records having each old status value are displayed. After the summary report is displayed, the program returns to the prompt for the project name.

2.10 UPDATE THE MODULE FUNCTION OF THE COMPONENT INFORMATION FILE (CIF) UTILITY (UPDMFN) DESCRIPTION

Program Function. UPDMFN computes and updates the FORTRAN module function value for records of a given CIF in the SEL data base.

Data Sets Accessed. The following data sets are accessed:

Data Set	Operation
Encoding Dictionary	Read only
CIF File (by project)	Read, update
CIF Transaction File	Write (append) only

Program Invocation. Execution of UPDMFN may be initiated by either logging on with UIC [204,3] and entering "@UPDMFN" or logging on with another UIC and entering "@[204,3]UPDMFN".

Program Operation. After the UPDMFN utility is entered, the operator is prompted for the project name. The operator must enter a valid project name or "S" for stop. "S" stops program execution. A null response also stops program execution. When an invalid project name is entered, the program displays an error message on the terminal and prompts for project name again.

When the project name is valid, the program performs the following activities without operator intervention: For each record in the CIF, when the module function field is blank and the number of executable statements is greater than zero, the module function value is computed and stored in the appropriate field of the record. Otherwise, the module function value in the record is not changed. After all records in the specified CIF have been processed, a summary report, which contains the number of records read, number of records changed, and number of errors encountered, is displayed on the terminal. After the summary report is displayed, the program returns to the prompt for another project name.

SECTION 3 - SYSTEM DESCRIPTION

The DBAM system is composed of the programs and data sets described in the following subsections.

3.1 SYSTEM ORGANIZATION

System activities are organized onto three levels:

- Function--The type of activity to be performed (e.g., UPDATE, or ARCHIV)
- Subfunction--The application of that activity to a specific file (e.g., update component information file (UPDCIF))
- Mode--The particular operation in progress on a file (e.g., ADD, CHANGE, or DELETE)

Only the UPDATE function has subfunctions (e.g., UPDCIF or UPDENC), and only the UPDATE subfunctions have modes (e.g., ADD or CHANGE). Six other utility functions are available in addition to the UPDATE function. Each utility function and UPDATE subfunction is an independently executable module. (Their operation was described in detail in Sections 2.4 through 2.10.) The executable modules of DBAM are composed of indirect command files and FORTRAN subroutines as indicated below.

3.1.1 INDIRECT COMMAND FILES

The highest level of system control is exercised by the indirect command files (Figure 3-1). The activities they perform include selection of the function to be performed, selection of the type of file, initiation of execution of FORTRAN programs, and integration of RMS utilities. Indirect command files are not compiled; they are interpreted code and system directives. Minimally, the call to the executable task image associated with each function is imbedded in an indirect command file of the same name. Listings of

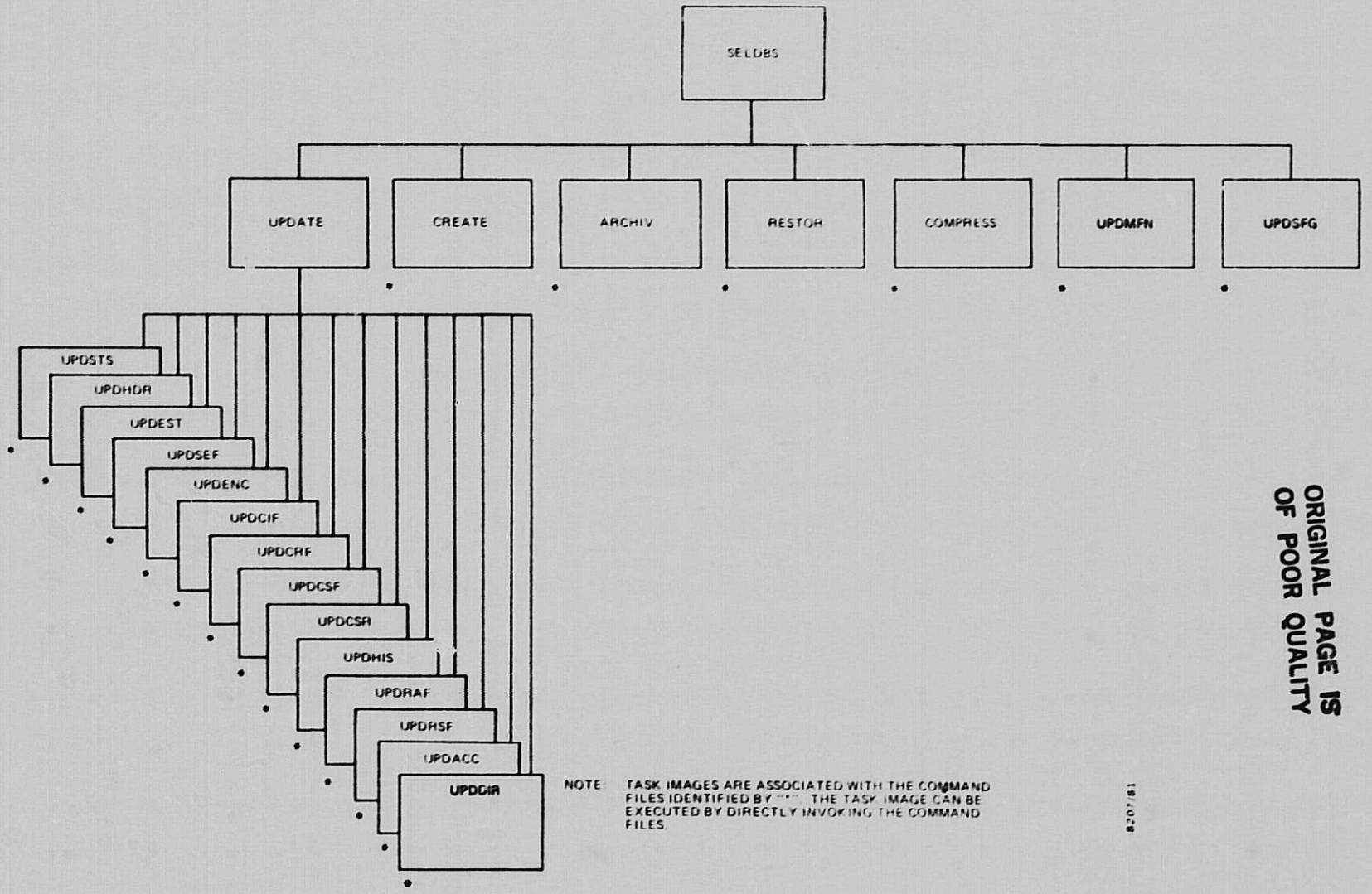


Figure 3-1. SEL DBAM Indirect Command File Organization

the DBAM indirect command files are presented in Appendix D. Other services may also be connected by these indirect command files.

3.1.2 INTERACTIVE UPDATES

The special nature of the Record Management System (RMS)-indexed file structure prevents SEL files from being accessed by the usual sequential methods. Interactive update programs take advantage of the RMS-indexed feature to provide fast access to the data. Every record in a file is identified by a unique field (primary key) and possibly by other fields (alternate keys). These keys are used to manipulate RMS-indexed records directly.

The 13 UPDATE subfunctions update the data base. Figures A-1 through A-13 in Appendix A illustrate the baseline diagrams for these subfunctions. The module functions are described in Tables B-1 through B-13 and in Table B-20 in Appendix B. Three modes (ADD, CHANGE, DELETE) within each subfunction are supported for all SEL files except the Comments File. The additional capability of reading data from the SAP Output File is also available in the UPDCIF function.

Comments can only be added or deleted; they cannot be changed. Comments are not logically independent of the forms to which they pertain, and so they are updated as part of a form update. No separate UPDATE function is available for Comment Files.

3.1.3 FILE MAINTENANCE UTILITIES

Four utilities--CREATE, ARCHIV, RESTOR, and COMPRESS--support the maintenance of the global data base structure. These utilities are indirect command files that use RMS system utilities in addition to application programs to perform their function. Descriptions of the utilities follow (the

RMS utility used is noted in parentheses). Figures A-14 through A-17 in Appendix A illustrate the baseline diagrams for these functions. The individual module functions are described in Tables B-14 through B-17 and in Table B-20 in Appendix B. Reference 6 describes the use of the RMS utilities.

The CREATE (DEF) uti ity enters a new project name in the Encoding Dictionary, creates all data files specific to that project, and adds records for those files to the File Name and Status File. The six Header Files are the only data sets that must be defined by separate procedures.

The ARCHIV (BCK) utility makes a tape copy of all or a list of data base files. The "last backup date" field in the File Name and Status File is updated for every file saved. The Transaction Files are also reinitialized if the entire data base is archived. The date and tape identification of every ARCHIV transaction is recorded on a log data set.

The RESTOR (RST) utility recovers all or a list of data sets from a tape. Previous versions of Transaction Files are merged with the reinstalled files if "ALL" is specified.

The COMPRESS (DEF, IFL) utility compresses the data base files by sorting the file indexes and reformatting the data into contiguous blocks.

3.1.4 INDEPENDENT UTILITIES

Two utilities--UPDMFN and UPDSFG--are included in DBAM to perform some special functions. Figures A-18 and A-19 in Appendix A illustrate the baseline diagrams for these functions. The individual module functions are described in Tables B-18, B-19, and B-20 in Appendix B.

The UPDMFN utility computes and updates the FORTRAN module function value for records of specified CIFs. The UPDSFG

utility updates the status flag field of every record in a selected data base file to a particular value. UPDSFG cannot be used to update status flags in header files.

3.2 DATA SET DESCRIPTIONS

The data sets used by DBAM fall into the three groups described below. All files, unless otherwise noted, are located on disk DB1 in UIC [204,1].

3.2.1 DATA BASE FILES

Data base files contain the information used in the interpretation and analyses that are the goals of the SEL. The files containing descriptions of the structure of these files are presented in Appendix E. The detailed formats of the files are presented in Reference 3. All data base files are RMS-indexed files with fixed-length, formatted records. (The SEF and SAP files are exceptions as noted below.)

There are six header files in the data base. Three of these files contain information describing other files or their content: the Encoding Dictionary (ENCODE.HDR), the File Name and Status File (STAT.HDR), and the Subjective Evaluations Directory File (DIR.HDR). The other three header files contain summary information describing the projects: the Phase Dates File (HEADER.HDR), the Estimated Statistics File (EST.HDR), and the Subjective Evaluations Data File (SEF.HDR). The SEF is the only data base file containing variable-length records.

There are nine "form" data file types in the data base. Each project may be represented by up to nine files, one of each type. These files are named "project.abv", where project is the name of the project and abv is the three-character file type abbreviation (Table 1-1). The files contain either form, computer, component, or comment data.

The SAP Output File is the only sequential organization data base file. This file contains the component data extracted from project source code by the SAP (Reference 2).

3.2.2 TRANSACTION FILES

Transaction files receive afterimages of all updates made to seven types of form data files. (There is no transaction file for either the CMT or Accounting Information (ACC) type of form data file.) A transaction file has the same format as the corresponding form data file but has additional bytes to specify the type of transaction (A = ADD, D = DELETE, C = CHANGE) and the date of the transaction. These files are named "[204,1]TRANS.abv", where abv is the three-character form type (Table 1-1). These files have fixed length and sequential formats and are the only files associated with the SEL data base that are located on disk DB0.

3.2.3 OTHER FILES

Several files are used by the maintenance utilities to pass data among indirect command files, programs, and RMS utilities. CREATE uses "[204,3]PRONAM.DAT" to hold the name of the project for which files are being created. ARCHIV and RESTOR use "[204,3]FILNAM.DAT" to save the specification of files to be archived or restored. A log of ARCHIV transactions is maintained on "[204,3]BCKLOG.DAT". Error messages generated by UPDCIF while processing SAP data are saved on "[204,3]MESSAGE.SAP". All of these data sets have variable-length sequential formats.

3.3 TASK-BUILD PROCEDURES

All programs described in this document were implemented in structured FORTRAN. Consequently, all sources must pass through the FORTRAN Preprocessor (FPP) (Reference 7) before compilation. The locations of all data sets required to prepare a task image (load module) are listed in Table 3-1.

Table 3-1. Locations of Component Data Sets

File Type	Qualifier	UIC
Source	FPP	[204,15]
Object	OBJ	[204,15]
Overlay	ODL	[204,15]
Task image	TSK	[204,5]
Indirect task builder commands	TKB	[204,15]
Task image generation commands	CMD	[204,15]

The component (subroutine) name is used as the file name of the source and object data sets. The program (function) name is used as the file name of the task images. Overlay Descriptor Language data set names have the format "XX.ODL" and indirect task builder command data set names have the format "XX.TKB", where XX is the subsystem prefix (Table 3-2). Task-build options used in preparing task images are listed in Table 3-2. Task image generation command files perform all precompile, compile, and task build operations and have names with the format "XXGEN.CMD", where XX is the subsystem prefix. Appendix C lists the Overlay Descriptor Language needed to build these task images.

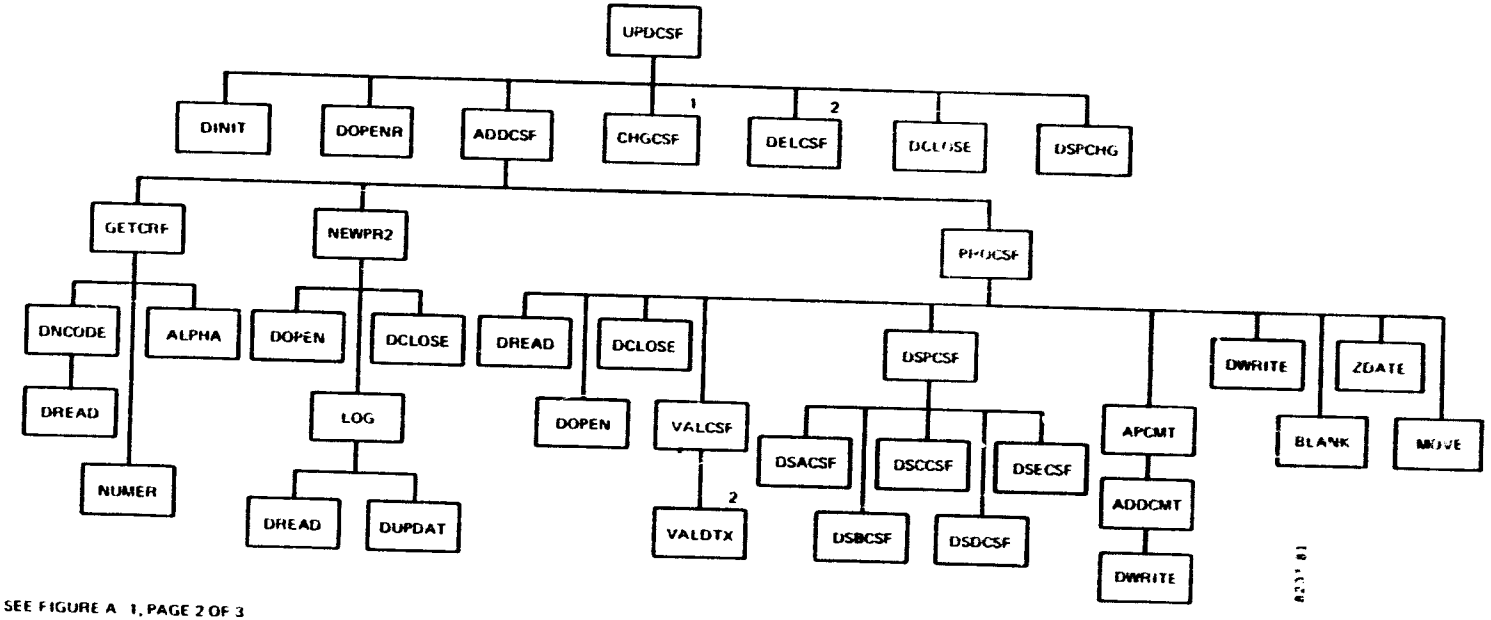
Table 3-2. Task-Build Options for DBAM Functions

Program	Subsystem Prefix	Active Files (ACTFIL)	Maximum Number of Units (UNITS)	Maximum Buffer Size (MAXBUF)
UPDSEF	SE	3	13	578
UPDEST	ES	2	12	
UPDHDR	HD	2	12	
UPDHIS	HI	4	12	
UPDRSF	RS	4	13	
UPDRAF	RA	4	14	
UPDCSR	CT	4	13	
UPDCSF	CM	4	14	263
UPDCIF	CI	4	14	
UPDCRF	CH	4	14	
UPDSTS	ST	2	12	
UPDENC	EN	2	12	
CREFIL (CREATE)	CE	4	11	
ARCFIL (ARCHIV)	AR	4	11	263
RESFIL (RESTOR)	RE	4	13	263
STS (COM-PRESS)	CO	1	1	
UPDDIR	SD	2	1	
UPDMFN	MF	3	3	
UPDSFG	SF	4	3	263

APPENDIX A - BASELINE DIAGRAMS

Appendix A contains baseline diagrams for the DBAM programs. Figures A-1 through A-13 and show the 13 update subfunctions; Figures A-14 through A-19 show the 6 utility functions.

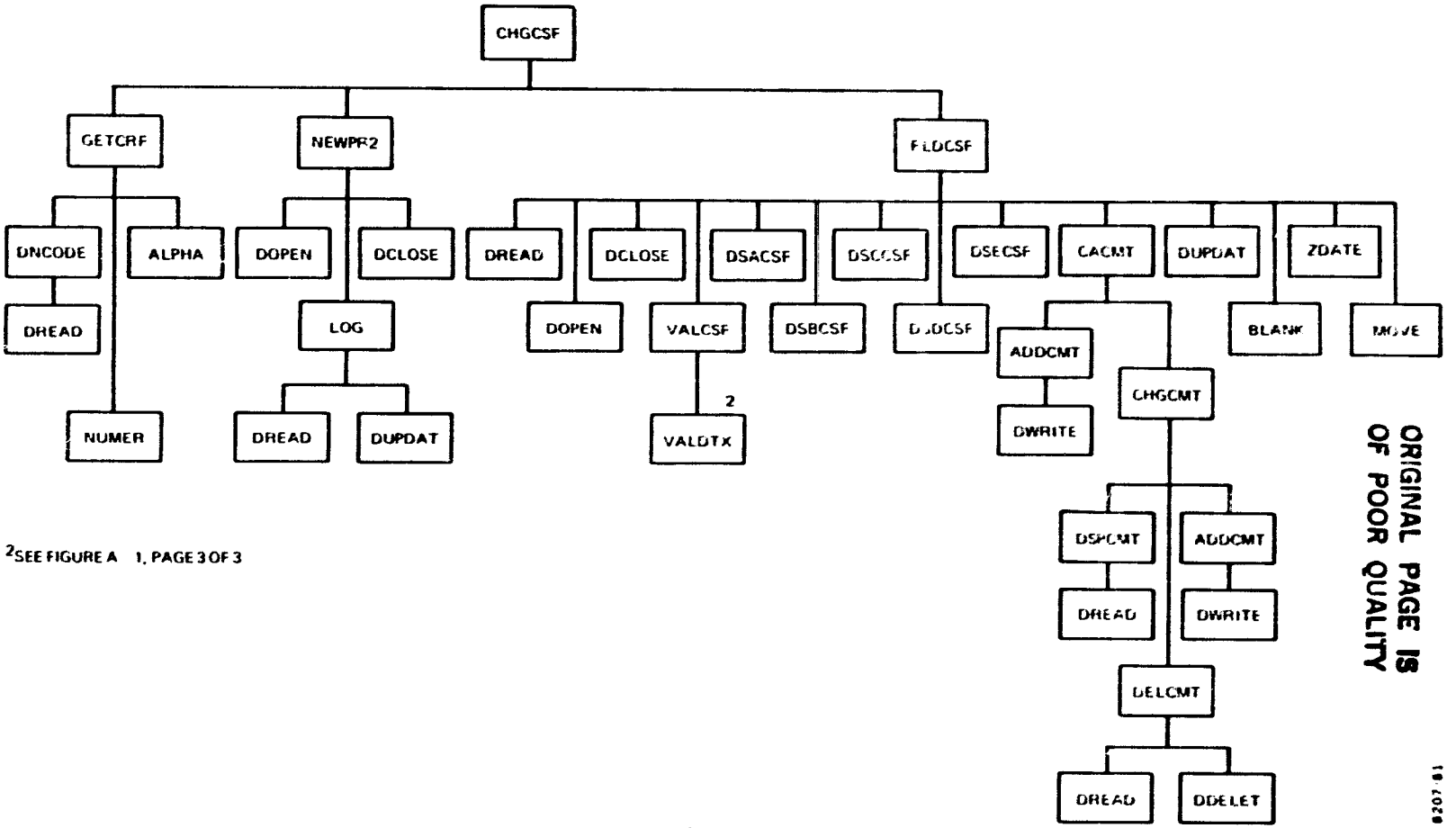
Figure Number	Title
A-1	Component Summary Form Update Subfunction
A-2	Change Report Form Update Subfunction
A-3	Component Information File Update Subfunction
A-4	Component Status Report Update Subfunction
A-5	Run Analysis Form Update Subfunction
A-6	Resource Summary Form Update Subfunction
A-7	Growth History Update Subfunction
A-8	File Name and Status File Update Subfunction
A-9	Encoding Dictionary Update Subfunction
A-10	Estimated Statistics File Update Subfunction
A-11	Subjective Evaluations File Update Subfunction
A-12	Phase Dates (Header) Update Subfunction
A-13	Subjective Evaluations Directory File Update Subfunction
A-14	Create Function
A-15	Archive Function
A-16	Restore Function
A-17	Compress Function
A-18	Update Status Flag Function
A-19	Update the FORTRAN Module Function of the Component Information File Function



[1] SEE FIGURE A 1, PAGE 2 OF 3

[2] SEE FIGURE A 1, PAGE 3 OF 3

Figure A-1. Baseline Diagram for Component Summary Form Update Subfunction (1 of 3)



ORIGINAL PAGE IS
OF POOR QUALITY

Figure A-1. Baseline Diagram for Component Summary Form Update Subfunction (2 of ?)

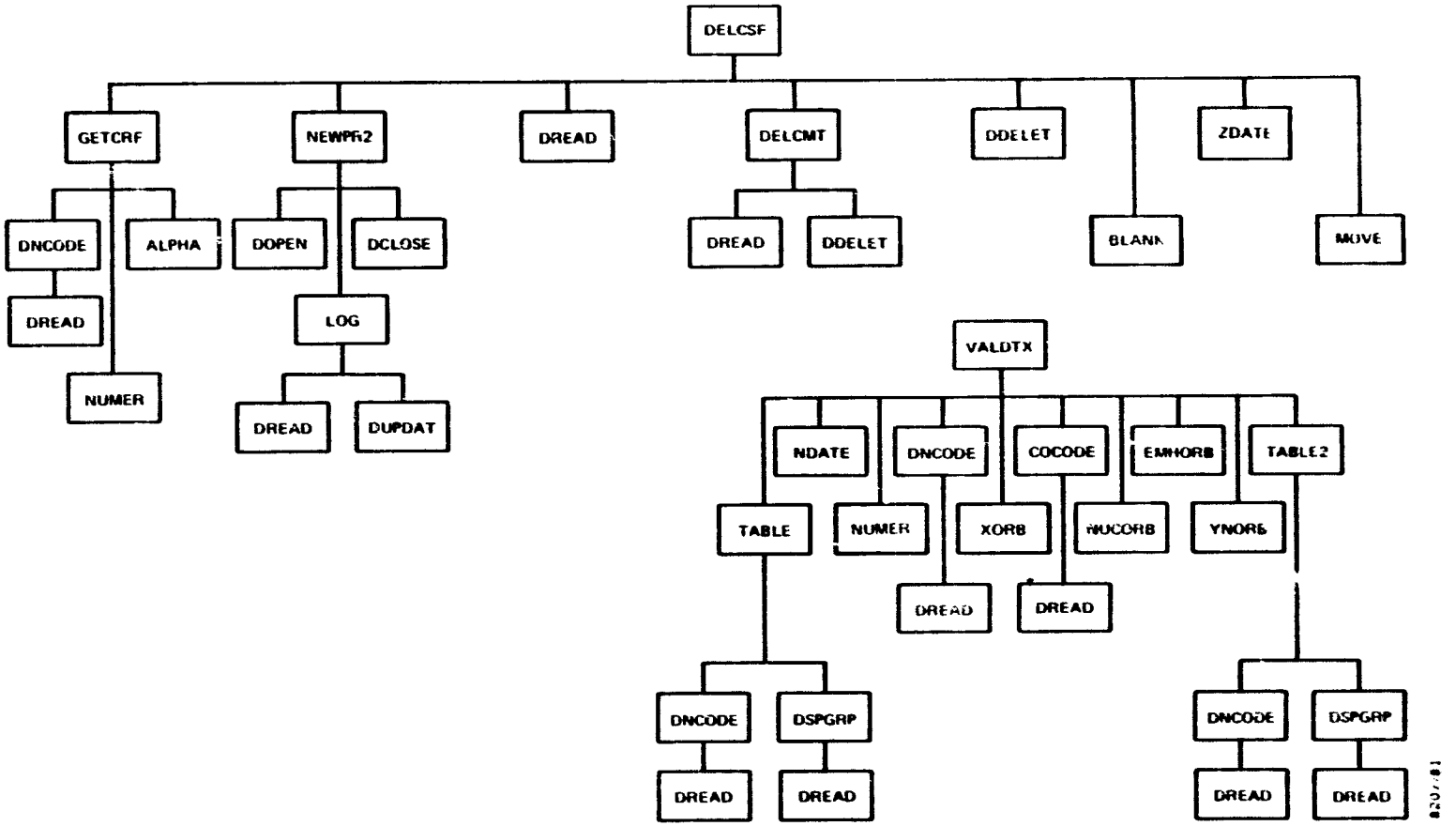


Figure A-1. Baseline Diagram for Component Summary Form Update Subfunction (3 of 3)

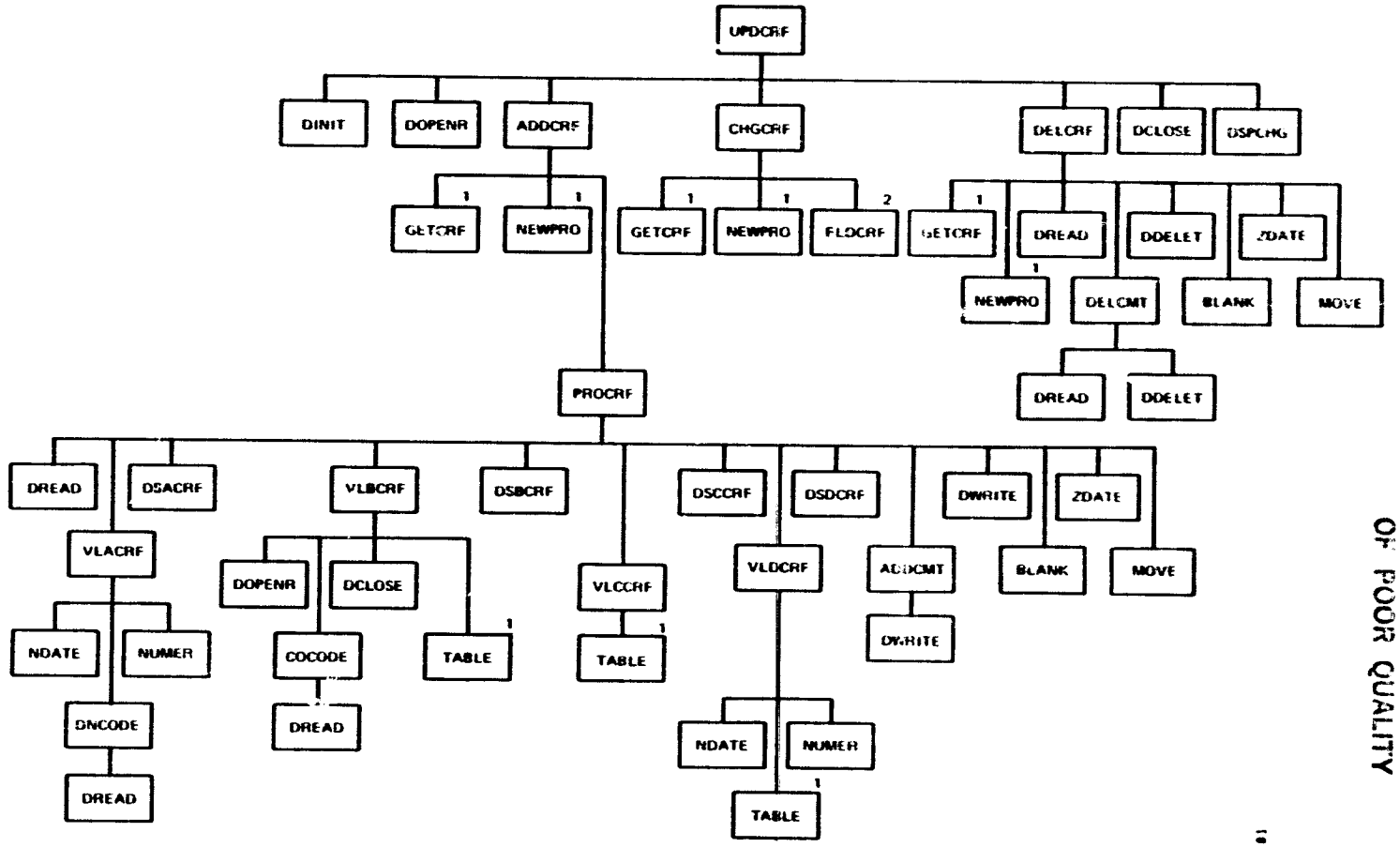


Figure A-2. Baseline [illegible]agram for Change Report Form Update Subfunction (1 of 3)

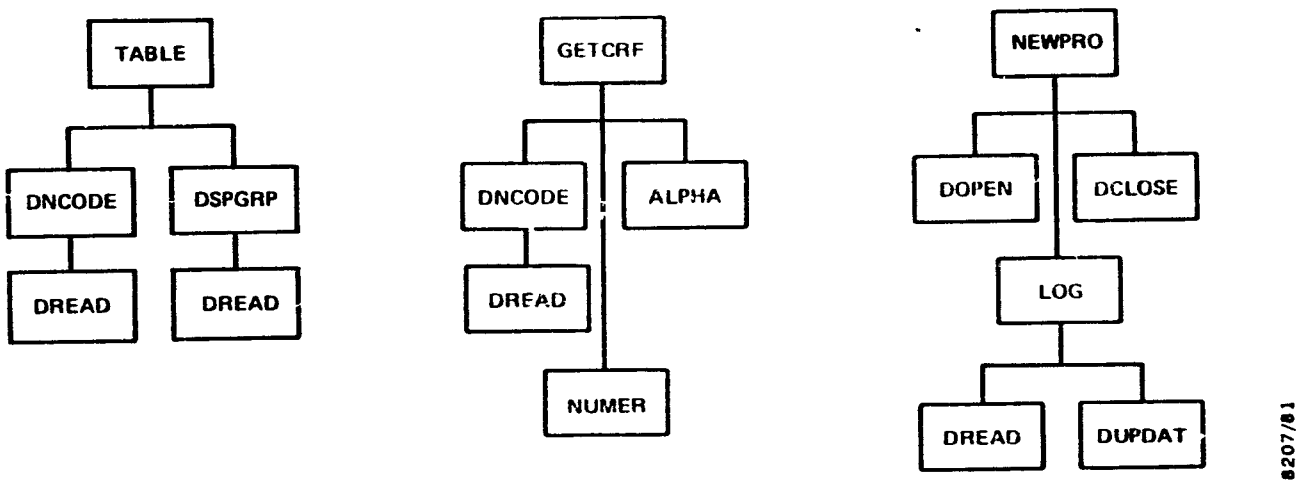


Figure A-2. Baseline Diagram for Change Report Form Update Subfunction (2 of 3)

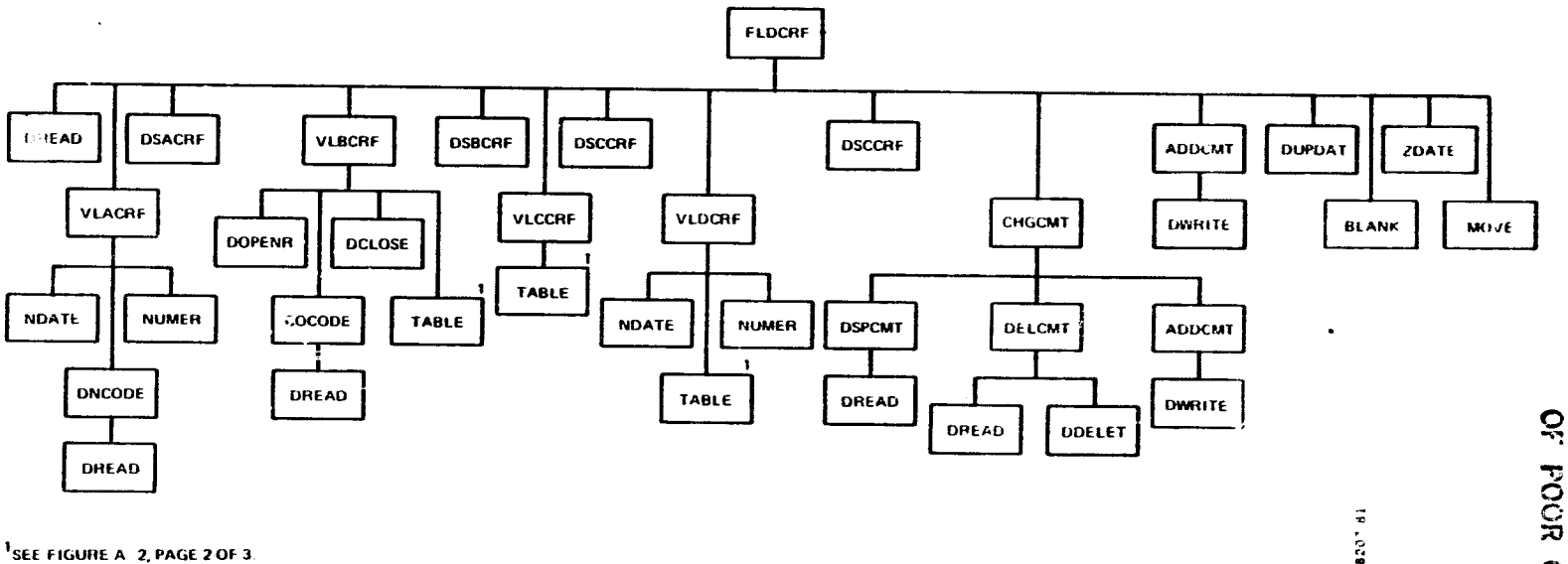
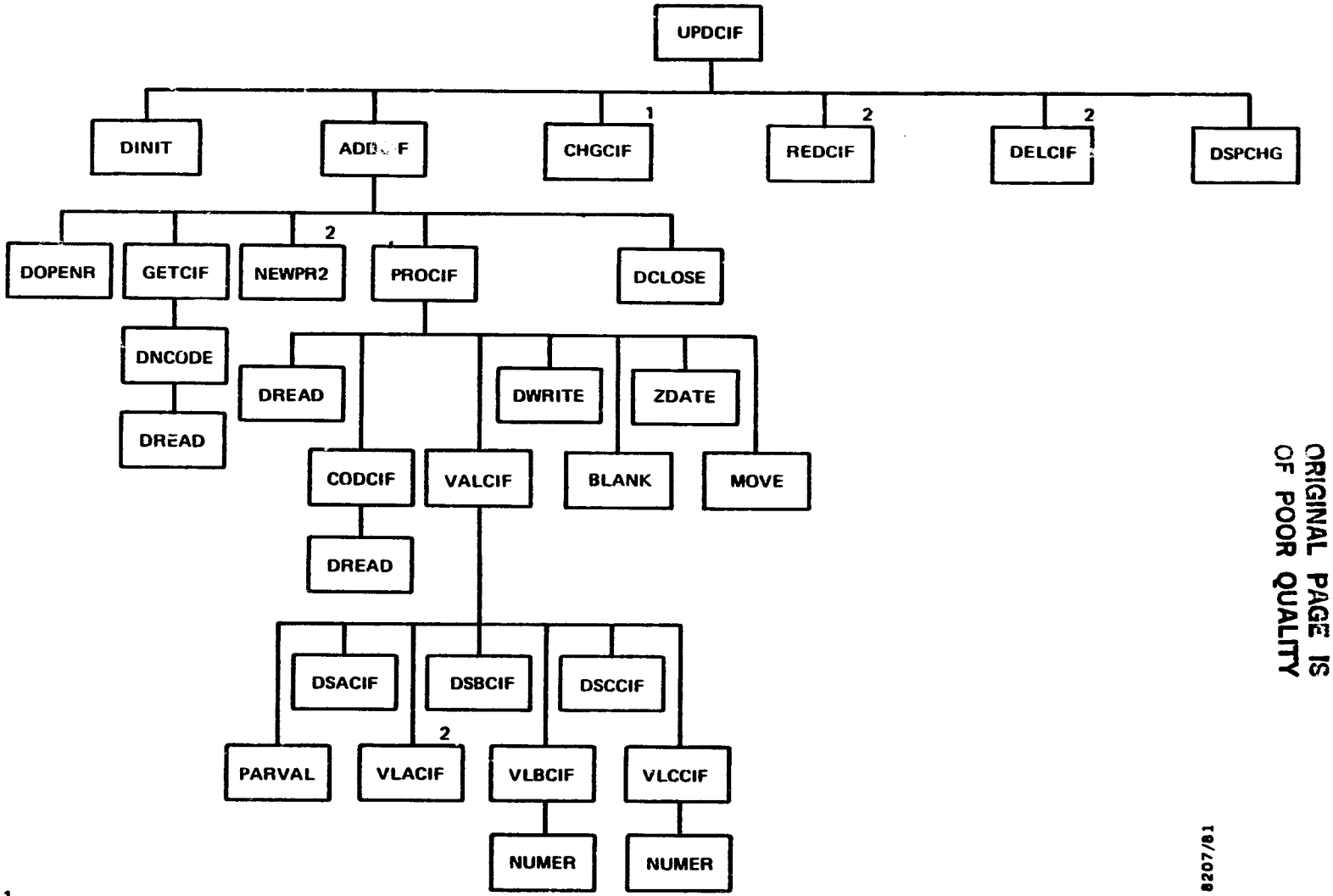


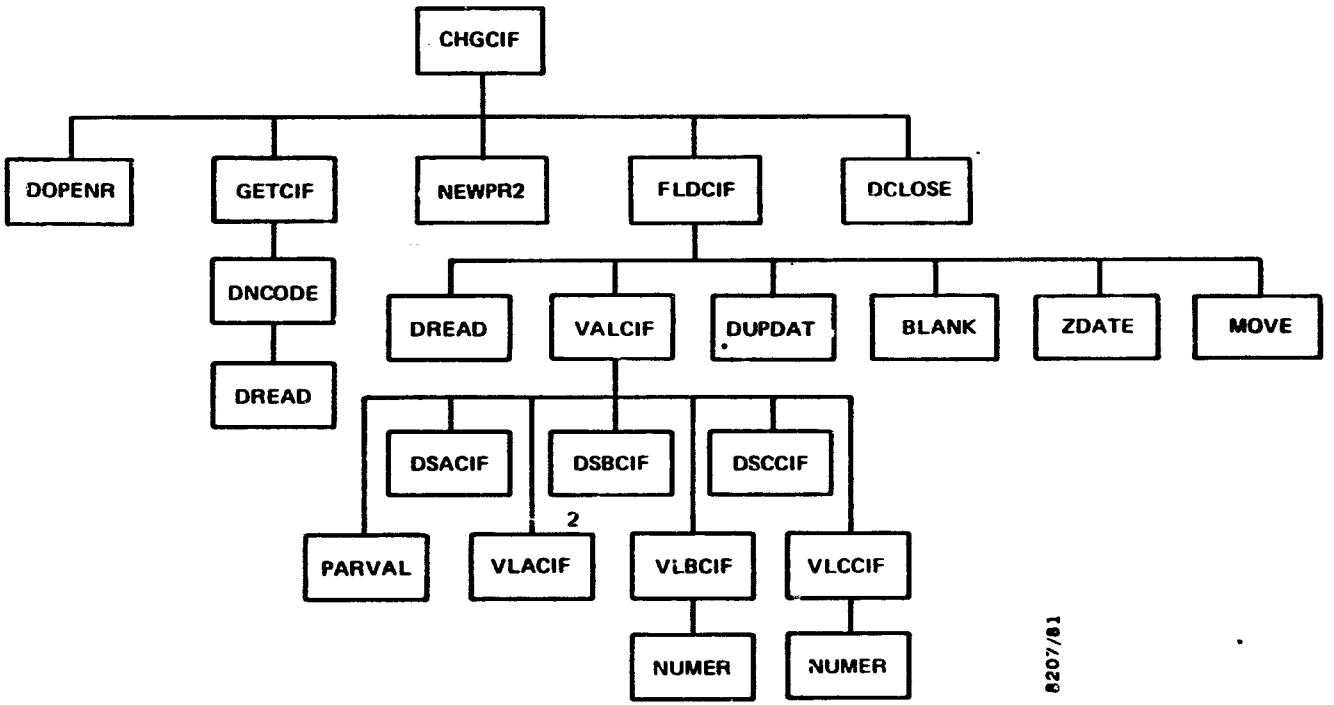
Figure A-2. Baseline Diagram for Change Report Form Update Subfunction (3 of 3)



[1] SEE FIGURE A–3, PAGE 2 OF 3.

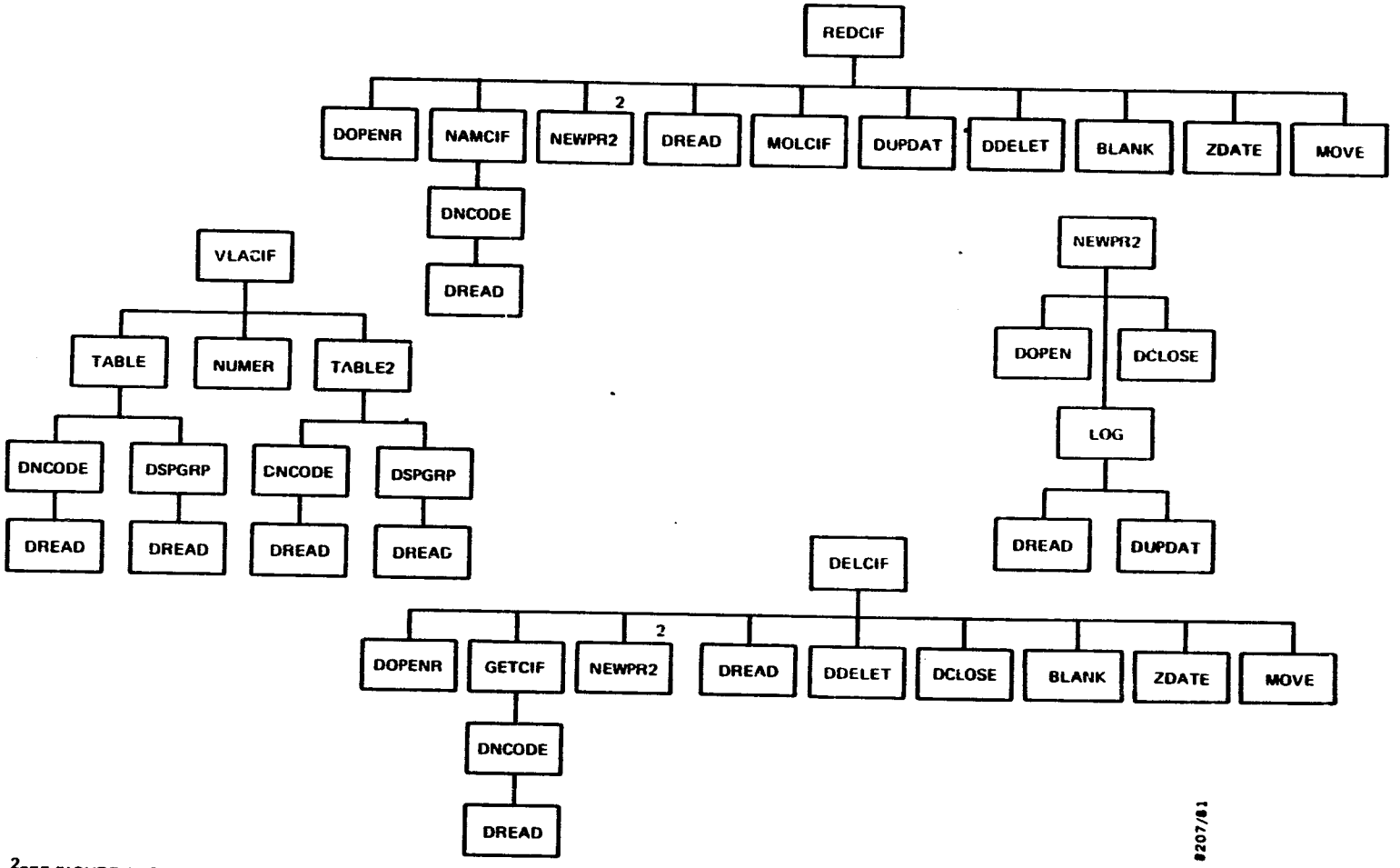
[2] SEE FIGURE A–3, PAGE 3 OF 3.

Figure A-3. Baseline Diagram for Component Information File Update Subfunction (1 of 3)



[2]SEE FIGURE A–3, PAGE 3 OF 3.

Figure A-3. Baseline Diagram for Component Information File Update Subfunction (2 of 3)



[2]SEE FIGURE A-3, PAGE 3 OF 3.

Figure A-3. Baseline Diagram for Component Information File Update Subfunction (3 of 3)

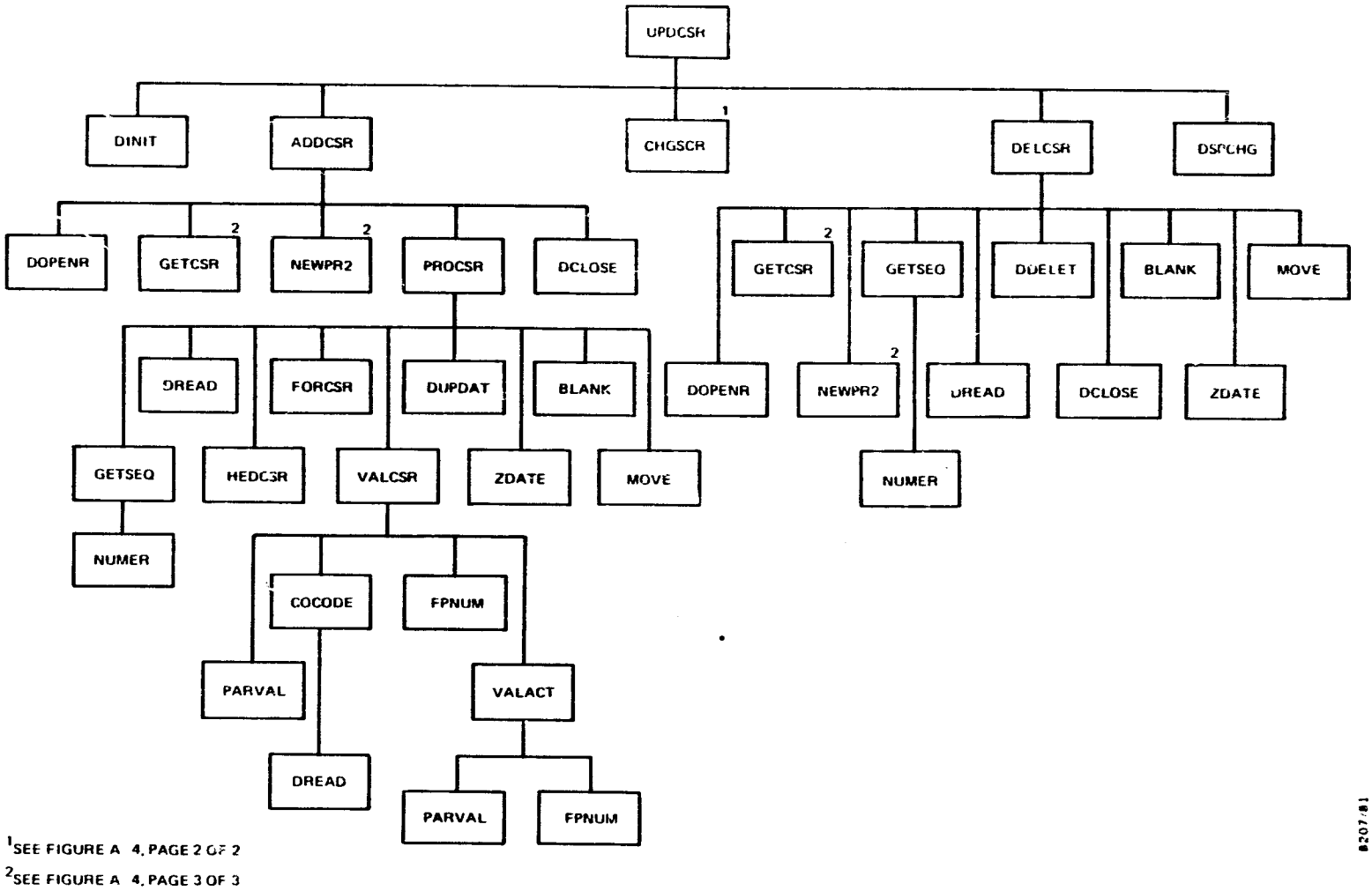
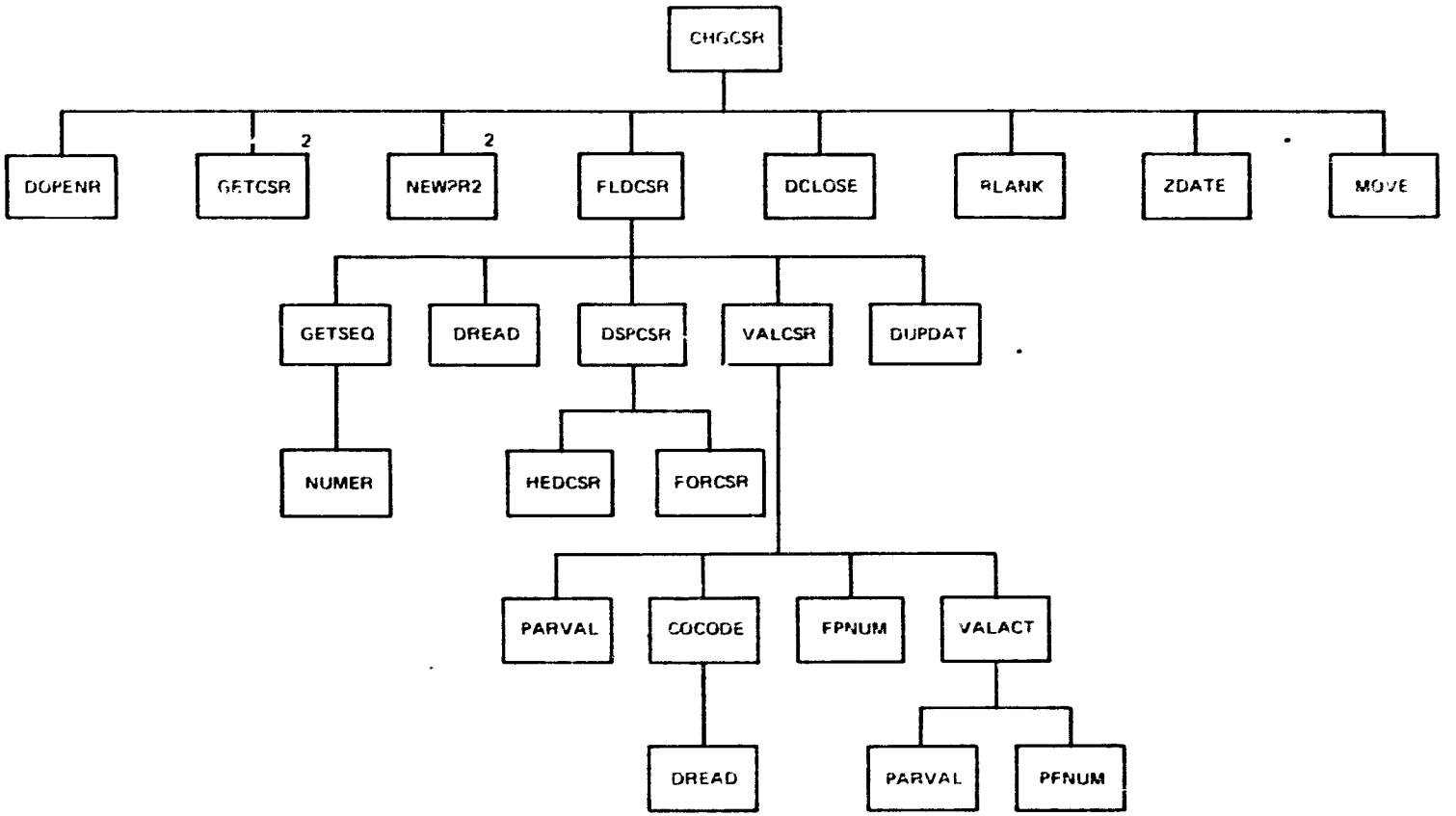


Figure A-4. Baseline Diagram for Component Status Report Update Subfunction (1 of 3)



[2]SEE FIGURE A-4, PAGE 3 OF 3

Figure A-4. Baseline Diagram for Component Status Report Update Subfunction (2 of 3)

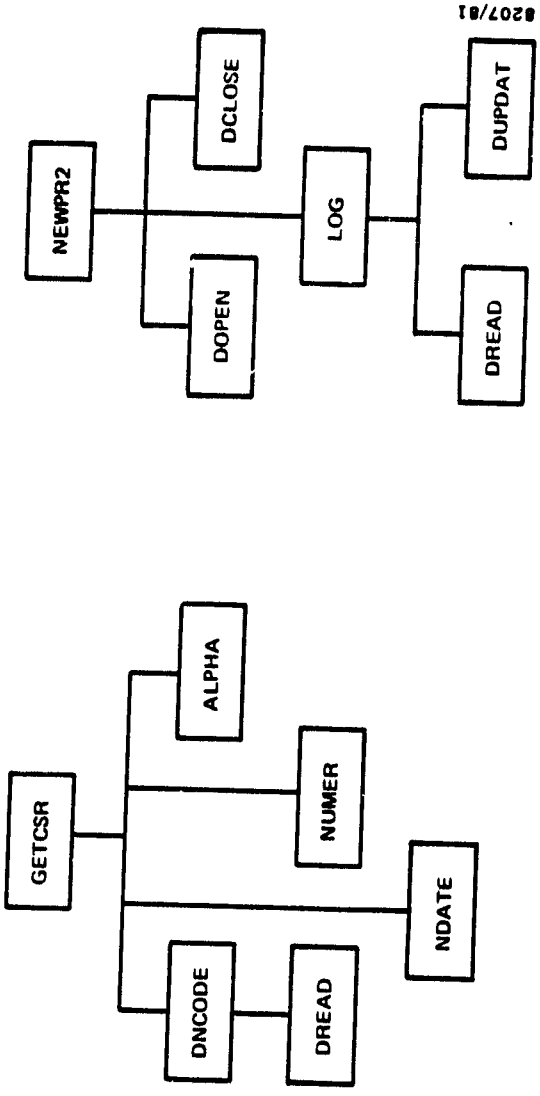
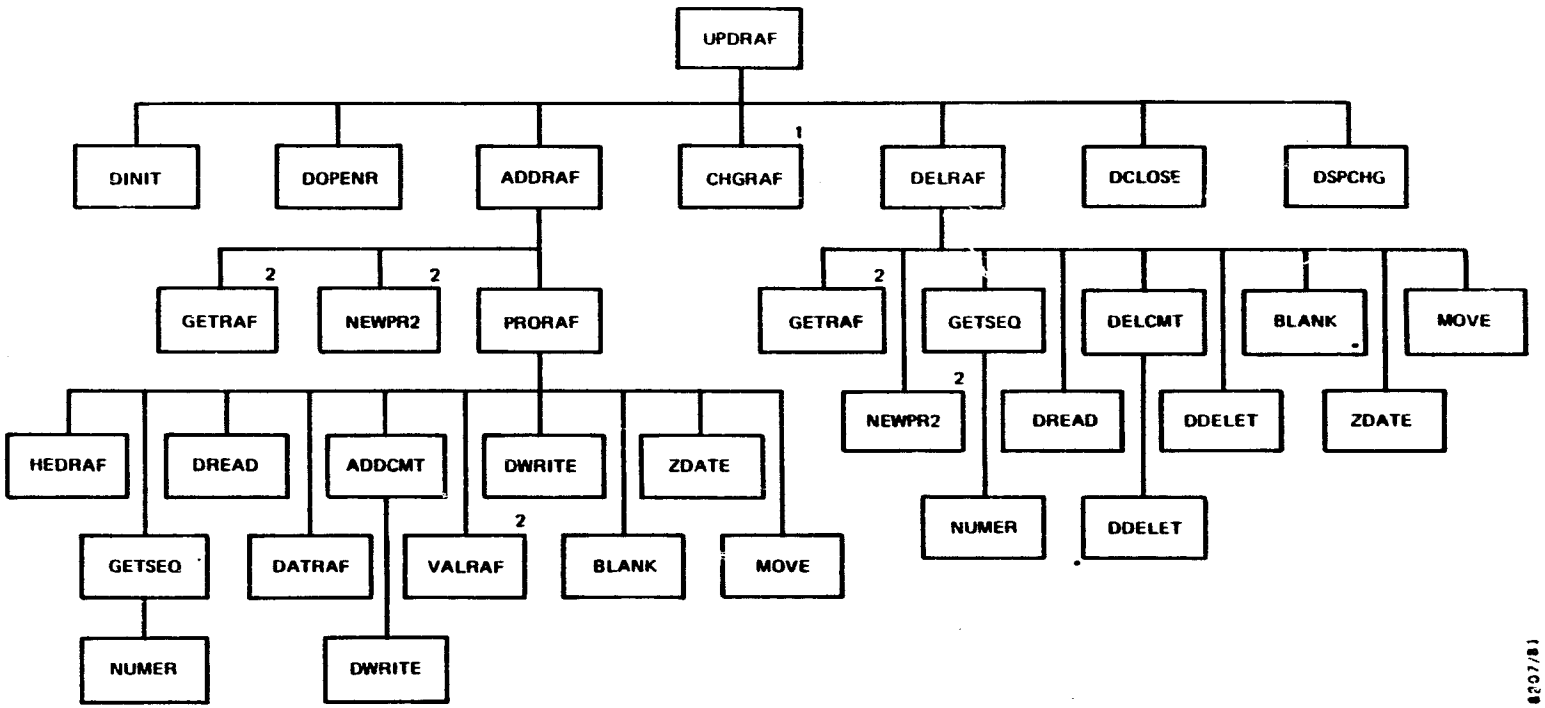


Figure A-4. Baseline Diagram for Component Status Report Update Subfunction (3 of 3)



[1]SEE FIGURE A-5, PAGE 2 OF 3.

[2]SEE FIGURE A-5, PAGE 3 OF 3.

Figure A-5. Baseline Diagram for Run Analysis Form Update Subfunction (1 of 3)

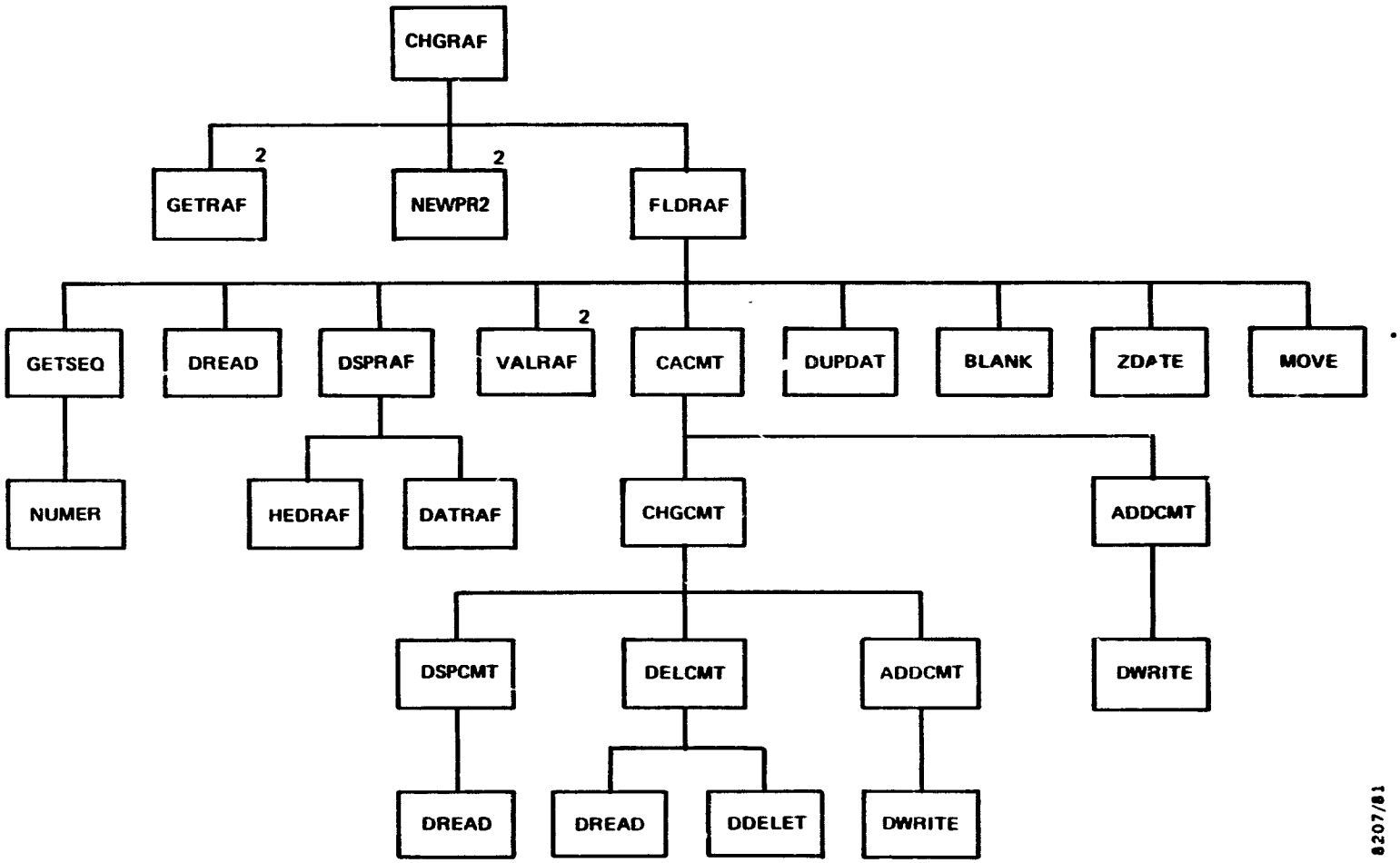


Figure A-5. Baseline Diagram for Run Analysis Form Update Subfunction (2 of 3)

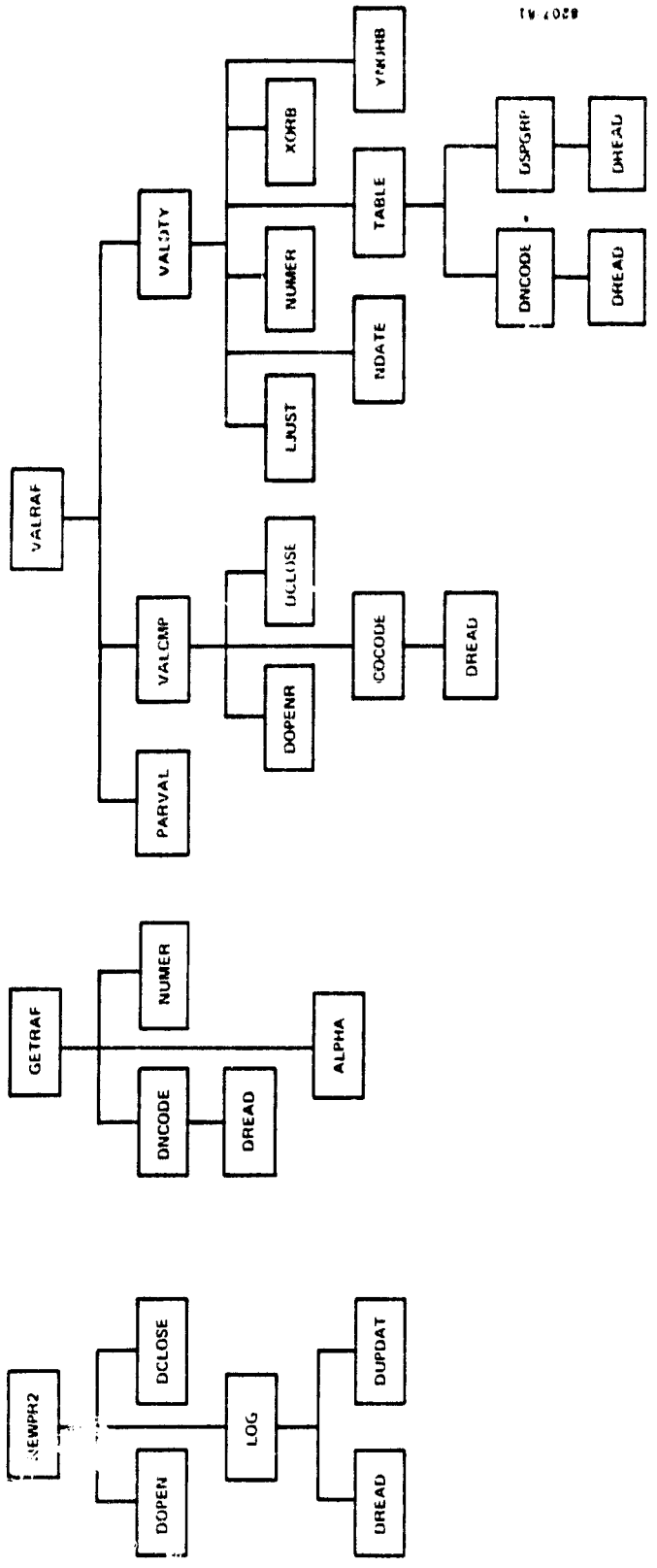
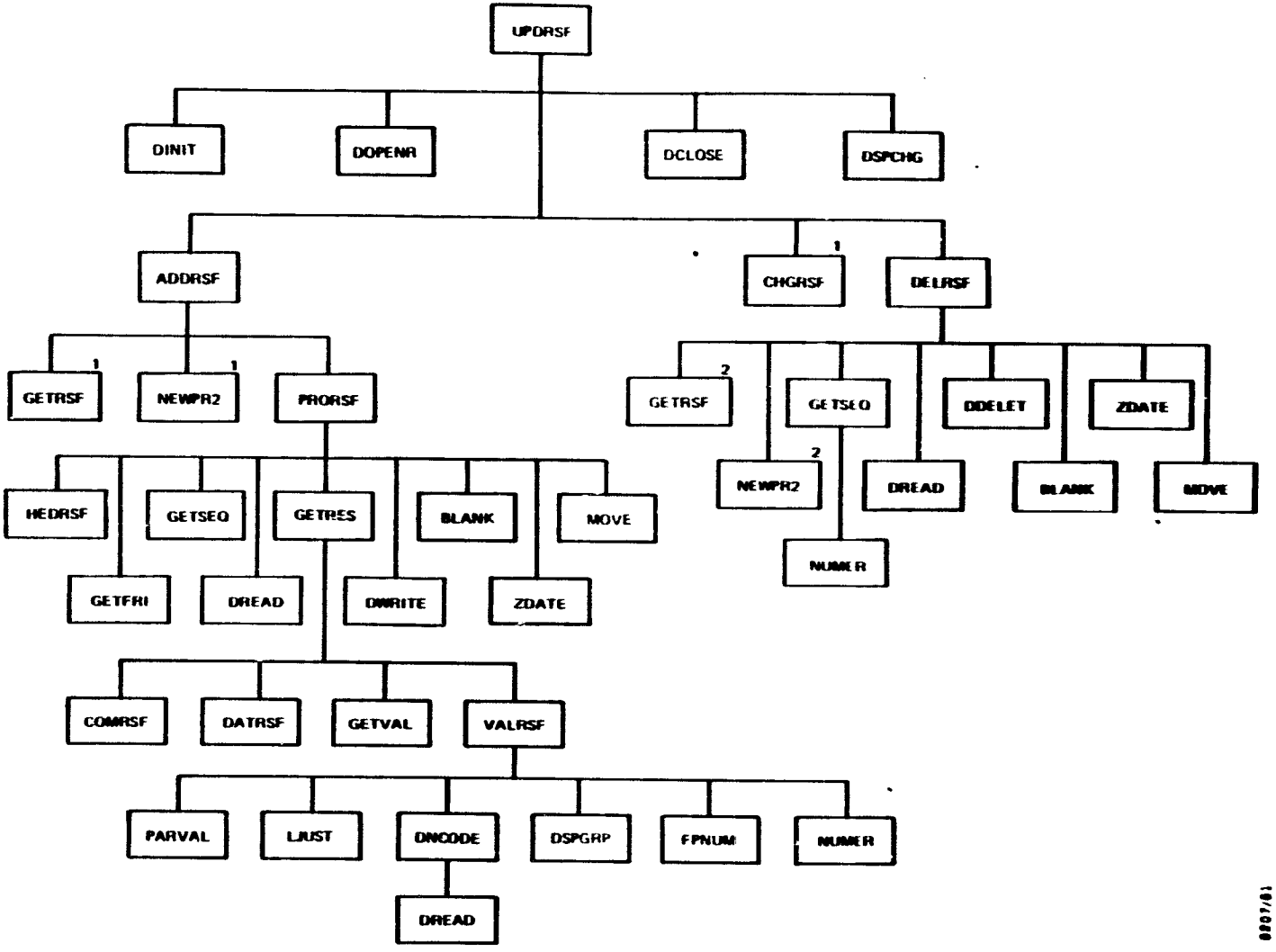


Figure A-5. Baseline Diagram for Run Analysis Form Update Subfunction (3 of 3)



[1] SEE FIGURE A 6, PAGE 2 OF 2

Figure A-6. Baseline Diagram for Resource Summary Form Update Subfunction (1 of 2)

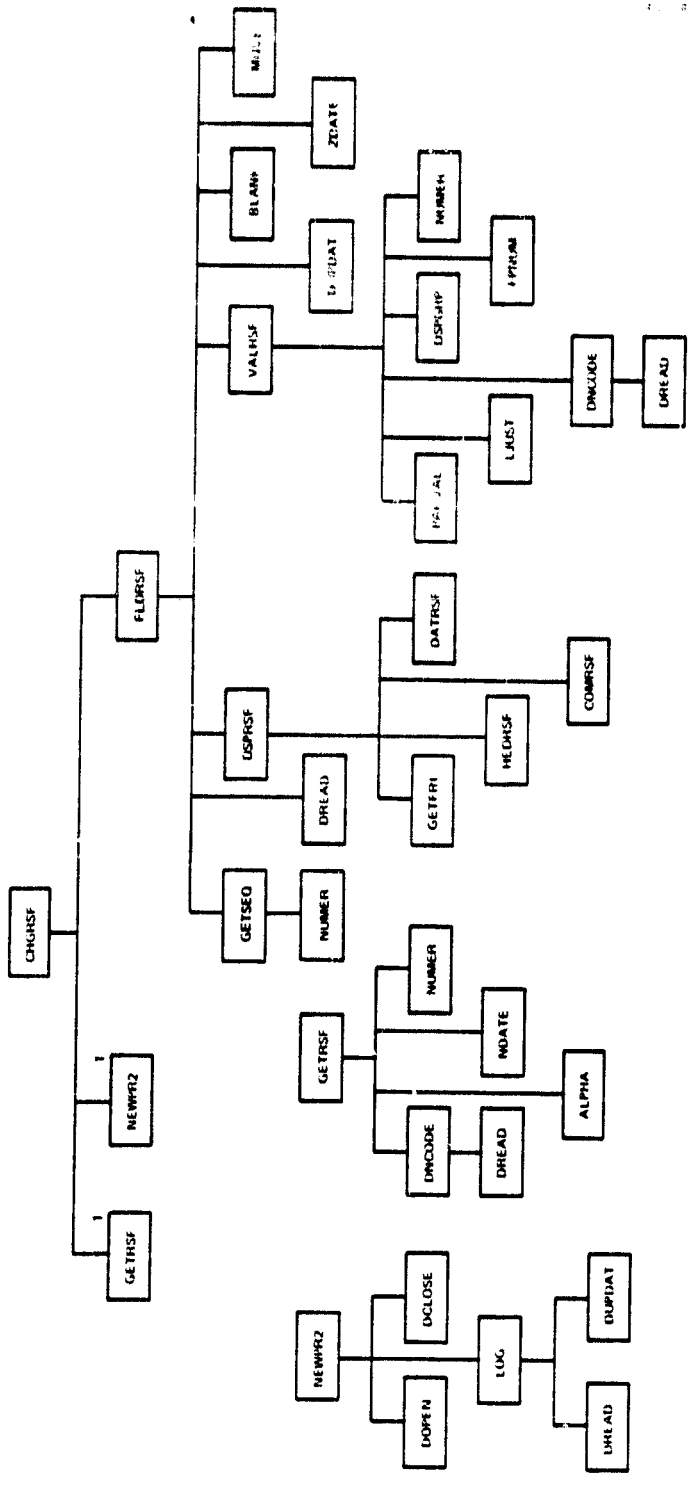


Figure A-6. Baseline Diagram for Resource Summary Form Update Subfunction (2 of 2)

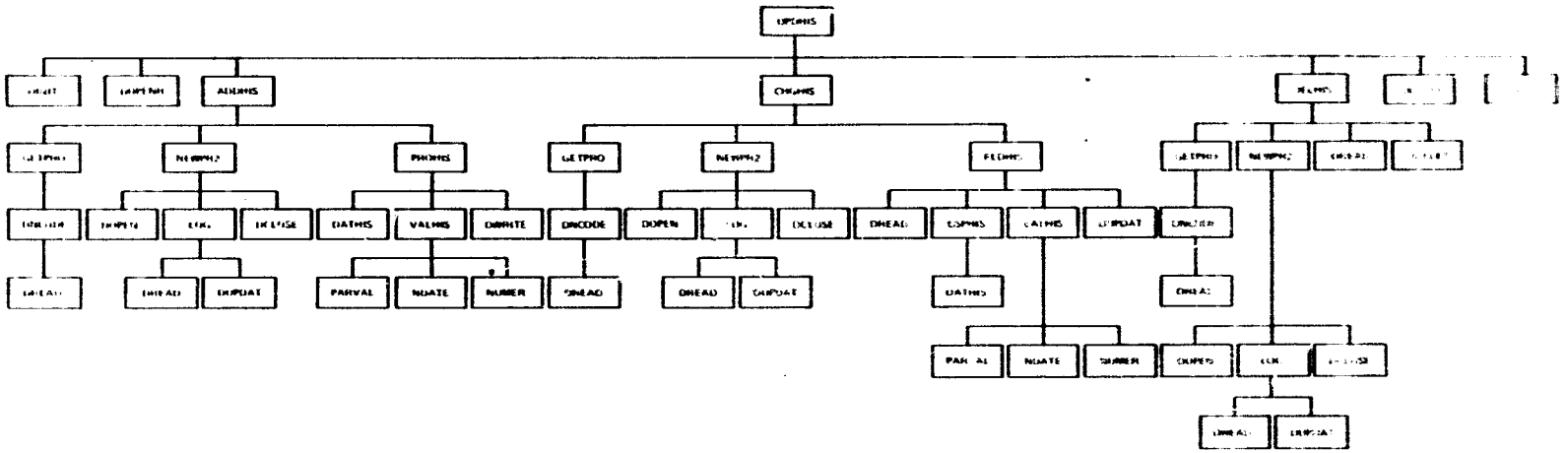


Figure A-7. Baseline Diagram for Growth History Update Subfunction

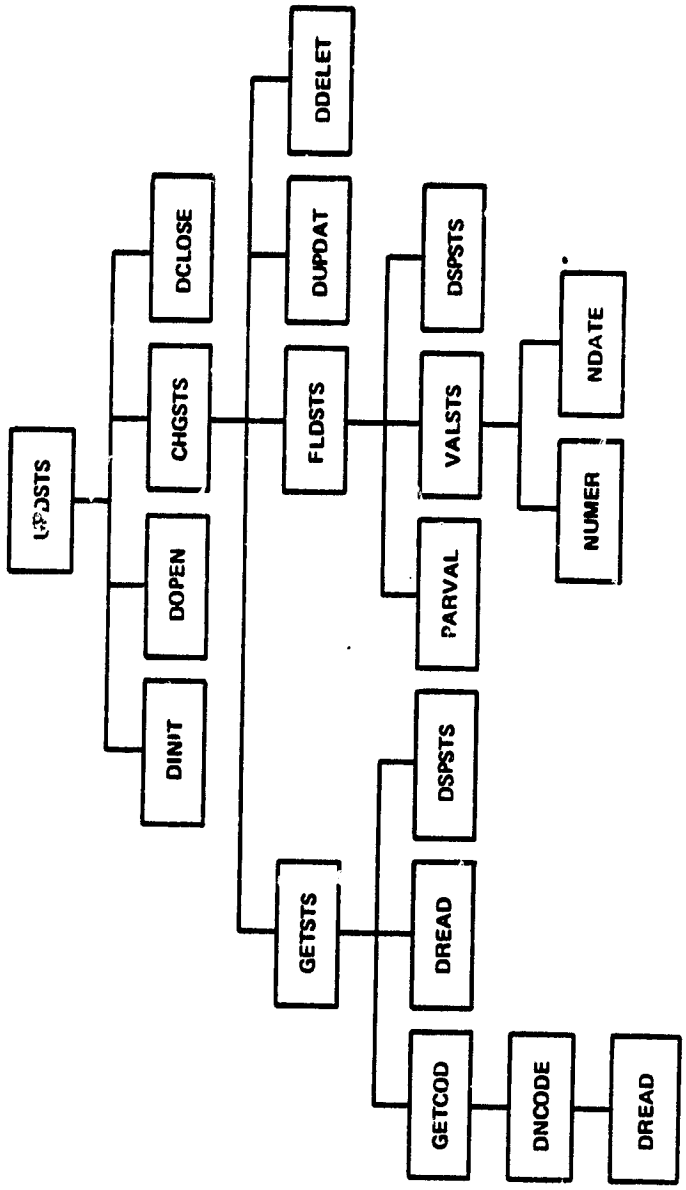


Figure A-8. Baseline Diagram for File Name and Status File Update Subfunction

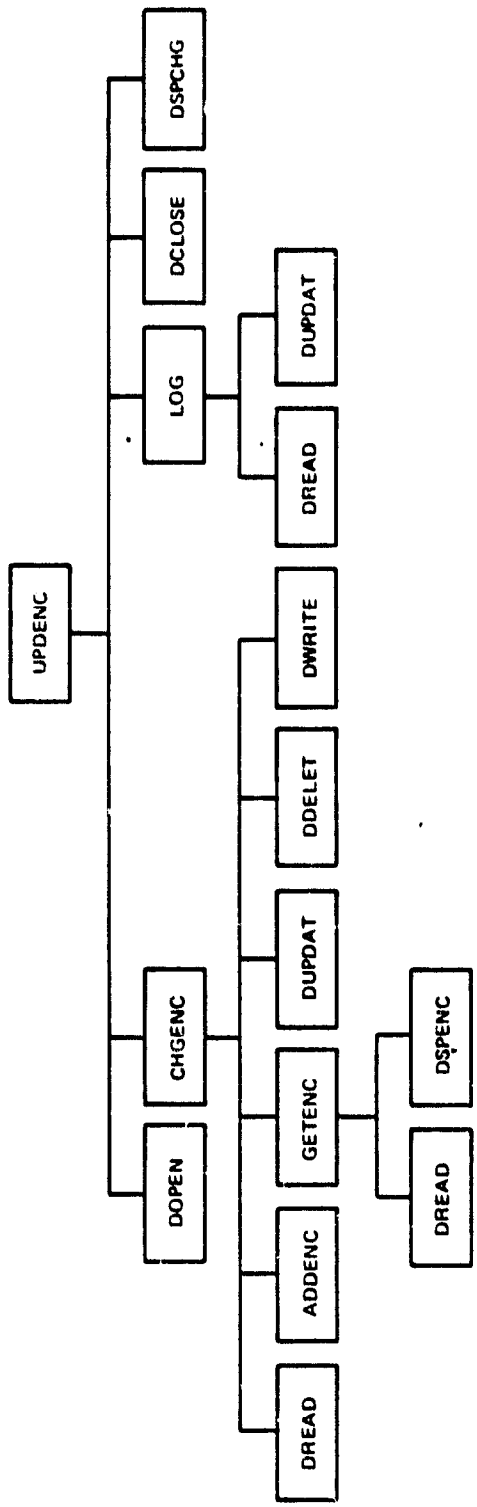


Figure A-9. Baseline Diagram for Encoding Dictionary Update Subfunction

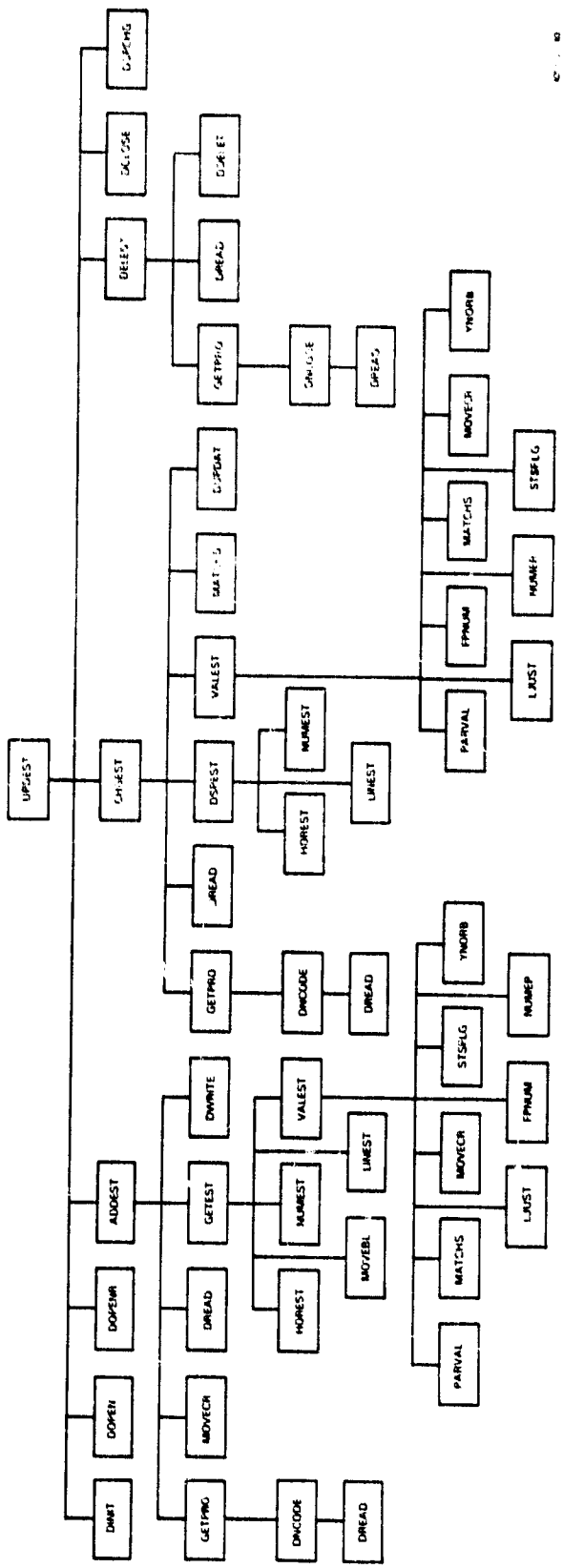


Figure A-10. Baseline Diagram for Estimated Statistics File Update Subfunction

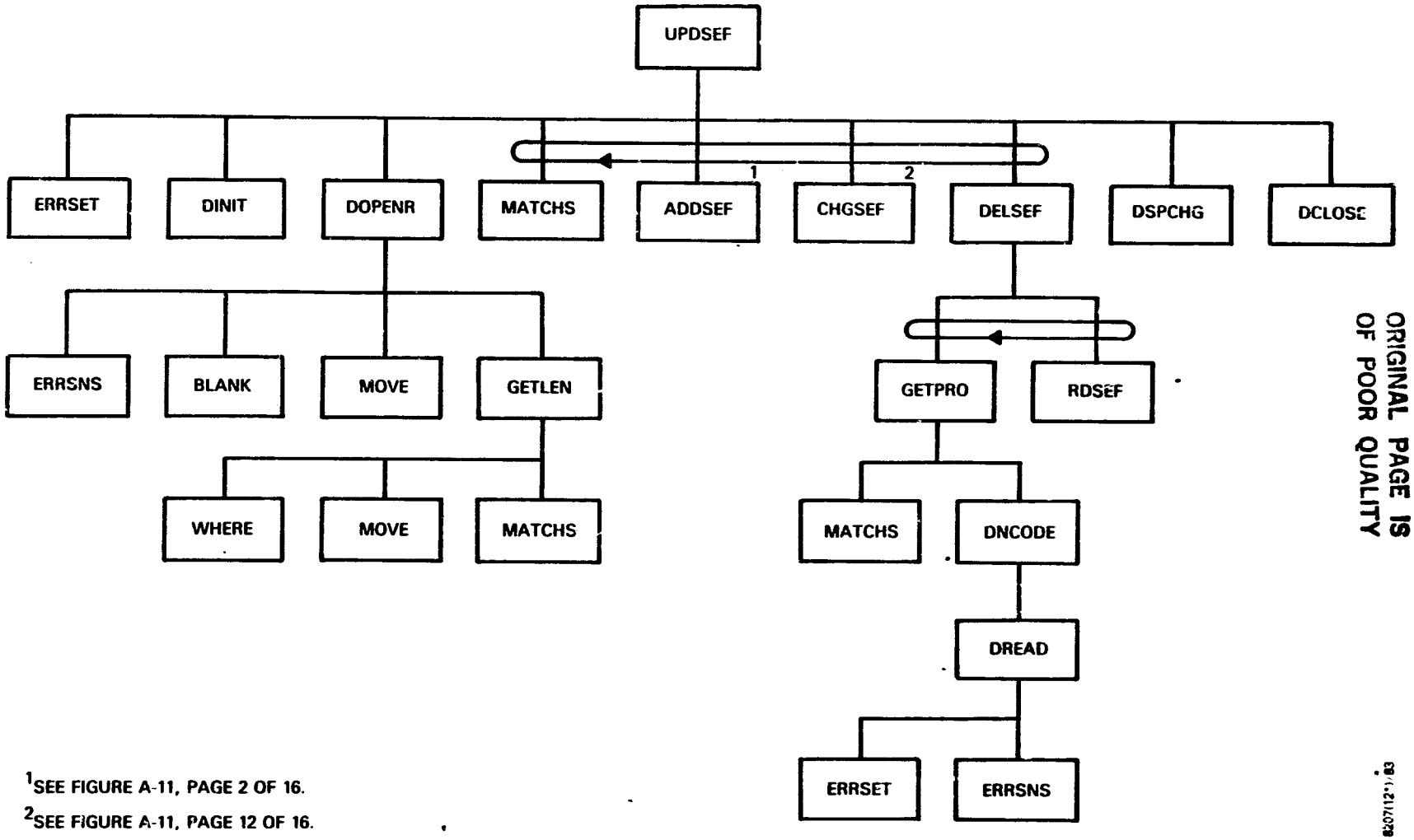
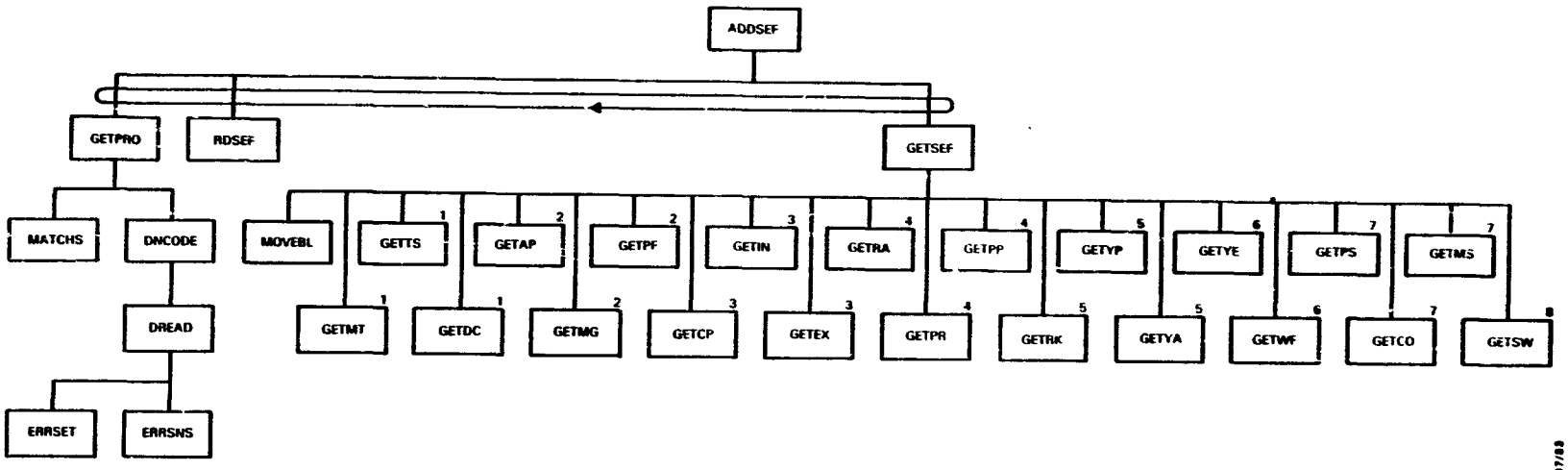


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (1 of 16)



[1] SEE FIGURE A-11, PAGE 3 OF 16.
[2] SEE FIGURE A-11, PAGE 4 OF 16.
[3] SEE FIGURE A-11, PAGE 5 OF 16.
[4] SEE FIGURE A-11, PAGE 6 OF 16.
[5] SEE FIGURE A-11, PAGE 7 OF 16.
[6] SEE FIGURE A-11, PAGE 8 OF 16.
[7] SEE FIGURE A-11, PAGE 9 OF 16.
[8] SEE FIGURE A-11, PAGE 10 OF 16.

Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (2 of 16)

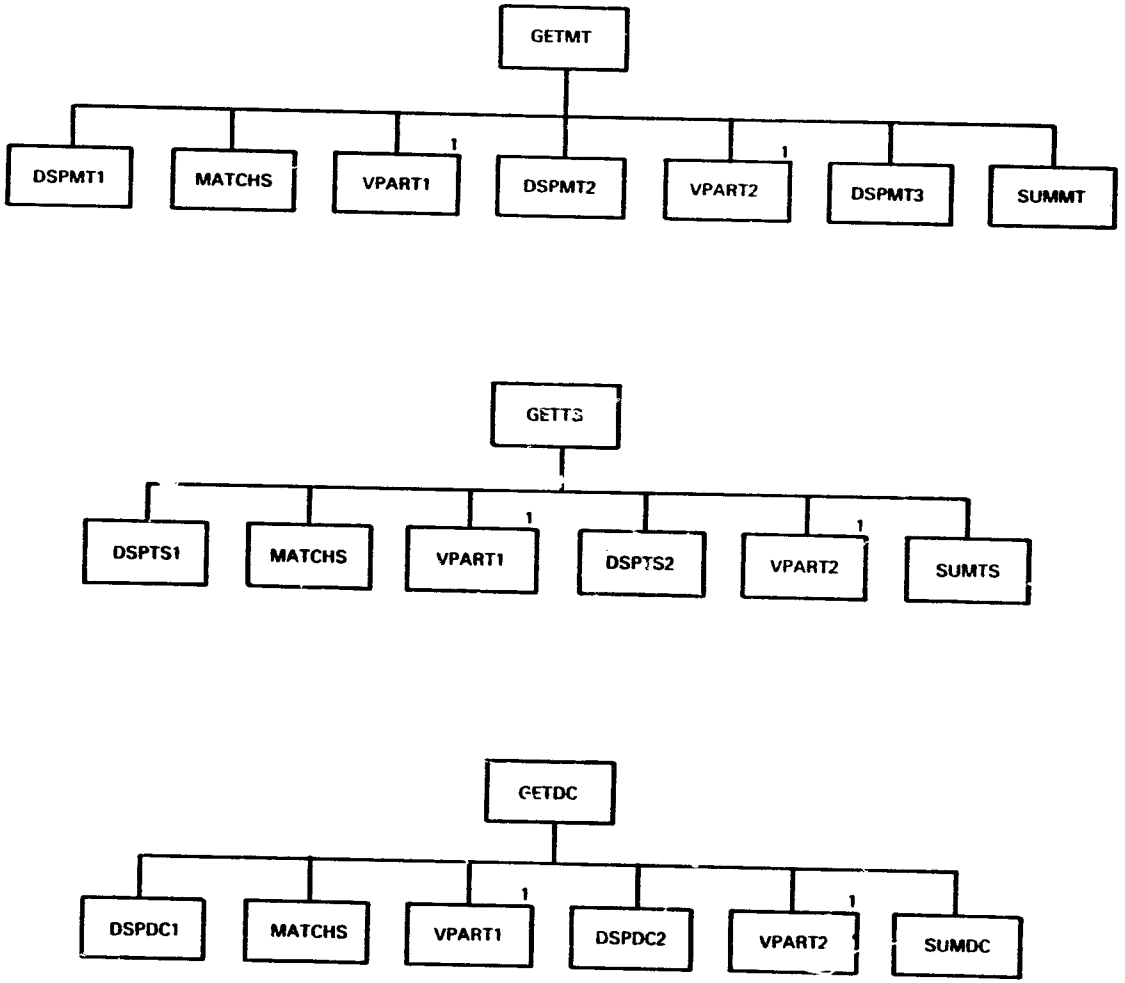


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (3 of 16)

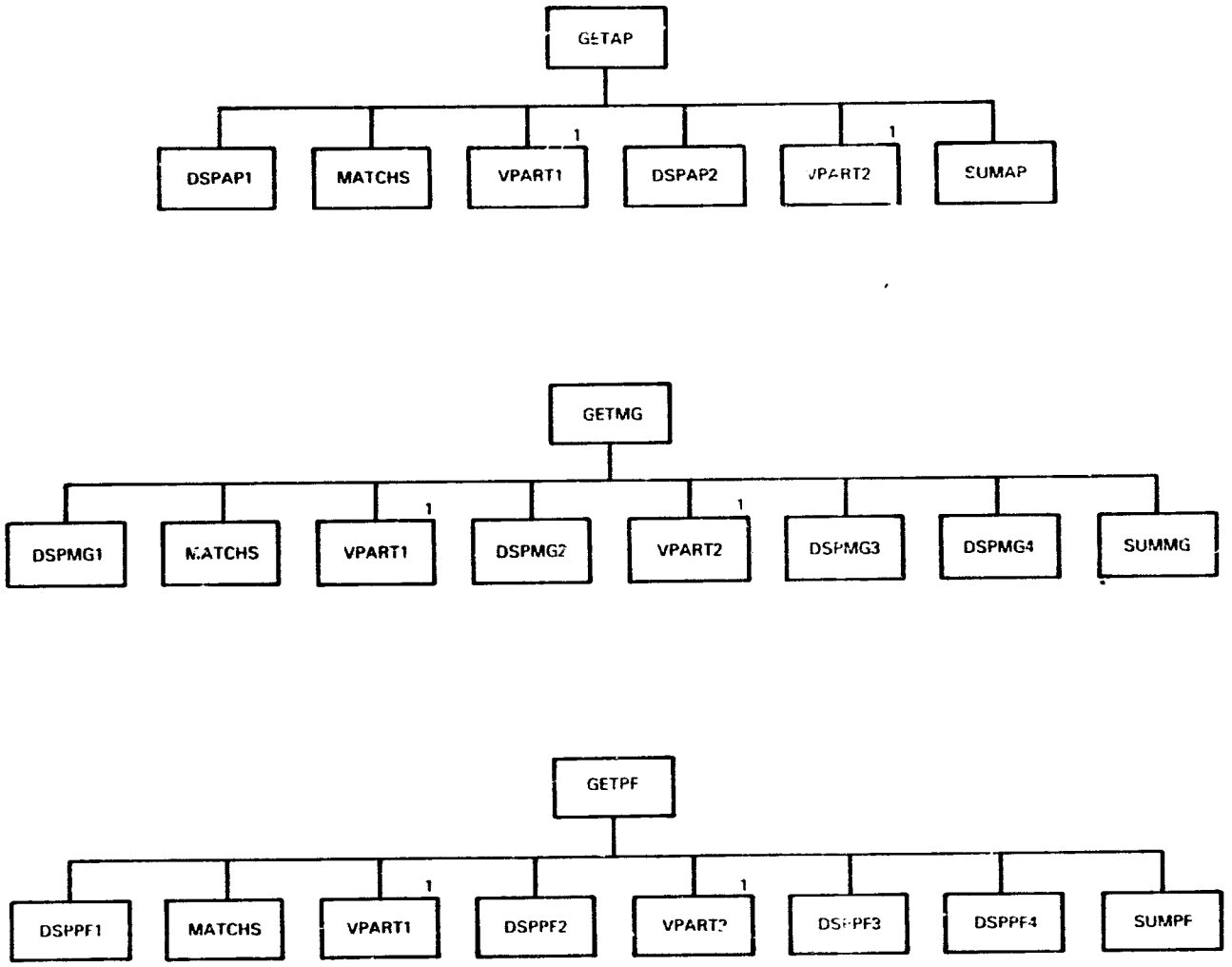


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (4 of 16)

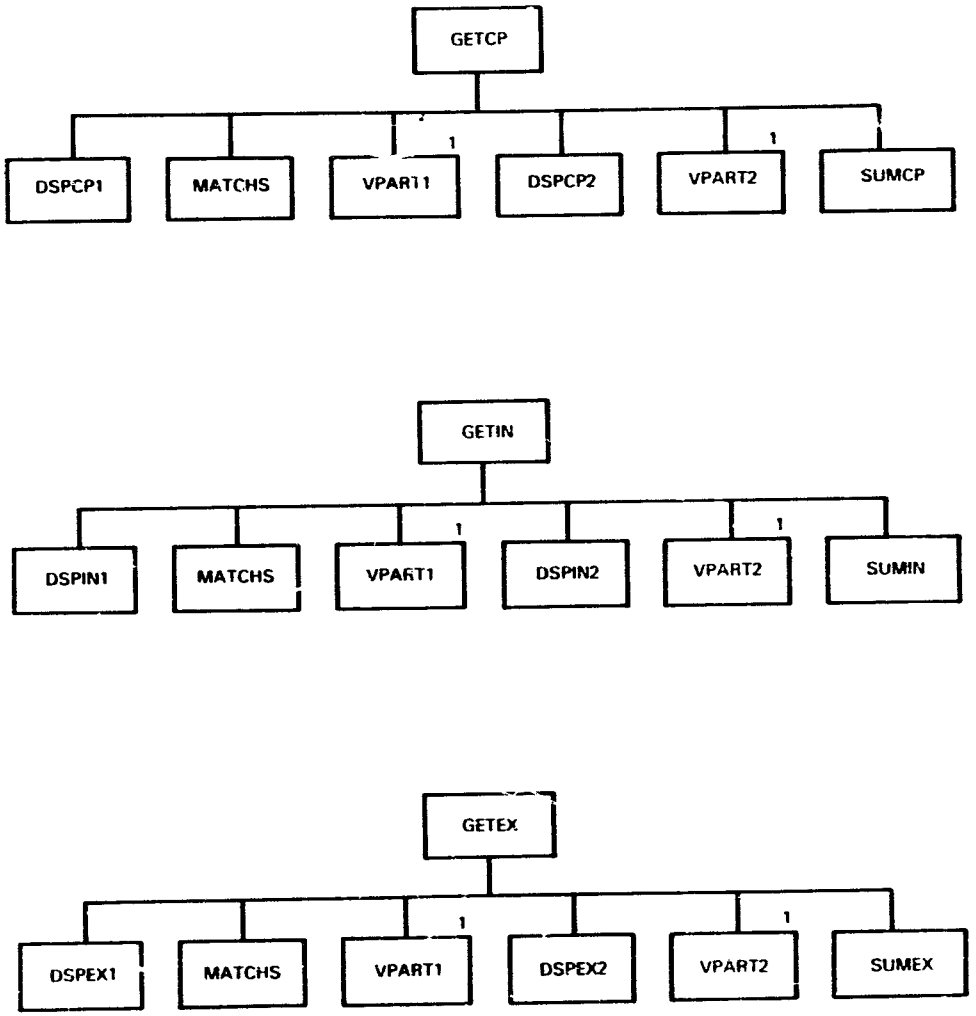


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (5 of 16)

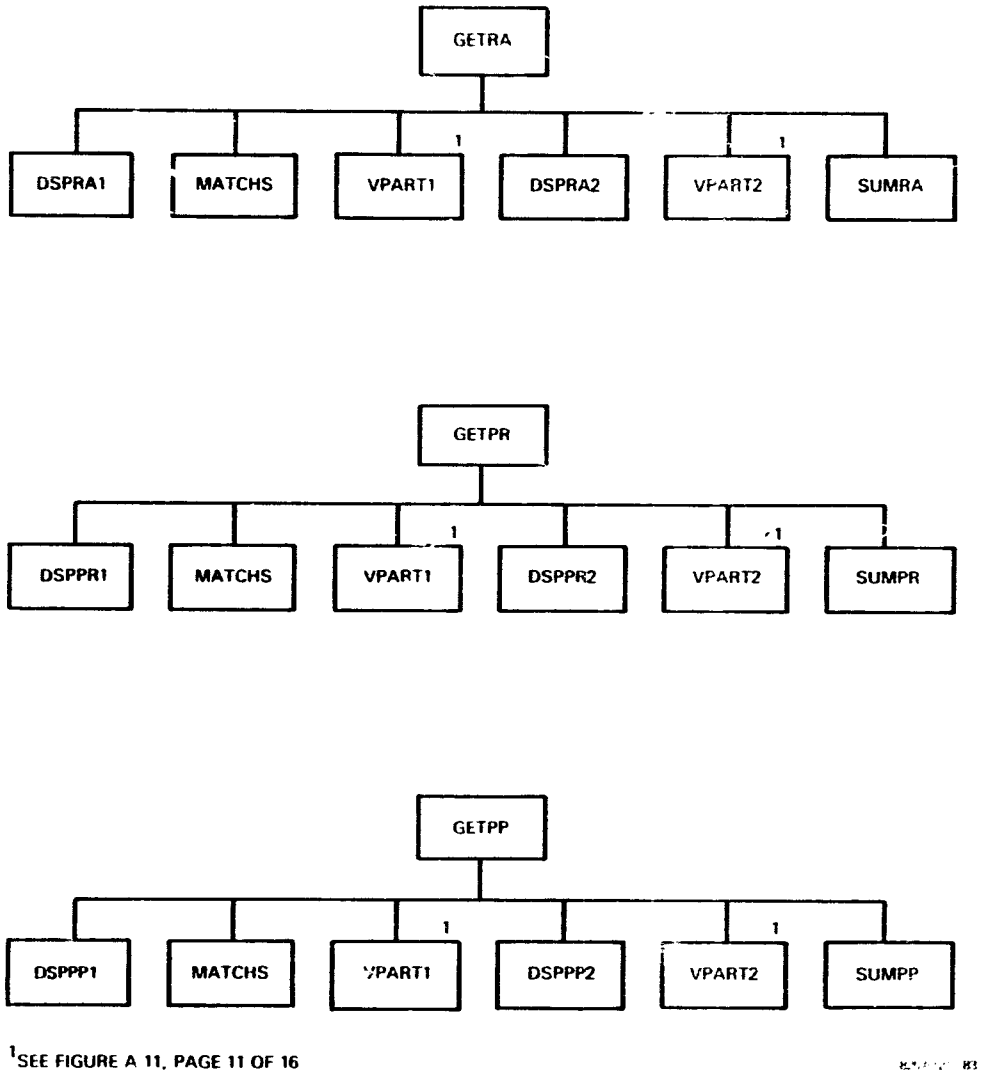


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (6 of 16)

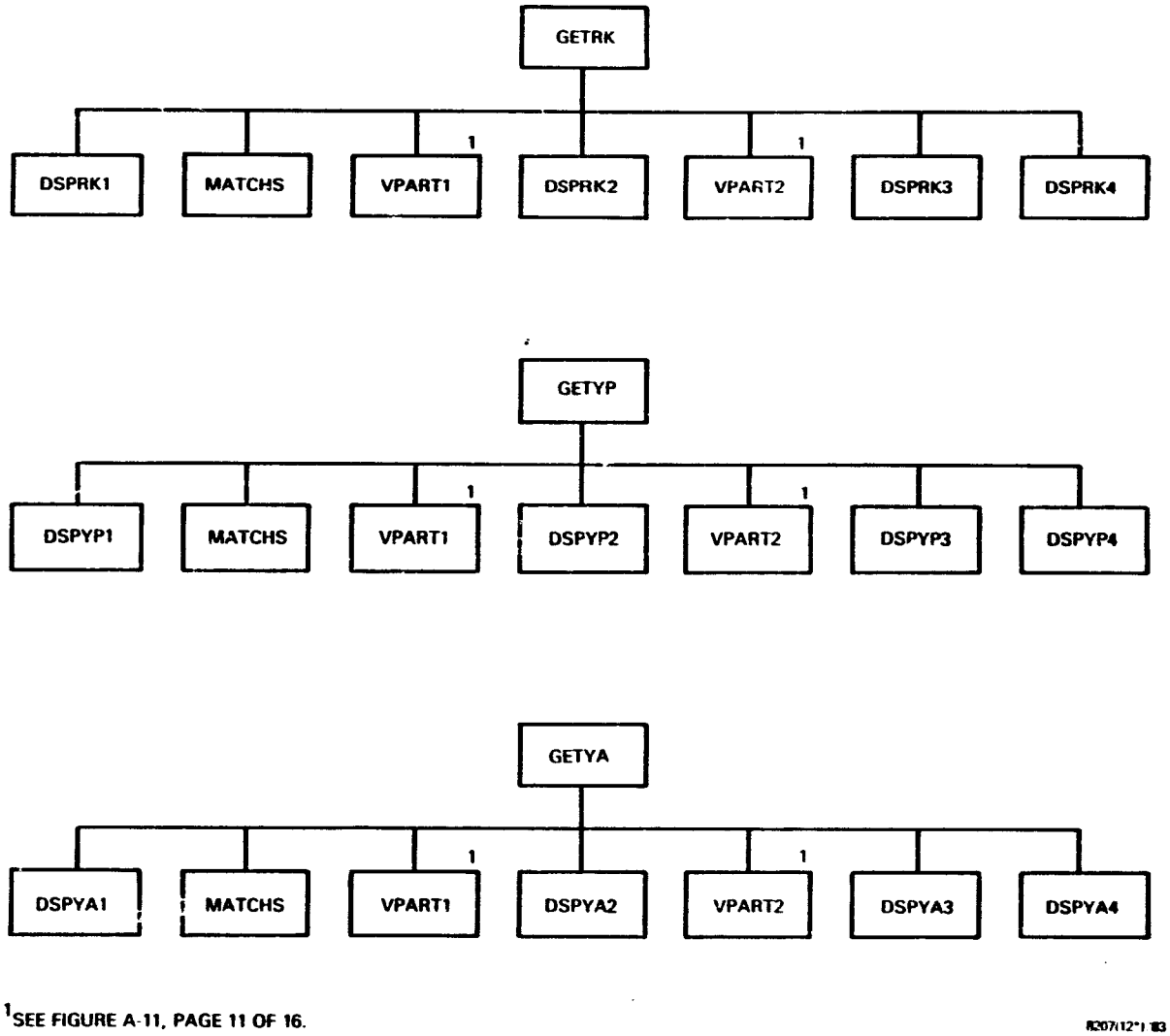


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (7 of 16)

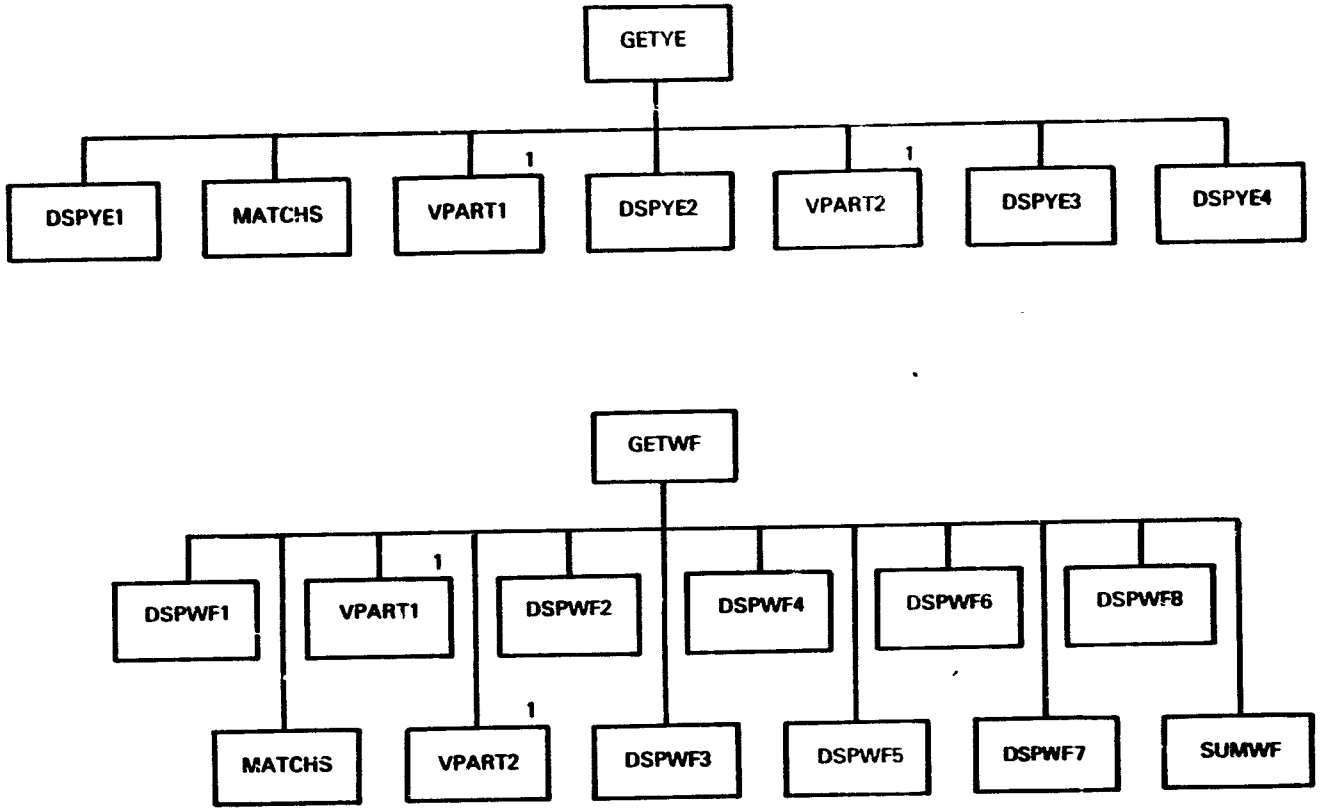


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (8 of 16)

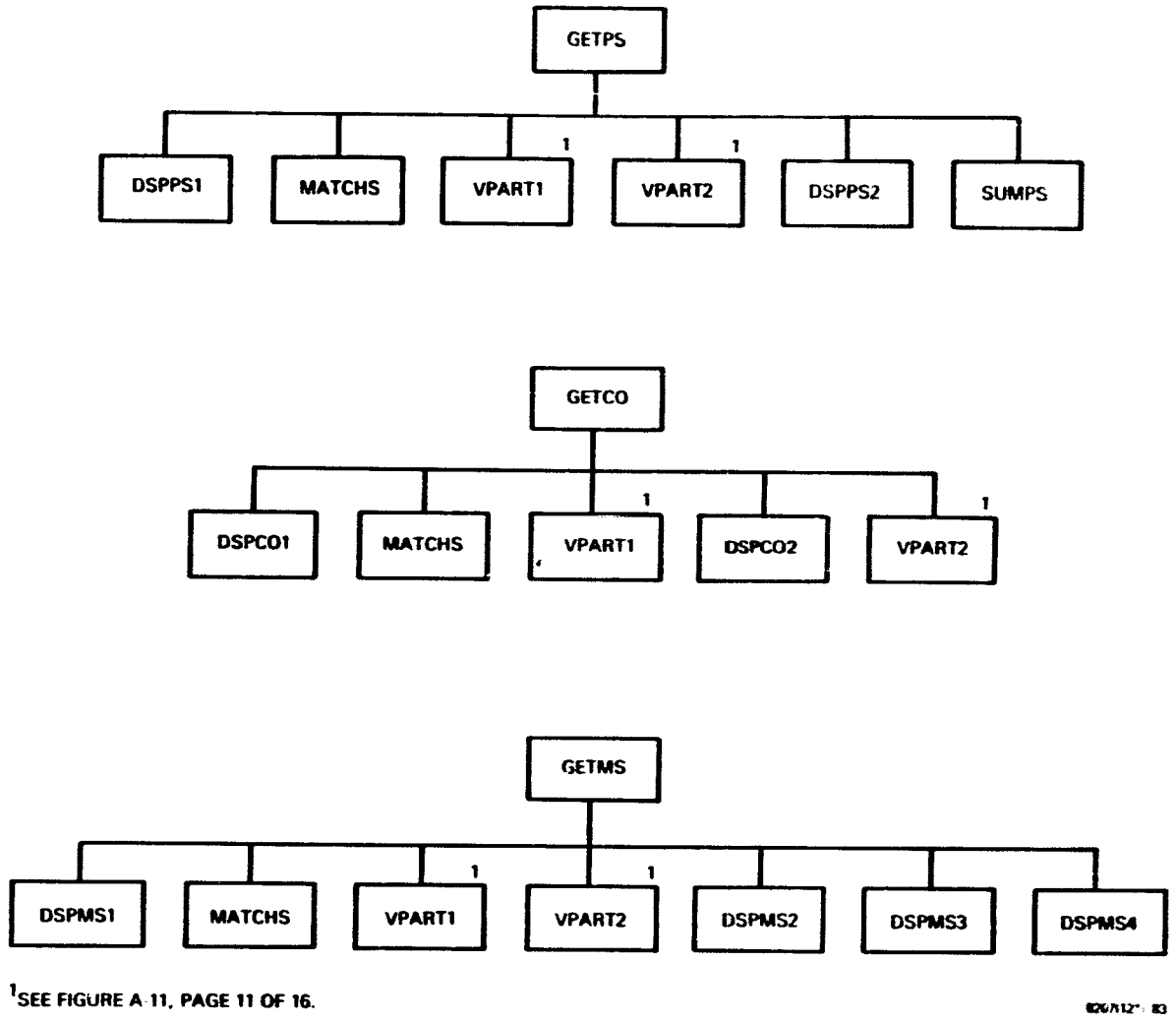
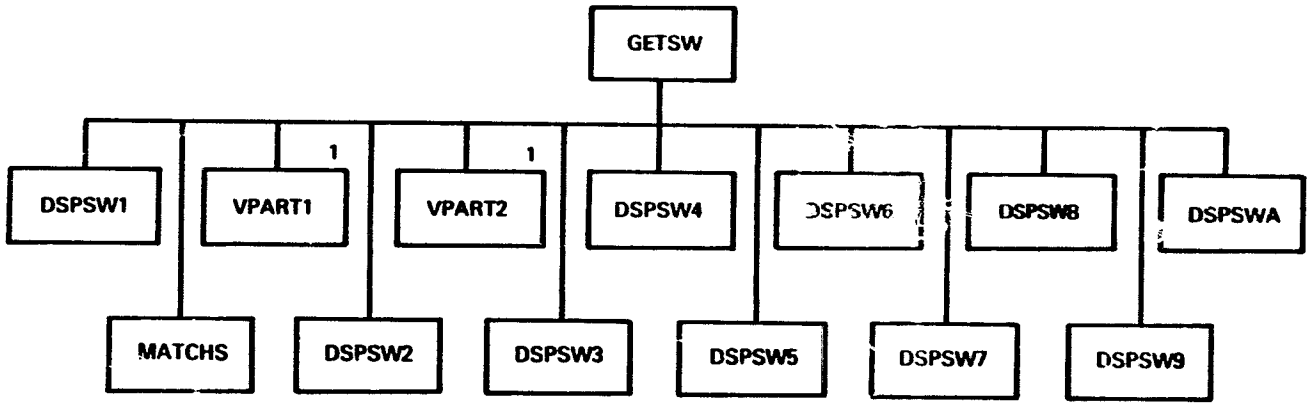


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (9 of 16)



[1]SEE FIGURE A-11, PAGE 11 OF 16.

Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (10 of 16)

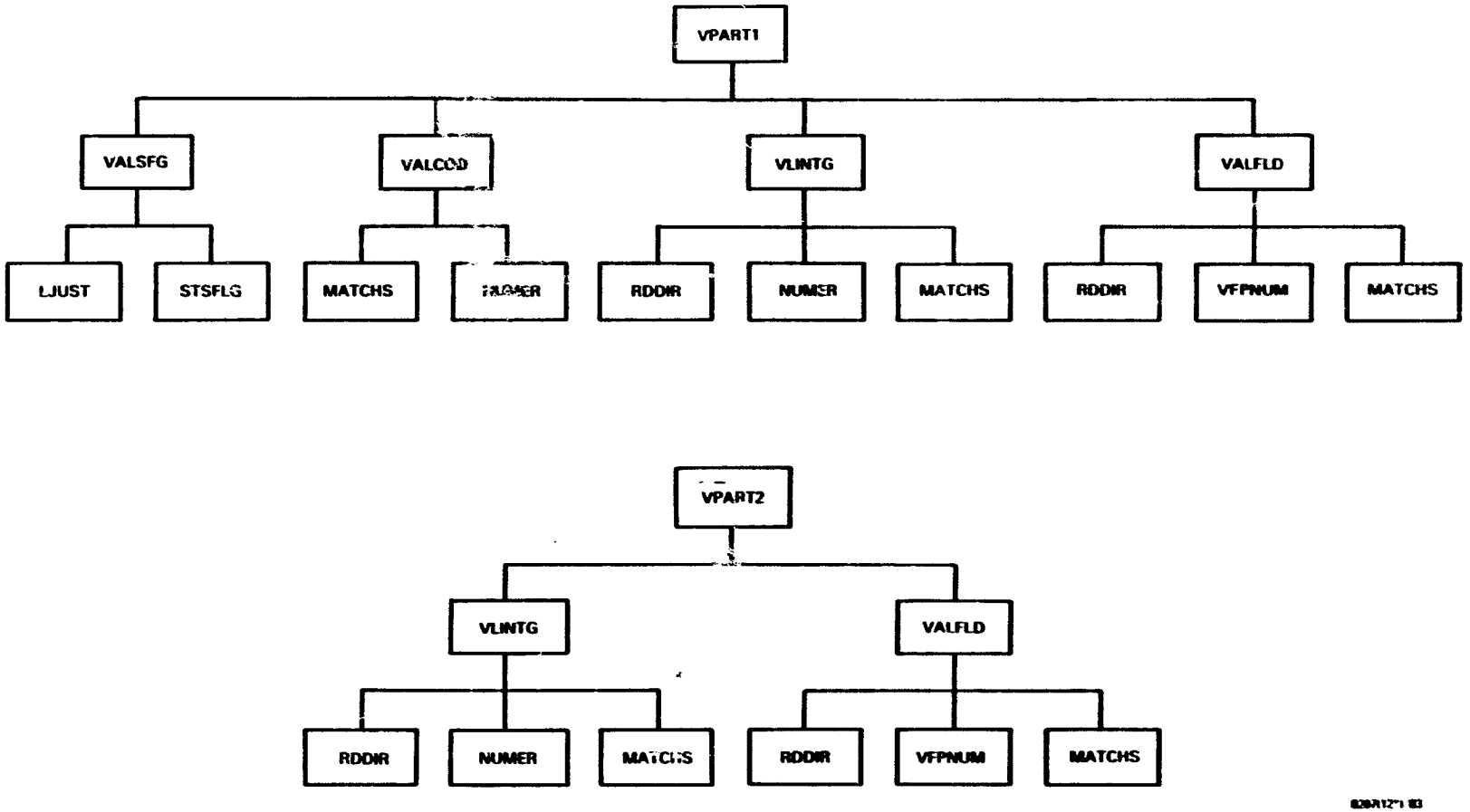
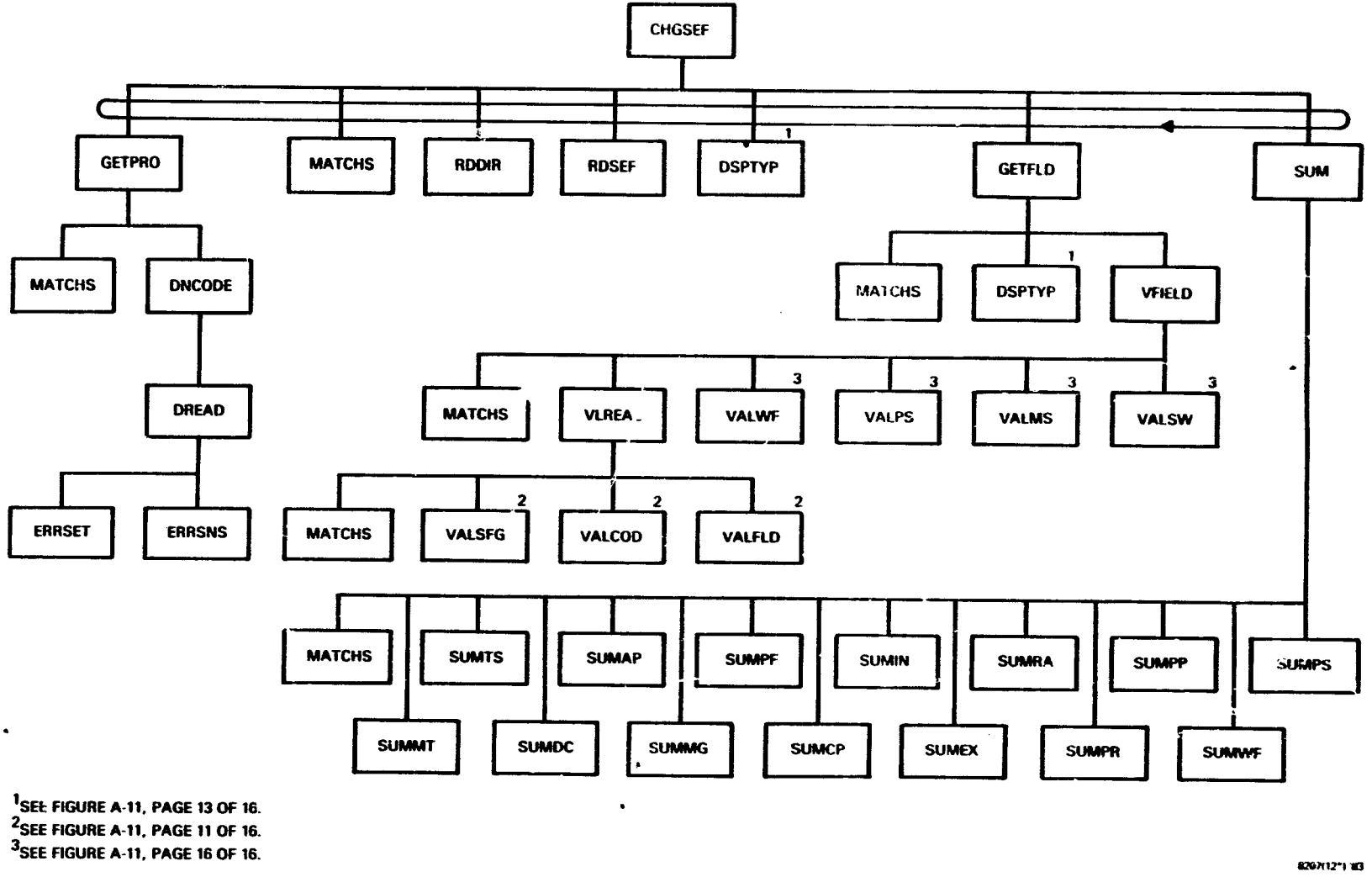
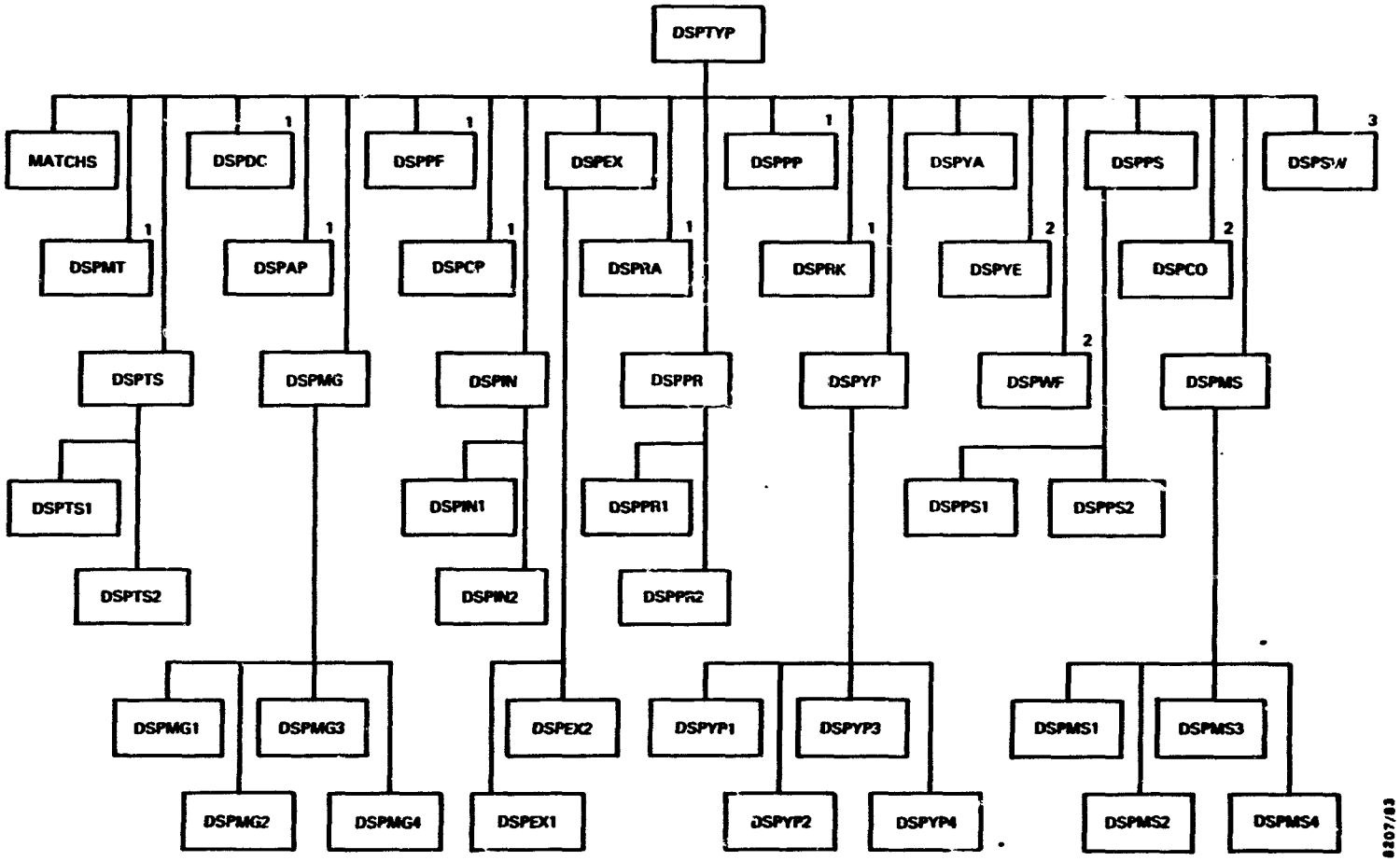


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (11 of 16)



[1]SEE FIGURE A-11, PAGE 13 OF 16.
[2]SEE FIGURE A-11, PAGE 11 OF 16.
[3]SEE FIGURE A-11, PAGE 16 OF 16.

Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (12 of 16)



[1]SEE FIGURE A-11, PAGE 14 OF 16.

[2]SEE FIGURE A-11, PAGE 15 OF 16.

[3]SEE FIGURE A-11, PAGE 16 OF 16.

Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (13 of 16)

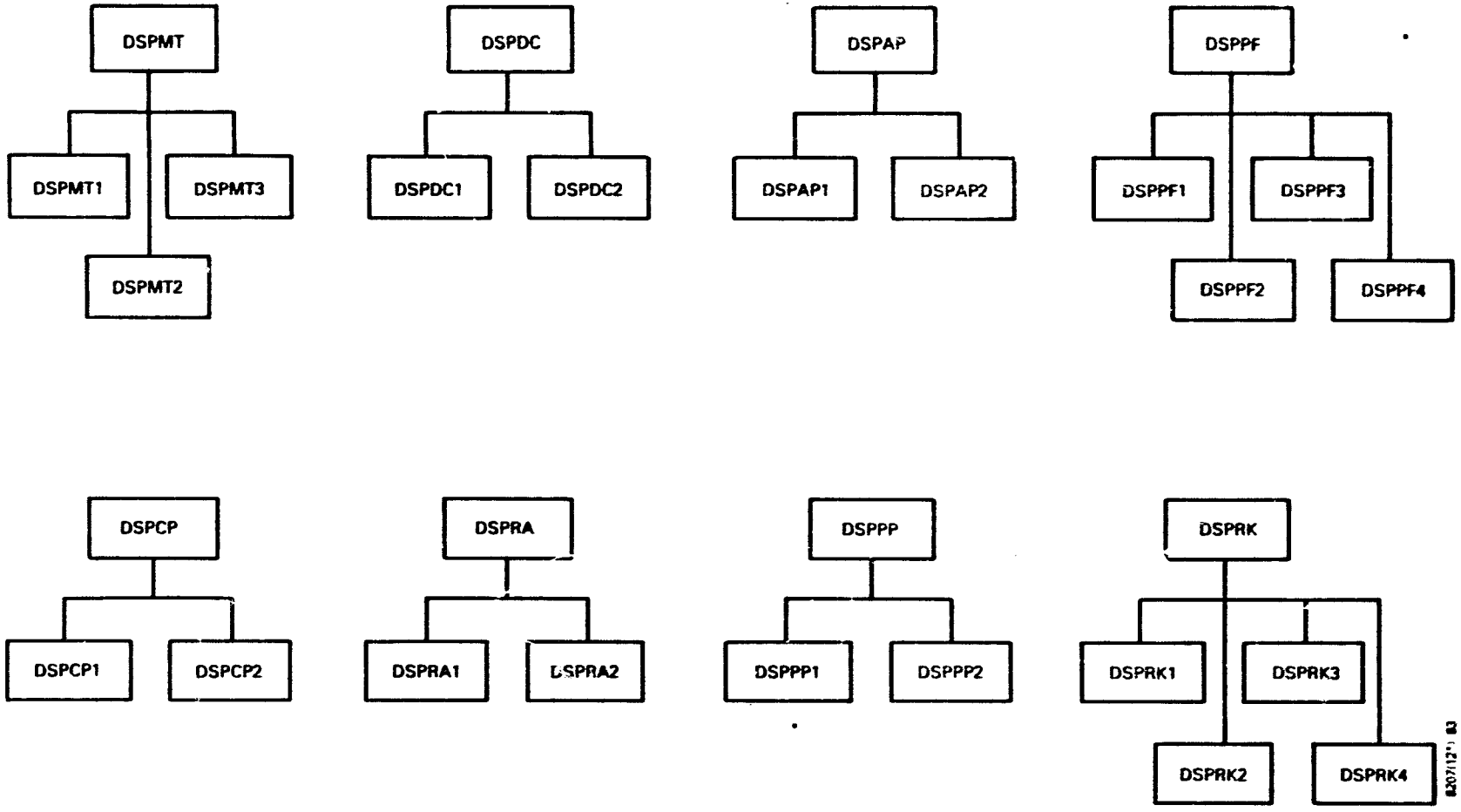


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (14 of 16)

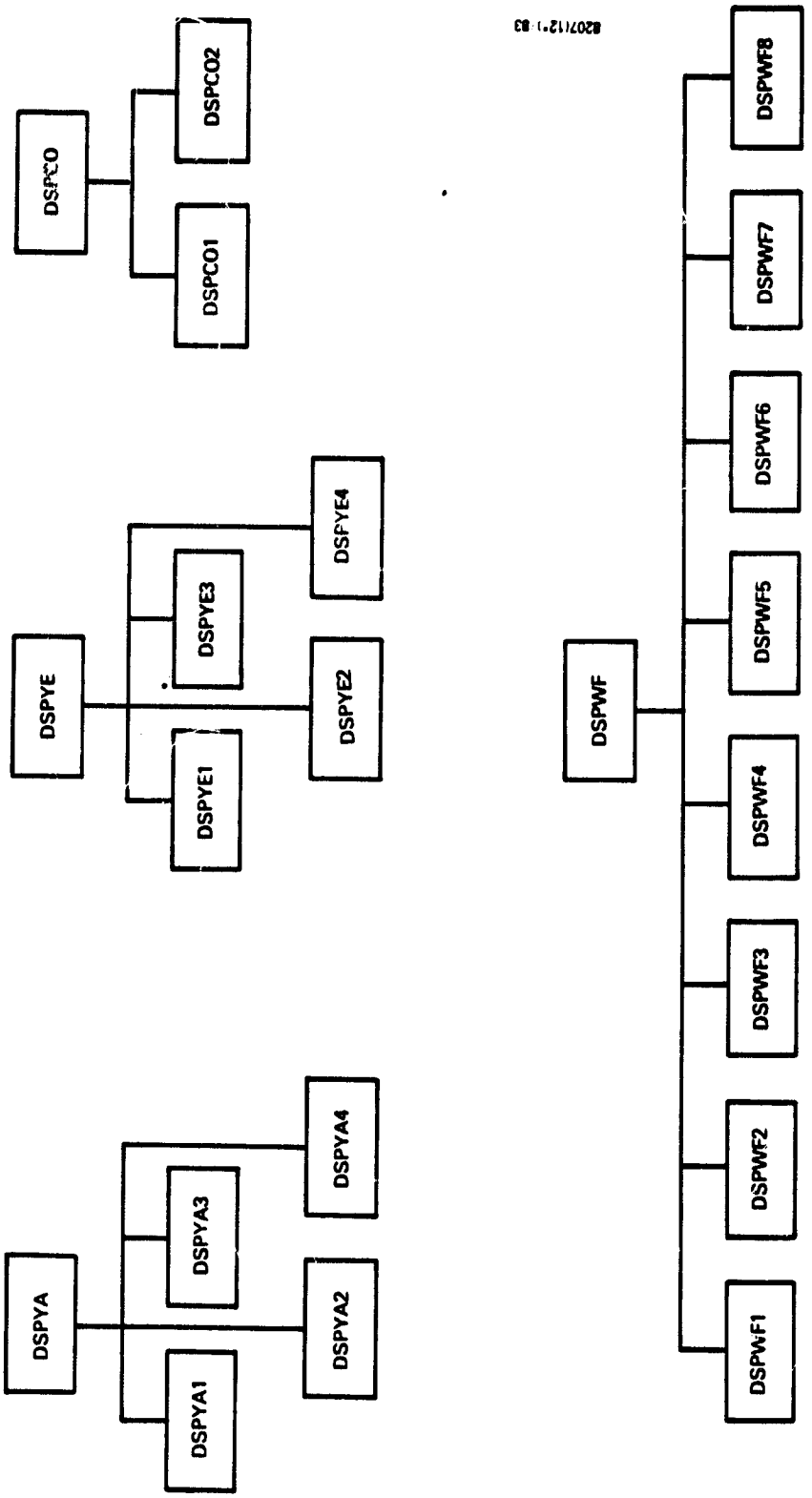


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (15 of 16)

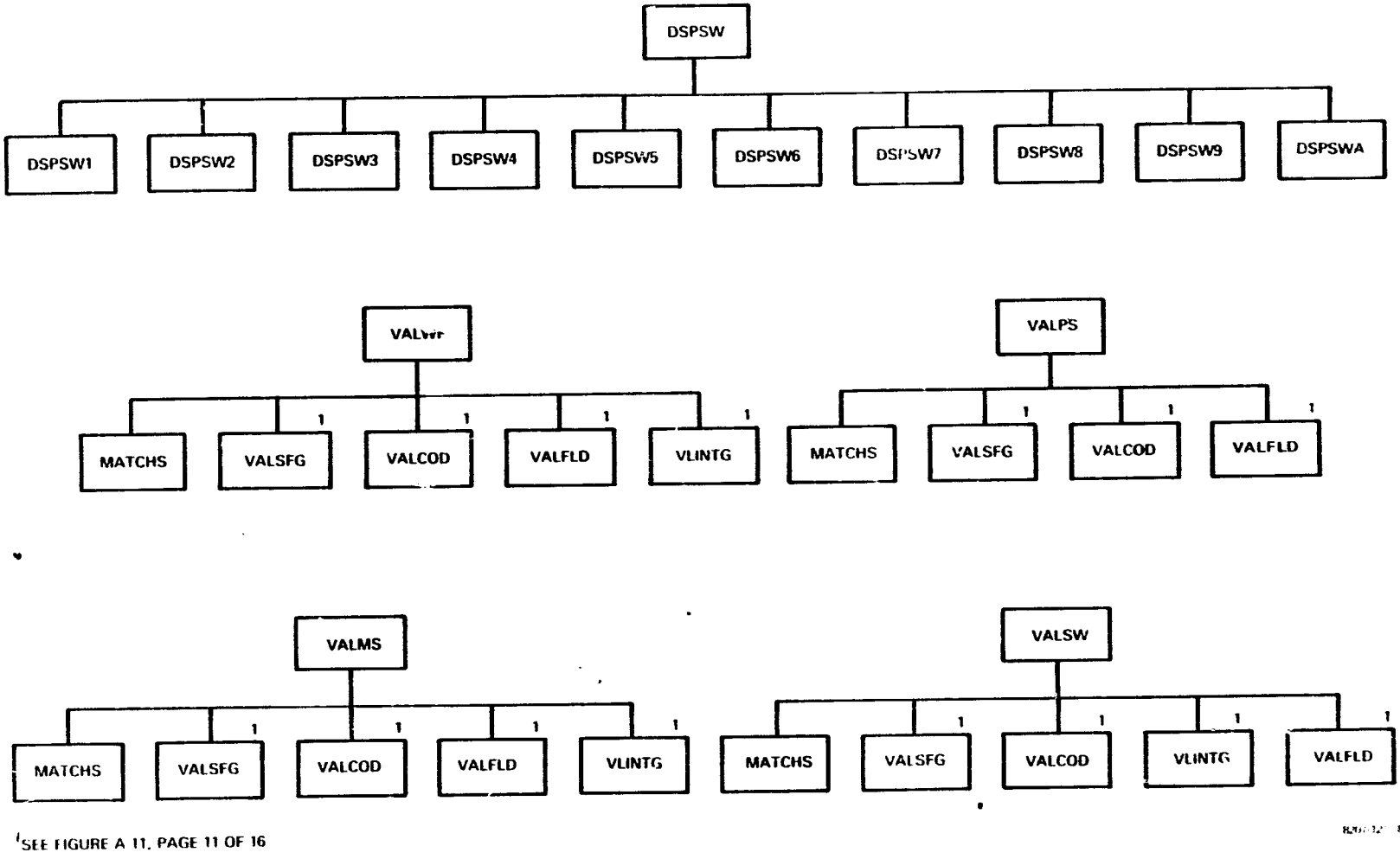


Figure A-11. Baseline Diagram for Subjective Evaluations File Update Function (16 of 16)

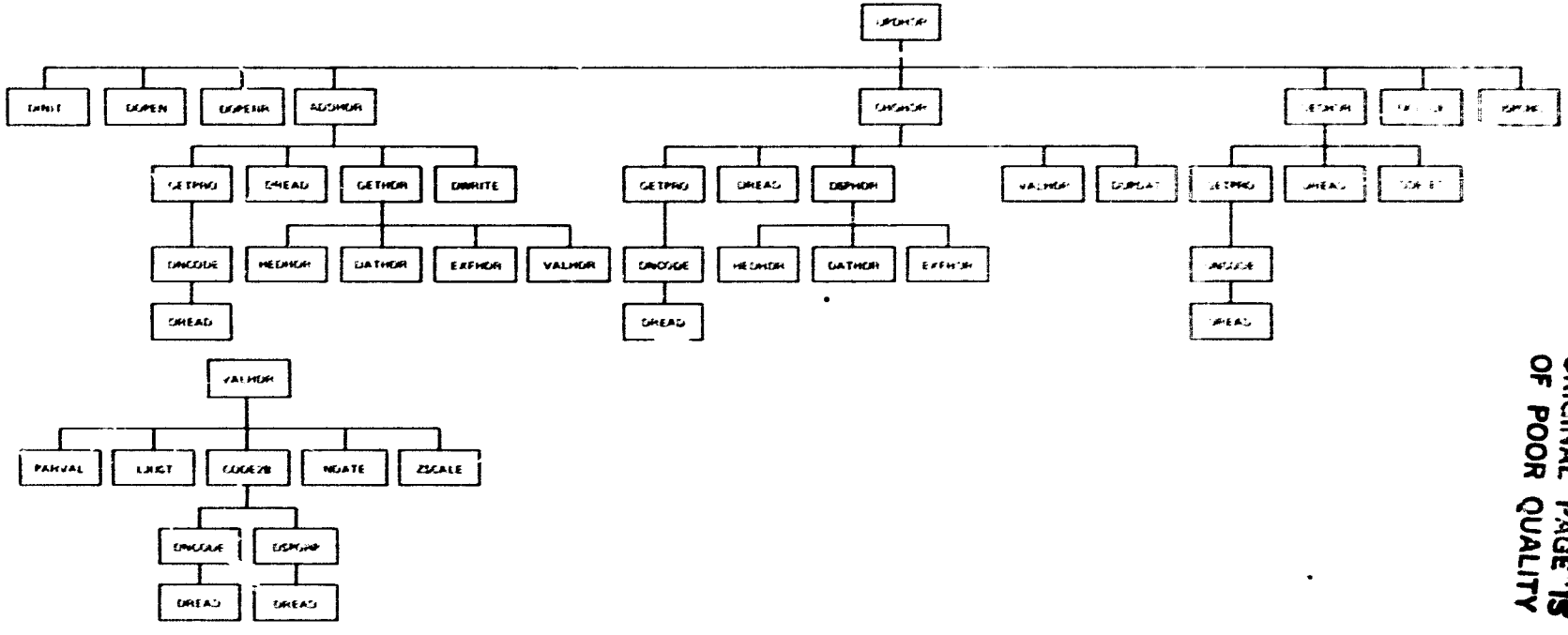


Figure A-12. Baseline Diagram for Phase Dates (Header) Update Subfunction

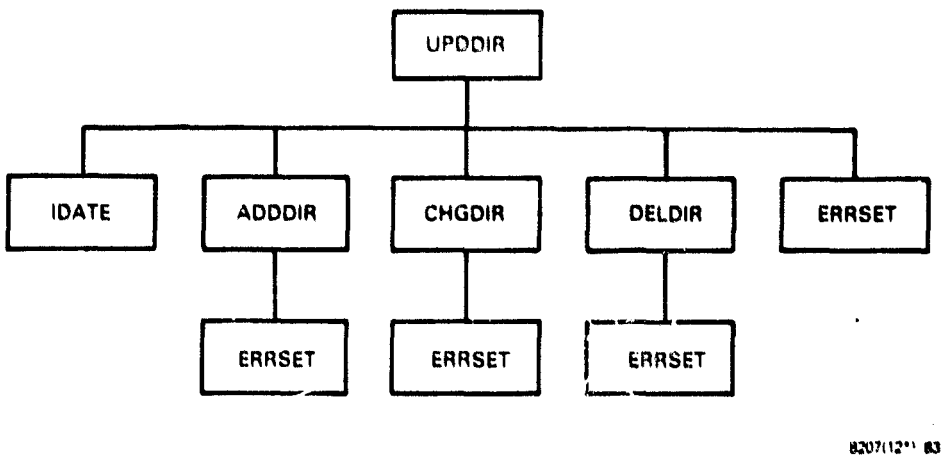


Figure A-13. Baseline Diagram for Subjective Evaluations Directory File Update Function

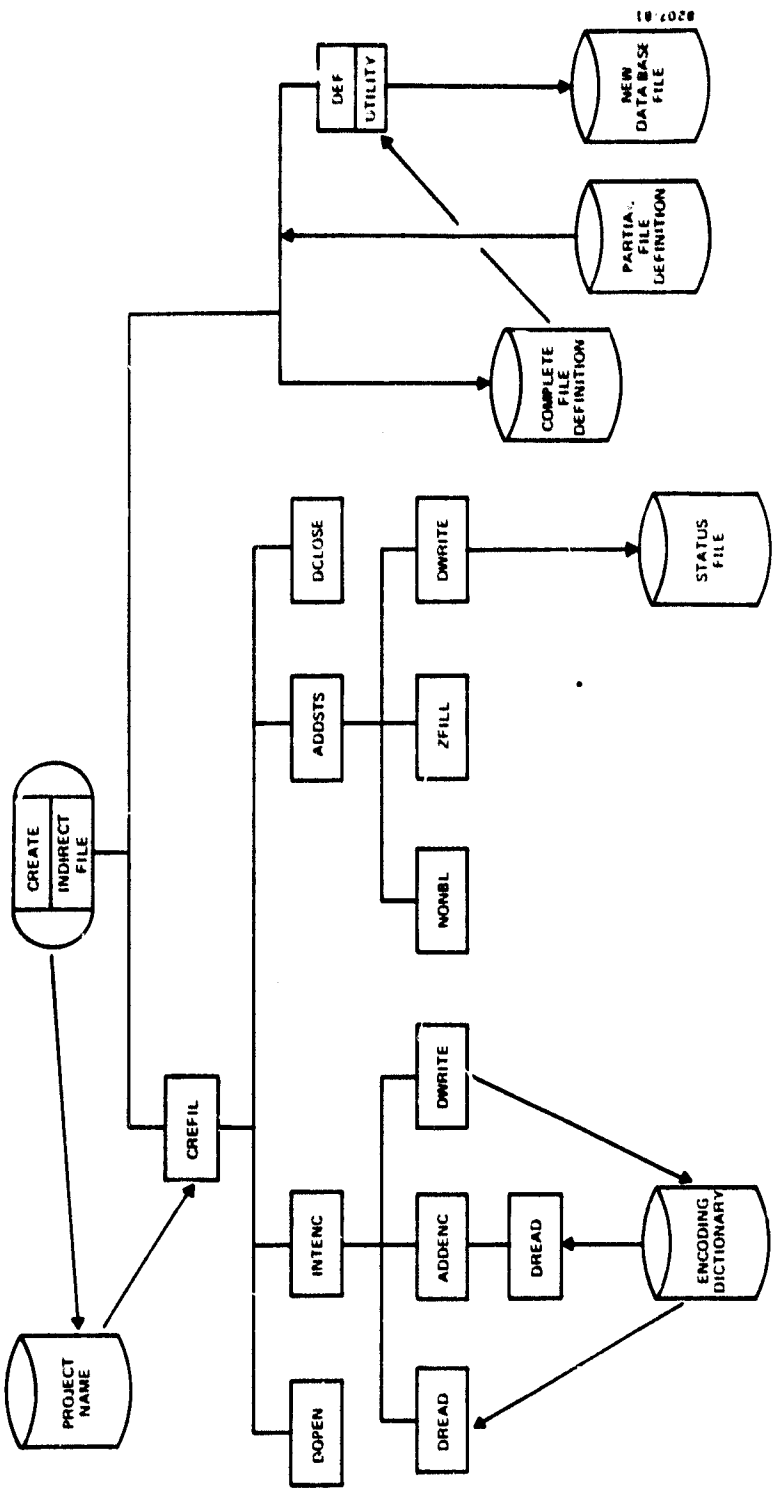


Figure A-14. Baseline Diagram for Create Function

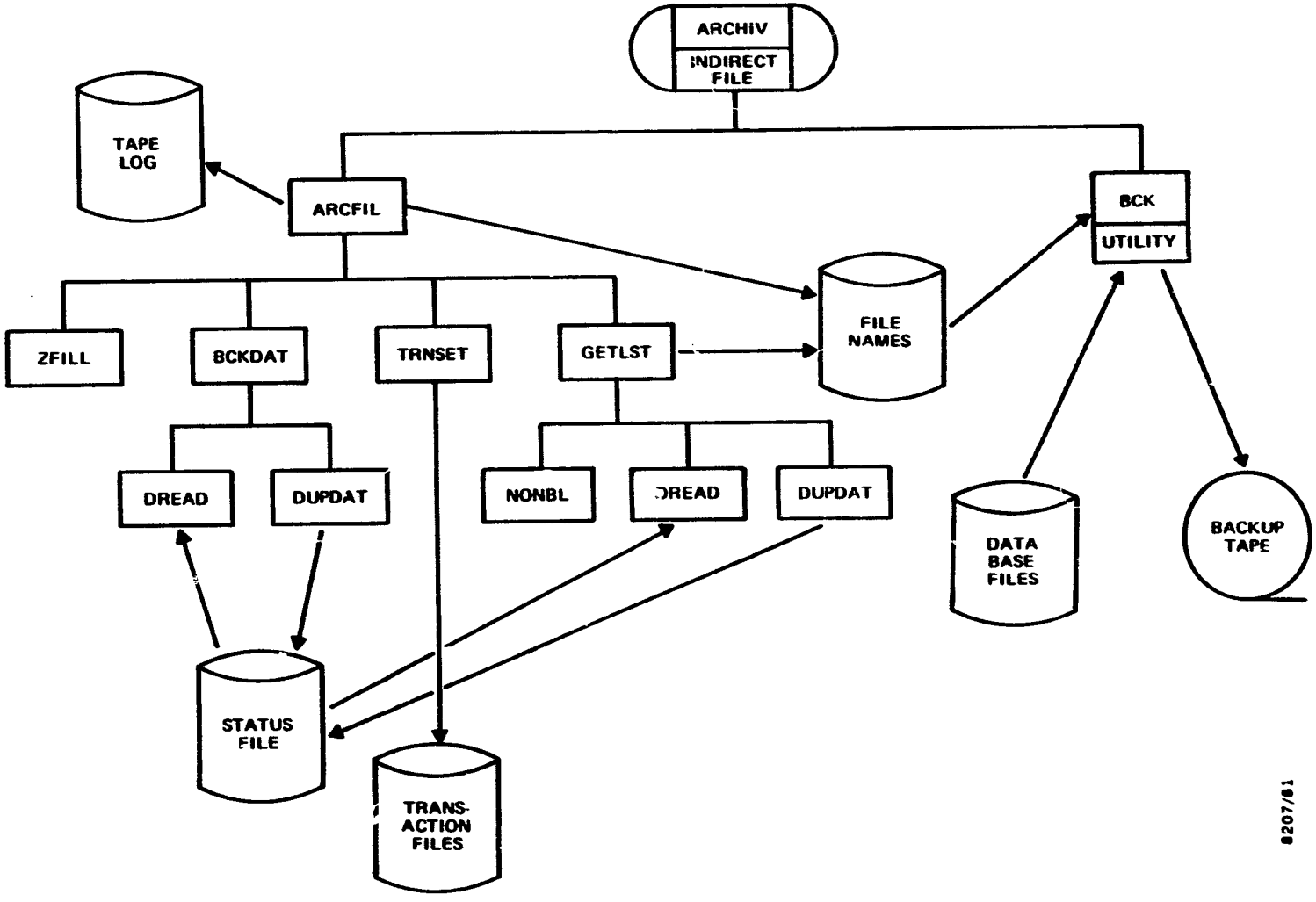


Figure A-15. Baseline Diagram for Archive Function

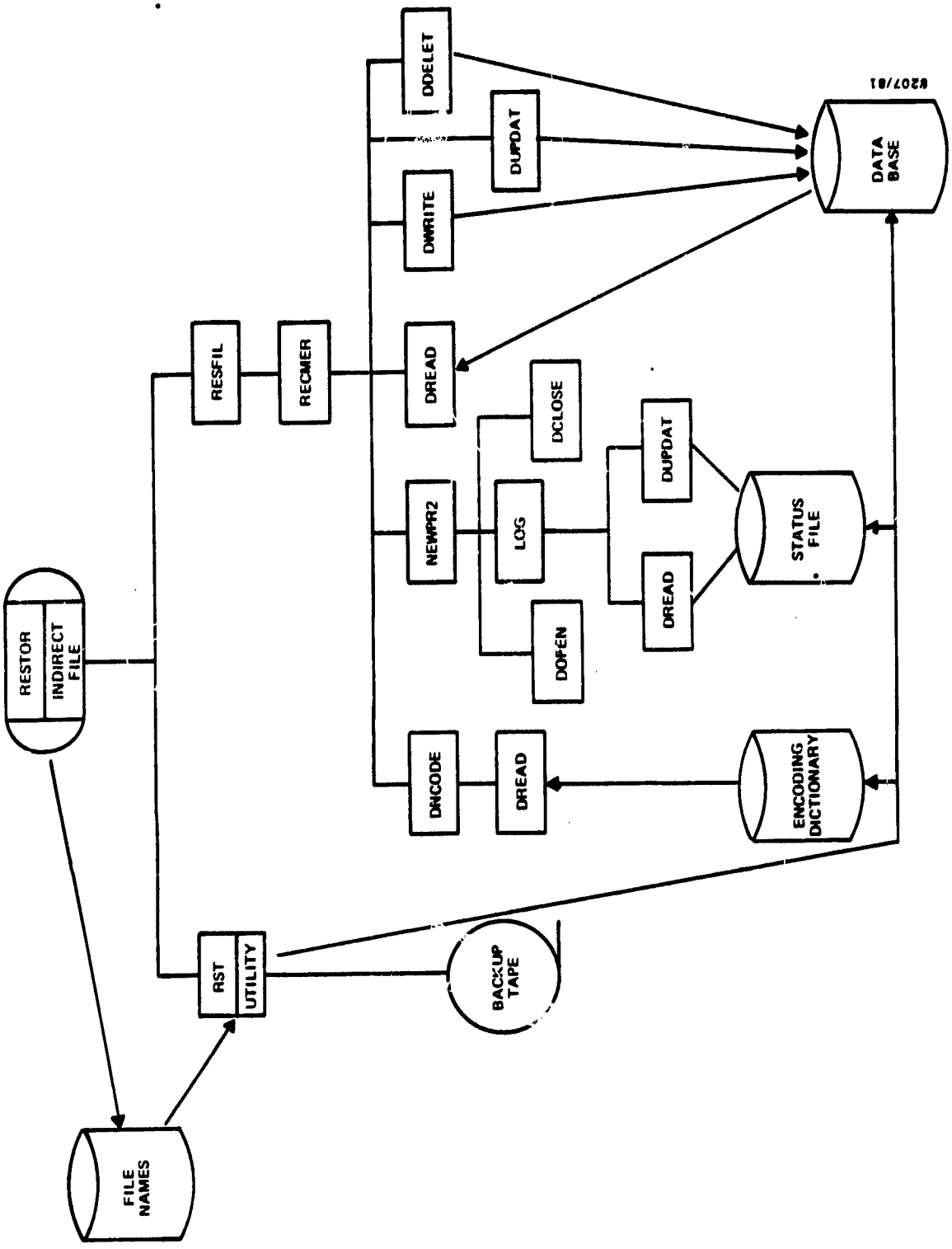


Figure A-16. Baseline Diagram for Restore Function

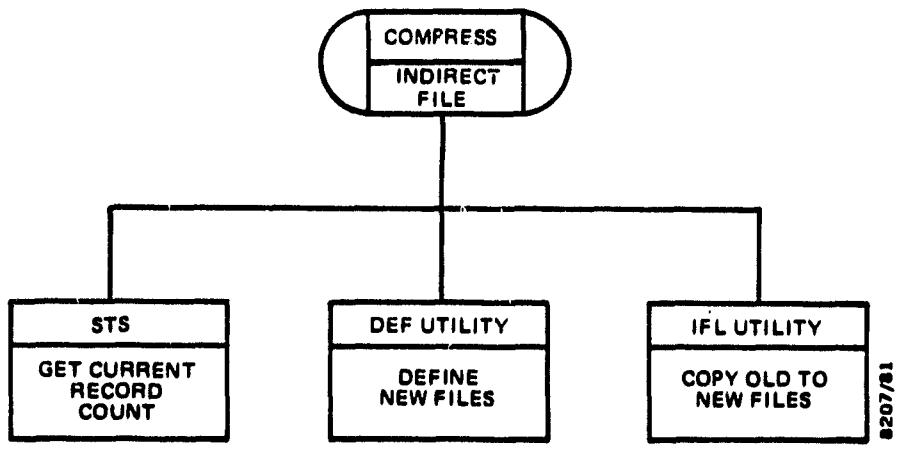


Figure A-17. Baseline Diagram for Compress Function

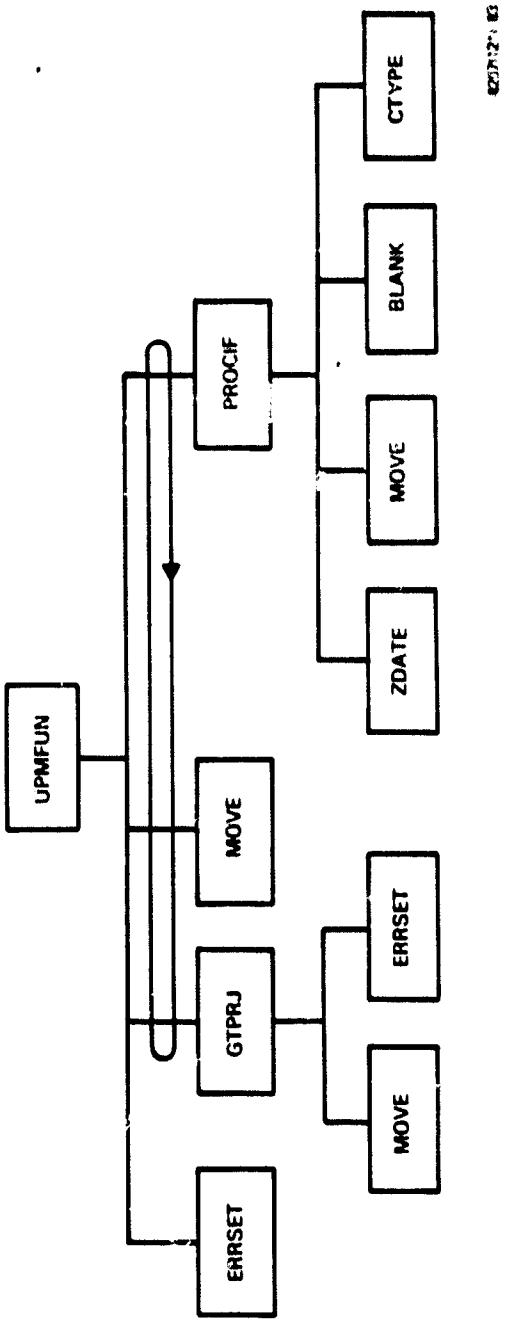


Figure A-18. Baseline Diagram for FORTRAN Module Function of the Component Information File Update Function

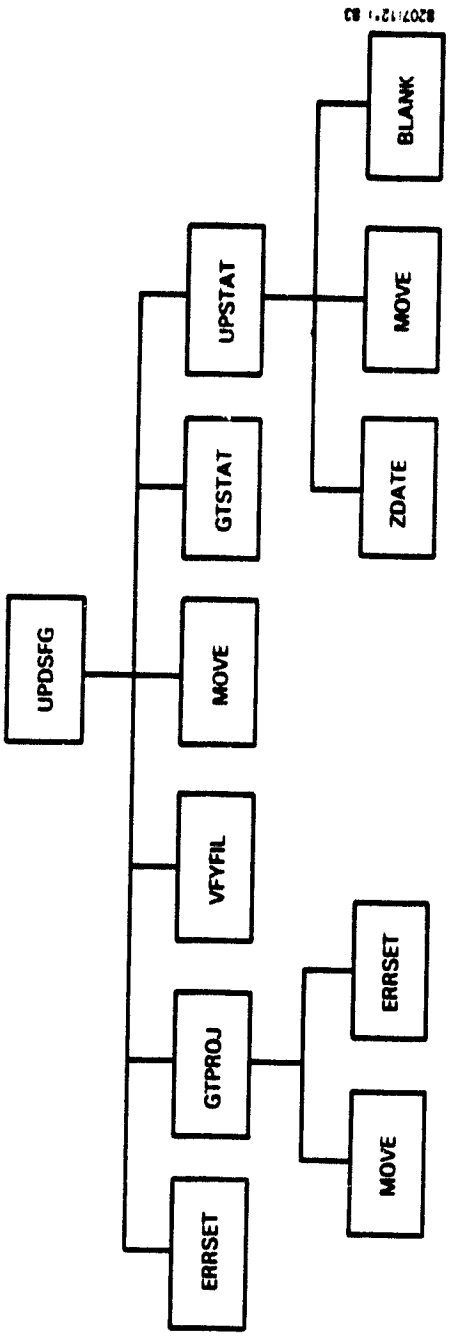


Figure A-19. Baseline Diagram for Status Flag Update Function

APPENDIX B - COMPONENT DESCRIPTIONS

Appendix B contains brief descriptions of all components written for DBAM. These are organized by subfunction (task image). Those components that are common to several subfunctions are listed in the last table.

Table B-1. UPDCSF Component Descriptions

Module	Description
ADDCSF	ADD mode driver for CSF; controls data set access
APCMT	Adds comments to comment file
CHGCSF	CHANGE mode driver for CSF; controls data set access
DELCSF	DELETE mode driver for CSF; controls data set access; deletes identified record
DSACSF	Displays first segment of CSF record
DSBCSF	Displays second segment of CSF record
DSCCSF	Displays third segment of CSF record
DSDCSF	Displays fourth segment of CSF record
DSECSF	Displays fifth segment of CSF record
DSFCSF	Displays sixth segment of CSF record
DSPCSF	Display driver for CSF data
EMHORB	Verifies that a 1-byte field contains "E", "M", "H", or a blank
FLDCSF	Prompts for changes and rewrites a CSF record
GETCRF	Prompts for and validates the project name and form number
NUCORB	Verifies that a 1-byte field contains "N", "U", "C", or a blank
PROCSF	Constructs and adds a new CSF record from operator input
UPDCSF	Update Component Summary Form File subfunction driver; selects mode
VALCSF	Passes parameters from validation control tables
VALDTX	Validation driver; selects validation type

Table B-2. UPDCRF Component Descriptions

Module	Description
ADDCRF	ADD mode driver for CRF; controls data set access
CHGCRF	CHANGE mode driver for CRF; controls data set access
DELCRF	DELETE mode driver for CRF; controls data set access; deletes identified record
DSACRF	Displays first segment of CRF record
DSBCRF	Displays second segment of CRF record
DSCCRF	Displays third segment of CRF record
DSDCRF	Displays fourth segment of CRF record
FLDCRF	Identifies fields to be changed in the CRF record
GETCRF	Prompts for and validates project name and form number
PROCRF	Drives CRF record updates
UPDCRF	Update Change Report Form File subfunction driver; selects mode
VLACRF	Validates first segment of CRF data
VLBCRF	Validates second segment of CRF data
VLCCRF	Validates third segment of CRF data
VLDCRF	Validates fourth segment of CRF data

Table B-3. UPDCIF Component Descriptions

Module	Description
ADDCIF	ADD mode driver for CIF; controls data set access
CHGCIF	CHANGE mode driver for CIF; controls data set access
CODCIF	Finds an unused component code in the CIF
DELCIF	DELETE mode driver for CIF; controls data set access; deletes identified record
DSACIF	Displays first segment of CIF record
DSBCIF	Displays second segment of CIF record
DSCCIF	Displays third segment of CIF record
FLDCIF	Prompts for changes and rewrites a CIF record
GETCIF	Prompts for and validates project name and component name
MOVCIF	Moves data from a SAP record to a CIF record
NAMCIF	Reads and validates project name and component name from SAP record
PROCIF	Constructs and adds a new CIF record from operator input
REDCIF	Reads SAP records and matches them with CIF records
UPDCIF	Update Component Information File subfunction driver; selects mode
VALCIF	CIF validation driver
VLACIF	Validates first segment of CIF data
VLBCIF	Validates second segment of CIF data
VLCCIF	Validates third segment of CIF data

Table B-4. UPDCSR Component Descriptions

Module	Description
ADDCSR	ADD mode driver for CSR; controls data set access
CHGCSR	CHANGE mode driver for CSR; controls data set access
DELCSR	DELETE mode driver for CSR; controls data set access; deletes indicated record
DSPCSR	Display driver for CSR data
FLDCSR	Prompts for changes and rewrites a CSR record
FORCSR	Displays titles for data part of CSR record
GETCSR	Prompts for and validates project name form number, and date
HEDCSR	Displays titles for record identification part of CSR record
PROCSR	Constructs and adds a new CSR record from operator input
UPDCSR	Update Component Status Report File subfunction driver; selects mode
VALACT	Validates other-activity field of CSR record
VALCSR	Validates data fields of CSR record

Table B-5. UPDRAF Component Descriptions

Module	Description
ADDRAF	ADD mode driver for RAF; controls data set access
CHGRAF	CHANGE mode driver for RAF, controls data set access
DATRAF	Displays titles for data part of RAF record
DELRAF	DELETE mode driver for RAF; controls data set access; deletes identified record
DSPRAF	Display driver for RAF data
FLDRAF	Prompts for changes and rewrites an RAF record
GETRAF	Prompts for and validates project name, form number, programmer, and machine
HEDRAF	Displays titles for record identification part of RAF record
PRORAF	Constructs and adds an RAF record from operator input
UPDRAF	Update Run Analysis Form File subfunction driver; selects mode
VALCMP	Verifies that a character string is a valid component name
VALDTY	RAF validation driver; selects validation type
VALRAF	Passes parameters from validation control tables

Table B-6. UPDRSF Component Descriptions

Module	Description
ADDRSF	ADD mode driver for RSF; controls data set access
CHGRSF	CHANGE mode driver for RSF; controls data set access
COMRSF	Displays titles for RSF computer usage data
DATRSF	Displays titles for RSF manpower and other hours data
DELRSF	DELETE mode driver for RSF; controls data set access; deletes identified record
DSPRSF	Display driver for RSF data
FLDRSF	Prompts for changes and rewrites an RSF record
GETFRI	Calculates dates at 1-week intervals from a starting date
GETRES	Prompts driver for RSF add mode
GETRSF	Prompts for and validates project name and form number
GETVAL	Terminal input driver for RSF ADD mode
HEDRSF	Displays titles for record identification area of RSF record
PRORSF	Constructs and adds a new RSF record from operator input
UPDRSF	Update Resource Summary Form File subfunction driver; selects mode
VALRSF	RSF validation driver; selects validation type

Table B-7. UPDHIS Component Descriptions

Module	Description
ADDHIS	ADD mode driver for HIS; controls data set access
CHGHIS	CHANGE mode driver for HIS; controls data set access
DATHIS	Displays titles for HIS data
DELHIS	DELETE mode driver for HIS; controls data set access; deletes identified record
DSPHIS	Display HIS record
FLDHIS	Prompts for changes and rewrites HIS record
PROHIS	Constructs and adds new HIS record from operator input
UPDHIS	Update Growth History File subfunction driver; selects mode
VALHIS	HIS validation driver; selects validation type

Table B-8. UPDSTS Component Descriptions

Module	Description
CHGSTS	Lower-level driver for STS update activities
DSPSTS	Displays an STS record
FLDSTS	Prompts for changes and rewrites or deletes an STS record
GETCOD	Retrieves the file and project codes from the Encoding Dictionary
GETSTS	Reads and displays a specified STS record
UPDSTS	Update File Name and Status File sub-function driver; opens and closes data sets
VALSTS	STS validation driver; selects validation type

Table B-9. UPDENC Component Description

Module	Description
ADDENC	Constructs a new ENC record from operator input
CHGENC	Adds, rewrites, or deletes a specified ENC record
DSPENC	Displays an ENC record
FLDENC	Changes the description field of an ENC record
GETENC	Process ENC record
UPDENC	Update Encoding Dictionary subfunction driver

Table B-10. UPDEST Component Descriptions

Module	Description
ADDEST	Constructs and adds a new EST record from operator input
CHGEST	Prompts for changes and rewrites an EST record
DELEST	Deletes a specified EST record
DSPEST	Display driver for EST data
GETEST	Prompts for and receives EST data
HOREST	Displays titles for EST hours data
NUMEST	Displays titles for EST numeric data
UPDEST	Update Estimated Statistics File subfunction driver
VALEST	EST validation driver; selects validation type

Table B-11. UPDSEF Component Descriptions (1 of 8)

Module	Description
ADDSEF	Constructs and adds new SEF records from operator input
CHGSEF	Prompts for changes and rewrites SEF records for SEF file
DELSEF	Deletes records of a given project from the SEF file
DSPTYP	Displays title and value of fields for a selected category of measure
GETFLD	Obtains field to be changed and its new value and validates it
GETSEF	Fills the seven SEF records of a given project for ADD mode
RDDIR	Reads one indexed record from the SEF directory file
SUM	Computes SUM for a selected category of measure
UPDSEF	Updates Subjective Evaluations File subfunction driver
VALCOD	Validates the evaluation code for the SEF record
VALFLD	Validates a fixed decimal number field for the SEF record
VALSFG	Validates status flag value for the SEF record
VFIELD	Validates the value of a given field for the SEF records
VFPNUM	Validates a fixed decimal number and positions it in the output area with given number of implied decimal place
VLINTG	Validates an integer number field for the SEF record
VLREAL	Validates a given category of measure that all fields are real numbers with same format
VPART1	Validates fields that are 4 bytes long for the part one of a given category of measure

Table B-11. UPDSEF Component Descriptions (2 of 8)

Module	Description
VPART2	Validates fields for the part two and above of a given category of measure
GETMT	Obtains and verifies data for the Practices and Techniques (MT) category
SUMMT	Computes sums on the MT category
DSPMT	Displays titles and data for the MT category
DSPMT1	Displays first line of titles for the MT category
DSPMT2	Displays second line of titles for the MT category
DSPMT3	Displays third line of titles for the MT category
GETTS	Obtains and verifies data for the Tools (TS) category
SUMTS	Computes sums on the TS category
DSPTS	Displays titles and data for the TS category
DSPTS1	Displays first line of titles for the TS category
DSPTS2	Displays second line of titles for the TS category
GETDC	Obtains and verifies data for the Documentation (DC) category
SUMDC	Computes sums on the DC category
DSPDC	Displays titles and data for the DC category
DSPDC1	Displays first line of titles for the DC category
DSPDC2	Displays second line of titles for the DC category
GETAP	Obtains and verifies data for the Experience with Application (AP) category
SUMAP	Computes sums on the AP category
DSPAP	Displays titles and data for the AP category

Table B-11. UPDSEF Component Descriptions (3 of 8)

Module	Description
DSPAP1	Displays first line of titles for the AP category
DSPAP2	Displays second line of titles for the AP category
GETMG	Obtains and verifies data for the Effectiveness of Management (MG) category
SUMMG	Computes sums on the MG category
DSPMG	Displays titles and data for the MG category
DSPMG1	Displays first line of titles for the MG category
DSPMG2	Displays second line of titles for the MG category
DSPMG3	Displays third line of titles for the MG category
DSPMG4	Displays fourth line of titles for the MG category
GETPF	Obtains and verifies data for the Performance of Team (PF) category
SUMPF	Computes sums on the PF category
DSPPF	Displays titles and data for the PF category
DSPPF1	Displays first line of titles for the PF category
DSPPF2	Displays second line of titles for the PF category
DSPPF3	Displays third line of titles for the PF category
DSPPF4	Displays fourth line of titles for the PF category
GETCP	Obtains and verifies data for the Complexity of Problem (CP) category
SUMCP	Computes sums on the CP category

Table B-11. UPDSEF Component Descriptions (4 of 8)

Module	Description
DSPCP	Displays titles and data for the CP category
DSPCP1	Displays first line of titles for the CP category
DSPCP2	Displays second line of titles for the CP category
GETIN	Obtains and verifies data for the Internal Influences on Project (IN) category
SUMIN	Computes sums on the IN category
DSPIN	Displays titles and data for the IN category
DSPIN1	Displays first line of titles for the IN category
DSPIN2	Displays second line of titles for the IN category
GETEX	Obtains and verifies data for the External Influences on Project (EX) category
SUMEX	Computes sums on the EX category
DSPEX	Displays titles and data for the EX category
DSPEX1	Displays first line of titles for the EX category
DSPEX2	Displays second line of titles for the EX category
GETRA	Obtains and verifies data for the Resources Available (RA) category
SUMRA	Computes sums on the RA category
DSPRA	Displays titles and data for the RA category
DSPRA1	Displays first line of titles for the RA category
DSPRA2	Displays second line of titles for the RA category
GETPR	Obtains and verifies data for the Software Product (PR) category

Table B-11. UPDSEF Component Descriptions (5 of 8)

Module	Description
SUMPR	Computes sums on the PR category
DSPPR	Displays titles and data for the PR category
DSPPR1	Displays first line of titles for the PR category
DSPPR2	Displays second line of titles for the PR category
GETPP	Obtains and verifies data for the Product/Process Performance (PP) category
SUMPP	Computes sums on the PP category
DSPPP	Displays titles and data for the PP category
DSPPP1	Displays first line of titles for the PP category
DSPPP2	Displays second line of titles for the PP category
GETRK	Obtains and verifies data for the Team Rank (RK) category
DSPRK	Displays titles and data for the RK category
DSPRK1	Displays first line of titles for the RK category
DSPRK2	Displays second line of titles for the RK category
DSPRK3	Displays third line of titles for the RK category
DSPRK4	Displays fourth line of titles for the RK category
GETYP	Obtains and verifies data for the Years of Professional Experience (YP) category
DSPYP	Displays titles and data for the YP category
DSPYP1	Displays first line of titles for the YP category
DSPYP2	Displays second line of titles for the YP category

Table B-11. UPDSEF Component Descriptions (6 of 8)

Module	Description
DSPYP3	Displays third line of titles for the YP category
DSPYP4	Displays fourth line of titles for the YP category
GETYA	Obtains and verifies data for the Years of Applicable Experience (YA) category
DSPYA	Displays titles and data for the YA category
DSPYA1	Displays first line of titles for the YA category
DSPYA2	Displays second line of titles for the YA category
DSPYA3	Displays third line of titles for the YA category
DSPYA4	Displays fourth line of titles for the YA category
GETYE	Obtains and verifies data for the Years of Environment Experience (YE) category
DSPYE	Displays title and data for the YE category
DSPYE1	Displays first line of titles for the YE category
DSPYE2	Displays second line of titles for the YE category
DSPYE3	Displays third line of titles for the YE category
DSPYE4	Displays fourth line of titles for the YE category
GETWF	Obtains and verifies data for the Walston-Felix Model (WF) category
SUMWF	Computes sums on the WF category
DSPWF	Displays titles and data for the WF category
DSPWF1	Displays the first line of titles for the WF category

Table B-11. UPDSEF Component Descriptions (7 of 8)

Module	Description
DSPWF2	Displays the second line of titles for the WF category
DSPWF3	Displays the third line of titles for the WF category
DSPWF4	Displays the fourth line of titles for the WF category
DSPWF5	Displays the fifth line of titles for the WF category
DSPWF6	Displays the sixth line of titles for the WF category
DSPWF7	Displays the seventh line of titles for the WF category
DSPWF8	Displays the eighth line of titles for the WF category
VALWF	Validates fields for the WF measures
GETPS	Obtains and verifies data for the Price S3 Model (PS) category
SUMPS	Computes sums on the PS category
DSPPS	Displays titles and data for the PS category
DSPPS1	Displays first line of titles for the PS category
DSPPS2	Displays second line of titles for the PS category
GETCO	Obtains and verifies data for the COCOMO Model (CO) category
DSPCO	Displays titles and data for the CO category
DSPCO1	Displays first line of titles for the CO category
DSPCO2	Displays second line of titles for the CO category
GETMS	Obtains and verifies data for the Miscellaneous (MS) category
DSPMS	Displays titles and data for the MS category

Table B-11. UPDSEF Component Descriptions (8 of 8)

Module	Description
DSPMS1	Displays the first line of titles for the MS category
DSPMS2	Displays the second line of titles for the MS category
DSPMS3	Displays the third line of title for the MS category
DSPMS4	Displays the fourth line of titles for the MS category
VALMS	Validates fields for the MS measures
GETSW	Obtains and verifies data for the Code Breakdown (SW) category
DSPSW	Displays titles and data for the SW category
DSPSW1	Displays the first line of titles for the SW category
DSPSW2	Displays the second line of titles for the SW category
DSPSW3	Displays the third line of titles for the SW category
DSPSW4	Displays the fourth line of titles for the SW category
DSPSW5	Displays the fifth line of titles for the SW category
DSPSW6	Displays the sixth line of titles for the SW category
DSPSW7	Displays the seventh line of titles for the SW category
DSPSW8	Displays the eighth line of titles for the SW category
DSPSW9	Displays the ninth line of titles for the SW category
DSPSWA	Displays the tenth line of titles for the SW category
VALSW	Validates fields for the SW measures

Table B-12. UPDHDR Component Descriptions

Module	Description
ADDHDR	Constructs and adds a new HDR record
CHGHDR	Prompts for changes and rewrites an HDR record
DATHDR	Displays titles for HDR date data
DELHDR	Deletes a specified HDR record
DSPHDR	Display driver for HDR data
EXFHDR	Displays titles for extra (unused) HDR date fields
GETHDR	Prompts for and receives HDR data for add mode
HEDHDR	Displays titles for non-date HDR data
UPDHDR	Update Phase Dates (Header) File sub-function driver
VALHDR	HDR validation driver; selects validation type

Table B-13. UPDDIR Component Descriptions

Module	Description
ADDDIR	Constructs a new DIR record from operator input
CHGDIR	Changes and rewrites an existing DIR record as requested by operator input
DELDIR	Deletes an existing DIR record as requested by operator input
UPDDIR	Update Subjective Evaluations Directory (DIR) File subfunction driver; updates the Status File (STAT.HDR)

Table B-14. CREATE Component Descriptions

Module	Description
ADDENC	Constructs a new Encoding Dictionary record
ADDSTS	Constructs and adds records to the File Name and Status File for all files to be created for the new project
CREATE (indirect file)	Creates new files
CREFIL (indirect file)	Update Header Files for new project driver
INTENC	Adds a new record (project) to the Encoding Dictionary

Table B-15. ARCHIV Component Descriptions

Module	Description
ARCFIL	Identifies files to be archived and tape number
ARCHIV (indirect file)	Copies data base files to tape
BCKDAT	Sets the last backup date to the current date for all File Name and Status File records
GETLST	Prompts for file names, verifies them, updates the last backup date, and writes them to a data set
TRNSET	Initializes new versions of all Transaction Files

Table B-16. RESTOR Component Descriptions

Module	Description
RECMER	Reads a record from a Transaction File and performs the indicated update on the indicated file
RESFIL	Merge Transaction Files driver; identifies versions to use
RESTOR (indirect file)	Restore function driver; identifies files to be restored

Table B-17. COMPRESS Component Descriptions

Module	Description
COMPRESS (indirect file)	Drives compress function; calls DEF utility to define new file and then calls IFL to copy old file to new file
STS	Retrieves the number of records of a given data base file from the File Name and Status File

Table B-18. UPDMFN Component Descriptions

Module	Description
GTPRJ	Obtains the project name from the user
PROCIF	Reads records from a given CIF, computes the FORTRAN module function, and stores it back to CIF file
UPMFUN	UPDMFN function driver; computes and stores the FORTRAN module function value for existing CIF record of a given CIF file

Table B-19. UPDSFG Component Descriptions

Module	Description
GTPROJ	Obtains project name from user
GTSTAT	Prompts user for the particular status value
UPDSFG	Update Status Flag function driver; updates all status fields in a given SEL data base file to a particular value with one command
UPSTAT	Updates all status fields with a new value for a given file and writes changes to transaction file
VFYFIL	Verifies the given file type

Table B-20. Recurring Component Descriptions (1 of 2)

Module	Description
ADDCMT	Adds a comment (one or more records) to the comment files
ALPHA	Verifies that a field is alphabetic
CACMT	Display and change comments driver for multiple comments
CHGCMT	Deletes an existing comment and replaces it with another comment
COCODE	Converts a character string to a component code with the Component Information Files
CODE2B	Verifies a 2-byte field with the Encoding Dictionary
DELCMT	Deletes a specified comment
DNCODE	Converts a name to a code or a code to a name with the Encoding Dictionary
DSPCHG	Displays summary counts of completed transactions
DSPCMT	Reads and displays a specified comment
DSPGRP	Displays all entries of a specified type from the Encoding Dictionary
FPNUM	Verifies that a field has a decimal numeric format
GETPRO	Prompts for and validates the project name
GETSEQ	Prompts for and validates or increments a sequence number
LJUST	Left-justifies a character string
LOG	Writes the current date into the last access date and updates the record count in a status record
NDATE	Verifies that string has a valid date format
NEWPRO	Opens and closes project data files and the corresponding comment files
NEWPR2	Opens and closes project data files

Table B-20. Recurring Component Descriptions (2 of 2)

Module	Description
NONBL	Counts non-blank characters in a string
NUMER	Verifies that a field has a numeric format
PARVAL	Parses a string into two parts using "=" as a delimeter
TABLE	Verifies a 1-byte field with the Encoding Dictionary
TABLE2	Verifies a repeated 1-byte field with the Encoding Dictionary
XORB	Verifies that a byte contains "X" or a blank
YNORB	Verifies that a byte contains "Y", "N", or a blank
ZFILL	Zero-fills a string

APPENDIX C - OVERLAY DESCRIPTION FILES

This appendix contains the Overlay Descriptor Language needed to task-build the task images listed in Table 3-2 (Section 3).

Figure Number	Task Image
C-1	ARCFIL
C-2	CREFIL
C-3	RESFIL
C-4	UPDENC
C-5	UPDEST
C-6	UPDHDR
C-7	UPDSEF
C-8	UPDSTS
C-9	UPDCIF
C-10	UPDCRF
C-11	UPDCSF
C-12	UPDCSR
C-13	UPDHIS
C-14	UPDRAF
C-15	UPDRSF
C-16	UPDDIR
C-17	UPDMFN
C-18	UPDSFG

```
;
;   @AR.ODL
;
;   THE OVERLAY STRUCTURE FOR THE DBAM ARCHIVE DATA BASE FILES (ARCHIV)
;   UTILITY    (P. LO  10/12/82)
;
        .ROOT RMSROT-OTSROT-$ROOT,OTSALL,RMSALL
$ROOT:  .FCTR [204,15]ARARCFIL-[204,15]ARTRNSET-[204,15]ARBCKDAT-$ROOT2
$ROOT2: .FCTR [204,15]ARGETLST-[204,15]DMZFILL -[204,15]DMNONBL -$ROOT3
$ROOT3: .FCTR [204, 7]UTMOVE   -[204, 7]UTCHAREQ-$ROOT4
$ROOT4: .FCTR [204, 7]UTMATCHS-[204, 7]UFRMSIAC/LB
;
@LB:[1,1]RMS11M
@LB:[1,1]RMS12X
;
        .END
```

Figure C-1. ARCFIL Overlay Description File

```
;
;   @CE.ODL
;
;   OVERLAY STRUCTURE FOR THE DBAM CREATE NEW PROJECT FILES (CREATE)
;   UTILITY    (P. LO  10/12/82)
;
        .ROOT RMSROT-OTSROT-$ROOT,OTSALL,RMSALL
$ROOT:  .FCTR [204,15]CECREFIL-[204,15]CEINTENC-[204,15]DMADTENC-$ROOT2
$ROOT2: .FCTR [204,15]CEADDSTS-[204,15]DMNONBL -[204,15]DMZFILL -$ROOT3
$ROOT3: .FCTR [204, 7]UTCHAREQ-$ROOT4
$ROOT4: .FCTR [204, 7]UTMATCHS-[204, 7]UFRMSIAC/LB
;
@LB:[1,1]RMS11M
@LB:[1,1]RMS12X
;
        .END
```

Figure C-2. CREFIL Overlay Description File

```
;
;   @RE.ODL
;
;   OVERLAY STRUCTURE FOR THE DBAM RESTORE DATA BASE FILES (RESTOR)
;   UTILITY    (P. LO  10/12/82)
;
        .ROOT RMSROT-OTSROT-$ROOT,OTSALL,RMSALL
$ROOT:  .FCTR [204,15]RERESFIL-[204,15]RERECMER-[204,15]DMNEWPR2-$ROOT2
$ROOT2: .FCTR [204,15]DMLOG    -[204,15]DMDNCODE-$ROOT3
$ROOT3: .FCTR [204, 7]UTCHAREQ-$ROOT4
$ROOT4: .FCTR [204, 7]UTMATCHS-[204, 7]UTMOVE  -[204, 7]UFRMSIAC/LB
;
@LB:[1,1]RMS11M
@LB:[1,1]RMS12X
;
        .END
```

Figure C-3. RESFIL Overlay Description File

```
;
;   @EN.ODL
;
;   OVERLAY STRUCTURE FOR THE DBAM UPDATE ENCODING DICTIONARY (UPDENC)
;   PROGRAM    (P. LO  10/12/82)
;
        .ROOT RMSROT-OTSROT-$ROOT,OTSALL,RMSALL
$ROOT:  .FCTR [204,15]ENUPDENC-[204,15]ENCHGENC-[204,15]ENGETENC-$ROOT2
$ROOT2: .FCTR [204,15]DMADDENC-[204,15]DMLOG    -[204,15]ENDSPENC-$ROOT3
$ROOT3: .FCTR [204,15]DMDSPCHG-$ROOT4
$ROOT4: .FCTR [204, 7]UTMATCHS-[204, 7]UFRMSIAC/LB
;
@LB:[1,1]RMS11M
@LB:[1,1]RMS12X
;
        .END
```

Figure C-4. UPDENC Overlay Description File

**ORIGINAL PAGE IS
OF POOR QUALITY**

```
;
;   @ES.ODL
;
;   OVERLAY STRUCTURE FOR THE DBAM UPDATE ESTIMATED STATISTICS FILE
;   (UPDEST) PROGRAM    (P. LO  10/12/82)
;
        .ROOT PMSPOT-OTSROT-$ROOT,$00,OTSALL,PMSALL
$ROOT:  .FCTR [204,15]ESUPDEST-[204,15]DMGETPRO-$ROOT2
$ROOT2: .FCTR [204,15]DMDSPCHG-[204,15]ESVALEST-$ROOT3
$ROOT3: .FCTR [204,15]DMPARVAL-[204,15]DMMOVEBL-[204,15]DMDNCODE-$ROOT4
$ROOT4: .FCTR [204, 7]UTMATCHS-[204, 7]UFRMSIAC/LB-$ROOT5
$ROOT5: .FCTR *($BB,$CC,$DD)
;
$BB:    .FCTR [204,15]ESCHGEST-[204, 7]UFRMSIAC/LB
$CC:    .FCTR [204,15]ESADDEST-[204,15]ESGETEST-[204, 7]UFRMSIAC/LB
$DD:    .FCTR [204,15]ESDELEST-[204, 7]UFRMSIAC/LB
;
$00:    .FCTR *($RR,$SS)
$RR:    .FCTR [204,15]DMSTSFLG-[204,15]DMLJUST -[204,15]DMNUMER -$RR2
$RR2:   .FCTR [204,15]DMFPNUM -[204,15]DMYNORB -[204, 7]UTCHAREQ
$SS:    .FCTR [204,15]ESDSPEST-[204,15]ESNUMEST-[204,15]ESHOREST-$SS2
$SS2:   .FCTR [204,15]ESLINEST
;
@LB:[1,1]RMS11M
@LB:[1,1]RMS12X
;
        .END
```

Figure C-5. UPDEST Overlay Description File

```
;
;   QHD.ODL
;
;   OVERLAY STRUCTURE FOR THE DBAM UPDATE PHASE DATES FILE (UPDHDR)
;   PROGRAM    (P. LO  10-13-82)
;
        .ROOT $ROOT-RMSROT-QTSROT-$BB, $DD, QTSALL,RMSALL
$ROOT:  .FCTR [204,15]HDUPDHDR-[204,15]DMGETPRO-$ROOT2
$ROOT2: .FCTR [204,15]DMDSPCHG-[204,15]HDVALHDR-$ROOT3
$ROOT3: .FCTR [204,15]DMPARVAL-[204,15]DMMOVEBL-[204,15]DMDNCODE-$ROOT4
$ROOT4: .FCTR [204, 7]UTMATCHS-[204, 7]UFRMSIAC-LB
;
$BB:    .FCTR (($BB1,$BB2,$BB3)
$BB1:   .FCTR [204,15]HDCHGHDR-[204, 7]UFRMSIAC-LB
$BB2:   .FCTR [204,15]HDADDHDR-[204,15]HDGETHDR-[204, 7]UFRMSIAC-LB
$BB3:   .FCTR [204,15]HDDELHDR-[204, 7]UFRMSIAC-LB
;
$DD:    .FCTR (($DD1,$DD2,$DD3)
$DD1:   .FCTR [204,15]DMSTSFLG-[204,15]DMNDATE -[204,15]DMLJUST -$DD1A
$DD1A:  .FCTR [204,15]DMCSCALE-[204, 7]UTCHAREQ
$DD2:   .FCTR [204,15]DMCODE2B-[204,15]DMDSPGRP
$DD3:   .FCTR [204,15]HDDSPHDR-[204,15]HDHEDHDR-[204,15]HDDATHDR-$DD3A
$DD3A:  .FCTR [204,15]HDEOFHDR
;
QLB:[1,1]RMS11M.ODL
QLB:[1,1]RMS12X.ODL
;
        .END
```

Figure C-6. UPDHDR Overlay Description File

ORIGINAL PAGE IS
OF POOR QUALITY

```
;
;   @SE.ODL
;
;   THE OVERLAY STRUCTURE FOR THE DBAM UPDATE SUBJECTIVE EVALUATIONS
;   FILE (UPDSEF) PROGRAM      (P. LO    11/15/82)
;
        .ROOT RMSROT-OTSROT-$ROOT,$QQ,OTSALL,RMSALL
$ROOT:  .FCTR [204,15]SEUPDSEF-[204,15]DMGETERR-[204,15]DMUNCODE-$ROOT2
$ROOT2: .FCTR [204,15]SEDSPCHG-[204, 7]UTMOVE  -[204, 7]UTMATCHS-$ROOT3
$ROOT3: .FCTR [204,15]SEVPART1-[204,15]SEVPART2-[204, 6]SFRDSEF -$ROOT4
$ROOT4: .FCTR [204,15]SERDDIR -[204, 7]UFRMSIAC/LB-$ROOT5
$ROOT5: .FCTR *($BB,$CC,$DD)
;
$BB:    .FCTR [204,15]SECHGSEF-[204,15]SEDSPTYP-[204,15]SEGETFLD-$BB1
$BB1:   .FCTR [204,15]SEVFIELD-[204,15]SESUM   -[204,15]SEVLREAL-$BB2
$BB2:   .FCTR *(B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,B13,B14,$BB3)
$BB3:   .FCTR *(B15,B16,B17,B18,B19,B20,B21)
B1:     .FCTR [204,15]SEDSPAP
B2:     .FCTR [204,15]SEDSPCO
B3:     .FCTR [204,15]SEDSPCP
B4:     .FCTR [204,15]SEDSPDC
B5:     .FCTR [204,15]SEDSPEX
B6:     .FCTR [204,15]SEDSPIN
B7:     .FCTR [204,15]SEDSPMG
B8:     .FCTR [204,15]SEDSPMS -[204,15]SEVALMS
B9:     .FCTR [204,15]SEDSPMT
B10:    .FCTR [204,15]SEDSPPF
B11:    .FCTR [204,15]SEDSPPP
B12:    .FCTR [204,15]SEDSPPR
B13:    .FCTR [204,15]SEDSPPS -[204,15]SEVALPS
B14:    .FCTR [204,15]SEDSPRA
B15:    .FCTR [204,15]SEDSPRK
B16:    .FCTR [204,15]SEDSPSW -[204,15]SEVALSW
B17:    .FCTR [204,15]SEDSPTS
B18:    .FCTR [204,15]SEDSPWF -[204,15]SEVALWF
B19:    .FCTR [204,15]SEDSPYA
B20:    .FCTR [204,15]SEDSPYE
B21:    .FCTR [204,15]SEDSPYP
;
$CC:    .FCTR [204,15]SEADDSEF-[204,15]SEGETSEF-[204,15]DMMOVEBL-$CC1
$CC1:   .FCTR [204, 7]UFRMSIAC/LB-$CC2
$CC2:   .FCTR *(C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,$CC3)
$CC3:   .FCTR *(C15,C16,C17,C18,C19,C20,C21)
C1:     .FCTR [204,15]SEGETAP
C2:     .FCTR [204,15]SEGETCO
C3:     .FCTR [204,15]SEGETCP
C4:     .FCTR [204,15]SEGETDC
C5:     .FCTR [204,15]SEGETEX
C6:     .FCTR [204,15]SEGETIN
C7:     .FCTR [204,15]SEGETMG
C8:     .FCTR [204,15]SEGETMS
C9:     .FCTR [204,15]SEGETMT
C10:    .FCTR [204,15]SEGETPF
C11:    .FCTR [204,15]SEGETPP
C12:    .FCTR [204,15]SEGETPR
```

Figure C-7. UPDSEF Overlay Description File (1 of 2)

```
C13:    .FCTR [204,15]SEDETPS
C14:    .FCTR [204,15]SEDETRA
C15:    .FCTR [204,15]SEDETRM
C16:    .FCTR [204,15]SEDETSW
C17:    .FCTR [204,15]SEDETTS
C18:    .FCTR [204,15]SEDETWF
C19:    .FCTR [204,15]SEDETYA
C20:    .FCTR [204,15]SEDETYE
C21:    .FCTR [204,15]SEDETYP
;
$DD:    .FCTR [204,15]SEDELSEF-[204, 7]UFRMSIAC/LB
;
$QQ:    .FCTR $RR-*($SS)
$RR:    .FCTR [204,15]DMSTSFLG-[204,15]DMLJUST -[204,15]DMNUMER -$RR2
$RR2:   .FCTR [204,15]SEVFPNUM-[204,15]SEVALSFG-$RR3
$RR3:   .FCTR [204,15]SEVALCOD-[204,15]SEVALFLD
;
$SS:    .FCTR *(S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11,S12,S13,S14,$SS2)
$SS2:   .FCTR *(S15,S16,S17,S18,S19,S20,S21)
S1:     .FCTR [204,15]SESUMAP -[204,15]SEDSPAP1-[204,15]SEDSPAP2
S2:     .FCTR [204,15]SEDSPCO1-[204,15]SEDSPCO2
S3:     .FCTR [204,15]SESUMCP -[204,15]SEDSPCP1-[204,15]SEDSPCP2
S4:     .FCTR [204,15]SESUMDC -[204,15]SEDSPDC1-[204,15]SEDSPDC2
S5:     .FCTR [204,15]SESUMEX -[204,15]SEDSPEX1-[204,15]SEDSPEX2
S6:     .FCTR [204,15]SESUMIN -[204,15]SEDSPIN1-[204,15]SEDSPIN2
S7:     .FCTR [204,15]SESUMMG -[204,15]SEDSPMG1-[204,15]SEDSPMG2-S7A
S7A:    .FCTR [204,15]SEDSPMG3-[204,15]SEDSPMG4
S8:     .FCTR [204,15]SEDSPMS1-[204,15]SEDSPMS2-[204,15]SEDSPMS2-S8A
S8A:    .FCTR [204,15]SEDSPMS4
S9:     .FCTR [204,15]SEDSPMT1-[204,15]SEDSPMT2-[204,15]SEDSPMT3-S9A
S9A:    .FCTR [204,15]SESUMMT
S10:    .FCTR [204,15]SESUMPF -[204,15]SEDSPPF1-[204,15]SEDSPPF2-S10A
S10A:   .FCTR [204,15]SEDSPPF3-[204,15]SEDSPPF4
S11:    .FCTR [204,15]SESUMPP -[204,15]SEDSPPP1-[204,15]SEDSPPP2
S12:    .FCTR [204,15]SESUMPR -[204,15]SEDSPPR1-[204,15]SEDSPPR2
S13:    .FCTR [204,15]SESUMPS -[204,15]SEDSPPS1-[204,15]SEDSPPS2
S14:    .FCTR [204,15]SESUMRA -[204,15]SEDSPRA1-[204,15]SEDSPRA2
S15:    .FCTR [204,15]SEDSPRK1-[204,15]SEDSPRK2-[204,15]SEDSPRK3-S15A
S15A:   .FCTR [204,15]SEDSPRK4
S16:    .FCTR [204,15]SEDSPSW1-[204,15]SEDSPSW2-[204,15]SEDSPSW3-S16A
S16A:   .FCTR [204,15]SEDSPSW4-[204,15]SEDSPSW5-[204,15]SEDSPSW6-S16B
S16B:   .FCTR [204,15]SEDSPSW7-[204,15]SEDSPSW8-[204,15]SEDSPSW9-S16C
S16C:   .FCTR [204,15]SEDSPSWA
S17:    .FCTR [204,15]SESUMTS -[204,15]SEDSPTS1-[204,15]SEDSPTS2
S18:    .FCTR [204,15]SESUMWF -[204,15]SEDSPWF1-[204,15]SEDSPWF2-S18A
S18A:   .FCTR [204,15]SEDSPWF3-[204,15]SEDSPWF4-[204,15]SEDSPWF5-S18B
S18B:   .FCTR [204,15]SEDSPWF6-[204,15]SEDSPWF7-[204,15]SEDSPWF8
S19:    .FCTR [204,15]SEDSPYA1-[204,15]SEDSPYA2-[204,15]SEDSPYA2-S19A
S19A:   .FCTR [204,15]SEDSPYA3
S20:    .FCTR [204,15]SEDSPYE1-[204,15]SEDSPYE2-[204,15]SEDSPYE3-S20A
S20A:   .FCTR [204,15]SEDSPYE4
S21:    .FCTR [204,15]SEDSPYP1-[204,15]SEDSPYP2-[204,15]SEDSPYP3-S21A
S21A:   .FCTR [204,15]SEDSPYP4
;
@LB:[1,1]RMS11M
@LB:[1,1]RMS12X
;
        .END
```

Figure C-7. UPDSEF Overlay Description File (2 of 2)

```
;
;    @ST.ODL
;
;    OVERLAY STRUCTURE FOR THE DBAM UPDATE FILE NAME AND STATUS FILE
;    (UPDSTS) PROGRAM     (P. LO  10/12/82)
;
         .ROOT RMSROT-OTSROT-$ROOT,OTSALL,RMSALL
$ROOT:   .FCTR [204,15]STUPDSTS-[204,15]STCHGSTS-[204,15]STDSPSTS-$ROOT2
$ROOT2:  .FCTR [204,15]DMPARVAL-[204,15]STGETSTS-$ROOT3
$ROOT3:  .FCTR [204,15]STGETCOD-[204,15]DMDNCODE-[204,15]STFLDSTS-$ROOT4
$ROOT4:  .FCTR [204,15]STVALSTS-[204,15]DMNUMER -[204,15]DMNDATE -$ROOT5
$ROOT5:  .FCTR [204, 7]UTMATCHS-[204, 7]UFRMSIAC/LB
;
@LB:[1,1]RMS1%M
@LB:[1,1]RMS12X
;
         .END
```

Figure C-8. UPDSTS Overlay Description File

```
;
;   @CI.ODL
;
;   OVERLAY STRUCTURE FOR THE DBAM UPDATE COMPONENT INFORMATION FILE
;   (UPDCIF) PROGRAM    (P. LO  10/13/82)
;
        .ROOT RMSROT-OTSROT-$ROOT,*$QQ,OTSALL,RMSALL
$ROOT:  .FCTR [204,15]CIUPDCIF-[204,15]CIGETCIF-[204,15]DMNEWPRO-$ROOT2
$ROOT2: .FCTR [204,15]DMDSPFLG-[204,15]DMCDATE -[204,15]DMLOG   -$ROOT4
$ROOT4: .FCTR [204,7]OUTBLANK -[204,7]OUTMOVE  -[204,7]OUTMATCHS-$ROOT5
$ROOT5: .FCTR [204,7]OUFRMSIAC LB-$ROOT6
$ROOT6: .FCTR *(,$BB,$CC,$DD,$EE)
;
$BB:    .FCTR [204,15]CICHGCIF-[204,15]CIFLDCIF-[204,7]OUFRMSIAC LB
$CC:    .FCTR [204,15]CIADDCIF-[204,15]CIPROCIF-[204,15]CICODCIF-$CC2
$CC2:   .FCTR [204,7]OUFRMSIAC/LB
$DD:    .FCTR [204,15]CIDELCIF-[204,7]OUFRMSIAC/LB
$EE:    .FCTR [204,15]CIREDCIF-[204,15]CINAMCIF-[204,15]CIMOVCIF-$EE2
$EE2:   .FCTR [204,6]PSCTYPE -[204,7]OUFRMSIAC/LB
;
        .NAME %
$00:    .FCTR %-[204,15]DMDMCODE-[204,15]DMDSPGRP-[204,15]DMTABLE -$QQ2
$QQ2:   .FCTR [204,15]DMTABLE2-[204,15]DMSTSFLG-[204,15]CIVALCIF-$QQ3
$QQ3:   .FCTR [204,15]DMPARVAL-[204,15]DMLJUST -[204,15]DMNUMER -$QQ4
$QQ4:   .FCTR [204,7]OUTCHARE0-$QQ5
$QQ5:   .FCTR *(,$RR,$SS,$TT)
;
$RR:    .FCTR [204,15]CIVLACIF-[204,15]CIDSACIF
$SS:    .FCTR [204,15]CIVLBCIF-[204,15]CIDSBCIF
$TT:    .FCTR [204,15]CIVLCCIF-[204,15]CIDSCCIF
;
@LB:[1,1]RMS11M
@LB:[1,1]RMS12N
;
        .END
```

Figure C-9. UPDCIF Overlay Description File

ORIGINAL PAGE IS
OF POOR QUALITY

```
;
;    @CH.ODL
;
;    OVERLAY STRUCTURE FOR THE DBAM UPDATE CHANGE REPORT FORM (UPDCRF)
;    PROGRAM    (P. LO  10/14/82)
;
        .ROOT [204,15]CHUPDCRF-RMSROT-OTSROT-AFCTR,*BFCTR,OTSALL,RMSALL
AFCTR:  .FCTR [204,15]DMGETCRF-[204,15]DMCFILL - (FCTR - ($D1,$D2,$D3)
.FCTR:  .FCTR [204,15]DMNCHPRO-[204,15]DMDSPCHG-[204,15]DMLOG    - FCTR2
.FCTR2: .FCTR [204, 7]UTMOVE   -[204, 7]UTMATCHS-[204, 7]UTBLANK - FCTR3
.FCTR3: .FCTR [204,15]DMCDATE  -[204, 7]UFRMSIAC LB
;
$D1:    .FCTR [204,15]CHCHGCRF-[204,15]CHFLDCRF-[204,15]DMCHGCMT-$D1A
$D1A:   .FCTR [204,15]DMDSPCMT-[204,15]DMDELCMT-[204,15]DMADDCMT-$D1B
$D1B:   .FCTR [204, 7]UFRMSIAC LB
$D2:    .FCTR [204,15]CHADDCRF-[204,15]CHPROCRF-[204,15]DMADDCMT-$D2A
$D2A:   .FCTR [204, 7]UFRMSIAC LB
$D3:    .FCTR [204,15]CHDELCRF-[204,15]DMDELCMT-[204, 7]UFRMSIAC LB
;
BFCTR:  .FCTR [204,15]DMDHCODE-[204,15]DMCOCODE-[204,15]DMDSPGRP-BFCTR2
BFCTR2: .FCTR [204,15]DMTABLE -[204,15]DMNDATE -CFCTR
CFCTR:  .FCTR [204,15]DMNUMER -*(DFCTR,EFCTR,FFCTR,GFCTR)
;
DFCTR:  .FCTR [204,15]CHVLACRF-[204,15]CHDSACRF
EFCTR:  .FCTR [204,15]CHVLBCRF-[204,15]CHDSBCRF
FFCTR:  .FCTR [204,15]CHVLCCRF-[204,15]CHDSCCRF
GFCTR:  .FCTR [204,15]CHVLDCRF-[204,15]DMSTSFLG-[204,15]CHDSDCRF
;
@LB:[1,1]RMS11M.ODL
@LB:[1,1]RMS12X.ODL
        .END
```

Figure C-10. UPDCRF Overlay Description File

```
;
;    @CM.ODL
;
;    OVERLAY STRUCTURE FOR THE DBAII UPDATE COMPONENT SUMMARY FORM
;    (UPDCSF) PROGRAM    (P. LO  10/14/82)
;
          .ROOT [204,15]CMUPDCSF-RMSROT-OTSROT-AFCTR,BFCTR,OTSALL,RMSALL
AFCTR:    .FCTR [204,15]DMGETCRF-XFCTR-*((#D1,#D2,#D3)
XFCTR:    .FCTR [204,15]DMMIEWPP2-[204,15]DMDSPCHG-[204,15]DMLOG    - FCTR2
XFCTR2:   .FCTR [204,15]CMVALCSF-[204,15]CMVALDTX-[204,15]DMCFILL - FCTR3
XFCTR3:   .FCTR [204, 7]UTMATCHS-[204, 7]UTMOVE   -[204, 7]UTBLANK - FCTR4
XFCTR4:   .FCTR [204,15]DMCDATE -[204, 7]UTJINTG  -[204, 7]UFRMSIAC LB
;
#D1:      .FCTR [204,15]CMCHGCSF-[204,15]CMFLDCSF-[204,15]DMCACMT -#D1A
#D1A:     .FCTR [204,15]DMCHGCMT-[204,15]DMDSPCMT-[204,15]DMDELCMT-#D1B
#D1B:     .FCTR [204,15]DMADDCMT-[204, 7]UFRMSIAC/LB
#D2:      .FCTR [204,15]CMADDCSF-[204,15]CMPROCSF-[204,15]CMAPCMT -#D2A
#D2A:     .FCTR [204,15]DMADDCMT-[204, 7]UFRMSIAC/LB
#D3:      .FCTR [204,15]CMDELCSF-[204,15]DMDELCMT-[204, 7]UFRMSIAC/LB
;
BFCTR:    .FCTR *((CFCTR,DFCTR,GFCTR)
CFCTR:    .FCTR [204,15]CMDSPCSF-*((#D10,#D11)
#D10:     .FCTR [204,15]CMDSACSF-[204,15]CMDSBCSF-[204,15]CMDSCCSF
#D11:     .FCTR [204,15]CMDSDCSF-[204,15]CMDSECSF-[204,15]CMDSFCSF
DFCTR:    .FCTR [204,15]DMNUMER -[204,15]DMNDATE -[204,15]DMCOCODE-EFCTR
EFCTR:    .FCTR [204,15]DMSTSFLG-[204,15]DMYNORB -[204,15]DMXORB  -EFCTR2
EFCTR2:   .FCTR [204,15]CMEMHORB-[204,15]CMNUCORB
GFCTR:    .FCTR [204,15]DMDNCODE-[204,15]DMDSPGRP-[204,15]DMTABLE2-GFCTR2
GFCTR2:   .FCTR [204,15]DMTABLE
;
QLB:[1,1]RMS11M.ODL
QLB:[1,1]RMS12X.ODL
;
          .END
```

Figure C-11. UPDCSF Overlay Description File

ORIGINAL PAGE IS
OF POOR QUALITY

```
;
;   QCT.ODL
;
;   OVERLAY STRUCTURE FOR THE DBAM UPDATE COMPONENT STATUS REPORT
;   (UPDCSR) PROGRAM    (P. LO  10/14/82)
;
        .ROOT $ROOT-RMSROT-OTSROT-$QQ, OTSALL, RMSALL
$ROOT:  .FCTR [204,15]CTUPDCSR-[204,15]CTGETCSR-[204,15]DMZFILL -$R1
$R1:    .FCTR [204,15]DMLJUST -[204,15]DMNEWPR2-[204,15]DMDSPCHG-$R2
$R2:    .FCTR [204,15]DMLOG   -[204,15]CTVALCSR-[204,15]DMGETSEQ-$R3
$R3:    .FCTR [204,15]CTVALACT-[204,15]DMPARVAL-$R4
$R4:    .FCTR [204,15]DMSTSFLG-[204,15]DMRMIND-[204,15]DMNDATE -$R5
$R5:    .FCTR [204,15]CTDSPCSR-[204,15]CTHEDCSR-[204,15]CTFORCSR-$R6
$R6:    .FCTR [204,15]DMFPNUM -[204,15]DMNUMER -[204,15]DMDNCODE-$R7
$R7:    .FCTR [204,15]DMCOCODE-[204,15]DMZDATE -$R10
$R10:   .FCTR [204, 7]UTMOVE  -[204, 7]UTMATCHS-[204, 7]UTBLANK -$R11
$R11:   .FCTR [204,15]CTEMPTAB-[204, 7]UFRMSIAC/LB
;
$QQ:    .FCTR *($RR,$SS,$TT)
$RR:    .FCTR [204,15]CTCHGCSR-[204,15]CTFLDCSR-[204, 7]UFRMSIAC/LB
$SS:    .FCTR [204,15]CTADDCSR-[204,15]CTPROCSR-[204, 7]UFRMSIAC/LB
$TT:    .FCTR [204,15]CTDELCSR-[204, 7]UFRMSIAC/LB
;
;
QLB:[1,1]RMS11M.ODL
QLB:[1,1]RMS12X.ODL
;
        .END
```

Figure C-12. UPDCSR Overlay Description File

```
;
;   QHI.ODL
;
;   OVERLAY STRUCTURE FOR THE DBAM UPDATE GROWTH HISTORY FILE (UPDHIS)
;   PROGRAM    (P. LO  10/14/82)
;
        .ROOT RMSROT-OTSROT-$ROOT,$QQ,OTSALL,RMSALL
$ROOT:  .FCTR [204,15]JHIUPDHIS-[204,15]JDMGETPRO-[204,15]JDMNEWPP2-$ROOT2
$ROOT2: .FCTR [204,15]JDMDSPCHG-[204,15]JDMILOG    -[204,15]JHIVALHIS-$ROOT3
$ROOT3: .FCTR [204,15]JDMPARVAL-[204,15]JDMZDATE   -[204,15]JDMMOVEBL-$ROOT4
$ROOT4: .FCTR [204, 7]JUTBLANK  -[204, 7]JUTMOVE   -[204, 7]JUTMATCHS-$ROOT5
$ROOT5: .FCTR [204, 7]JUFRMSIAC/LB-$ROOT6
$ROOT6: .FCTR *($BB,$CC,$DD)
;
$BB:    .FCTR [204,15]JHICHGHIS-[204,15]JHIFLDHIS-[204, 7]JUFRMSIAC/LB
$CC:    .FCTR [204,15]JHIADDHIS-[204,15]JHIPROHIS-[204, 7]JUFRMSIAC/LB
$DD:    .FCTR [204,15]JHIDELHIS-[204, 7]JUFRMSIAC/LB
;
$QQ:    .FCTR *($RR,$SS)
$RR:    .FCTR [204,15]JHIDSPHIS-[204,15]JHIDATHIS-[204,15]JDMLJUST -$RR2
$RR2:   .FCTR [204,15]JDMSTSFLG-[204,15]JDMNDATE -[204,15]JDMNUMER -$RR3
$RR3:   .FCTR [204, 7]JUTCHAREQ
$SS:    .FCTR [204,15]JDMDMCODE
;
@LB:[1,1]RMS11M
@LB:[1,1]RMS12X
;
        .END
```

Figure C-13. UPDHIS Overlay Description File

ORIGINAL PAGE IS
OF POOR QUALITY

```
;
;    @RA.ODL
;
;    OVERLAY STRUCTURE FOR THE DBAM UPDATE RUN ANALYSIS FORM (UPDRAF)
;    PROGRAM     (P. LO  10/18/82)
;
        .ROOT $ROOT-RMSROT-OTSROT-$AA, $CC, OTSALL, RMSALL
$ROOT:  .FCTR [204,15]RAUPDRAF-[204,15]RAGETRAF-[204,15]DMNEWPR2-$R2
$R2:    .FCTR [204,15]DMZFILL -[204,15]DMDSPCHG-[204,15]DMLOG   -$R3
$R3:    .FCTR [204,15]RAVALRAF-[204,15]DMGETSEQ-[204,15]RAVALCMP-$R4
$R4:    .FCTR [204,15]DMPARVAL-[204,15]DMZDATE -[204,15]RAVALDTY-$R6
$R6:    .FCTR [204,15]DMCACMT -[204,15]DMCHGCMT-[204,15]DMDSPCMT-$R7
$R7:    .FCTR [204,15]DMADDCMT-[204,15]DMDELCMT-$R8
$R8:    .FCTR [204, 7]UTMOVE  -[204, 7]UTMATCHS-[204, 7]UTBLANK -$R9
$R9:    .FCTR [204, 7]UFRMSIAC/LB
;
$AA:    .FCTR *($BB1,$BB2,$BB3)
$BB1:   .FCTR [204,15]RACHGRAF-[204,15]RAFLDRAF-[204, 7]UFRMSIAC/LB
$BB2:   .FCTR [204,15]RAADDRAF-[204,15]RAPRORAF-[204, 7]UFRMSIAC/LB
$BB3:   .FCTR [204,15]RADELRAF-[204, 7]UFRMSIAC/LB
;
$CC:    .FCTR *($EE,$FF,$GG)
;
;
$EE:    .FCTR [204,15]RADSPRAF-[204,15]RAHEDRAF-[204,15]RADATRAF-$EE2
$EE2:   .FCTR [204,15]DMCOCODE
;
$FF:    .FCTR [204,15]DMSTSFLG-[204,15]DMLJUST -[204,15]DMNUMER -$FF2
$FF2:   .FCTR [204,15]DMINDATE -[204,15]DMXORB  -[204,15]DMYNORB
;
$GG:    .FCTR [204,15]DMDNCODE-[204,15]DMTABLE -[204,15]DMDSPGRP
;
;
@LB:[1,1]RMS11M
@LB:[1,1]RMS12K
;
        .END
```

Figure C-14. UPDRAF Overlay Description File

```
;
;   QRS.ODL
;
;   OVERLAY STRUCTURE FOR THE DBAM UPDATE RESOURCE SUMMARY FORM (UPDRSF)
;   PROGRAM    (P. LO  10 18 82)
;
        .ROOT RMSROT-QTSROT-$ROOT, $DD, QTSALL, RMSALL
$ROOT:  .FCTR [204,15]RSUPDRSF-[204,15]RSGETRSF-[204,15]DMCFILL -$ROOT2
$ROOT2: .FCTR [204,15]DMMFMPP2-[204,15]DMDSPCHG-[204,15]DMLOG   -$ROOT3
$ROOT3: .FCTR [204,15]RSVALRSF-[204,15]DMGETSEQ-[204,15]DMPARVAL-$ROOT4
$ROOT4: .FCTR [204,15]DMPDMMD-[204,15]DMCDATE -[204,15]DMSTSFLG-$ROOT5
$ROOT5: .FCTR [204, 7]UTMATCHS-[204, 7]UTMOVE  -[204, 7]UTBLANK -$ROOT6
$ROOT6: .FCTR [204, 7]UFRMSIAC LB-$ROOT9
$ROOT9: .FCTR *(,$AA,$BB,$CC)
;
$AA:    .FCTR [204,15]RSCHGRSF-[204,15]RSFLDRSF
$BB:    .FCTR [204,15]RSADDRSF-[204,15]RSPRORSF-[204,15]RSGETRES-$BB2
$BB2:   .FCTR [204,15]RSGETVAL-[204, 7]UFRMSIAC LB
$CC:    .FCTR [204,15]RSDELPSF-[204, 7]UFRMSIAC LB
;
$DD:    .FCTR *(,$EE,$FF)
$EE:    .FCTR [204,15]RSDSPRS -[204,15]RSHEDRSF-[204,15]RSDATRSF-$EE2
$EE2:   .FCTR [204,15]RSCOMRSH-[204,15]DMMOVEBL-$EE3
$EE3:   .FCTR [204,15]DMLJUST -[204,15]RSGETFRI
$FF:    .FCTR [204,15]DMDNCODE-[204,15]DMDSPGRP-[204,15]DMNDATE -$FF2
$FF2:   .FCTR [204,15]DMNUMER -[204,15]DMFPNUM -$FF3
$FF3:   .FCTR [204, 7]UTCHAREQ
;
QLB:[1,1]RMS11M
 B:[1,1]RMS12X
;
        .END
```

Figure C-15. UPDRSF Overlay Description File

ORIGINAL PAGE IS
OF POOR QUALITY

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;       UPDATE SUBJECTIVE EVALUATION DIRECTORY (DIR) FILE
;       OVERLAY STRUCTURE.  (J. COOK   11/4/82)
;
            .ROOT $TREE1,OTSALL,RMSALL
$TREE1:     .FCTR $ROOT-RMSROT-OTSROT
;
 $ROOT:     .FCTR  [204,15]SDUPDDIR-[204,15]SDADDDIR-[204,15]SDCHGDIR-$R1
 $R1:       .FCTR  [204,15]SDDELDIR
;
@LB:[1,1]RMS11M.ODL
@LB:[1,1]RMS12X.ODL
            .END
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Figure C-16. UPDDIR Overlay Description File

```
;
;    QMF.ODL
;
;    OVERLAY STRUCTURE FOR DBAM UPDATE CIF FORTRAN MODULE FUNCTION
;    UTILITY   (4/30/82   BY P. LO)
;
            .ROOT RMSROT-OTSROT-$ROOT,OTSALL,RMSALL
$ROOT:      .FCTR [204,15]MFUPMFUN-[204,7]UTMOVE-$ROOT2
$ROOT2:     .FCTR *($GTPRJ,$PROCF)
;
$GTPRJ:     .FCTR [204,15]MFGTPRJ
;
$PROCF:     .FCTR [204.15]MFPROCIF-[204,7]UTBLANK-[204,15]DMZDATE-$PROC1
$PROC1:     .FCTR [204, 6]R5CTYPE
;
@LB:[1,1]RMS11M
@LB:[1,1]RMS12X
;
            .END
```

Figure C-17. UPDMFN Overlay Description File

```
;
;   QSF.ODL                                                                 1
;                                                                           2
;   OVERLAY STRUCTURE FOR DBAM UPDATE STATUS FIELD FUNCTION                 3
;                                                                           4
        .ROOT RMSROT-OTSROT-$ROOT,OTSALL,RMSALL                             5
$ROOT:  .FCTR [204,15]SFUPDSFG-[204,7]UTMOVE-$ROOT2                         6
$ROOT2: .FCTR *($GPR,$VFY,$GST,$UPS)                                        7
;                                                                           8
$GPR:   .FCTR [204,15]SFGTPROJ                                              9
;                                                                           10
$VFY:   .FCTR [204,15]SFVFYFIL                                              11
;                                                                           12
$GST:   .FCTR [204,15]SFGTSTAT                                              13
;                                                                           14
$UPS:   .FCTR [204,15]SFUPSTAT-[204,7]UTBLANK-[204,15]DMZDATE               15
;                                                                           16
QLB:[1,1]RMS11M                                                             17
QLB:[1,1]RMS12X                                                             18
;                                                                           19
        .END                                                                20
                                                                            21
```

Figure C-18. UPDSFG Overlay Description File

APPENDIX D - INDIRECT COMMAND FILE LISTINGS

Appendix D contains indirect file command listings.

Figure	Contents
D-1	DBAM Driver
D-2	Update Function
D-3	Archive Function
D-4	Create Function
D-5	Restore Function
D-6	Compress Function
D-7	Update FORTRAN Module Function for CIF File Function
D-8	Update Status Flag Function

```
;         ************************************
;         * SOFTWARE ENGINEERING LABORATORY *
;         *   DATA BASE MANAGEMENT SYSTEM   *
;         ************************************
;
;DETAILED OPERATOR INSTRUCTIONS MAY BE OBTAINED BY LISTING
;'SELDBS.HLP' OR BY CONSULTING THE USER'S GUIDE.
;
;***************** START SEL DBAM *********************
.100: .ASKS OPER ENTER FUNCTION TYPE
;
   .IF OPER <> 'ADD' .GOTO 200
      @ADDALL
      .GOTO 100
   .200: .IF OPER <> 'UPDATE' .GOTO 300
      @UPDATE
      .GOTO 100
   .300: .IF OPER <> 'CREATE' .GOTO 500
      @CREATE
      .GOTO 100
   .500: .IF OPER <> 'ARCHIVE' .GOTO 600
      @ARCHIV
      .GOTO 100
   .600: .IF OPER <> 'RESTORE' .GOTO 650
      @RESTOR
      .GOTO 100
   .650: .IF OPER <> 'COMPRESS' .GOTO 660
      @COMPRESS
      .GOTO 100
   .660: .IF OPER <> 'UPDMFN' .GOTO 670
      @UPDMFN
      .GOTO 100
   .670: .IF OPER <> 'UPDSFG' .GOTO 700
      @UPDSFG
      .GOTO 100
   .700: .IF OPER <> 'STOP' .GOTO 800
      .GOTO 900
.800: !*** INVALID FUNCTION SPECIFIED,
;           USE - 'ADD', BATCH PROCESSING OF ADDITIONS
;                 'UPDATE', INTERACTIVE DATA UPDATE
;                 'CREATE', ADD NEW PROJECT
;                 'ARCHIVE', SAVE DB ON TAPE
;                 'RESTORE', REPLACE DB FROM TAPE
;                 'COMPRESS', COMPRESS SPECIFIED DATA BASE FILES
;                 'UPDMFN', UPDATE FORTRAN MODULE FUNCTION FOR CIF
;                 'UPDSFG', UPDATE STATUS FLAG FIELDS WITH ONE COMMAND
;                 'STOP', TERMINATE PROGRAM
.; END WHILE
.GOTO 100
.900: !****************** END SEL DBAM *********************
```

Figure D-1. DBAM Driver Indirect Command File

```
!****************** UPDATE FUNCTION ******************
.100! .ASKS FORM ENTER FORM TYPE
!
.IF FORM <> 'RAF' .GOTO 200
@UPDRAF
.GOTO 900
.200! .IF FORM <> 'RSF' .GOTO 300
@UPDRSF
.GOTO 900
.300! .IF FORM <> 'CSR' .GOTO 400
@UPDCSR
.GOTO 900
.400! .IF FORM <> 'CSF' .GOTO 500
@UPDCSF
.GOTO 900
.500! .IF FORM <> 'CRF' .GOTO 600
@UPDCRF
.GOTO 900
.600! .IF FORM <> 'CIF' .GOTO 700
@UPDCIF
.GOTO 900
.700! .IF FORM <> 'HIS' .GOTO 720
@UPDHIS
.GOTO 900
.720! .IF FORM <> 'STS' .GOTO 730
@UPDSTS
.GOTO 900
.730! .IF FORM <> 'ENC' .GOTO 740
@UPDENC
.GOTO 900
.740! .IF FORM <> 'EST' .GOTO 750
@UPDEST
.GOTO 900
.750! .IF FORM <> 'SEF' .GOTO 760
@UPDSEF
.GOTO 900
.760! .IF FORM <> 'HDR' .GOTO 800
@UPDHDR
.GOTO 900
.800! .IF FORM <> 'DIR' .GOTO 850
@UPDDIR
.GOTO 900
.850! .IF FORM = 'STOP' .GOTO 900
!*** INVALID FORM TYPE SPECIFIED,
.GOTO 100
.900! !***************** RESTART SEL DRMS ******************
```

Figure D-2. Update Function Indirect Command File

ORIGINAL PAGE IS
OF POOR QUALITY

```
SET TERM/WID=80
.ENABLE SUBSTITUTION
@[204,4]CHEC UPDARCHIV START
.ASKS LABEL [ MOUNT TAPE ON MM0 ]    ENTER TAPE-ID (TO 6 LETTERS)
ALL MM0:
.IF <EXSTAT> EQ 1 ;    INI MM0:'LABEL'/DENS=1600
.IF <EXSTAT> EQ 1      MOU MM0:'LABEL'/DENS=1600
.IF <EXSTAT> EQ 1      RUNNING D.B. ACTIVITY REPORT (READING TRANSACTION
.IF <EXSTAT> EQ 1      RUN [204,5]RC/TASK=RC
.IF <EXSTAT> EQ 1    ; PLEASE PRINT RECENT.RPT AND DELIVER COPY TO JERRY
.IF <EXSTAT> EQ 1      RUN [204,5]DMARCFIL
.IF <EXSTAT> EQ 1      BCK @FILNAM.DAT
.IF <EXSTAT> EQ 1      BCK MM0:TRANS.DAT=DB0:[204,1]TRANS.*/SL:ARCHIV2.LS
@[204,4]CHEC ARCHIV END
PRI ARCHIV.LST
DMO MM0:
DEALLOCATE MM0:
```

Figure D-3. Archive Function Indirect Command File

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL
.ENABLE QUIET

.SETS $NAME P1
.SETS $VTYP 'CIFCMTCRFCSFCSRHISRAFRSF'
.SETN $NTYP 8.
.SETS $UIC1 '[204,1]'      ; DATA BASE FILES
.SETS $UIC2 '[204,4]'      ; DEFINE COMMAND FILE UIC
.SETF $PROMT
.IF $NAME EQ ''      .SETT $PROMT
.IFT $PROMT      !!   ********  CREATE FILES FOR NEW PROJECT  ********
.SETF $ERROR
.SETF $EOF
.SETN $ALLOC 5.
.SETN $EXTEN 8.
.SETN $DFILL 512.
.SETN $IFILL 950.
.SETN $FILLA 1000.
.SETN $FILLB 1000.
.SETN $FILLC 1000.
.SETN $FILLD 1000.

.WHILE1: .; MORE PROJECTS TO CREATE
   .IFT $PROMT      .ASKS $NAME ENTER NEW PROJECT NAME OR 'STOP'
   .IF $NAME EQ 'STOP'      .GOTO ELSE1
      .OPEN #1 [204,3]PRONAM.DAT
      .DATA #1 '$NAME'
      .CLOSE #1
      !!  UPDATING ENCODING DICTIONARY AND FILE STATUS FILE
      RUN [204,5]DMCREFIL
      .SETF $ERROR
      .;ASK $ERROR ARE THERE ANY ERRORS SO FAR
      .IFT $ERROR      .GOTO ENDIF2
         .SETN ITYP 1.
         .WHILE2: .; ITYP LE 24   (3 * $NTYP)
            .SETS $TYP $VTYP[ITYP:ITYP+2.]
            .SETS $DBF2 $UIC1+$NAME+'.'+$TYP
            .TESTFILE '$DBF2'
            .IF <FILERR> EQ 1      .GOTO ELSE4
               ! CREATING '$DBF2'
               @'$UIC2'DEFINE'$TYP'.CMD
               .GOTO ENDIF4
            .ELSE4:
               .SETT $ERROR
               !! FILE '$DBF2' ALREADY EXISTS; NEW FILE NOT CREATED
            .ENDIF4:
            .SETN ITYP ITYP+3.
         .;END WHILE MORE FILES TO CREATE FOR CURRENT PROJECT
         .IF ITYP LE 24.      .GOTO WHILE2
      .ENDIF2:
      .GOTO ENDIF1
   .ELSE1: .;  'STOP' ENTERED
      .SETT $EOF
   .ENDIF1:
.;END WHILE NOT EOF
.IFT $PROMT .AND .IFF $EOF .AND .IFF $ERROR      .GOTO WHILE1

.IFF $ERROR      SRD 99%.TMP/DE
!!
!!    END OF CREATE
!!
.STOP:
```

Figure D-4. Create Function Indirect Command File

```
SET TERM/MCR
!*************** BEGIN RESTORE *****************
!
!PLEASE READY YOUR TAPE FROM THE SYSTEM CONSOLE
!
!>MOU MM0:XXXXXX/DENS=1600
!
!WHERE 'XXXXXX' IS THE TAPE IDENTIFICATION
!
.ENABLE SUBSTITUTION
@[204,4]CHEC RESTOR START
.OPEN FILNAM.DAT
.100: .ASKS FNAME ENTER FILES (LIST/ALL)
.IF FNAME <> 'ALL' .GOTO 200
.DATA *.*=MM0:[204,1]SELDBS.DAT
.CLOSE
RST @FILNAM.DAT
!RUN [204,14]DMRESFIL
RUN [204,5]DMRESFIL
.GOTO 900
.200: .IF FNAME = '' .GOTO 900
.IF FNAME <> 'LIST' .GOTO 100
.!DATA [204,1]*.*=MM0:[204,1]SELDBS.DAT/SE:-
.DATA *.*=MM0:[204,1]SELDBS.DAT/SE:-
.300: .ASKS FNAME ENTER FILE NAME TO SAVE
.IF FNAME = '' .GOTO 700
.DATA 'FNAME'
.GOTO 300
!.700: .DATA [204,3]NOTFIL.DAT
.700: .DATA
.CLOSE
RST @FILNAM.DAT
.900: !***************** END RESTORE ******************
@[204,4]CHEC RESTOR END
```

Figure D-5. Restore Function Indirect Command File

```
.;  THIS INDIRECT FILE (PROGRAM) COMPRESSES THE SEL DATA BASE
.;                                                  DAVE WYCKOFF   JUL 81
.;
.; BASELINE
.; --------
.;
.; CPR                10
.;    HELP           20
.;    CMP            30
.;       FILDIR      40
.;       CMPFIL      50
.;          DEFNEW   60
.;
.ENABLE SUBSTITUTION
.ENABLE GLOBAL
.ENABLE QUIET

@[204,4]CHEC COMPRESS START
.SETS $P1 P1
.SETS $P2 P2
.SETS $VTYP "ACCATMCIFCMTCRFCSFCSRENCESTGPSHDRHISRAFRSFSEFSTS"
.SETS $UIC1 "[204,1]"     ; FOR NOW   THEN [204,1]
.SETS $UIC2 "[204,4]"     ; FOR NOW   THEN [204,3]
.SETN $NFILE 1.
.SETS L "L"
.SETN $ERTP1 0.
.SETN $ERTP2 10.
.SETF $ERROR

.PHASE  UIC  "[,]" DUMMY GROUP MEMBER DUMMY
.IF GROUP EQ "204"        .GOTO ENDF10
   !! UIC '.UIC>' CANNOT COMPRESS THE DATA BASE.  YOU MUST USE A 204 UIC.
   .STOP
.ENDF10:

.SETF $PROMT
.IF $P1 EQ ""       .SETT $PROMT
.IFT $PROMT      !! DATA BASE COMPRESS UTILITY.  ENTER HELP FOR HELP.
.IFT $PROMT      !!    A BACKUP TAPE SHOULD BE RUN BEFORE THIS COMPRESS.
.OPENA #1 MSG.DAT
.SETS DA "[204,3]CPR.PRO"
.TESTFILE 'DA'
.PARSE <FILSPC> ";" DUMMY VERS
.ENABLE DATA #1
DATA BASE COMPRESS       VERSION 'VERS'        '<DATE>'  '<TIME>'
        ACT  #REC  BLK  FILLS
.DISABLE DATA #1
.WHIL12: .; MORE USER COMMANDS AND NO PARAMETERS ON COMMAND LINE
   .IFT $PROMT      .ASKS $P1 COMPRESS>
   .IF $P1 NE "HELP" .AND .IF $P1 NE "H"     .GOTO ELSE14
      .GOSUB HELP
      .GOTO ENDF14
   .ELSE14:
      .GOSUB CMP
   .ENDF14:
.;END WHILE
.IFT $PROMT      .GOTO WHIL12
.CLOSE #1
!!
!!  END OF COMPRESS.
!!   A SUMMARY OF RESULTS IS CONTAINED IN MSG.DAT.
!!
.GOTO STOP

.HELP: .; THIS SUBROUTINE WRITES HELP INFORMATION TO THE TERMINAL.

   !!
   !!  IF YOU WANT TO COMPRESS A SINGLE FILE, ENTER THAT NAME.
   !!  WILDCARDS (*) MAY BE USED TO SPECIFY ALL PROJECTS OR ALL
   !!  FORM TYPES.  FOR EXAMPLE:
```

Figure D-6. Compress Function Indirect Command File (1 of 6)

```
        !!          * COMPRESS>[S]: DEA.RAF
        !!          * COMPRESS [S]: *.RAF
        !!          * COMPRESS [S]: DEA.*
        !!         >* COMPRESS>[S]: *.*
        !!
        !!   THESE MAY BE ENTERED ON THE COMMAND LINE.  FOR EXAMPLE:
        !!
        !!          @[204,n]COMPRESS *.RAF
        !!
.RETURN

.CMP: .; THIS SUBROUTINE READS THE DIRECTORY FILE OF SELECTED DB FILES AND
.;       COMPRESSES EACH OF THEM.

   .;  CREATE A DIRECTORY FILE CONTAINING NAMES OF FILES TO BE COMPRESSED
   .GOSUB FILDIR
   .IFT $ERROR       .GOTO ENDF30
      .OPENR #2 99DIR.TMP
      .;    DUMMY READ TO GET PAST THE HEADER
      .READ #2 DUMMY
      .IFT <EOF>          .GOTO ELSE32
         .; PIP DIR HEADER = 'DUMMY'
         .SETF $EOF
         .WHIL34: .; MORE FILES TO COMPRESS
            .READ #2 RECORD
            .SETS R RECORD[5.;12.]
            .IF R EQ "TOTAL OF" .OR .IFT <EOF>       .GOTO ELSE36      ; (SAY EOF)
               .; COMP  REC FROM PIP FILE IS 'RECORD'
               .SETS REC RECORD[3.;25.]
               .PARSE REC " " $NAME DUMMY
               .PARSE $NAME ".;" $PROJ $TYP $VER51
               .SETS $DBF1 "[204,1]"+$NAME
               .SETS $DBF2 $UIC1+$PROJ+"."+$TYP
               .; $NAME = '$NAME'
               !!
               !!      COMPRESSING FILE '$DBF2'
               !!
               .READ #2 REC2
               .; REC2 = 'REC2'
               .PARSE REC2 "." $ALC DUMMY
               .SETS $CHR $ALC
               .GOSUB CHRINT
               .SETN $ALNOW $NUM
               .; $ALC = '$ALC'     $ALNOW = '$ALNOW'
               .SETF LOCKED
               .SETS TT REC2[10,;10.]
               .; REC(10) = 'TT'
               .IF L EQ REC2[10.;10.]      .SETT LOCKED
               .IFT LOCKED       .GOTO ELSE38
                  .SETS CHAR REC2[9.;9.]
                  .;   IF COMPRESSING NON-CONTIGUOUS FILES ONLY AND NON-CONTIG.
                  .IF CHAR EQ "C" .AND .IF $P2 EQ "NOCONTIG"      .GOTO ELSE37
                     .GOSUB CMPFIL
                     .GOTO ENDF37
                  .ELSE37:
                     !!   FILE '$DBF2' IS ALREADY CONTIGUOUS; IGNORED
                  .ENDF37:
                  .GOTO ENDF38
               .ELSE38: .; FILE IS LOCKED
                  !!  FILE '$NAME' IS LOCKED (AND THEREFORE IGNORED)
                  .DATA #1  '<TIME>'          FILE '$NAME' IS LOCKED, NOT COMPRESSED
               .ENDF38:
               .GOTO ENDF36:
            .ELSE36:
               .SETT $EOF
            .ENDF36:
         .IFF $EOF      .GOTO WHIL34
         .GOTO ENDF32
      .ELSE32: .; ZERO FILES SELECTED
         !!  NO FILES SELECTED
         .DATA #1  '<TIME>'          NO FILES SELECTED
      .ENDF32:
      .CLOSE #2
      $RD 99*.TMP/DE
   .ENDF30:
.RETURN
```

Figure D-6. Compress Function Indirect Command File (2 of 6)

```
.FILDIR: .; THIS SUBROUTINE CREATES OF PIP DIRECTORY OF SELECTED DB FILES TO
.;         BE COMPRESSED

   .SETF $ERROR
   .; 'FILDIR'  $P1 = '$P1'
   .IF $P1 EQ ""      .GOTO ELSE40
      .PARSE $P1 "." $PROJ $TYP
      .TEST $PROJ
      .IF <STRLEN> GT 9.      .GOTO ELSE42
         .TEST $TYP
         .IF <STRLEN> LT 1. .OR. .IF <STRLEN> GT 3.      .GOTO ELSE44
            .;    $PROJ = '$PROJ'   $TYP = '$TYP'
            PIP 99DIR.TMP = [204,1]'$P1' LI
            .GOTO ENDF44
         .ELSE44:
            .SETT $ERROR
            !!  INVALID STRING '$P1'
         .ENDF44:
         .GOTO ENDF42
      .ELSE42:
         .SETT $ERROR
         !!   INVALID STRING '$P1'
      .ENDF42:
      .GOTO ENDF40
   .ELSE40: .; NULL LINE ENTERED; IGNORE IT
      .SETT $ERROR
   .ENDF40:
.RETURN

.CMPFIL: .. THIS SUBROUTINE DOES THE COMPRESS ON THE SINGLE GIVEN FILE.

   .TEST $VERS1
   .SETF $ACTIV
   .IF <STRLEN> LE 2      .SETT $ACTIV
   .TESTFILE '$DBF1'
   .IF <FILERR> NE 1      .GOTO ELSE50  ; (FILE NOT FOUND)
      .SETS $TY $TYP
      .IF $TYP NE "HDR"      .GOTO ENDF56
         .SETS $TY "X"
         .IF $PROJ EQ "ENCODE"      .SETS $TY "ENC"
         .IF $PROJ EQ "EST"         .SETS $TY "EST"
         .IF $PROJ EQ "HEADER"      .SETS $TY "HDR"
         .IF $PROJ EQ "SEF"         .SETS $TY "SEF"
         .IF $PROJ EQ "STAT"        .SETS $TY "STS"
         .IF $TY EQ "X"      !   INVALID FILE NAME GIVEN:  '$NAME'
      .ENDF56:
      .SETS DEFDSN $UIC2+"DEFINE"+$TY+".CMD"
      .TESTFILE 'DEFDSN'
      .IF <FILERR> NE 1      .GOTO ELSE52
         .; CREATE NEW FILE
         .GOSUB DEFNEW
         .IF $P2 EQ "DSP"      DSP '$DBF2'/FU
         .IFT $ERROR      .GOTO ENDF54
            .;  COPY OLD VERSION TO NEW VERSION (EFFECTIVELY COMPRESSING IT)
            .IF $P2 NE "NOCNV"      IFL '$DBF2'/LO = '$DBF1'/NOSO
            !!  BEFORE AND AFTER FILES:
            SRD '$DBF2'
            .SETS FLAG "I"
            .IFT $ACTIV      .SETS FLAG "A"
            .DATA #1  '<TIME>' 'FLAG' '$NFILE'  '$NREC'  '$ALLOC' '$DFILL' '$IFILL' '$DBF2'
         .ENDF54:
         .GOTO ENDF52
      .ELSE52: .; DEFINE COMMAND FILE NOT FOUND
         !! FILE DEFINITION COMMAND FILE 'DEFDSN' NOT FOUND
         .DATA #1  '<TIME>'          FILE DEFINITION COMMAND FILE 'DEFDSN' NOT FOUND
      .ENDF52:
      .GOTO ENDF50
   .ELSE50: .; NO DATA BASE FILE  (SHOULD NEVER HAPPEN)
      !!  FILE '$DBF1' NOT FOUND      (SOMETHING WRONG)
      .DATA #1  '<TIME>'          FILE '$DBF1' NOT FOUND
   .ENDF50:
   .INC $NFILE
.RETURN
```

Figure D-6. Compress Function Indirect Command File (3 of 6)

ORIGINAL PAGE IS
OF POOR QUALITY

```
..DFNEW: ;  THIS SUBROUTINE DEFINES A NEW FILE FOR THE GIVEN FILE TYPE.

   .;   DFNEW   $DBFS = '$DBFS'      $TYP = '$TYP'   $ACTIV = '$ACTIV'
   .;           $ALLOW = '$ALLOW'    $UIC2 = '$UIC2'
   .SETF $ERROR
   .SETF FOUND
   .SETS $DB $PROJ+","+$TYP
   .; RUN STS (FORTRAN) PROGRAM TO GET CURRENT NUMBER OF RECORDS IN 99REC.TMP
   .OPEN #0 99PRJ.TMP
   .DATA #0 '$DB'
   .CLOSE #0
   RUN [20,1]STS
   .OPENR #0 99REC.TMP
   .READ #0 RECORD
   .CLOSE #0
   .; CONVERT CHARACTER TO INTEGER
   .SETS $CHR RECORD[1.:8.]
   .GOSUB CHRINT
   .SETN $NREC $NUM
   .SETN $RECL 0.
   .SETN $NKEY 0.
   .;   CHR RECORD FROM 99REC.TMP = 'RECORD'    $NREC = '$NREC'
  .IF $NREC EQ 0      .GOTO EL1

   .IF $TYP NE "ACC"      .GOTO ENDF61
      .SETT FOUND
      .SETN $RECL 87.
      .SETN $NKEY 1.
   .ENDF61:

   .IF $TYP NE "CIF"      .GOTO ENDF63
      .SETT FOUND
      .SETN $RECL 80.
      .SETN $NKEY 3.
   .ENDF63:

   .IF $TYP NE "CMT"      .GOTO ENDF64
      .SETT FOUND
      .SETN $RECL 104.
      .SETN $NKEY 1.
   .ENDF64:

   .IF $TYP NE "CRF"      .GOTO ENDF65
      .SETT FOUND
      .SETN $RECL 101.
      .SETN $NKEY 1.
   .ENDF65:

   .IF $TYP NE "CSF"      .GOTO ENDF66
      .SETT FOUND
      .SETN $RECL 250.
      .SETN $NKEY 2.
   .ENDF66:

   .IF $TYP NE "CSR"      .GOTO ENDF67
      .SETT FOUND
      .SETN $RECL 79.
      .SETN $NKEY 3.
   .ENDF67:

   .IF $TYP NE "HIS"      .GOTO ENDF72
      .SETT FOUND
      .SETN $RECL 29.
      .SETN $NKEY 1.
   .ENDF72
```

Figure D-6. Compress Function Indirect Command File
(4 of 6)

```
.IF $TYP NE 'RAF'      .GOTO ENDF73
   .SETT FOUND
   .SETN $RECL 53.
   .SETN $KEY 1.
.ENDF73:

.IF $TYP NE 'RSF'      .GOTO ENDF74
   .SETT FOUND
   .SETN $RECL 115.
   .SETN $KEY 1.
.ENDF74:

.IF $TYP EQ 'HDR'      .SETT FOUND

.IFF FOUND      .GOTO ELSE98
   .IFF $ACTIV      .GOTO ELSE99
      .SETN $DFILL 1009.-(3.*($RECL+7.))
      .SETN $IFILL  950.
      .SETN NR $NREC/10.+1.
      .SETN $ALL    NR*($RECL+7.)/($DFILL/27.)+$EXTR1
      .SETN $ALLOC ($ALL/10.+1.)*10.
      .SETN $EXTEN 30.
      .SETN $FILLA  950.
      .SETN $FILLB  950.
      .SETN $FILLC  950.
      .SETN $FILLD  950.
      .GOTO ENDF99
   .ELSE99: .; INACTIVE
      .SETN NR $NREC/10.+1
      .SETN $ALL    NR*($RECL+7.)/40.+((($NKEY-1)*$EXTR2)
      .SETN $ALLOC ($ALL/10.+1.)*10.
      .SETN $EXTEN 10.
      .SETN $DFILL 1009.
      .SETN $IFILL 1009.
      .SETN $FILLA 1009.
      .SETN $FILLB 1009.
      .SETN $FILLC 1009.
      .SETN $FILLD 1009.
   .ENDF99:
   @'DEFDSN'
   .TESTFILE '$DBF2'
   .PARSE (FILSPEC) *.* DUMMY $VERS2
   .; $VERS1 = '$VERS1'    $VERS2 = '$VERS2'
   .IF $VERS2 GT $VERS1      .GOTO ENDF97
      .SETT $ERROR
      || NO NEW FILE CREATED; FILE '$DBF2' NOT COMPRESSED
      .DATA #1  '<TIME>'    '$NFILE'  '$NREC'  '$ALLOC' '$DFILL' '$IFILL' '$DBF2'  DEF PROB, NOT DONE
   .ENDF97:
   .GOTO ENDF98
.ELSE98:
   .SETT $ERROR
   || INVALID FILE TYP: '$TYP'
.ENDF98:
.GOTO EF1
.EL1: .; ELSE NO RECORDS READ ON FILE STS FILE
.SETT $ERROR
|| FILE '$DBF1' HAS A REC COUNT=0 ON THE FILE STATUS FILE.
|| SO THIS FILE WAS NOT COMPRESSED.
.DATA #1  '<TIME>'    '$NFILE'  '$DBF1' RECORD COUNT IS 0; NOT COMPRESSED
.EF1:

.RETURN

.CHRINT: .; THIS SUBROUTINE CONVERTS A CHARACTER STRING TO A DECIMAL INTEGER.

   .OPEN #3 99INT.TMP
   .ENABLE DATA #3
      .ENABLE SUBSTITUTION
      .ENABLE GLOBAL
      .SETN $NUM '$CHR'.
      .RETURN
```

Figure D-6. Compress Function Indirect Command File (5 of 6)

```
.DISABLE DATA #3
   .CLOSE #3
   @99INT.TMP
   .;   $NUM = '$NUM'    $CHR = '$CHR'
.RETURN

.STOP:
   @[204,4]CHEC COMPRESS END
   .RETURN
```

Figure D-6. Compress Function Indirect Command File (6 of 6)

```
;
;   @UPDMFN.CMD
;
;   UPDATE THE FORTRAN MODULE FUNCTION OF A COMPONENT INFORMATION FILE
;   (P. LO    5/18/82)
;
@[204,4]CHEC UPDMFN START
RUN [204,5]DMUPDMFN/TASK=UPDMFN
@[204,4]CHEC UPDMFN END
```

Figure D-7. Update FORTRAN Module Function for CIF File Function Indirect Command File

```
;
;   @UPDSFG.CMD
;
;   UPDATE STATUS FLAG OF A SEL DATA BASE FILE  (P. LO   4/23/82)
;
@[204,4]CHEC UPDSFG START
RUN [204,5]DMUPDSFG/TASK=UPDSFG
@[204,4]CHEC UPDSFG END
```

Figure D-8. Update Status Flag Function Indirect Command File

APPENDIX E - FILE DEFINITIONS

Appendix E contains listings of the file definitions.

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.; CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6/29/81
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
;$DBF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
    67          ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE    (PRIMARY KEY)
;               ;;    KEY LOCATION
     2
;               ;;    KEY LENGTH
     8
;               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
NO              ;;    MORE KEYS?
;               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
YES             ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #1)
 '$ALLOC'       ;;       INITIAL ALLOCATION (BLOCKS)
     2          ;;       BUCKET SIZE
 '$EXTEN'       ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #2)
    10          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
    10          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #3)
    10          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
    10          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
NO              ;;       MORE AREAS?
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
     0          ;;    DATA BUCKET AREA # FOR THIS KEY    (PRIMARY AND ONLY KEY)
 '$DFILL'       ;;       FILL NUMBER FOR DATA BUCKETS
     1          ;;    INDEX BUCKET AREA #
 '$IFILL'       ;;       FILL NUMBER FOR INDEX BUCKETS
     2          ;;    LOW INDEX
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;               ;;    OWNER
RWED
;               ;;    GROUP
RWED
;               ;;    SYSTEM
RWED
;               ;;    WORLD
R
.DISABLE DATA
.CLOSE #3

.; DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-1. Accounting Information (ACC) File Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.; CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6/29/81
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
 '$DBF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
    80          ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE    (PRIMARY KEY)
;               ;;    KEY LOCATION
     2
;               ;;    KEY LENGTH
     8
;               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
YES             ;;    MORE KEYS?
STR             ;;    DATA TYPE    (SECONDAY KEY)    (FIRST ALT)
;               ;;    KEY LOCATION
     2
;               ;;    KEY LENGTH
     2
;               ;;    KEY NAME (IF ANY)

YES             ;;    DUPE KEYS?
YES             ;;    ALLOW KEYS TO CHANGE
NO              ;;    DEFINE NULL KEY VALUE
YES             ;;    MORE KEYS?
STR             ;;    DATA TYPE    (TERTIARY KEY)
;               ;;    KEY LOCATION
    10
;               ;;    KEY LENGTH
     3
;               ;;    KEY NAME (IF ANY)

YES             ;;    DUPE KEYS?
YES             ;;    ALLOW KEYS TO CHANGE
NO              ;;    DEFINE NULL KEY VALUE
NO              ;;    MORE KEYS?
;               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
NO              ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #5)
 '$ALLOC'       ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
 '$EXTEN'       ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
 '$DFILL'       ;;       FILL NUMBER FOR DATA BUCKETS
 '$IFILL'       ;;       FILL NUMBER FOR INDEX BUCKETS
 '$FILLA'       ;;       FILL NUMBER FOR DATA BUCKETS   (ALT KEY 1)
 '$FILLB'       ;;       FILL NUMBER FOR INDEX BUCKETS
 '$FILLC'       ;;       FILL NUMBER FOR DATA BUCKETS   (ALT KEY 2)
 '$FILLD'       ;;       FILL NUMBER FOR INDEX BUCKETS
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;               ;;    OWNER
RWED
;               ;;    GROUP
RWED
;               ;;    SYSTEM
RWED
;               ;;    WORLD
R
.DISABLE DATA
.CLOSE #3

.; DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-2. Component Information File (CIF) Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.; CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6/29/81
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
'$DEF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
   104          ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE   (PRIMARY KEY)
;               ;;    KEY LOCATION
     0
;               ;;    KEY LENGTH
    10
;               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
NO              ;;    MORE KEYS?
;               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
YES             ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?   (AREA #1)
  '$ALLOC'      ;;       INITIAL ALLOCATION (BLOCKS)
     2          ;;       BUCKET SIZE
  '$EXTEN'      ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?   (AREA #2)
    10          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
    10          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?   (AREA #3)
    10          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
    10          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
NO              ;;       MORE AREAS?
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
     0          ;;    DATA BUCKET AREA # FOR THIS KEY   (PRIMARY AND ONLY KEY)
  '$DFILL'      ;;       FILL NUMBER FOR DATA BUCKETS
     1          ;;    INDEX BUCKET AREA #
  '$IFILL'      ;;       FILL NUMBER FOR INDEX BUCKETS
     2          ;;    LOW INDEX
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;               ;;    OWNER
RWED
;               ;;    GROUP
RWED
;               ;;    SYSTEM
RWED
;               ;;    WORLD
R
.DISABLE DATA
.CLOSE #3

.; DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-3. Comments (CMT) File Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.; CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6-29-81
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
;$DBF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
   101          ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE   (PRIMARY KEY)
;               ;;    KEY LOCATION
     0
;               ;;    KEY LENGTH
     6
;               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
NO              ;;    MORE KEYS?
;               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
YES             ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?   (AREA #1)
 '$ALLOC'       ;;       INITIAL ALLOCATION (BLOCKS)
     2          ;;       BUCKET SIZE
 '$EXTEN'       ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?   (AREA #2)
    10          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
    10          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?   (AREA #3)
    10          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
    10          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
NO              ;;       MORE AREAS?
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
     0          ;;    DATA BUCKET AREA # FOR THIS KEY   (PRIMARY AND ONLY KEY)
 '$DFILL'       ;;       FILL NUMBER FOR DATA BUCKETS
     1          ;;    INDEX BUCKET AREA #
 '$IFILL'       ;;       FILL NUMBER FOR INDEX BUCKETS
     2          ;;    LOW INDEX
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;               ;;    OWNER
RWED
;               ;;    GROUP
RWED
;               ;;    SYSTEM
RWED
;               ;;    WORLD
R
.DISABLE DATA
.CLOSE #3

.; DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-4. Change Report Form (CRF) File Definition

ORIGINAL PAGE IS
OF POOR QUALITY

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.; CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6/29/81
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
'$DBF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
   250          ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE    (PRIMARY KEY)
;               ;;    KEY LOCATION
     0
;               ;;    KEY LENGTH
     6
;               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
YES             ;;    MORE KEYS?
STR             ;;    DATA TYPE    (SECONDAY KEY)    (FIRST ALT)
;               ;;    KEY LOCATION
    25
;               ;;    KEY LENGTH
     3
;               ;;    KEY NAME (IF ANY)

YES             ;;    DUPE KEYS?
YES             ;;    ALLOW KEYS TO CHANGE
NO              ;;    DEFINE NULL KEY VALUE
NO              ;;    MORE KEYS?
;               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
NO              ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #5)
  '$ALLOC'      ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
  '$EXTEN'      ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
  '$DFILL'      ;;       FILL NUMBER FOR DATA BUCKETS
  '$IFILL'      ;;       FILL NUMBER FOR INDEX BUCKETS
  '$FILLA'      ;;       FILL NUMBER FOR DATA BUCKETS   (ALT KEY 1)
  '$FILLB'      ;;       FILL NUMBER FOR INDEX BUCKETS
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;               ;;    OWNER
RWED
;               ;;    GROUP
RWED
;               ;;    SYSTEM
RWED
;               ;;    WORLD
R
.DISABLE DATA
.CLOSE #3

.; DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-5. Component Summary Form (CSF) File Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.; CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6/29/81
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
'$DBF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
    79          ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE    (PRIMARY KEY)
;               ;;    KEY LOCATION
     0
;               ;;    KEY LENGTH
     8
;               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
YES             ;;    MORE KEYS?
STR             ;;    DATA TYPE    (SECONDAY KEY)    (FIRST ALT)
;               ;;    KEY LOCATION
    21
;               ;;    KEY LENGTH
     3
;               ;;    KEY NAME (IF ANY)

YES             ;;    DUPE KEYS?
YES             ;;    ALLOW KEYS TO CHANGE
NO              ;;    DEFINE NULL KEY VALUE
YES             ;;    MORE KEYS?
STR             ;;    DATA TYPE    (TERTIARY KEY)
;               ;;    KEY LOCATION
    10
;               ;;    KEY LENGTH
     5
;               ;;    KEY NAME (IF ANY)

YES             ;;    DUPE KEYS?
YES             ;;    ALLOW KEYS TO CHANGE
NO              ;;    DEFINE NULL KEY VALUE
NO              ;;    MORE KEYS?
;               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
NO              ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #5)
  '$ALLOC'      ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
  '$EXTEN'      ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
  '$DFILL'      ;;       FILL NUMBER FOR DATA BUCKETS
  '$IFILL'      ;;       FILL NUMBER FOR INDEX BUCKETS
  '$FILLA'      ;;       FILL NUMBER FOR DATA BUCKETS  (ALT KEY 1)
  '$FILLB'      ;;       FILL NUMBER FOR INDEX BUCKETS
  '$FILLC'      ;;       FILL NUMBER FOR DATA BUCKETS  (ALT KEY 2)
  '$FILLD'      ;;       FILL NUMBER FOR INDEX BUCKETS
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;               ;;    OWNER
RWED
;               ;;    GROUP
RWED
;               ;;    SYSTEM
RWED
;               ;;    WORLD

.DISABLE DATA
.CLOSE #3

.; DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-6. Component Status Report (CSR) File Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.; CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6/29/81
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
  '$DBF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
    60          ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE    (PRIMARY KEY)
;               ;;    KEY LOCATION
     0
;               ;;    KEY LENGTH
     8
;               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
YES             ;;    MORE KEYS?
STR             ;;    DATA TYPE    (SECONDARY KEY)    (FIRST ALT)
;               ;;    KEY LOCATION
  (0,8)
;               ;;    KEY LENGTH
  (3,8)
;               ;;    KEY NAME (IF ANY)

YES             ;;    DUPE KEYS?
YES             ;;    ALLOW KEYS TO CHANGE
NO              ;;    DEFINE NULL KEY VALUE
NO              ;;    MORE KEYS?
;               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
NO              ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #5)
   100          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
    10          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
    804         ;;       FILL NUMBER FOR DATA BUCKETS
    950         ;;       FILL NUMBER FOR INDEX BUCKETS
    950         ;;       FILL NUMBER FOR DATA BUCKETS  (ALT KEY 1)
    950         ;;       FILL NUMBER FOR INDEX BUCKETS
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;               ;;    OWNER
RWED
;               ;;    GROUP
RWED
;               ;;    SYSTEM
RWED
;               ;;    WORLD
R
.DISABLE DATA
.CLOSE #3

.; DEFINE NEW INDEXED FILE
DEF 99DEF.TMP

.RETURN
```

Figure E-7. Encoding Dictionary (ENC) File Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.! CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
!             !! DCW   6/29/81
!             !! THE FIRST QUESTION ASKS FOR T   FILE SPECIFICATION.
 '$DBF2'
NO
!             !! THE NEXT QUESTIONS D L WITH FILE ORGANIZATION & RECORD ATTR
IDX           !!    FILE ORGANIZATION
FIX           !!    RECORD FORMAT
   120        !!    RECORD SIZE
YES           !!    CARRIAGE RETURN CONTROL
!             !! THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR           !!    DATA TYPE   (PRIMARY KEY)
!             !!    KEY LOCATION
     0
!             !!    KEY LENGTH
     2
!             !!    KEY NAME (IF ANY)

NO            !!    DUPE KEYS?
YES           !!    MORE KEYS?
STR           !!    DATA TYPE   (SECONDAY KEY)   (FIRST ALT)
!             !!    KEY LOCATION
     0
!             !!    KEY LENGTH
    . 8       .
!             !!    KEY NAME (IF ANY)

YES           !!    DUPE KEYS?
YES           !!    ALLOW KEYS TO CHANGE
NO            !!    DEFINE NULL KEY VALUE
NO            !!    MORE KEYS?
!             !! THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
NO            !!    DEFINE AREAS?
NO            !!       PLACEMENT CONTROL?   (AREA #5)
    30        !!       INITIAL ALLOCATION
   . 2        !!       BUCKET SIZE
    10        !!       DEF EXTENSION
YES           !!       CONTIGUOUS
!             !! THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
    854       !!       FILL NUMBER FOR DATA BUCKETS
    950       !!       FILL NUMBER FOR INDEX BUCKETS
    950       !!       FILL NUMBER FOR DATA BUCKETS  (ALT KEY 1)
    950       !!       FILL NUMBER FOR INDEX BUCKETS
!             !! THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
!             !!    OWNER
RWED
!             !!    GROUP
RWED
!             !!    SYSTEM
RWED
!             !!    WORLD
R
.DISABLE DATA
.CLOSE #3

.! DEFINE NEW INDEXED FILE
DEF @99DEF.TMP
```

Figure E-8. Estimated Statistics (EST) File Definition

ORIGINAL PAGE IS
OF POOR QUALITY

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.! CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;             !! DCW   6/29/81
;             !! THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
 '$DBF2'
NO
;             !! THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX           !!    FILE ORGANIZATION
FIX           !!    RECORD FORMAT
   112        !!    RECORD SIZE
YES           !!    CARRIAGE RETURN CONTROL
;             !! THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR           !!    DATA TYPE    (PRIMARY KEY)
;             !!    KEY LOCATION
     8
;             !!    KEY LENGTH
     2
;             !!    KEY NAME (IF ANY)

NO            !!    DUPE KEYS?
YES           !!    MORE KEYS?
STR           !!    DATA TYPE    (SECONDAY KEY)    (FIRST ALT)
;             !!    KEY LOCATION
     0
;             !!    KEY LENGTH
     8
;             !!    KEY NAME (IF ANY)

YES           !!    DUPE KEYS?
YES           !!    ALLOW KEYS TO CHANGE
NO            !!    DEFINE NULL KEY VALUE
NO            !!    MORE KEYS?
;             !! THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
NO            !!    DEFINE AREAS?
NO            !!       PLACEMENT CONTROL?    (AREA #5)
    30        !!       INITIAL ALLOCATION
     2        !!       BUCKET SIZE
    10        !!       DEF EXTENSION
YES           !!       CONTIGUOUS
;             !! THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
   833        !!       FILL NUMBER FOR DATA BUCKETS
   950        !!       FILL NUMBER FOR INDEX BUCKETS
   950        !!       FILL NUMBER FOR DATA BUCKETS  (ALT KEY 1)
   950        !!       FILL NUMBER FOR INDEX BUCKETS
;             !! THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;             !!    OWNER
RWED
;             !!    GROUP
RWED
;             !!    SYSTEM
RWED
;             !!    WORLD
R
.DISABLE DATA
.CLOSE #3

.! DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-9. Phase Dates (HDR) File Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.; CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6-29-81   *
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
; $DBF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
    29          ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE   (PRIMARY KEY)
;               ;;    KEY LOCATION
     2
;               ;;    KEY LENGTH
     6
;               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
NO              ;;    MORE KEYS?
;               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
YES             ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?   (AREA #1)
 '$ALLOC'       ;;       INITIAL ALLOCATION (BLOCKS)
     2          ;;       BUCKET SIZE
 '$EXTEN'       ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?   (AREA #2)
     2          ;;       INITIAL ALLOCATION
     3          ;;       BUCKET SIZE
     2          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?   (AREA #3)
     2          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
     2          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
NO              ;;       MORE AREAS?
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
    0           ;;    DATA BUCKET AREA # FOR THIS KEY   (PRIMARY AND ONLY KEY)
 '$DFILL'       ;;       FILL NUMBER FOR DATA BUCKETS
     1          ;;    INDEX BUCKET AREA #
 '$IFILL'       ;;       FILL NUMBER FOR INDEX BUCKETS
     2          ;;    LOW INDEX
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;               ;;    OWNER
RWED
;               ;;    GROUP
RWED
;               ;;    SYSTEM
RWED
;               ;;    WORLD
R
.DISABLE DATA
.CLOSE #3

.; DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-10. Growth History (HIS) File Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.; CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6/29/81
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
'$DBF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
     53         ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE    (PRIMARY KEY)
:               ;;    KEY LOCATION
     0
:               ;;    KEY LENGTH
     8
;               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
NO              ;;    MORE KEYS?
;               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
YES             ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #1)
  '$ALLOC'      ;;       INITIAL ALLOCATION (BLOCKS)
     2          ;;       BUCKET SIZE
  '$EXTEN'      ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #2)
     10         ;;       INITIAL ALLOCATION
      2         ;;       BUCKET SIZE
     10         ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #3)
     10         ;;       INITIAL ALLOCATION
      2         ;;       BUCKET SIZE
     10         ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
NO              ;;       MORE AREAS?
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
      0         ;;    DATA BUCKET AREA # FOR THIS KEY    (PRIMARY AND ONLY KEY)
  '$DFILL'      ;;       FILL NUMBER FOR DATA BUCKETS
      1         ;;    INDEX BUCKET AREA #
  '$IFILL'      ;;       FILL NUMBER FOR INDEX BUCKETS
      2         ;;    LOW INDEX
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;               ;;    OWNER
RWED
;               ;;    GROUP
RWED
;               ;;    SYSTEM
RWED
;               ;;    WORLD
R
.DISABLE DATA
.CLOSE #3

.; DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-11. Run Analysis Form (RAF) File Definition

ORIGINAL PAGE IS
OF POOR QUALITY

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.; CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6/29/81
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
 $DBF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
    115         ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE    (PRIMARY KEY)
;               ;;    KEY LOCATION
     0
;               ;;    KEY LENGTH
     8
;               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
NO              ;;    MORE KEYS?
;               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
YES             ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #1)
 '$ALLOC'       ;;       INITIAL ALLOCATION (BLOCKS)
     2          ;;       BUCKET SIZE
 '$EXTEN'       ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #2)
    10          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
    10          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
YES             ;;    MORE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #3)
    10          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
    10          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS AREAS?
NO              ;;       MORE AREAS?
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
     0          ;;    DATA BUCKET AREA # FOR THIS KEY    (PRIMARY AND ONLY KEY)
 '$DFILL'       ;;       FILL NUMBER FOR DATA BUCKETS
     1          ;;    INDEX BUCKET AREA #
 '$IFILL'       ;;       FILL NUMBER FOR INDEX BUCKETS
     2          ;;    LOW INDEX
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
;               ;;    OWNER
RWED
;               ;;    GROUP
RWED
;               ;;    SYSTEM
RWED
;               ;;    WORLD
R
.DISABLE DATA
.CLOSE #3

.; DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-12. Resource Summary Form (RSF) File Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.! CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 @99DEF.TMP
.ENABLE DATA #3
;               ;; DCW   6/29/81
;               ;; THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
'$DBF2'
NO
;               ;; THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             ;;    FILE ORGANIZATION
FIX             ;;    RECORD FORMAT
    52          ;;    RECORD SIZE
YES             ;;    CARRIAGE RETURN CONTROL
;               ;; THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             ;;    DATA TYPE    (PRIMARY KEY)
:               ;;    KEY LOCATION
     0
;               ;;    KEY LENGTH
     4
:               ;;    KEY NAME (IF ANY)

NO              ;;    DUPE KEYS?
YES             ;;    MORE KEYS?
STR             ;;    DATA TYPE    (SECONDAY KEY)    (FIRST ALT)
;               ;;    KEY LOCATION
     4
;               ;;    KEY LENGTH
    25
;               ;;    KEY NAME (IF ANY) .

YES             ;;    DUPE KEYS?
YES             ;;    ALLOW KEYS TO CHANGE
NO              ;;    DEFINE NULL KEY VALUE
NO              ;;    MORE KEYS?
:               ;; THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
NO              ;;    DEFINE AREAS?
NO              ;;       PLACEMENT CONTROL?    (AREA #5)
    70          ;;       INITIAL ALLOCATION
     2          ;;       BUCKET SIZE
    10          ;;       DEF EXTENSION
YES             ;;       CONTIGUOUS
;               ;; THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
   885          ;;       FILL NUMBER FOR DATA BUCKETS
   950          ;;       FILL NUMBER FOR INDEX BUCKETS
   950          ;;       FILL NUMBER FOR DATA BUCKETS  (ALT KEY 1)
   950          ;;       FILL NUMBER FOR INDEX BUCKETS
;               ;; THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
:               ;;    OWNER
RWED
:               ;;    GROUP
RWED
:               ;;    SYSTEM
RWED
:               ;;    WORLD
R
.DISABLE DATA
.CLOSE #3

.! DEFINE NEW INDEXED FILE
DEF @99DEF.TMP

.RETURN
```

Figure E-13. File Name and Status (STS) File Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.! CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
!               !! PLL   12/13/82
!               !! THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
'$DBF2'
NO
!               !! THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX             !!    FILE ORGANIZATION
VAR             !!    RECORD FORMAT
   578          !!    RECORD SIZE
YES             !!    CARRIAGE RETURN CONTROL
!               !! THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR             !!    DATA TYPE   (PRIMARY KEY)
!               !!    KEY LOCATION
     0
!               !!    KEY LENGTH
     3
!               !!    KEY NAME (IF ANY)

NO              !!    DUPE KEYS?
NO              !!    MORE KEYS?
!               !! THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
NO              !!    DEFINE AREAS?
NO              !!       PLACEMENT CONTROL?   (AREA #5)
   150          !!       INITIAL ALLOCATION
     2          !!       BUCKET SIZE
    40          !!       DEF EXTENSION
NO              !!       CONTIGUOUS
!               !! THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
   950          !!       FILL NUMBER FOR DATA BUCKETS
   950          !!       FILL NUMBER FOR INDEX BUCKETS
   950          !!       FILL NUMBER FOR DATA BUCKETS  (ALT KEY 1)
   950          !!       FILL NUMBER FOR INDEX BUCKETS
!               !! THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
!               !!    OWNER
RWED
!               !!    GROUP
RWED
!               !!    SYSTEM
RWED
!               !!    WORLD
R
.DISABLE DATA
.CLOSE #3

.! DEFINE NEW INDEXED FILE
DEF @99DEF.TMP
```

Figure E-14. Subjective Evaluations (SEF) File Definition

```
.ENABLE SUBSTITUTION
.ENABLE GLOBAL

.! CREATE DEFINE COMMAND FILE WITH APPROPRIATE PARAMETERS
.OPEN #3 99DEF.TMP
.ENABLE DATA #3
!             !! PLL   12/13/82
!             !! THE FIRST QUESTION ASKS FOR THE FILE SPECIFICATION.
 '$DBF2'
NO
!             !! THE NEXT QUESTIONS DEAL WITH FILE ORGANIZATION & RECORD ATTR
IDX           !!    FILE ORGANIZATION
FIX           !!    RECORD FORMAT
   100        !!    RECORD SIZE
YES           !!    CARRIAGE RETURN CONTROL
!             !! THE FOLLOWING QUESTIONS DEAL WITH KEYS
STR           !!    DATA TYPE   (PRIMARY KEY)
!             !!    KEY LOCATION
     0
!             !!    KEY LENGTH
     4
!             !!    KEY NAME (IF ANY)

NO            !!    DUPE KEYS?
YES           !!    MORE KEYS?
STR           !!    DATA TYPE   (SECONDAY KEY)   (FIRST ALT)
!             !!    KEY LOCATION
     4
!             !!    KEY LENGTH
     8
!             !!    KEY NAME (IF ANY)

YES           !!    DUPE KEYS?
YES           !!    ALLOW KEYS TO CHANGE
NO            !!    DEFINE NULL KEY VALUE
NO            !!    MORE KEYS?
!             !! THE NEXT QUESTIONS DEAL WITH ALLOCATION AND PLACEMENT ATTRIBUTES
NO            !!    DEFINE AREAS?
NO.           !!       PLACEMENT CONTROL?   (AREA #5)
    40        !!       INITIAL ALLOCATION
     2        !!       BUCKET SIZE
    10        !!       DEF EXTENSION
NO            !!       CONTIGUOUS
!             !! THE NEXT QUESTIONS DEAL WITH ASSOC KEYS TO AREAS + FILL SIZES
    856       !!       FILL NUMBER FOR DATA BUCKETS
    950       !!       FILL NUMBER FOR INDEX BUCKETS
    950       !!       FILL NUMBER FOR DATA BUCKETS  (ALT KEY 1)
    950       !!       FILL NUMBER FOR INDEX BUCKETS
!             !! THE FOLLOWING QUESTIONS DEAL WITH FILE PROTECTION
!             !!    OWNER
RWED
!             !!    GROUP
RWED
!             !!    SYSTEM
RWED
!             !!    WORLD
R
.DISABLE DATA
.CLOSE #3

.! DEFINE NEW INDEXED FILE
DEF @99DEF.TMP
```

Figure E-15. Subjective Evaluations Directory (DIR) File Definition

APPENDIX F - SAMPLE DATA COLLECTION FORMS

The forms reproduced here are used by the SEL at the Goddard Space Flight Center to collect data on development projects. The terms used in these forms are defined in Section F.2.

F.1 SAMPLE DATA COLLECTION FORMS AND INSTRUCTIONS

This section contains sample data collection forms and instructions for their use. The instructions precede the forms. The following forms are included:

1. General Project Summary (GPS)
2. Resource Summary Form (RSF)
3. Component Summary Form (CSF)
4. Component Status Report (CSR)
5. Run Analysis Form (RAF)
6. Change Report Form (CRF) and Attitude Maintenance Change Report (ATM)

INSTRUCTIONS FOR COMPLETING THE GENERAL PROJECT SUMMARY - FORM 580-1 (2/77)

This form is used to classify the project and will be used in conjunction with the other reporting forms to measure the estimated versus actual development progress. It should be filled out by the project manager at the beginning of the project, at each major milestone, and at the end. Numbers and dates used at the initiation of the project are assumed to be estimated; intermediate reports should change estimates to actuals (if known) and update estimates. The final report should accurately describe the system development life cycle.

A. PROJECT DESCRIPTION

Description. Give an overview of the project.

Inputs. Specifications and requirements (etc.) of project. Give the format of these.

Requirements. How requirements are established and changed.

Products Developed. List all items developed for the project (e.g., operational system, testing system, simulator, etc.).

Products Delivered. List all items required to be delivered (e.g., source of the operational system, object code of the operational system, design documents, etc.).

B. RESOURCES

Target Computer System. System for which software was developed.

Development Computer System. System on which software was developed.

Constraints. List any size or time constraints for the finished product. Do you anticipate any problems in meeting these constraints?

Useful Items From Similar Projects:

1. List previous projects, which will contribute various aspects to this project.
2. For each project, give the percent of the current project it makes up in each of the 3 listed aspects.
3. For each of the 3 listed aspects (specification, design, code) check what level of modifications are necessary.

C. TIME

Start Date. First date of work, including design and modification of the specifications.

End Date. Delivery date.

Estimated Lifetime. Estimate the operational life of the system.

Mission Date. Scheduled operation date of the system (write unknown if not known or undecided yet on any of these dates). Date project must be operational.

Confidence Level. Give the percent probability you think the end date is realistic. (e.g., 100% means certain delivery on that date, 0% means no chance of delivery.)

D. COST

Cost. Total amount of money the project costs, including both contract and in-house costs.

Maximum Available. Maximum amount available, independent of what estimated cost is.

Confidence Level. Rate percent reliability in cost estimate.

How Determined. At initiation how is it estimated, at completion how is it calculated.

Personnel. Give the number of full time equivalent persons required at inception of the project, 1/3 of the way into the project, 2/3 of the way into the project, at the completion of the project.

Total Person Months. Give the total number of months that full time equivalent personnel (managers, designers, programmers, keypunchers, editors, secretaries, etc.) are assigned to the project. Do not include all overhead items such as vacation and sick leave.

Computer Time. Give the total number of hours on all systems normalized to one machine (e.g., the IBM 360/75) and name the machine.

E. SIZE

Size of the System. Include the total amount of machine space needed for all instructions generated on the project plus the space for data, library routines (e.g., FORTRAN I/O package) and other code already available. Break down size into data space and instruction space.

Confidence Level. Rate percent reliability in size estimates.

Total Number of Source Statements. Give the number of FORTRAN, ALC, or any other language instructions generated specifically for this project.

Structure of System. Give overall structure of system. Is it a single load module, is it an overlay structure, or is it a set of independent programs? For overlay and separate programs, give the number and average size of each.

Define Your Concept of a Module. Give the criteria you are using to divide the software into modules.

Estimated Number of Modules. Include only the number of new modules to be written.

Range in Module Size. Give the number of instructions in the minimum, maximum and average module and the language in which they are written as a reference.

Number of Different I/O Formats Used. Give the number of distinct external data sets that are required for the system including card reader, printer, graphics device, and temporary files.

F. COMPUTER ACCESS

A librarian is a person who can be used to perform any of the clerical functions associated with programming, including those given on the chart. Check the appropriate boxes for those persons who have access to the computer to perform the given functions. Give the percentage of time spent by each in batch and interactive access to the computer.

G. TECHNIQUES EMPLOYED

For "level," specify to what level of detail in the finished project the technique is used. (e.g., subroutine, module, segments of 1000 lines, top level, etc.)

Specifications

Functional - Components are described as a set of functions, each component performing a certain activity.

Procedural - Components are specified in some algorithmic manner (e.g., using a PDL).

English - Components are specified using an English Language prose statement of the problem.

Formal - Some other formal system is used to specify the components.

Design and Development

Top Down - The implementation of the system one level at a time, with the current level and expansion of the yet to be defined subroutines at the previous higher level.

Bottom Up - The implementation of the system starting with the lowest level routines and proceeding one level at a time to the higher level routines.

Iterative Enhancement - The implementation of successive implementations, each producing a usable subset of the final product until the entire system is fully developed.

Hardest First - The implementation of the most difficult aspects of the system first.

Other - Describe the strategy used if it is not a combination of any of the above.

None Specified - No particular strategy has been specified.

Coding. The final encoding of the implementation in an executable programming language.

Structured Code With Simulated Constructs - The language does not support structured control structures (e.g., FORTRAN) but they are simulated with the existing structures; please state the structured control structures you are using (e.g., WHILE, CASE, IF).

Structured Control Constructs - The language supports structured control structures (e.g., a FORTRAN preprocessor) please list structures you are using.

Other Standard - Describe any other standard you are using.

None Specified - No particular strategy has been specified.

Validation/Verification. Testing: execution of the system, via a set of test cases.

Top Down - Stubs or dummy procedures are written to handle the yet to be implemented aspects of the system and testing begins with the top level routines and proceeds as new levels are added to the system.

Bottom Up - Check out of a module at a time using test drivers and starting at the bottom level modules first.

Structure Driven - Using structure of program to determine test date (e.g., every statement of program executed at least once).

Specification Driven - Using specifications of program to determine test data (e.g., all input/output relationships hold for a set of test data).

Other - Describe any other strategy you are using.

None Specified - No testing strategy has been specified.

Validation/Verification. Inspection: visual examination of the code or design.

Code Reading - Visual inspection of the code or design by other programmers.

Walk Throughs - Formal meeting sessions for the review of code and design by the various members of the project, for technical rather than management purposes.

Proofs - Formal proofs of the design or code; please specify the techniques used, e.g., axiomatic, predicate transforms, functional, etc.

None Specified - No inspection techniques have been specified.

There is some space given to permit the further explanation of any of the strategies that may be used.

H. FORMAL NOTATIONS USED AT VARIOUS LEVELS AND PHASES

Give the phases (e.g., design, implementation, testing, etc.) and levels (subroutine, module, segments of 1000 lines, top level, etc.) at which any type of formalism (flowchart, PDL, etc.) will be used in the development of the system.

I. AUTOMATED TOOLS USED

Name all automated tools used, including automated versions of the formalisms given above and compilers for the programming languages used, and at which phase and at what level they are used. Include any products that may be developed as part of this project (e.g., simulator).

J. ORGANIZATION

Describe how the personnel are subdivided with respect to responsibilities into teams or groups, giving titles, brief job descriptions, the number of people satisfying that title and their names and organizational affiliations if known.

K. STANDARDS

List all standards used, whether they are required or optional, and the title of the document describing the standard.

L. MILESTONES

Give the phase at which management may check on progress of the development of the system (e.g., specification, design, implementation of version 1, etc.). State also the date at which it should take place (at completion of the project), how it is to be determined that the milestone was reached, who will be responsible for reviewing the progress at that point and what the review procedure will be. Also give the resources used since the last milestone. For

size of system give the current size of the system at that milestone. Each milestone has 2 confidence levels, one for time estimates and one for resource expenditures. For estimated future milestone, the first confidence level for the probability of reaching the milestone at that date. The second is for the accuracy of the resources used. For past milestones, the first confidence level is normally 100% (actual date) while the second is an estimate on the accuracy of the accounting system.

M. DOCUMENTATION

For each time of documentation developed, state the type of documentation, its purpose, the date it should be completed, its size and list any tools used in its production. (At the beginning of the project these should be estimates, at the end of the project, they should be accurate figures.)

N. PROBLEMS

Give the three most difficult problems you expect to encounter managing this project. Please be as specific as possible.

O. QUALITY ASSURANCE

To what do you attribute your confidence in the completed system. Be as specific as possible.

GENERAL PROJECT SUMMARY

PROJECT NAME _____ DATE _____

A. PROJECT DESCRIPTION

Description _____

_____

Form of Input _____

Requirements _____

Products Developed _____

_____

Products Delivered _____

_____

B. RESOURCES

Target Computer Systems _____ Development Computer Systems _____

Constraints: Execution Time _____ Size _____

Other _____

Any Problems in Meeting Constraints? _____

_____

Useful Items from Similar Projects:

Project	Specification				Design				Code			
	%	Major	Minor	None	%	Major	Minor	None	%	Major	Minor	None

C. TIME

Start Date _____ End Date _____ Estimated Lifetime _____ Mission Date _____

Confidence Level _____

D. Cost

Cost $ _____ Maximum Available $ _____ Confidence Level _____

How Cost Determined _____

Personnel: Inception _____ 1/3 Way _____ 2/3 Way _____ Completion _____

Total Person Months _____

Other Costs: Computer Time _____ (hrs) Documentation $ _____

Other ( ) _____ Other ( ) _____

E. SIZE

Size of System _____ Words. _____ Data Words _____ Instructions

Maximum Space Available _____ Words. Confidence Level _____

Total Number of Source Statements: FORTRAN _____ ALC _____

Other ( ) _____

Structure of System (Check One):

___ Single Overlay

___ Overlay Structure (Number of Overlays _____ Avg. Size _____)

___ Independent Programs (Number of Programs _____ Avg. Size _____)

Define Your Concept of a Module _____

_____

Number of Modules _____ Range in Module Size: Min. _____ Max. _____ Avg. _____

Number of Different I/O Formats _____

SEC-1 (2/77)

F. COMPUTER ACCESS (Check All That Apply. Who Has Access to What.)

	Librarian	Programmer
Keying in New Source Code		
Keying in Update of Source Code		
Inclusion of Code Into System		
Submitting Compilations		
Module Testing		
Integration Testing		
Utility Runs (Tape Backup, Etc.)		

Give Percentages for Types of Access:

	Librarian	Programmer
% Batch		
% Interactive		

G. TECHNIQUES EMPLOYED (Check All That Apply and Give Level at Which Used.)

Specification:	Used	Level		Used	Level
Functional			Procedural		
English			Formal		

Design:					
Top Down			Bottom Up		
Iterative Enhance.			Hardest First		
Other:			None Used		

Development:					
Top Down			Bottom Up		
Iterative Enhance.			Hardest First		
Other:			None Used		

Coding:					
Simulating Construct			Structured Code		
Other:			None		

Validation/Verification: Testing					
Top Down (Stubs)			Bottom Up (Drivers)		
Other:			Specification Driven		
Structure Driven			None		

Validation/Verification: Inspection					
Code Reading			Walk Through		
Proof:			None		

H. FORMALISMS USED

	Used	Level	Phases
PDL			
HIPO			
Flowcharts			
Baseline Diag. (Tree Ch.)			
HOS			
Functions			
Other:			
Other:			

I. AUTOMATED TOOLS USED

Name	Phases in Which Used	Level

J. ORGANIZATION

How are the Personnel Organized: _____

_____

Project Personnel:

Title	Job Description	Number	Names and Affiliations (If Known)

K. STANDARDS

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

SSD-1 (2/77) Continuation

ORIGINAL PAGE IS
OF POOR QUALITY

L. MILESTONES

Phase ____________ Estimated Date ____________ Confidence Level ______
How Determined ____________
Reviewers ____________
Reporting Procedure ____________
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ____________ Estimated Date ____________ Confidence Level ______
How Determined ____________
Reviewers ____________
Reporting Procedure ____________
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ____________ Estimated Date ____________ Confidence Level ______
How Determined ____________
Reviewers ____________
Reporting Procedure ____________
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ____________ Estimated Date ____________ Confidence Level ______
How Determined ____________
Reviewers ____________
Reporting Procedure ____________
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ____________ Estimated Date ____________ Confidence Level ______
How Determined ____________
Reviewers ____________
Reporting Procedure ____________
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ____________ Estimated Date ____________ Confidence Level ______
How Determined ____________
Reviewers ____________
Reporting Procedure ____________
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ____________ Estimated Date ____________ Confidence Level ______
How Determined ____________
Reviewers ____________
Reporting Procedure ____________
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ____________ Estimated Date ____________ Confidence Level ______
How Determined ____________
Reviewers ____________
Reporting Procedure ____________
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

580-1 (2/77) Continuation

M. DOCUMENTATION

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

N. PROBLEMS

State the three most difficult problems you expect to encounter in completing the project. (1 = most difficult)

1. _____

2. _____

3. _____

O. QUALITY ASSURANCE

State the three most important aspects of the design, development and testing of the system to which you attribute your confidence in the completed system. (1 = most important)

1. _____

2. _____

3. _____

PERSON FILLING OUT FORM _____

980-1 (2/77) Continuation

INSTRUCTIONS FOR COMPLETING THE RESOURCE SUMMARY

This form keeps track of the project costs on a weekly basis. It should be filled out by the project manager every week of the project duration.

PROJECT. Give project name.

DATE. List date form turned in.

NAME. Name of project manager.

WEEK OF. List date of each successive Friday.

MANPOWER. List all personnel on the project on separate lines. Give the number of hours each spent that week on the project.

% OF MANAGEMENT. Add the % of time this person spent managing the project during this reporting period. A new form should be used if this % changes.

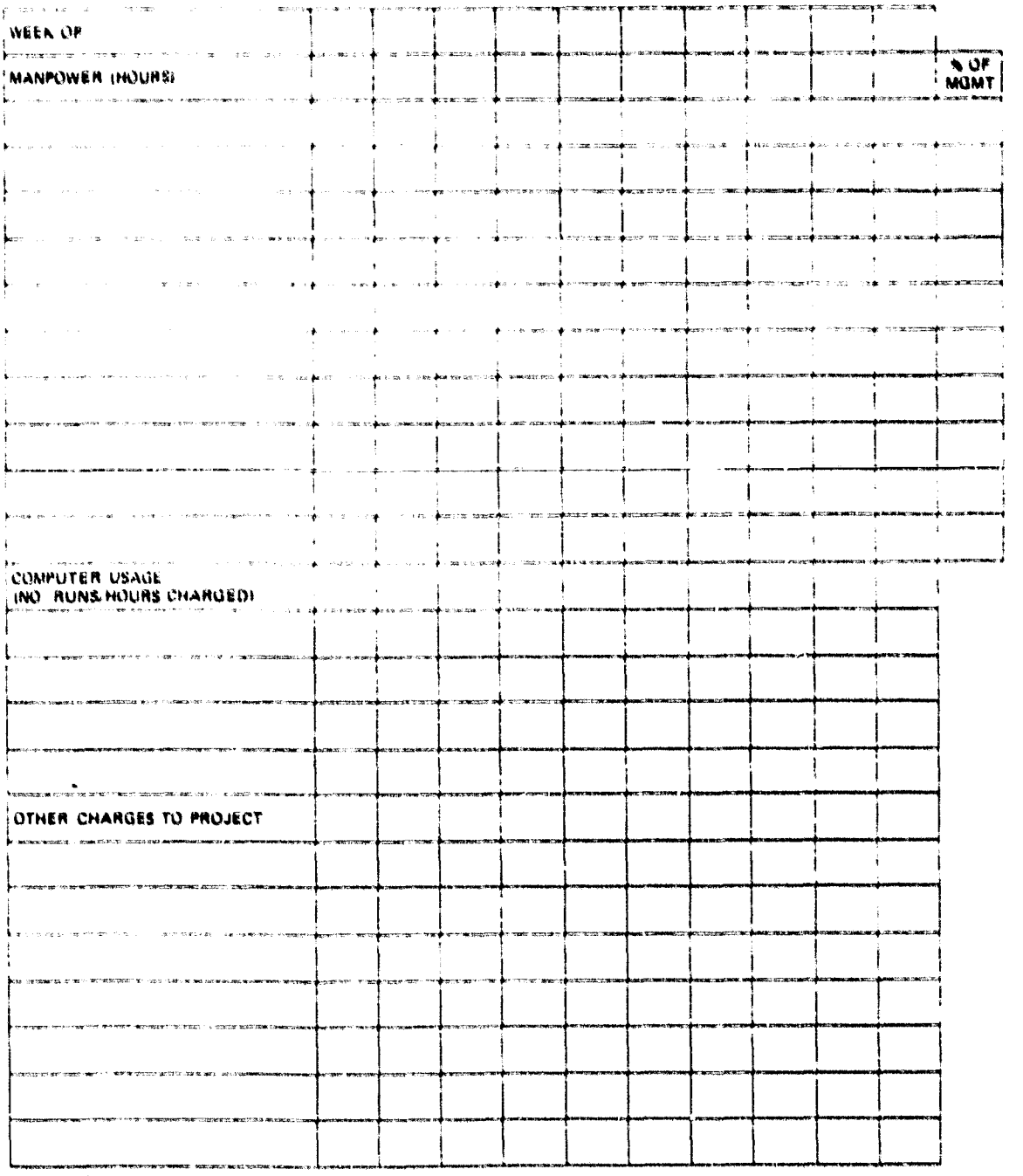
COMPUTER USAGE. List all machines used on the projec or each machine give the number of runs during each week and the amount of computer time used.

OTHER. List any other charges to the project.

RESOURCE SUMMARY

PROJECT _____ DATE _____

NAME _____



500-3 (6/70)

INSTRUCTIONS FOR COMPLETING THE COMPONENT SUMMARY

This form is used to keep track of the components of a system. A component is a piece of the system identified by name or common function (e.g., an entry in a tree chart or baseline diagram for the system at any point in time, or a shared section of data such as a COMMON block). With the information on this form combined with the information on the Component Status Report, the structure and status of the system and its development can be monitored.

This form should be filled out for each component at the time that the component is defined, at the time it is completed, and at any point in time when a major modification to the component is made. It should be filled out by the person responsible for the component.

PROJECT. Give project name.

DATE. Give date form filled out.

NAME OF COMPONENT. Give name (up to 8 characters) by which the component will be referred to in other forms.

BRIEF DESCRIPTION. State function of component.

TYPE OF SOFTWARE. Check all classifications that apply. All common blocks are separate components.

STATUS OF COMPONENT. Check whether this is a new component, whether it is a component under development (e.g., a previous component summary has already been submitted), or whether the component is now complete.

A. CODE SPECIFICATIONS. Give the form of design for this component, and tell to what level of detail the specifications are given.

Functional—Components are described as a set of functions, each component performing a certain action.

Procedural—Components are specified in some algorithmic manner (e.g., using a PDL).

English—Components are specified using an English Language prose statement of the problem.

Formal—Some other formal system is used to specify the components.

Relative to the one developing the component, rate the precision of the specifications. Very precise means that no additional analysis on the problem is needed, precise means that only easy or trivial ideas have to be developed, and imprecise means that much work still remains in developing this component and its basic structure.

B. INTERFACES

Give the relative position of this component in the system. Give the number and list the names of all components that call this component, and are called by this component. Also, give the names of any components or other items this component shares with other components (e.g., COMMON blocks, external data). The components directly descended from this component refers to the tree chart or the system. If the interfaces are not yet complete, check "Not Fully Specified".

C. PROGRAMMING LANGUAGES

List languages (or assembly languages) to be used to implement this component. If more than one, list percentages of each (in lines of source code). If there are any constraints on the component (e.g., size, execution time) list them. Also give estimated size of finished component in terms of source statements, (estimate size with comments and without comments) and resulting machine languages (including data areas, but not COMMON blocks).

Useful Items From Similar Projects

1. List previous components and projects which contribute various aspects to this component.
2. For each such component, give the percent of each of the three listed aspects it makes up (e.g., a component may be 50% of design but only 25% of code due to changed interfaces, etc.).
3. For each of the three listed aspects, check what level of modifications are necessary.

D. COMPLEXITY

Rate your belief in the complexity of the implementation. Also approximate the number (by %) of assignment type statements (input statements are included), and control statements (those that alter the flow of control, e.g., IF, CALL, GOTO). The sum of these two may not be 100% (e.g., CONTINUE, DIMENSION and REAL statements will not be counted). I/O and declarations should be listed as other.

E. RESOURCES TO IMPLEMENT

For each of the three listed phases (Design, Code, Test), estimate computer runs, time needed, hours to implement, and estimated completion date. If not known, or no estimate can be given, write "unknown".

F. ORIGIN OF COMPONENT

If this component is independent of any other component of the system (e.g., is a low level component which is designed first, or is the root node of the tree chart) then check yes, otherwise check no.

If no is checked, then explain why the component was added. (Usually only one reason will be checked, although more may be checked, if appropriate).

A lower level elaboration of a higher level component means that an existing component was expanded to include new components (e.g., expanding tree chart). List the higher level component time.

Added as a driver or interface means that a calling program was added to call existing components. List these called components.

A redesign of an existing component means that new capabilities were added to an already existing component. Write its name.

A renaming of an older component. Give the old name.

A regrouping of existing material means that several components were redesigned with a new component resulting from this redesign. Give the old component names.

Type of addition. Why was this component added to the system at this time? Check the appropriate reason. (Normally, only one should be checked, although more can be if appropriate.)

G. ADDITIONAL COMMENTS. Add any other comments that will help explain the purpose, design, and complexity of this component.

H. PERSON RESPONSIBLE. Include name of person responsible for implementing component.

I. PERSON FILLING OUT FORM. Give name of person filling out form. This normally is the same name as in H.

COMPONENT SUMMARY

PROJECT _____ DATE _____

NAME OF COMPONENT _____ CREATION DATE _____

BRIEF DESCRIPTION _____

STATUS OF COMPONENT NEW _____ UNDER DEVEL _____ COMPLETED _____

TYPE OF SOFTWARE (Check All That Apply)

- ____ I/O Processing
- ____ Algorithmic
- ____ Logic Control
- ____ Systems Related
- ____ DATA/COMMON Block
- ____ Other

A. CODE SPECIFICATIONS (Check All That Apply)

FORM OF DESIGN	LEVEL OF DETAIL				
	Component	Subcomponent	Basic Block Segment	Stmt	Other
Functional					
Procedural					
English					
Formal					
Other ( )					

Precision of Code Specification Very Precise _____ Precise _____ Imprecise _____

B. INTERFACES

Number Components Called _____ Names _____
Not Fully Specified _____

Number Calling This Component _____ Names _____
Not Fully Specified _____

Number Shared Items _____ Names _____
Not Fully Specified _____

Number of Components Directly Descended from This Component _____ Names _____
Not Fully Specified _____

C. PROGRAMMING LANGUAGES

Languages Used and Percentages _____ (____) _____ (____)

CONSTRAINT PROBLEM EXPECTED:

	Constraint Present	Component Meets Constraint
Memory Space		
Execution Time		
Other ( )		

Size: Source Statements (Including Comments) _____ Machine Bytes _____
Source Statements (Not Including Comments) _____

Useful Items From Similar Projects

Component	Project	Specification				Design				Code			
		%	Major	Minor	None	%	Major	Minor	None	%	Major	Minor	None

S80-5 (6/78)

D. COMPLEXITY

Complexity of Function Easy _____ Moderate _____ Hard _____

_____ % Assignment Statements _____ % Control Statements _____ % Other Statements (e.g., Data Decl. I/O)

E. RESOURCES TO IMPLEMENT

	Runs	Computer Time (min)	Effort (hrs)	Est. Completion Date
Design				
Code				
Test				

F. Is this component independent of the existing components? _____ Yes _____ No

If No, describe relation of this component to the existing system:

_____ inserted as a lower level elaboration of higher level components (names) _____

_____ added as a driver or interface for existing components (names) _____

_____ redesign (to add new capability) of existing components (names) _____

_____ renaming of existing component (name) _____

_____ regrouping of existing material from several components (names) _____

_____ other _____

Type of Addition:

_____ error correction

_____ planned enhancement

_____ implementation of requirements change

_____ improvement of clarity, maintainability, or documentation

_____ other (explain below)

_____ improvement of user service

_____ utility for development purposes only

_____ optimization of time/space/accuracy

_____ adaptation to environment change

G. ADDITIONAL COMMENTS

H. PERSON RESPONSIBLE FOR IMPLEMENTING COMPONENT _____

I. PERSON FILLING OUT FORM _____

SEL-8 (6/78)

INSTRUCTIONS FOR COMPLETING THE COMPONENT STATUS REPORT

This form is to be used to accurately keep track of the development of each component in the system. A Component Summary Report should exist for each component mentioned. The form is to be turned in at the end of each week. Please fill out either daily or once each week. If daily, then a given component may be listed several times during the course of a week. For each component list the number of hours spent on each of the listed activities. This form should be filled out by persons working on the project.

PROJECT. Name of the project.

PROGRAMMER. Name of programmer.

DATE. Date report turned in. Usually the date of a Friday.

COMPONENT. Name of component. Either a part of the system structure for which there is a component summary form, or one of the following:

JCL. Developing command language instructions.

Overlay. Developing system overlay structure.

User Guide. User's Guide Documentation.

System Description. System Description Documentation.

DESIGN

Create. Writing of a component design.

Read. Reading (by peer) of design to look for errors. (e.g., peer review)

Formal Review. Formal meeting of several individuals for purpose of explaining design. Also include time spent in preparing for review. All those attending review should list components discussed in their own Component Status Report for that week.

CODE/DEVELOPMENT

Code. Writing executable instructions and desk checking program.

Read. Code reading by peer. Similar to Design Read above.

Formal Review. Review of coded components. Similar to Design Review above.

TESTING

Unit. Unit testing. Test run with test data on single module.

Integ. Integration testing of several components.

Review. Review of testing status.

OTHER. Any other aspect related to a component of the project not already covered other than Design, Code Development, Test (e.g., Documentation of a specific component). List type of activity, and hours spent on that activity. A set of activities has been listed for which time may be charged to the overall project:

Travel. Time spent on official travel related to this project, (including trips to and from GSFC).

Forms. Time spent on filling out reporting forms.

Meetings. Time spent in meetings which are not design or code review meetings.

Training. Training activities identified for project.

Acc Test. Acceptance Testing activities.

COMPONENT STATUS REPORT

PROJECT _____ DATE _____

PROGRAMMER _____

COMPONENT	DESIGN			CODE DEVELOPMENT			TEST			OTHER	
	CREATE	READ	FORMAL REVIEW	CODE	READ	FORMAL REVIEW	UNIT	INTEG	REVIEW	ACTIVITY	HRS
										Travel	
										Forms	
										Meetings	
										Acc Test	
										Training	
JCL											
Overlay											
User Guide											
System Doc											

580-4 (6/78)

INSTRUCTIONS FOR COMPLETING THE COMPUTER PROGRAM RUN ANALYSIS FORM

This form will be used to monitor the activities for which the computer is used in the course of a project life cycle. An entry should be made for each computer run-including all activities performed when the computer is used in an interactive mode.

PROGRAMMER. Write down name of person preparing computer runs. This may not necessarily be the person running the program (i.e., librarian).

PROJECT. Write down project name. Use a different form for each project.

COMPUTER. Indicate the machine on which these runs were made (e.g., S/360, PDP-11, Σ8).

DATE. Date form turned in.

JOB ID. Identification of job.

RUN DATE. Date run submitted in format MM-DD (month-day).

INTERACTIVE. Place an X if the run was submitted from an interactive terminal.

RUN PURPOSE. Place an X in all boxes that describe this run.

Unit Test. A purpose of the run is to test one or more components without the rest of the system being configured into the load module. A run which uses a 'test driver' would fall into this category.

System Test. This run executes a load module which contains all of the currently available system in order to test one or more components in a full system configuration.

Benchmark Test. This is a recertification type run. A run that has successfully executed in the past is now rerun to verify that certain capabilities still exist.

Maintenance/Utility. A purpose of this run is to perform a 'library-type' function. Examples are runs that update source, create backups, delete/compress/copy data sets.

Compile/Assembly/Link. A purpose of the run is to check for errors in the compile, assembly and/or link steps. A run which includes one or more of these steps simply as a prerequisite to a system execution would not fall into this category.

Debug Run. This run was submitted in order to investigate a known error.

Other. This run has a purpose which does not fall into one of the other categories. Examples are runs which access other systems in order to aid in the design, development and/or testing of the project under study.

COMPONENTS OF INTEREST. List all components important to this run (e.g., components being tested, compiled, copied, etc.)

FIRST RUN. Place an X here if this is the first time any of the listed components have been processed by the computer for the purpose of run specified.

MEETS OBJECTIVES. This is a subjective evaluation of whether the run satisfied your objectives. Runs that terminate in errors may be satisfactory if the objective was to locate errors or to test for correctness; runs that terminate normally may be unsatisfactory if the purpose was to locate an error known to be present. Thus this question is independent of whether the program contained any errors or not.

RUN RESULTS. Check the box that best describes the results of this run. Normally only one box is checked, although more than one may be checked if appropriate.

Good Run. Program ran to termination with no known errors.

Setup Error. Error in creating program deck.

Submit Error. Deck submitted incorrectly, resources unavailable, keypunch error, or general submission error.

JCL Error. JCL statement incorrect. (JCL cards mistyped should be listed under submit errors.)

Other Setup Error. Such as insufficient space or time specified for job step. This should not be caused by program error.

Machine Error. Errors outside of the control of the programmer.

Hardware Error. Machine malfunction.

Software Error. System crash or system program error (e.g., error in FORTRAN compiler).

Program Error. Error caused by the submitted program.

Compile Error. The source program contains an error which is found by the compiler or assembler.

Link Error. The loader or linkage editor finds an error.

Execute Error. System error messages are generated during the execution step, possibly causing an abend.

User Generated Error. The program terminates in a programmer generated error message which is not a system error.

Ran to Completion. The program terminated with no error message; however, the results are incorrect signifying that there is something wrong with the program.

COMMENTS. If you believe that your answers to these questions do not adequately characterize this run, you may add any additional comments that you wish. Also use this space to indicate if the run was lost before you had a chance to evaluate results.

COMPUTER PROGRAM RUN ANALYSIS

PROGRAMMER _____ PROJECT _____ COMPUTER _____ DATE ___/___/___

JOB ID	DATE MM DD	INTERACTIVE	RUN PURPOSE							COMPONENTS OF INTEREST	First Run	Run Met Objectives	Run Did Not Meet Objectives	RUN RESULTS										COMMENT (e.g., Run Lost, No Results)
			Unit Test	System Test	Benchmark Test	Maintenance/Utility	Compile/Assembly/Link	Debug Run	Other					Good Run	Setup Error			Machine Error		Program Error				
														Good Run	Submit Error	JCL Error	Other Setup Error	Hardware Error	Software Error	Compile Error	Link Error	Execute Error	User Generated Msg.	Ran to Completion	

500-7 (6/79)

INSTRUCTIONS FOR COMPLETING THE CHANGE REPORT FORM

This form is used to keep track of all changes made to a system. A change is any alteration to the design, documentation, or code generated for a project. Each change can be thought of as a step in the process of transforming the original software design into a complete working system. The initial creation of sections of fresh code or design is not a change.

One change report form should be filled out for each change. Where several changes are made simultaneously for different reasons a separate form should be completed for each reason.

NUMBER. A unique identifier per form per day consisting of initials followed by a sequence number. The initials should be those of the person filling out the form. The sequence number should be a positive integer indicating the number of forms filled out so far during the day. Number DMW01 indicates the first form of the day filled out by DMW, DMW02 is the second form that day, etc.

PROJECT NAME. The name of the development project.

CURRENT DATE. The date on which an entry is first made on the form, even if the form is not completed on that day.

SECTION A—IDENTIFICATION

REASON. Explain why the change is being made.

DESCRIPTION. Describe the change that is being made. This should not be on the variable name or bit level, but should be sufficiently abstract so that the function of the changed code can be determined, e.g., "the input buffer was cleared," rather than "array buff was set to zero."

EFFECT. What components (or documents) are changed? List the names of all components and documents modified as part of the change, including version numbers.

EFFORT: What additional components (or documents) were examined in determining what change was needed? List all components and documents that were examined, but were not actually changed, in deciding what change to make, how to make it, and where to make it. This list should not overlap with the list of components and documents actually changed.

DATES OF CHANGE. Need for change determined on. Give the date on which it was first realized that a change was needed.

Change started on. Give the date on which the change was started.

What was the effort in person-time required to understand and implement the change?

Give the best available estimate of the total time needed to understand what change had to be made and how to make it, including the implementation time. This should include the time of all persons involved in making the change. As an example, if two people each worked 6 hours on the change, the space marked "one day to 3 days" should be checked.

SECTION B—TYPE OF CHANGE

Check the one box that best describes the change. If none of the change descriptions seem to fit, check other and give a detailed description of the change in Section E. If several of the descriptions seem equally appropriate, more than one box may be checked.

Error Correction. A change made to correct an error in previous work. If this box is checked Sections C and D of the change report form should be completed.

Planned Enhancement. The insertion of a body of code into a program stub that was initially created as a dummy for testing purposes, or adding capability to an already existing component as part of a planned incremental development.

Implementation of Requirements Change. Altering the system to conform to a change in requirements imposed by the customer.

Improvement of Clarity, Maintainability, or Documentation. Changes made to improve code quality, such as improving indentation of code, resequencing labels for readability, adding or updating documentation or correcting literary errors in it, suppressing redundant information or replacing multiply-occurring sections of code with procedure calls. Corrections of violations of programming standards, and design improvements that should have been visible in the functional specifications of components of the system are to be treated as error corrections. Documentation updates made concomitantly with a change should be treated as a part of that change and classified with the primary cause of the change.

Improvement of User Services. During system development, individual programmers may find that with very little extra work they can provide the user with additional facilities on top of the functional requirements of the system. Such changes are classed as improvements to user services.

Insertion/Deletion of Debug Code. Changes made to the program text specifically to provide additional information during test runs so that errors can be isolated.

Optimize Time/Space/Accuracy. An optimization is a localized adjustment of the program whose main purpose is to reduce its execution time or memory requirements, or to obtain results of greater numerical accuracy by tuning the algorithms used to the specific problem being solved.

Adaptation to Environment Change. The "boundary" of a software system is defined to include just those programs whose development and maintenance is being monitored as part of the software engineering laboratory project. A change whose cause lies outside this boundary (e.g., in response to an operating system, compiler, or hardware change) is regarded as environmentally caused.

Was more than one component affected by the change? A component is defined to be directly involved in a change if it contains subroutines that are changed and it contains no subcomponents containing those subroutines. Check yes if the change directly involves more than one component of the system, no otherwise. It may be the case that a change to one subroutine/component will require some future adjustment in other components (these components may not even have been coded yet, or their adaptation may be postponed). In such cases, the effects of the change involve more than one component even though only one module was noted as changed on this form.

SECTION C—TYPE OF ERROR

Check the one box that best describes the error. If none of the error descriptions seem to fit, check other and give a detailed description of the error in Section E.

Requirements Incorrect or Misinterpreted. Requirements may be incorrect (inconsistent or ambiguous), or their meaning may be misinterpreted. In either case, an error of this type, if undetected early, may propagate through design and into code. Even if undetected until acceptance testing (or maintenance), errors resulting from incorrect or misinterpreted requirements should be classified in the requirements error category.

Functional Specifications Incorrect or Misinterpreted. Functional specifications are taken to be a specification of a component as a set of functions defining the output for any input. Similar to requirements, specifications may be either incorrect or misinterpreted. Errors in the specifications that occur as a result of misunderstandings of requirements are classified as misinterpreted requirements errors and not incorrect specifications. Specification errors that result from misunderstandings among those writing the specifications are classified as incorrect specifications. Errors in code or design or documents resulting from incorrect or misinterpreted specifications should be classified in the specifications error category.

Design Error Involving Several Components. A design decision is a choice of organization of a component into subcomponents, including the specification of the interfaces among the subcomponents. A design error is a design decision that results in one of the following:

- interfaces that contain insufficient, unnecessary, or redundant information;
- a set of subcomponents that do not satisfy the specifications of the component (i.e., one or more of the subcomponents do not have the capabilities needed to satisfy the use intended for the component).

Note that a design error may result from incorrect or misinterpreted requirements or specifications. In such cases, the error should not be classified as a design error, but as a requirements or specification error.

Error in the Design or Implementation of a Single Component. Most simple, localized programming mistakes fall into this category. It contains those cases where the organization of the system into components and their interfaces is correct, but a particular component does not behave according to its intended use (i.e., does not correspond to its specification). This may occur because the algorithm used in designing the component is incorrect, or because the implementation of the algorithm is incorrect. If the algorithm has a written specification prior to code generation, and the specification is incorrect or misinterpreted, the error is not classified as a design or implementation error, but as a specification error. If the erroneous algorithm has no written specification, or if the implementation of the algorithm has errors not attributable to any other category, then the error is classified as an error in the design or implementation of a single component.

Misunderstanding of External Environment, Except Language. Check this box if the error resulted from mistaken assumptions about the hardware or software environment in which the program operates (i.e., that software outside the "boundary" of the project—see "adaptation to environment change" in Section B). Included here are mistaken assumptions about how the operating system works, about how the hardware is controlled, about response of peripherals to various commands, about the operation of the library system, about the interface to special display hardware or software, etc.

Error in Use of Programming Language/Compiler. Errors in the use of the language/compiler are those errors that result from some misunderstanding of how the compiler works, how the language provided run-time support system operates, or some misunderstanding of particular language features. Not included in this category are clerical errors (e.g., typos) that lead to compilation errors.

Clerical Error Clerical errors are those errors that occur in the mechanical translation of an item from one format to another (e.g., one coding sheet to another) or from one medium to another (e.g., coding sheets to cards). No interpretation or semantic translation is involved in such a process.

FOR DESIGN OR IMPLEMENTATION ERRORS ONLY

This section should be filled out only if the error was a design error involving several components, or if it was an error in the design or implementation of a single component. Errors that occur in the design of a system, subsystem, set of components, or single component, or in the implementation of a single component, may be categorized in one of two ways. Either there was an error in the use of data, or there was an error in the function of a component (such as an algorithmic or computational error resulting in program behavior not corresponding to the intended use of the program). Data use errors can be characterized as either incorrect values for data items or improper assumptions about the structure of data items (e.g., array size or dimensions, or ordering of items on a list). Errors involving the function of a component include control and computational errors, such as incorrect sequencing of statements, omitted statements (where such are not clerical errors), improperly computed expressions, omitted capabilities of the component(s), etc.

SECTION D—VALIDATION AND REPAIR

What were the activities used to validate the program, to detect the error, and find its cause?

The purpose of this section is to discover how it became known that an error existing and how the cause of the error was determined. A check should be put in the first column for each method used for validating the component(s) where the error was found. A check should be put in the second column on the same line as the method by which the symptoms of this particular error was first noted. The third and fourth columns refer to activities used to find the cause of the error once it was known that the error existed. In the third column, check all techniques used in trying to find the cause of the error. In the fourth column, check those techniques that yielded the information needed to find the cause. In some cases, such as some errors found by code reading, the technique(s) used to find the error and discover its cause will be the same. Note that error messages have been divided into two categories: those produced by the support system (e.g., compiler, operating system), and those designed into the code for the specific purposes of the project. Testing has also been divided into two categories: test runs made prior to acceptance testing (pre-acceptance test runs), and acceptance tests. If activities other than those listed in the table were used in finding the error or discovering its cause, check other in the appropriate column, and describe the activities used in Section E. This table inevitably has some redundancy: a check in column 2 must always have a corresponding check in column 1, similarly with columns 4 and 3.

What was the time used to isolate the cause?

Check the space that most closely approximates the time required to isolate the cause of the error. This should be the total of the time that was spent in the activities tried to find the cause. If the cause of the error was never found, and a workaround was used, check the appropriate box. If the cause was never found and a workaround was not used, explain the circumstances in Section E.

Was this error related to a previous change?

Changes to software may result in errors because of one or more of several reasons:

- the change was incorrectly implemented, i.e., did not conform to its specification,
- the change invalidated an assumption made elsewhere in the software,
- an assumption made about the rest of the software in the design of the change was incorrect.

An error is related to a previous change if it results from one of the above three conditions. Errors that are uncovered by changes, i.e., an error masked by another that is revealed when the latter is corrected, do not belong in this category. If the error is related to a previous change, give the number and date of the change report form of the related change. When did the error enter the system?

Check the box that most closely represents the phase in the erroneous components' development in which the error was introduced.

SECTION E—ADDITIONAL INFORMATION

This section is intended to permit further explanation of any items you feel may be significant in categorizing the change (including error correction). If the "other" category was checked in any of the previous sections of the form, a fuller explanation should be given here. Do not hesitate to give a full description of the error or change or any doubts you may have in classifying it. The accuracy of our analysis is dependent on the amount and accuracy of the data you provide for us. The study we are performing is an attempt to do a careful, detailed investigation of the processes that go on during software development, the kinds of changes and errors that occur during development, and the reasons for their occurrence. With your help, we hope to gain enough insight into the design, coding, and testing of programs so that proposed techniques for coping with software changes and reducing the number of errors can be evaluated. Your cooperation and patience in completing the change report form each time you make a change to a document or program are needed and appreciated.

NUMBER

CHANGE REPORT FORM

PROJECT NAME CURRENT DATE

SECTION A IDENTIFICATION

REASON Why was the change made?

DESCRIPTION What change was made?

EFFECT What components or documents are changed? (Include version)

EFFORT What additional components or documents were examined in determining what change was needed?

	(Month	Day	Year)
Need for change determined on			
Change started on			

What was the effort in person time required to understand and implement the change?

____ 1 hour or less ____ 1 hour to 1 day ____ 1 day to 3 days ____ more than 3 days

SECTION B - TYPE OF CHANGE (How is this change best characterized?)

- ☐ Error correction
- ☐ Planned enhancement
- ☐ Implementation of requirements change
- ☐ Improvement of clarity, maintainability, or documentation
- ☐ Improvement of user services
- ☐ Insertion/deletion of debug code
- ☐ Optimization of time/space, accuracy
- ☐ Adaptation to environment change
- ☐ Other (Explain in E)

Was more than one component affected by the change? Yes ____ No ____

FOR ERROR CORRECTIONS ONLY

SECTION C - TYPE OF ERROR (How is this error best characterized?)

- ☐ Requirements incorrect or misinterpreted
- ☐ Functional specifications incorrect or misinterpreted
- ☐ Design error involving several components
- ☐ Error in the design or implementation of a single component
- ☐ Misunderstanding of external environment, except language
- ☐ Error in use of programming language/compiler
- ☐ Clerical error
- ☐ Other (Explain in E)

FOR DESIGN OR IMPLEMENTATION ERRORS ONLY

If the error was in design or implementation

The error was a mistaken assumption about the value or structure of data ____

The error was a mistake in control logic or computation of an expression ____

SDS 8 (6/78)

FOR ERROR CORRECTIONS ONLY

SECTION D - VALIDATION AND REPAIR

What activities were used to validate the program, detect the error, and find its cause?

	Activities Used for Program Validation	Activities Successful in Detecting Error Symptoms	Activities Tried to Find Cause	Activities Successful in Finding Cause
Pre-acceptance test runs				
Acceptance testing				
Post-acceptance use				
Inspection of output				
Code reading by programmer				
Code reading by other person				
Talks with other programmers				
Special debug code				
System error messages				
Project specific error messages				
Reading documentation				
Trace				
Dump				
Cross-reference/attribute list				
Proof technique				
Other (Explain in E)				

What was the time used to isolate the cause?

____one hour or less, ____one hour to one day, ____more than one day, ____never found

If never found, was a workaround used?____Yes____No (Explain in E)

Was this error related to a previous change?

____Yes (Change Report #/Date____) ____No ____Can't tell

When did the error enter the system?

____requirements ____functional specs ____design ____coding and test ____other ____can't tell

SECTION E - ADDITIONAL INFORMATION

Please give any information that may be helpful in categorizing the error or change, and understanding its cause and its ramifications.

Name:____ Authorized:____ Date:____

SEL-2 (6/78)

Current Date _____

_____Attitude System Maintenance Report

Project Name _____Need for Change determined on (Mo., Day, Yr.)_____

Describe Change _____

_____

_____

_____

What components/subroutines/modules are changed _____

_____

_____

CHANGE (NON-ERROR) (fill out this section if change is NOT an error correction)
This change is being made because of a change in: (Check all that apply)

_____ requirements
_____ new information/data
_____ specification
_____ design
_____ other (specify): _____

_____hardware environment
_____software environment
_____optimization

ERROR ONLY (fill out this section if change IS an error correction)
The following activities were used in error detection or isolation: (Check all that apply) (Put D for detection, I for isolation)

_____ normal use
_____ test runs
_____ code reading
_____ reading documentation
_____ other (Specify): _____

_____trace/dump
_____cross reference/attitude list
_____system error messages
_____project specific error messages

Which of the following best describes the error:

_____ requirements error
_____ design error
_____ error in translating design or specification to code
_____ other: Describe_____

_____specification error
_____clerical error

Was this error related to a previous maintenance change _____yes _____no _____can't tell

Please give any information that may be helpful in categorizing and understanding the change on the reverse side of this form.

Person filling out this form _____
Approved _____Date _____

Change started on date (month, day, year) _____

Time spent on this change:
_____ less than 1 day _____1 day to a week _____more than a week

F.2 SEL GLOSSARY OF TERMS USED WITH DATA COLLECTION FORMS

This section defines the terms used in the software engineering data collection forms reproduced in Section F.1. A more extensive glossary (based substantially on this one) is found in Reference 8.

assignment statements	All statements that change the value of a variable as their main purpose (e.g., assignment or READ statements, but the assignment of the DO loop variable in a DO statement should not be included).
attitude/orbit	Any component that is directly related to either the attitude determination (or control) task or to the orbit determination (or control) task falls into this category. This should include full systems in general (such as GTDS or ISEE-B Attitude) as well as specific modules such as Deterministic Attitude or DCCONES.
attribute list	A compiler-generated list of the identifiers used by a program that describes the characteristics of those identifiers and shows the source statements where they are first defined (or first used) and, for variables, their (relative) storage locations.
automated tools	Any programs whose purpose is to aid in software development (e.g., compiler, text editor, or dump or trace facility). This includes compilers but not standard operating system software (e.g., linkage editor).
baseline diagram	A structured chart listing all components in a system in which a connection from a higher component to a lower one indicates that the higher component calls the lower one.
batch	Use of a computer in which the entire job is read into the machine before the processing begins and in which there is no provision for interaction with the submitter during execution of the job. (Interactive usage is always via a terminal; batch usage may be via a terminal or a card deck.)

bottom-up	The design (or implementation) of the system starting with the lowest level routines and proceeding to the higher level routines that use the lower levels.
business/ financial	The second of the four major categories applies to components related to some accounting task, financial data formatting, business data retrieval or reporting, or possibly personnel data management. Very few of the components being studied will fall into this class.
change	A modification to design, code, or documentation. A change might be made to correct an error, to improve system performance, to add capability, to improve appearance, or to implement a requirements change, for example.
clerical	The process of copying an item from one format to another or from one medium to another, which involves no interpretation or semantic translation.
code reading	Visual inspection of the source code by persons other than the creator of the code.
command/ control	This class of components includes those used either to generate vehicle commands or to transmit these commands from the control center.
complexity	Measures the difficulty of implementing a component, independent of the implementer's experience. Easy (or simple) means that any good programmer can write down the correct code with little thought. Hard (or complex) means that much thought is involved in the design. (Compare this with "precise"; e.g., easy and imprecise may mean a vague specification, but once the approach is decided upon, the code is easy to write.)
component	A piece of the system identified by name or common function (e.g., separately compilable function, an entry in a tree chart or baseline diagram for the system at any point in time, or a shared section of data such as a COMMON block).

computer time	For batch usage, this is the billable time for all runs. For interactive usage, it is the number of hours spent at a terminal.
confidence level	Percentage probability that a given number is correct: 100 percent means that the number is absolute certainty; 0 percent means that the number must be incorrect.
constraints	Restrictions on resource availability (execution time, memory allocation) imposed by specifications.
constraints, space	All restrictions caused by space problems. On the Component Summary Report form, list each restriction separately (e.g., maximum number of words that component may occupy at one time or maximum disk space available during execution time or for program storage).
constraints, time	All restrictions caused by various machine and calendar time problems. On the Component Summary Report form, list each restriction separately (e.g., maximum execution time for component to process and respond to some input condition or time to complete a component or milestone).
control statements	All statements that potentially alter the sequence of executed instructions (e.g., GOTO, IF, RETURN, or DO).
correction	A change made to correct an error.
cosmetic	Changes in the source program that have little effect on the performance of program (e.g., correct comments, move code around as long as it does not alter the algorithm implemented, or change the name of a local variable).
create	The creation and recording of the idea.
creation date	Date that the component was first named (e.g., date it first appeared on a tree chart).
cross-reference	List of the identifiers used by a program showing (by means of indices or statement numbers) which statements of the program define and reference those identifiers.

data base applications	This category is to include components that retrieve, write to, or format information for a well-defined formatted bank of information available to the system. The user must decide whether or not the data set is to be considered a data base. An example of an acceptable data base would be the ADL file, SLP file, or Geodetics file, whereas a sequential telemetry file or tape would not be.
design	A description of what the system must do, its components, the interfaces among those components, and the system's interface(s) to the external environment.
design phase	The creation and recording of the design, including discussion about strategy with peers. This phase does not include the development of any code at the programming language level. It does include the creation of specifications for subcomponents of the current component.
design reading	Visual inspection of the design by persons other than the creator of the design.
development phase	The development and recording of code and inline comments based on the design. This phase includes the modification of code caused by design changes or errors found in testing. It does not include any time spent in entering the code into the computer.
documentation	Written material, other than source code statements, that describes a system or any of its components.
dump	Record of the state of the memory space used by a program at some point in its execution. A dump may include all or part of the program's memory space (including registers).
end date	Date that a project is scheduled to be completed.
English (or informal) specifications	Specifications given as readable English text, as opposed to some formal notation.

error	Discrepancy between a specification and its implementation. The specification might be requirements, design specifications, or coding specifications.
external environment	Combination of hardware and software used to maintain and execute the software, including the computer on which the software executes, the operating system for that computer, support libraries, text editors, and compilers.
formal specifications	Some specification technique based upon a strict set of rules for describing the specification and usually involving the use of an unambiguously defined notation (e.g., mathematical functions or formal PDL).
function	Mathematical notation used to specify the set of input, the set of output, and the relationship between input and output.
functional specifications	Specification of a component as a set of functions defining the output for any input. The specification emphasizes what the program is to do rather than how to do it. However, an algorithmic specification can be considered functional if it is not used to dictate the actual algorithm to be used. (See procedural specifications.)
hardest first	Design (or implementation) of the most difficult aspects of the system first.
HIPO (Hierarchical Input Process Output)	Graphical technique that defines each component by its transformation on its input data sets to its output data sets.
implementation	Implementation of a program is either a machine-executable form of the program or a form of the program that can be automatically translated (e.g., by compiler or assembler) into machine-executable form.
integration test	Test of several modules to check that the interfaces are defined correctly.
integration test, full	Test of the entire system (i.e., top-level component).

integration test, partial	Test of any set of modules but not the entire system.
intended use of	Result of invoking a program or segment of a program, including the actions performed by that program when invoked. Invocation may be by subroutine or function call or by a branch to a segment of code.
interface	Set of data passed between two or more programs or segments of programs and the assumptions made by each program about how the others operate.
interactive	Use of a computer via a terminal in which each line of input is immediately processed by the computer.
iterative enhancement	Design (or implementation) of successive versions, each producing a usable subset of the final product until the entire system is fully developed.
level	Unit corresponding to some partitioning of the final product (e.g., a single line of code, 10 lines of code, 25 lines of code, subroutine, or module). If the system is hierarchically structured, each component is at a higher level than its subcomponents, and the system may be described as the highest level component (the component at level 1), the component at level 2, or the lowest level component.
level, lowest	Smallest unit identified by the activity (e.g., code reading to the single statement, top-down design to the module level, or top-down design to level 3).
librarian	A clerk whose responsibilities include processing source statements but not writing them, (e.g., maintaining libraries, updating code, or producing tape backups).
machine words	Number of words in a main memory that a component occupies at one time.
manpower	Sum, over the number of people, of the number of hours per person charged to the contract.

mathematical/ numerical	This category is meant to be a more specific category than the scientific class. It contains those components that reflect a specific algebraic expression or mathematical algorithm. Such components as a dot product routine or a numerical integrator are in this category.
maximum space	Total number of machine words that the system may occupy at one time.
mission date	Date that system must be operational.
module test	Test of a single module.
none used	No explicit technique was specified to be used.
onboard processing	All components that are built for the purpose of satisfying some onboard processing need belong to this class. Although the component may be built and tested on a computer that is not the real flight computer, it should be classified as onboard if the final destination is the OBC (onboard computer).
optimization	Changes in the source code to improve program performance (e.g., run faster or use less space). Optimization changes are not error corrections; however, if a change is made to use less space to conform to the specified space constraint, then the term "error" applies.
PDL	Program design language (often called pseudocode). Used in the design and coding phases of a project, PDL is a language that contains a fixed set of control statements and a formal or informal way of defining and operating on data structures. PDL code may or may not be machine-readable, and for this study it is not considered as documentation, but as an integral part of the finished source program.

procedural specifications	Specification of a component in some algorithmic manner (e.g., using PDL or a flowchart). The specification says how the program is to work. (See functional specifications.)
proof technique	Method for formally demonstrating that a piece of software performs according to its specifications. Proof techniques usually use some form of mathematical notation to describe the result of executing a program.
range in module size	Number of source statements in a module, including comments.
read	The reading by peers of the recordings of the current phase to look for errors, invent tests, and so on.
real-time	This class includes components that are a direct function of events occurring at, or near, the current time. Typical components would be the Attitude Control Monitors. Since parts of most of the telemetry processors are required to process data as it is received, they too may be considered real-time components.
requirements	System specification written by the user to define a system to a developer. The developer uses these specifications in designing, implementing, and testing the system.
review	Formal meeting of several individuals for the purpose of explaining design (management review). Also includes the time spent in preparing for the review. All those attending a review should list the components discussed in their own Component Summary Report for that week.
scientific	A component may be in this category if it is related to some mathematical algorithm, engineering problem, law of physics, or celestial mechanics problem. Most of the full systems developed will fall into this category, whereas the various pieces of modules may fall into some of the other classes.

segment	Contiguous piece of code that is unnamed and, hence, cannot be referred to as a single entity in a program statement. A segment could be one or several lines of a subroutine, part of a data area, or an arbitrary contiguous section of memory.
shared items	Data and programs, accessible by several components, such as COMMON blocks, external files, and library subroutines.
simulating constructs	Statements that are used to simulate structured control structures when the language to be used does not contain structured control structures.
source instructions	See source statements.
source statements	All statements readable by and read by the compiler. This includes executable statements (e.g., assignment, IF, and GO TO); nonexecutable statements (e.g., DIMENSION, REAL, and END); and comments.
specification	Description of the input, output, and essential function(s) to be performed by a component of the system. The specification is produced by the organization that is to develop the system; that is, at the top level, it can be thought of as the contractor's interpretation of the requirements.
specification, imprecise	The input, output, and function of the component are loosely defined. Much of what is required is assumed rather than specified. The specification relies heavily on programmer experience and verbal communication to get an unambiguous interpretation and a full understanding of what is needed.
specification, precise	The input, output, and function of the component are well defined. There are underlying assumptions not specified, but it is assumed that any programmer working on the project, with experience on a similar project, will understand these assumptions. It is possible to arrive at an ambiguous interpretation or misunderstanding

specification, precise (Cont'd)	of the specifications if the reader does not have enough experience with the problem or does not obtain further verbal communication.
specification, very precise	Completely defined description of the input, output, and function of a component. The implementer of a very precise specification need make few, if any, assumptions. It is almost impossible to arrive at an ambiguous interpretation or misunderstanding of the specifications.
specification-driven	Using the specifications of the program to determine test data (e.g., test data is generated by examining the input/output requirements and specifications).
standards	Any specifications that refer to the method of development of the source program itself, and not to the problem to be implemented (e.g., using structured code, at most 100-line subroutines, or all names prefixed with subsystem name).
start date	Date on which initial work on a project began.
string processing	This includes components that perform operations on lists of characters. Normally, this class is assumed to include functions of compilers, hash code string hook-up, and array comparisons.
structure-driven	Using the structure of the program to determine test data (e.g., generating data to ensure that each branch of a program is executed at least once).
structure of data	Organization of a composite data item consisting of several variables or other array items. Examples of such composite data items are arrays (both singly- and multiply-dimensioned), strings, complex variables and constants, records on a disk file (each record containing several words), and multiple-word entries in a table.
structured code	The language supports structured control structures (e.g., a FORTRAN preprocessor).

systems	By system-related software, one includes any package designed to affect, modify, extend, or change the normal available processing procedure of the operating system. This could include such components as error tracing or extended I/O such as DAIO.
system size	Total number of machine words needed for all instructions generated on the project plus space for data, library routines, and other code. This is the total size of the system without using any overlay structure.
table handler	Includes components that are specifically designed to generate or interpret information in a table format such as the Generalized Telemetry Processor.
telemetry/ tracking	Includes all components that are specifically required to interface (either read, write, or format) with telemetry or tracking data.
testing phase	Design of tests, testing strategies, and the running of such tests. This phase does not include the writing of any code (even for debugging purposes), which should be recorded under coding.
top-down	Design (or implementation) of the system, starting with a single component, one level at a time, by expanding each component reference as an algorithm possibly calling other new components.
trace	Record of program execution showing the sequence of subroutine and function calls and, sometimes, the value of selected variables. Code used in producing a trace is automatically inserted into a program, usually by the compiler, sometimes by other support software.
type of software	The four major classifications of most of the applicable software being developed are: scientific, business/financial, systems, and utility. These classifications may be refined into the categories of: string processing, data base applications, real-time, and table

type of software (Cont'd)	handler. A further refinement includes the categories of: attitude/orbit, telemetry/tracking, command/control, mathematical, and numerical onboard.
utility	Any component that is generated to satisfy some general support function required by other applications software may be considered a utility. This class of components usually contains software that does not fit into any of the other three categories. Although components can fall into two of the primary categories (e.g., scientific and utility), it will be easier to use only the more descriptive of the categories (e.g., vector cross-product--scientific; data unpacking--utility).
value of data	The number and kind of number (e.g., integer, floating-point, or ASCII-encoded character) stored in a local variable or data area, parameter, common variable, or system-wide data item.
walkthrough	Formal meeting sessions for the review of source code and design by the various members of the project for technical rather than management purposes. The purpose is for error detection and not correction.
workaround	The method used to counteract the effects of an error in a program when the cause of the error and, consequently, the location of the statements containing the error is not known or is inaccessible (e.g., a compiler error).

REFERENCES

1. Software Engineering Laboratory, SEL-82-003, Software Engineering Laboratory (SEL) Data Base Reporting Software User's Guide and System Description, P. Lo and S. Eslinger, May 1983

2. --, SEL-78-102, FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 1), W. J. Decker and W. A. Taylor, September 1982

3. --, SEL-81-102, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide (Revision 1), P. Lo and D. Wyckoff, March 1983

4. Digital Equipment Corporation, AA-0002A-TC, IAS/RSX-11M RMS-11M MACRO Programmer's Reference Manual, 1977

5. --, AA-L672A-TC, RSX-11M/M-PLUS Command Language Manual, November 1981

6. --, AA-D083A-TC, RSX-11M RSX-11 Utilities User's Guide (Updated for V3.1), December 1977

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

8. Data and Analysis Center for Software, GLOS-1, The DACS Glossary, A Bibliography of Software Engineering Terms, October 1979

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-77-001, The Software Engineering Laboratory, V. R. Basili, M. V. Zelkowitz, F. E. McGarry, et al., May 1977

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

SEL-78-001, FORTRAN Static Source Code Analyzer (SAP) Design and Module Descriptions, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, February 1978

†SEL-78-002, FORTRAN Static Source Code Analyzer (SAP) User's Guide, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, February 1978

SEL-78-102, FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 1), W. J. Decker and W. A. Taylor, September 1982

SEL-78-003, Evaluation of Draper NAVPAK Software Design, K. Tasaki and F. E. McGarry, June 1978

SEL-78-004, Structured FORTRAN Preprocessor (SFORT) PDP-11/70 User's Guide, D. S. Wilson and B. Chu, September 1978

†This document superseded by revised document.

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

SEL-78-006, GSFC Software Engineering Research Requirements Analysis Study, P. A. Scheffer and C. E. Velez, November 1978

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

SEL-79-001, SIMPL-D Data Base Reference Manual, M. V. Zelkowitz, July 1979

SEL-79-002, The Software Engineering Laboratory: Relationship Equations, K. Freburger and V. R. Basili, May 1979

SEL-79-003, Common Software Module Repository (CSMR) System Description and User's Guide, C. E. Goorevich, A. L. Green, and S. R. Waligora, August 1979

SEL-79-004, Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment, C. E. Goorevich, A. L. Green, and W. J. Decker, September 1979

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

SEL-80-001, Functional Requirements/Specifications for Code 580 Configuration Analysis Tool (CAT), F. K. Banks, A. L. Green, and C. E. Goorevich, February 1980

SEL-80-002, Multi-Level Expression Design Language-Requirement Level (MEDL-R) System Evaluation, W. J. Decker and C. E. Goorevich, May 1980

SEL-80-003, Multimission Modular Spacecraft Ground Support Software System (MMS/GSSS) State-of-the-Art Computer Systems/Compatibility Study, T. Welden, M. McClellan, and P. Liebertz, May 1980

SEL-80-004, System Description and User's Guide for Code 580 Configuration Analysis Tool (CAT), F. K. Banks, W. J. Decker, J. G. Garrahan, et al., October 1980

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

SEL-80-007, An Appraisal of Selected Cost/Resource Estimation Models for Software Systems, J. F. Cook and F. E. McGarry, December 1980

†SEL-81-001, Guide to Data Collection, V. E. Church, D. N. Card, F. E. McGarry, et al., September 1981

SEL-81-101, Guide to Data Collection, V. E. Church, D. N. Card, F. E. McGarry, et al., August 1982

†SEL-81-002, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide, D. C. Wyckoff, G. Page, and F. E. McGarry, September 1981

SEL-81-102, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide Revision 1, P. Lo and D. Wykoff, March 1983

†SEL-81-003, Data Base Maintenance System (DBAM) User's Guide and System Description, D. N. Card, D. C. Wyckoff, and G. Page, September 1981

SEL-81-103, Software Engineering Laboratory (SEL) Data Base Maintenance System (DBAM) User's Guide and System Description, P. Lo and D. N. Card, April 1983

†SEL-81-004, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., September 1981

SEL-81-104, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., February 1982

†SEL-81-005, Standard Approach to Software Development, V. E. Church, F. E. McGarry, G. Page, et al., September 1981

†SEL-81-105, Recommended Approach to Software Development, S. Eslinger, F. E. McGarry, and G. Page, May 1982

SEL-81-205, Recommended Approach to Software Development, F. E. McGarry, G. Page, S. Eslinger et al., April 1983

SEL-81-006, Software Engineering Laboratory (SEL) Document Library (DOCLIB) System Description and User's Guide, W. Taylor and W. J. Decker, December 1981

†This document superseded by revised document.

†SEL-81-007, Software Engineering Laboratory (SEL) Compendium of Tools, W. J. Decker, E. J. Smith, A. L. Green, et al., February 1981

SEL-81-107, Software Engineering Laboratory (SEL) Compendium of Tools, W. J. Decker, W. A. Taylor, and E. J. Smith, February 1982

SEL-81-008, Cost and Reliability Estimation Models (CAREM) User's Guide, J. F. Cook and E. Edwards, February 1981

SEL-81-009, Software Engineering Laboratory Programmer Workbench Phase 1 Evaluation, W. J. Decker and F. E. McGarry, March 1981

SEL-81-010, Performance and Evaluation of an Independent Software Verification and Integration Process, G. Page and F. E. McGarry, May 1981

SEL-81-011, Evaluating Software Development by Analysis of Change Data, D. M. Weiss, November 1981

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-82-001, Evaluation of Management Measures of Software Development, G. Page, D. N. Card, and F. E. McGarry, September 1982, vols. 1 and 2

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

†This document superseded by revised document.

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-82-007, Proceedings From the Seventh Annual Software Engineering Workshop, December 1982

SEL-82-008, Evaluating Software Development by Analysis of Changes: The Data From the Software Engineering Laboratory, V. R. Basili and D. M. Weiss, December 1982

SEL-Related Literature

††Bailey, J. W., and V. R. Basili, "A Meta-Model for Software Development Resource Expenditures," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

Banks, F. K., "Configuration Analysis Tool (CAT) Design," Computer Sciences Corporation, Technical Memorandum, March 1980

††Basili, V. R., "Models and Metrics for Software Management and Engineering," ASME Advances in Computer Technology, January 1980, vol. 1

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

Basili, V. R., Tutorial on Models and Metrics for Software Management and Engineering. New York: Computer Societies Press, 1980 (also designated SEL-80-008)

††Basili, V. R., and J. Beane, "Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problems?", Journal of Systems and Software, February 1981, vol. 2, no. 1

††Basili, V. R., and K. Freburger, "Programming Measurement and Estimation in the Software Engineering Laboratory," Journal of Systems and Software, February 1981, vol. 2, no. 1

---

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

Basili, V. R., and B. T. Perricone, Software Errors and Complexity: An Empirical Investigation, University of Maryland, Technical Report TR-1195, August 1982

††Basili, V. R., and T. Phillips, "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," Proceedings of the ACM SIGMETRICS Symposium/Workshop: Quality Metrics, March 1981

Basili, V. R., R. W. Selby, and T. Phillips, Metric Analysis and Data Validation Across FORTRAN Projects, University of Maryland, Technical Report, November 1982

Basili, V. R., and R. Reiter, "Evaluating Automatable Measures for Software Development," Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity and Cost, October 1979

Basili, V.R., and D. M. Weiss, A Methodology for Collecting Valid Software Engineering Data, University of Maryland, Technical Report TR-1235, December 1982

Basili, V. R., and M. V. Zelkowitz, "Designing a Software Measurement Experiment," Proceedings of the Software Life Cycle Management Workshop, September 1977

††Basili, V. R., and M. V. Zelkowitz, "Operation of the Software Engineering Laboratory," Proceedings of the Second Software Life Cycle Management Workshop, August 1978

††Basili, V. R., and M. V. Zelkowitz, "Measuring Software Development Characteristics in the Local Environment," Computers and Structures, August 1978, vol. 10

Basili, V. R., and M. V. Zelkowitz, "Analyzing Medium Scale Software Development," Proceedings of the Third International Conference on Software Engineering. New York: Computer Societies Press, 1978

††Basili, V. R., and M. V. Zelkowitz, "The Software Engineering Laboratory: Objectives," Proceedings of the Fifteenth Annual Conference on Computer Personnel Research, August 1977

Card, D. N., "Early Estimation of Resource Expenditures and Program Size," Computer Sciences Corporation, Technical Memorandum, June 1982

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

Card, D. N., "Comparison of Regression Modeling Techniques for Resource Estimation," Computer Sciences Corporation, Technical Memorandum, November 1982

Card, D. N., and V. E. Church, "Analysis Software Requirements for the Data Retrieval System," Computer Sciences Corporation, Technical Memorandum, March 1983

Card, D. N., and V. E. Church, "A Plan of Analysis for Software Engineering Laboratory (SEL) Data," Computer Sciences Corporation, Technical Memorandum, March 1983

Card, D. N., and M. G. Rohleder, "Report of Data Expansion Efforts," Computer Sciences Corporation, Technical Memorandum, September 1982

††Chen, E., and M. V. Zelkowitz, "Use of Cluster Analysis To Evaluate Software Engineering Methodologies," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

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

Page, G., "Software Engineering Course Evaluation," Computer Sciences Corporation, Technical Memorandum, December 1977

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

Parr, F., and D. Weiss, "Concepts Used in the Change Report Form," NASA, Goddard Space Flight Center, Technical Memorandum, May 1978

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

Scheffer, P. A., and C. E. Velez, "GSFC NAVPAK Design Higher Order Languages Study: Addendum," Martin Marietta Corporation, Technical Memorandum, September 1977

Turner, C., and G. Caron, A Comparison of RADC and NASA/SEL Software Development Data, Data and Analysis Center for Software, Special Publication, May 1981

Turner, C., G. Caron, and G. Brement, NASA/SEL Data Compendium, Data and Analysis Center for Software, Special Publication, April 1981

Weiss, D. M., "Error and Change Analysis," Naval Research Laboratory, Technical Memorandum, December 1977

Williamson, I. M., "Resource Model Testing and Information," Naval Research Laboratory, Technical Memorandum, July 1979

††Zelkowitz, M. V., "Resource Estimation for Medium Scale Software Projects," Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science. New York: Computer Societies Press, 1979

Zelkowitz, M. V., "Data Collection and Evaluation for Experimental Computer Science Research," Empirical Foundations for Computer and Information Science (proceedings), November 1982

Zelkowitz, M. V., and V. R. Basili, "Operational Aspects of a Software Measurement Facility," Proceedings of the Software Life Cycle Management Workshop, September 1977

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