60498800

CONTROL DATA CORPORATION

DMS-170

# DATA BASE UTILITIES VERSION 1 REFERENCE MANUAL

CDC® OPERATING SYSTEMS:
NOS 1
NOS/BE 1

| REVISION RECORD | |
|---|---|
| REVISION | DESCRIPTION |
| A | Original release. |
| (11-01-75) | |
| B | This revision documents the use of COBOL 5 (feature CP 113) in the DMS-170 environment. Minor |
| (03-05-76) | technical corrections are included. |
| C | This revision documents version 1.1 of the data base utilities. Restore logging for the restore utility |
| (12-06-76) | (feature CP136) is added. Editorial changes reflect the use of the data base utilities by other data |
| | base management programs. PSR level 439. |
| D | This revision documents version 1.2 of the data base utilities. The use of CYBER Record Manager |
| (03-31-78) | Basic Access Methods Version 1.5 and Advanced Access Methods Version 2.0 is reflected. This |
| | revision also includes minor technical corrections. |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| Publication No. 60498800 | |

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

**CONTROL DATA CORPORATION**
*Publications and Graphics Division*
**215 MOFFETT PARK DRIVE**
**SUNNYVALE, CALIFORNIA 94086**

or use Comment Sheet in the back of this manual

# LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

| Page | Revision |
|---|---|
| Cover | – |
| Title Page | – |
| ii | D |
| iii/iv | D |
| v, vi | D |
| vii | D |
| viii | C |
| 1-1 | D |
| 1-2 | A |
| 1-3 thru 1-5 | C |
| 2-1 | D |
| 2-2 | C |
| 3-1 thru 3-7 | C |
| 3-8 | D |
| 3-9 thru 3-14 | C |
| 3-15 | D |
| 4-1, 4-2 | C |
| 4-3 | D |
| 4-4 thru 4-7 | C |
| A-1 | A |
| A-2 | D |
| A-3, A-4 | A |
| B-1 thru B-7 | D |
| C-1 thru C-5 | D |
| D-1 | A |
| D-2 | C |
| E-1 | D |
| Index-1 | C |
| Index-2 | D |
| Comment Sheet | D |
| Return Env. | – |
| Back Cover | – |

| Page | Revision |
|---|---|

| Page | Revision |
|---|---|

# PREFACE

The data base utilities constitute one essential area of service associated with a full-scale data management system. The tools and methodology needed to maintain the integrity of a data base exist in the logging facility and the recover and restore utilities, which are the data base utilities furnished in this release.

The data base utilities are part of the DMS-170 data management system. As described in this publication, the Data Base Utilities Version 1.2 operates under control of the following operating systems:

NOS 1 for the CONTROL DATA® CYBER 170 Models 171, 172, 173, 174, 175, CYBER 70 Models 71, 72, 73, 74 and 6000 Series Computer Systems

NOS/BE 1 for the CDC® CYBER 170 Series, CYBER 70 Models 71, 72, 73, 74 and 6000 Series Computer Systems

They interface with the CYBER Record Manager Advanced Access Methods Version 2 (except extended indexed sequential and extended MIP routines) and the Basic Access Methods Version 1.5.

The information in this manual is intended primarily for the data administrator - the programming or managerial person charged with the responsibility of defining, creating, controlling, and monitoring a data base. Proficiency in systems and applications programming and a familiarity with data management concepts are assumed. The term data base management program is used throughout this manual to refer to both CDCS Version 1 and QUERY UPDATE Version 3, the data base management programs that currently interface with the data base utilities.

Other manuals containing information related to the utilities are included in the list below.

| Title | Publication Number |
|---|---|
| CYBER Database Control System Version 1 Reference Manual | 60498700 |
| DDL Version 2 Reference Manual Volume 1: Schema Definition | 60498400 |
| DDL Version 2 Reference Manual Volume 2: COBOL Sub-Schema Definition | 60498500 |
| DDL Version 2 Reference Manual Volume 3: QUERY UPDATE Sub-Schema Definition | 60498600 |
| COBOL Version 4 Reference Manual | 60496800 |
| COBOL Version 5 Reference Manual | 60497100 |
| CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual | 60495700 |
| CYBER Record Manager Advanced Access Methods Version 2 Reference Manual | 60499300 |

| Title | Publication Number |
|---|---|
| FORM Version 1 Reference Manual | 60496200 |
| QUERY UPDATE Version 3 Reference Manual | 60498300 |
| NOS 1 Reference Manual (Volume 1) | 60435400 |
| NOS 1 Reference Manual (Volume 2) | 60445300 |
| NOS/BE 1 Reference Manual | 60493800 |
| Data Administrator User's Guide | 60499100 |

CDC manuals can be ordered from Control Data Literature and Distribution Services, mailing address Box 0, Minneapolis, Minnesota, 55440.

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

# CONTENTS

## FIGURES

## TABLES

# SYSTEM ORGANIZATION 1

The software modules of DMS-170 represent an advanced-concept data base management system. Inherent in DMS-170 are the means to:

- Create a common-user data base in which files can be used singly or joined in relationships.
- Provide and maintain a variety of data structures for specific users.
- Control, monitor, and interpret requests from applications programs to access one file or several related files at a time.
- Establish file integrity and security.
- Guarantee maximum efficiency for accessing, modifying, and processing data base information.

The DMS-170 functional configuration is shown in figure 1-1. The Data Description Language (DDL), the CYBER Database Control System (CDCS), and the CYBER Record Manager comprise the DMS-170 environment. COBOL applications programs and the QUERY UPDATE conversational language utilize the components of DMS-170 to access data in the data base.

## DATA DESCRIPTION LANGUAGE (DDL)

DDL is a high-level language that parallels the COBOL language convention of grouping English terms into sentence-type statements. All DDL statements are generated by a data administrator, who is responsible for preparing and maintaining the data base. DDL statements are used to describe the characteristics and structure of individual items in the data base in two distinct, but related, formats: the schema and the sub-schema. The schema describes the entire data base; the sub-schema describes the portion of the data base accessible to one or more applications programs.

### SCHEMA

The schema is a detailed description of the entire data base. The description is generated by DDL statements that name the schema, organize the schema into addressable storage units (files) called areas, describe each record type together with the characteristics of the data comprising the record, and join files in meaningful relationships. The DDL statements are used as input to the DDL compiler, which produces an object schema, or schema directory.

The system uses the schema directory to relate applications programs' symbolic references to the actual data in the data base. Only one schema exists for a data base.

### SUB-SCHEMA

The sub-schema is a detailed description of the portion of the data base that is available to an applications program. The description is generated by DDL statements that identify the schema when applicable, name the sub-schema, specify the areas needed, define the content and structure of the applicable records, and indicate any changes in data format required by the applications program; DDL statements are used to identify relations among files to be used and to specify record qualification for relation processing.

An applications program uses the sub-schema directory to obtain descriptions of applicable data. Any number of sub-schemas can exist for a data base.

## CYBER DATABASE CONTROL SYSTEM (CDCS)

CDCS is the DMS-170 controlling module that monitors and interprets all data base access requests from applications programs that are using the schema. CDCS accepts calls from the applications programs, interrogates the schema and sub-schema for compatibility, translates data formats from the programs'

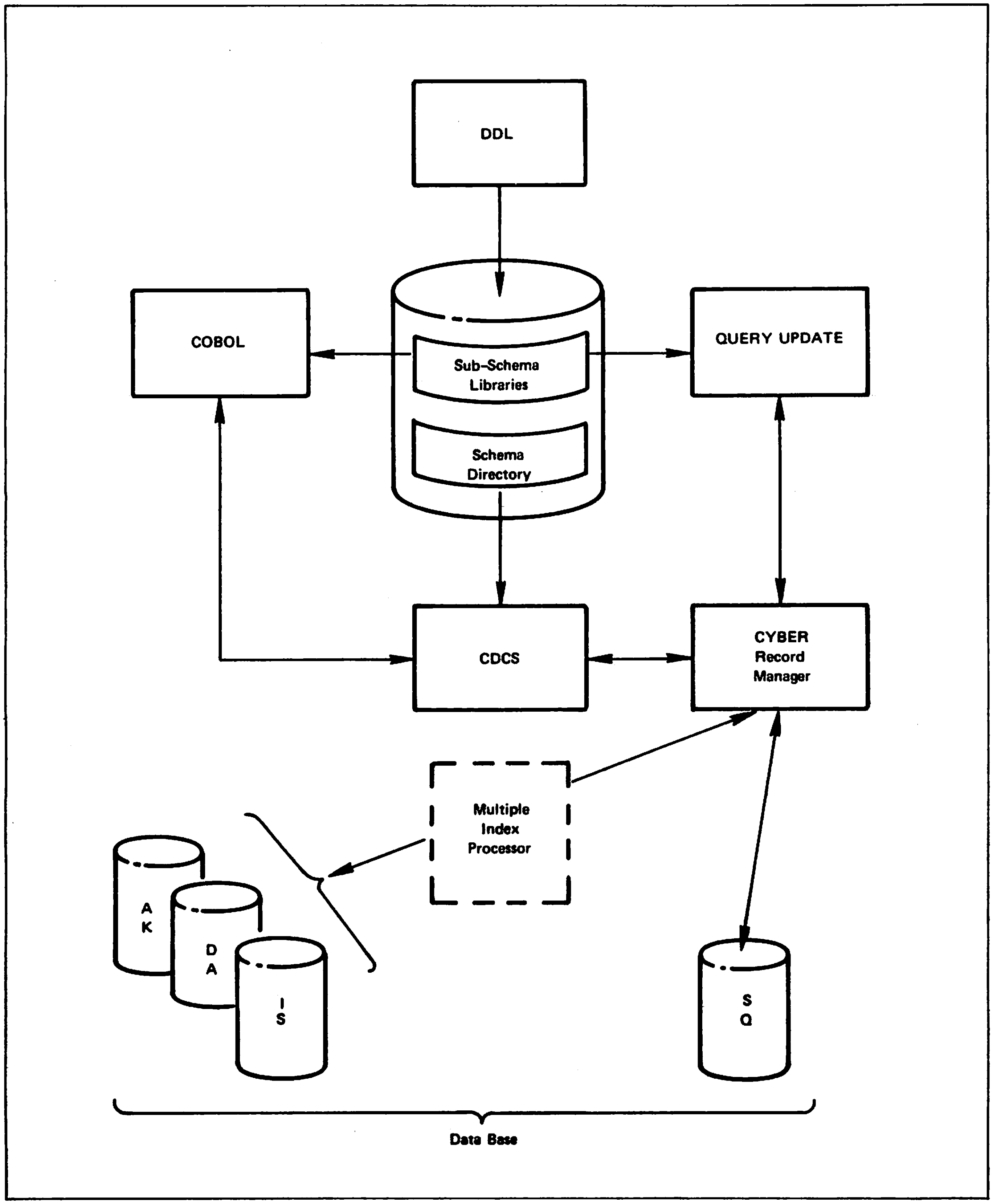


Figure 1-1. DMS-170 Functional Configuration

language to the internal format of the data, and determines the requirements for ultimate input/output processing.

### DATA MANAGEMENT SERVICES

Data independence is achieved through mapping between the schema and sub-schema. CDCS resolves the differences between data types specified in the schema and sub-schema through conversion, and reconciles structural variations that result from reordering, omission, and renaming of data in the sub-schema.

Data validation, if specified in the schema, is performed by CDCS before data is entered in the data base to prevent erroneous data from being stored. Data items must conform to stated picture specifications and fall within legal ranges when range checking is specified.

A data base procedure linkage is furnished by CDCS to permit special-purpose subprograms written by the data administrator to be called when specific situations occur during CDCS processing. Data base procedures are specified in the schema and initiated by CDCS. These routines can be used to perform a variety of operations: data validation, data conversions not supported by CDCS, calculation of actual or virtual items, specific processing on retrieval or update of records, and special handling of error conditions detected within CDCS.

The relational data base facility of DMS-170 allows COBOL users to retrieve data from several files joined logically in a structure called a relation. CDCS controls the manipulation of the files defined for a relation and the selection of record occurrences through the use of the schema and sub-schema directories.

### DATA BASE UTILITIES

The data base utilities supplement the services of CDCS and are used by both CDCS and QUERY UPDATE. The logging facility and the recover and restore off-line utilities provide the tools required to deal with an emergency situation concerning a lost, partially destroyed, or invalid data base.

The logging facility can be used to record detailed information about specific types of data base access as processing occurs. The disk or tape log file produced is processed during recovery, restoration, or statistical analysis at a later date. Logging options are defined in the schema or in the sub-schema (in the case of QUERY UPDATE) for each data base area. The types of log file entries that can be recorded are the transaction itself, a record image before the transaction, a record image after the transaction, or any combination of the three. When logging is required for a particular area at execution time, the log processing module is called to perform the operation.

The log file is processed by the recover and restore off-line utilities. The recover utility is initiated to recreate a lost or partially destroyed data base from the after image log records and a previously dumped backup copy of the data base. The restore utility uses before image log records to remedy the problem of invalid data in the data base. Selection of log records is dependent upon area, user, checkpoint, and date and time parameters included in the utility control statement.

Among the utilities that support CYBER Record Manager are routines to manipulate records, reorganize files, obtain statistical information, and create data files and index files. The operating systems provide additional utilities that are useful in data base management for dumping, loading, and copying files.

### INPUT/OUTPUT PROCESSING

CDCS directs all input/output processing to CYBER Record Manager. All operations concerning the physical storage and retrieval of data to and from the data base are handled by CYBER Record Manager.

## CYBER RECORD MANAGER

CYBER Record Manager performs execution-time input/output processing for DMS-170. Parameters relevant to CYBER Record Manager are contained in the schema directory. These parameters are specified through FILE control statements and the data control entry of the schema source program. The storage structure attributes are passed as parameters in I/O calls from CDCS to CYBER Record Manager.

CYBER Record Manager supports a variety of record types; the most commonly used types include fixed length, trailer count, zero byte, and character count. CYBER Record Manager also supports a variety of block formats and handles the blocking and deblocking of records to and from a physical device. File organizations within the data management environment are conventional as opposed to being unique within CDCS; therefore, programs written in COBOL 4, COBOL 5, FORTRAN Extended, and COMPASS can access files defined for CDCS use.

## FILE ORGANIZATION

File organization is specified in the operating system FILE control statement during DDL compilation of the schema; this information is stored in the schema directory. Four file organizations supported by CYBER Record Manager can be accessed through CDCS: sequential, indexed sequential, direct access, and actual key.

### Sequential Files

Records for sequential files are stored in the same physical order in which they are generated. Records are accessed by issuing successive reads. Sequential file organization is used most effectively with files that are to be read sequentially in the order in which they were created and that are to have records added at the end of the file.

### Indexed Sequential Files

Records for indexed sequential files are stored in ascending order by key. Records can be accessed randomly by key; records also can be accessed sequentially since the records are always in logical order. Indexed sequential file organization is used most effectively with very large mass storage files that need to be accessed both randomly and sequentially.

### Direct Access Files

Records for direct access files are stored randomly in fixed length blocks. The number of the block to receive a record is determined by a calculation performed by the system on the record key. Records can be accessed randomly by key; records can also be accessed sequentially in no specific order for read-only processing. Direct access file organization is used most effectively when rapid random access of large mass storage files is required.

### Actual Key Files

Actual key files contain records whose key values are assigned by the system. The key value specifies the block and the position within the block in which the record is stored. Records can be accessed randomly by actual key; records also can be accessed sequentially, but in no specific order. Actual key file organization is used most effectively for large mass storage files when the user can keep track of system-assigned keys and when performance and file growth characteristics are of primary concern.

## MULTIPLE INDEX PROCESSING

Multiple index processing is performed when alternate keys are defined for indexed sequential, direct access, or actual key files. CYBER Record Manager creates an index for each alternate key defined for a data file and automatically updates the indexes whenever the data file is updated. When data is read by alternate key, the index for that alternate key is searched for the desired value. The record is then obtained via the primary key, which is stored in the index. Subsequent records can be retrieved by performing a sequential read by alternate key.

## COBOL

A COBOL applications program written in COBOL 4 or COBOL 5 operates within DMS-170 whenever a COBOL sub-schema is specified in the source program. Conventional COBOL I/O verbs are used to access files or relations in the data base. CDCS interrogates the schema and sub-schema and translates the calls into requests to CYBER Record Manager for input/output processing. When the program is executing, the COBOL object time routines route I/O calls for data base files to CDCS.

## QUERY UPDATE

QUERY UPDATE is a special nonprocedural, interactive language that performs operations involving data storage and retrieval. The language is intended for non-programming personnel. Search, retrieval, update, and display operations can be performed through simple commands. Files can be joined in relations so that data from more than one file can be displayed to the user with a single query. A comprehensive report writing capability is an integral part of QUERY UPDATE.

QUERY UPDATE, like COBOL, functions within DMS-170 whenever a QUERY UPDATE sub-schema is specified. The two DMS-170 components, DDL and CYBER Record Manager, are used to generate a sub-schema and to handle input/output processing, respectively. The same files can be accessed through both QUERY UPDATE and CDCS if the QUERY UPDATE sub-schema describes the data as it resides in the data base. QUERY UPDATE and CDCS both use the logging facility and the off-line data base utilities.

The QUERY UPDATE user begins a session by entering the name of the sub-schema, the specific area of the data base, or the data base relation that is to be accessed. All input/output processing is performed automatically by CYBER Record Manager according to the information contained in the sub-schema.

# UTILITIES DESCRIPTION 2

The data base utilities constitute a set of tools required by the data administrator to maintain the integrity of the data base. The means to guard against loss of all or part of the data base, to monitor data base usage, and to perform data base recovery in an emergency situation are inherent in the data base utilities.

## UTILITIES PROVIDED

The logging facility and the recover and restore off-line utilities comprise the data base utilities. Logging is described in section 3; the recover and restore utilities are both described in section 4.

A log file of entries, which contains pertinent information on input/output operations processed, can be produced for a data base file through the logging facility. The recover and restore utilities depend on the logging feature because they require a historical record of all user interactions with a data base file. The log file actually serves a dual purpose: the log entries function as input to the recover and restore utilities and secondly, the log file provides a method of gathering usage statistics for analysis of data base areas.

The recover and restore utilities are called off-line to accomplish recovery or restoration processing. Selective recovery or restoration by area, user, checkpoint, and date and time is possible.

Several other utilities are applicable to the data base environment. FORM, a file management utility, can be used to manipulate records and reorganize files. CYBER Record Manager utilities associated with indexed sequential files include SISTAT and ESTMATE; utilities for key analysis and creation of direct access files are also provided. The IXGEN utility can generate an index file for alternate key access of an existing indexed sequential, direct access, or actual key file. Information about these utilities is contained in the FORM reference manual and the CYBER Record Manager Advanced Access Methods reference manual.

In addition to the data base utilities, FORM, and the CYBER Record Manager utilities, the operating systems provide related utility routines that are useful in data base management for dumping, loading, and copying files. The operating system reference manuals should be consulted for further information on these utilities.

## SYSTEM FLOW

Figure 2-1 shows the environment in which the logging facility and the recover and restore off-line utilities are used. Before the logging facility can be called, a log file name and logging options must be specified, a log file must be initialized as a permanent file if it is a disk log file, and the log file must be attached or requested. During the execution of a user program using the calling data base management program, as shown in the execution and logging phase, the log processing module is called to record log records on the mass storage or magnetic tape log files for the data base areas being accessed.

Backup copies of the data base can be made on a periodic basis using the operating system utility routine for dumping permanent files. Before initiating the recover or restore utility, the exact status of the data base can be determined by employing FORM, QUERY UPDATE, or a COBOL program to read the log file and to create a log report. The report can be used to decide which area or areas of the data base require recovery or restoration, as well as which applications programs must be rerun.

Control statements are used to call the recover or restore utility as shown in the recovery phase. A backup copy of the data base, which can be loaded using the operating system utility for loading permanent files, is required for recovery. If the data base is to be restored, a backup copy of the data need not be loaded. The restore utility works from the current state of the data base.

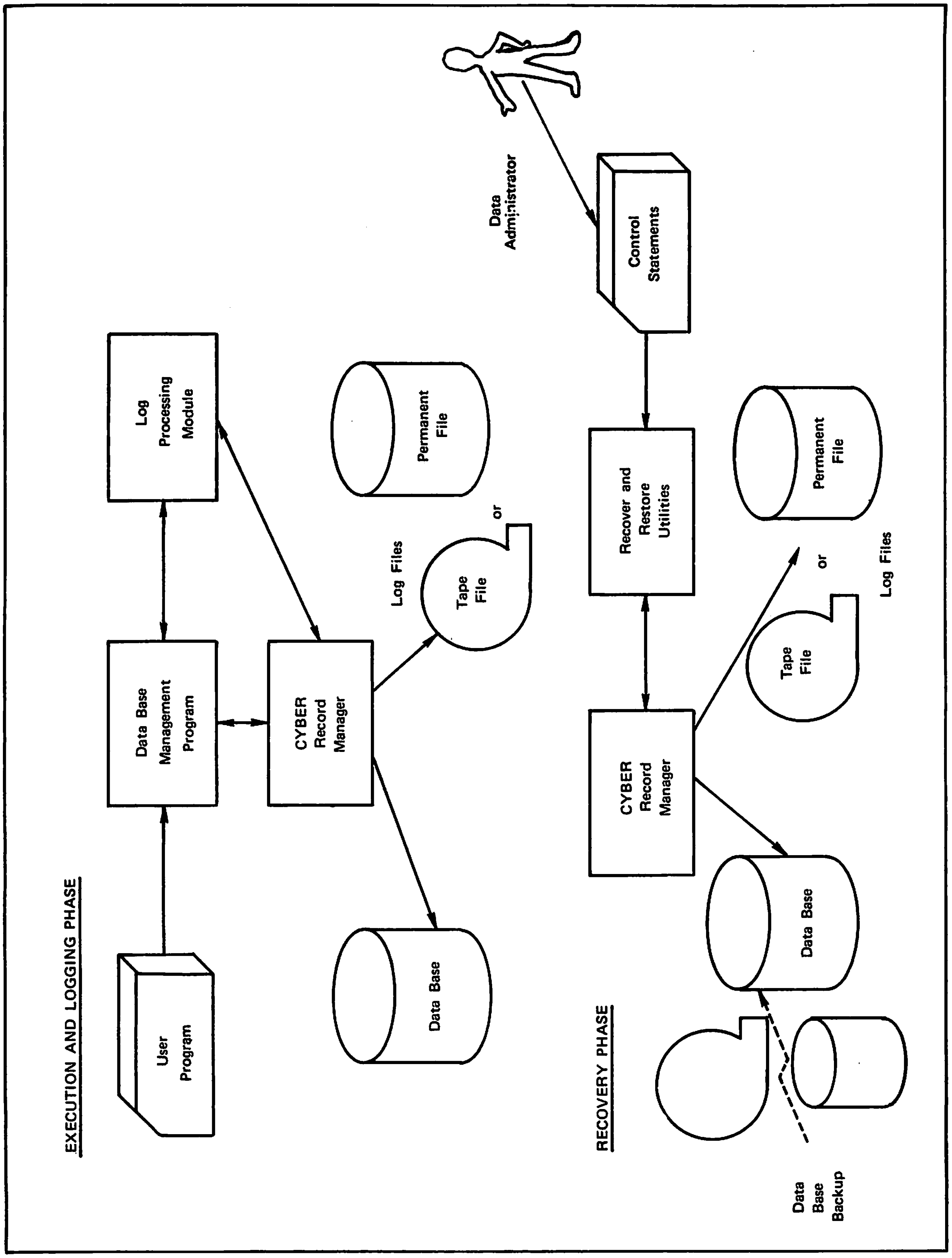


Figure 2-1. System Flow

# 3 LOGGING

The logging facility incorporates the concept of a log file whose entries provide a historical record of users' interactions with a data base area. The log file is an independent permanent file or magnetic tape file; it can serve one or more data base areas. The log processing module is called to generate a log file entry each time an input/output operation for which logging is supported is requested for the area assigned to the log file. The log file ultimately serves as input to the recover and restore utilities described in section 4.

Logging requirements are specified using the method established by the data base management program that calls the log processing module. For example, a data description language such as DDL can be used. The following types of log entries can be specified for logging:

- the transaction itself
- a record image before the transaction
- a record image after the transaction
- any combination of the above

Three additional types of entries are recorded in the log file without being specified: open, close, and checkpoint log entries.

The options of recording before and after images are intended for the express purpose of recovery and restoration. A before image is a copy of a record in a data base before the record has been modified; conversely, an after image is a copy of a record after it has been changed.

Transaction logging records a user program's requests for services from the data base management program, thereby providing a facility for collecting statistical information on each area. Information such as the frequency of data base access by a specific user or the particular kinds of operations performed on the data base can be obtained from the log file. Logging of users' activities in this way provides a comprehensive history of data base processing.

When a checkpoint is taken by an applications program, the checkpoint number can be recorded in a separate log entry in the log file, thus supplying further information for recovery purposes. The Checkpoint/restart system facility captures the total environment of a job at specified intervals so that a job can be restarted should a system failure interrupt its processing. Records comprising a checkpoint are written to an independent magnetic tape file, not a data base area or the log file.

The primary purpose of this section is to describe the structure of the log file. Additional information concerning the CYBER Record Manager interface with the log processing module, necessary control statements, and log file report generation is included.

## LOGGING MEDIA

A permanent file or a magnetic tape file can be chosen as the logging medium. The use of permanent files for logging is recommended if the anticipated volume of log record entries is not great. If the log file is a permanent file, it is regarded as a cumulative log and is positioned at end-of-information (EOI) when the file is opened. A cumulative log permits log entries to be accumulated for more than one user of the area serviced by the log file.

To maintain a degree of order, the contents of the log file can be transferred to a master log on a daily or periodic basis. For example, the master log could be a cumulative magnetic tape file of each day's permanent file entries. This procedure permits the log file to be purged and recataloged as a null log file for subsequent use. If a master log is to be used, appropriate administrative procedures need to be established for transferring data to the master log and for archiving master logs after a specified length of time has elapsed.

Magnetic tape files are the best choice for logging if the number of interactions with the data base, particularly update operations, is high. Unlike permanent file log files, a cumulative log is not generated for magnetic tape log files; log records are written from beginning-of-information (BOI) when the log file is opened. Tape log files can span more than one reel. Appropriate administrative procedures must be established for the maintenance of these files if master logs are to be produced.

An additional point to be considered when choosing a logging medium is the possible differences in performance and cost. If permanent files are selected as the logging medium, storage is more expensive and additional overhead associated with maintaining the data integrity of the permanent files is incurred.

## LOGGING RESTRICTIONS

Logging of before and after images is restricted to CYBER Record Manager files with the following file organizations:

| | |
|---|---|
| DA | direct access |
| AK | actual key |
| IS | indexed sequential |

Transaction logging can be defined for any of the above file organizations as well as sequential (SQ). A maximum of 30 areas and 15 log files can be processed during any one run in which the logging facility is utilized.

## LOG PROCESSING

A data base management program calls the log processing module when a log record is to be written on the log file. The circumstances for generating specific types of log records might differ in the data base management programs that call the log processing module (refer to the appropriate reference manual for details).

When an open command to activate an area for processing is issued, the file information table (FIT) for the area is included in the open log record. The information in the FIT is required at a later time by the recover and restore utilities to properly activate the areas that are to be recovered or restored.

Before image or after image records are recorded as a result of operations that modify the data base area. Transaction log records are recorded for specific types of operations on the data base area, determined by the data base management program using the logging facility. A checkpoint log entry is recorded when a checkpoint is taken by an applications program. Open and close log records are recorded upon completion of the open and close operations, respectively.

## LOG FILE STRUCTURE

The log file is a CYBER Record Manager sequential (SQ) file. The record type used is trailer count (T) and the blocking is character count (C). The log file can contain six types of log records or entries:

- Open
- Close
- Transaction
- Before Image
- After Image
- Checkpoint

Each log record consists of five words of information common to all log record types and a variable number of words of logging information applicable to a particular log record type. Entries in the log file are generated in a contiguous manner for each user. Each user has exclusive use of the log file when it is attached; it is not returned until job termination, unless the user returns it explicitly within the job sequence.

The log record types are entered in the following order: an open log record; one or more of the before image, after image, transaction, or checkpoint log records; and a close log record. Open, close, and checkpoint (if applicable) entries are recorded at the appropriate times.

Two additional entries, end-of-partition (EOP) and end-of-section (EOS), are generated in the log file for each sequence of log records to facilitate positioning of the log file during the recovery and restoration operations. An EOP entry is written when the log file is opened; the EOS entry is written before each open log record, after each close log record, and after each checkpoint log record. Figure 3-1 shows the arrangement of record types in a typical log file used for a single area.

If a single log file is used for more than one area, the order of recording log entries in the log file might resemble the sequence illustrated in figure 3-2. This case considers a user accessing three different areas (SALES, CUSTOMER, and PRODUCTS) during a single run.

A user has exclusive use of the log file when accessing the area associated with it. For example, if three applications programs SALES-REPORT, CUSTOMER-BILLING, and PROD-LIST access areas SALES, CUSTOMER, and PRODUCTS, respectively, and utilize a log file assigned to all three areas, each program would have to wait for the previous one to complete processing before it could gain exclusive use of the log file. In addition, a log file servicing several areas takes longer to process for recovery and restoration.

## LOG RECORD STRUCTURE

The first five words of each log record (words 0 through 4) contain information common to all log record types; the remainder of the record consists of information applicable to a specific log record type. The number of words contained in the log record varies according to the type of log record being generated. All fields in the log record hold display code. The contents of all unused fields are undefined.

### Word 0

Word 0, shown in figure 3-3, holds the record type code (one of the six possible log record types), the directive code (one of the eleven possible function requests processed by the log processing module), and the log record length (the number of characters minus 10).

An abbreviated description of word 0 is included in the description of each log record type. The actual record type code and directive code (if only one is applicable) are shown in word 0 of each log record type.

### Words 1-4

The next four words in the log record structure (words 1 through 4) contain information common to all record types. The format for these words is given in figure 3-4.

Word 1 differs in the checkpoint log record. The contents of character positions 6 through 10 of word 3 differ in the open, transaction, before image, and after image log records. Abbreviated descriptions of words 1 through 4 are included in the description of each log record type. Those character positions that differ are discussed fully under each log record type.

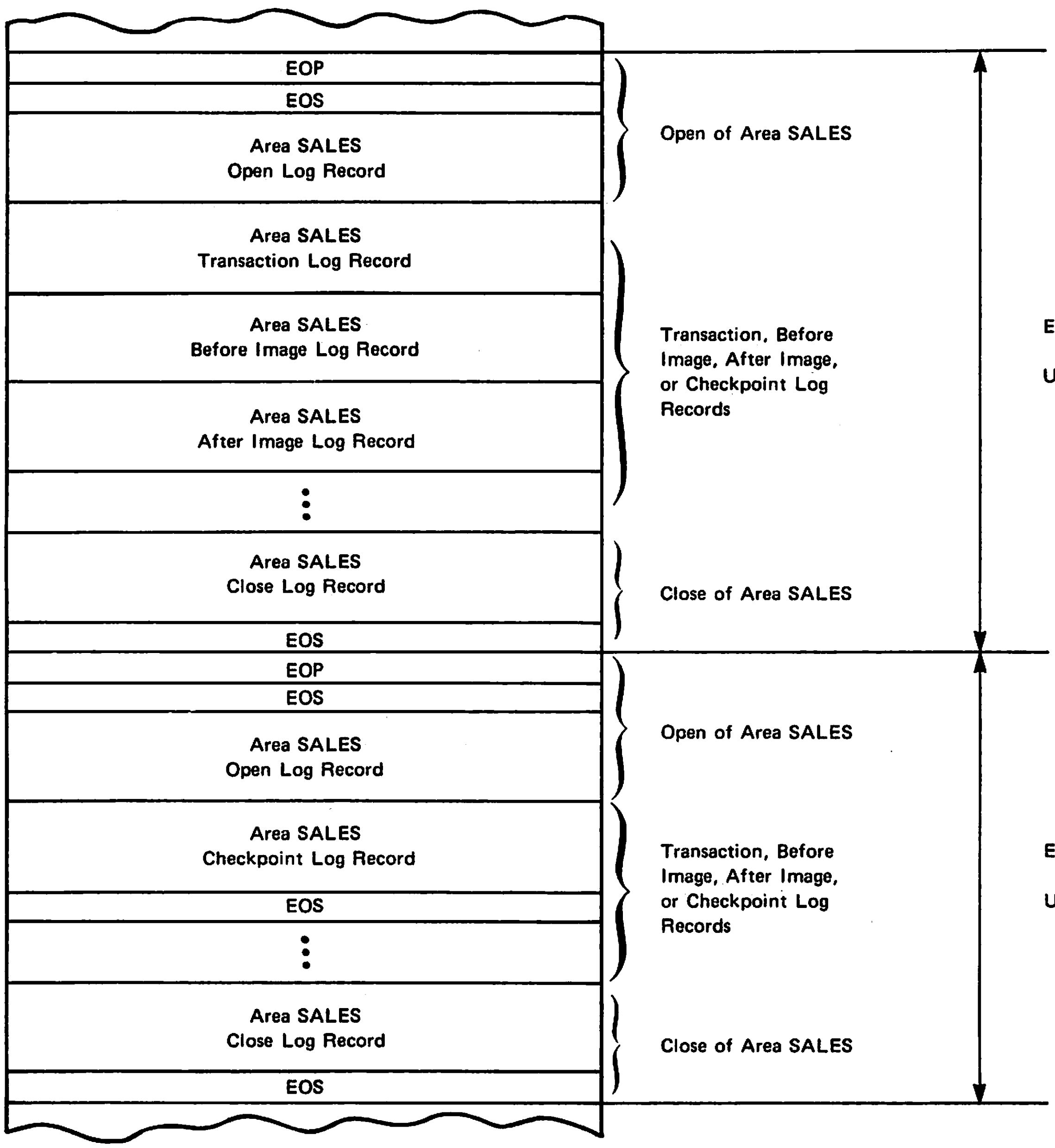
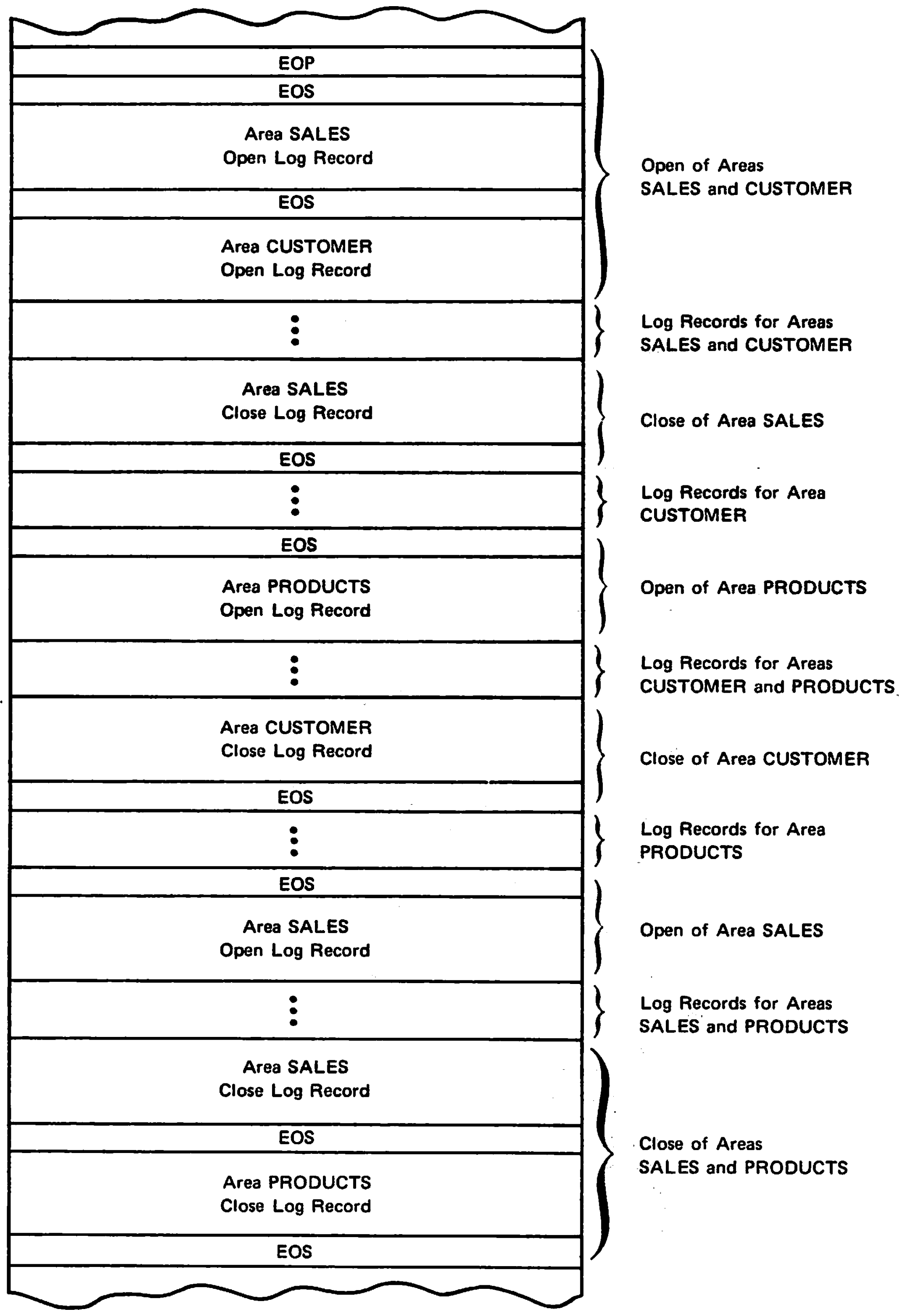

EOP
EOS
Area SALES Open Log Record
Area SALES Transaction Log Record
Area SALES Before Image Log Record
Area SALES After Image Log Record
Area SALES Close Log Record
EOS
EOP
EOS
Area SALES Open Log Record
Area SALES Checkpoint Log Record
EOS
Area SALES Close Log Record
EOS
Open of Area SALES
Transaction, Before Image, After Image, or Checkpoint Log Records
Close of Area SALES
Open of Area SALES
Transaction, Before Image, After Image, or Checkpoint Log Records
Close of Area SALES
Entries for User A
Entries for User B

Figure 3-1. One Area Assigned to a Log File



EOP
EOS
Area SALES Open Log Record
EOS
Area CUSTOMER Open Log Record
Open of Areas SALES and CUSTOMER
Log Records for Areas SALES and CUSTOMER
Area SALES Close Log Record
EOS
Close of Area SALES
Log Records for Area CUSTOMER
EOS
Area PRODUCTS Open Log Record
Open of Area PRODUCTS
Log Records for Areas CUSTOMER and PRODUCTS
Area CUSTOMER Close Log Record
EOS
Close of Area CUSTOMER
Log Records for Area PRODUCTS
EOS
Area SALES Open Log Record
Open of Area SALES
Log Records for Areas SALES and PRODUCTS
Area SALES Close Log Record
EOS
Area PRODUCTS Close Log Record
EOS
Close of Areas SALES and PRODUCTS

Figure 3-2. Three Areas Assigned to a Log File

## Log Record Types

The remainder of the log record follows one of the formats specified below, depending on the type of log record being recorded. Each record type is shown with applicable type codes and directives indicated in words 0 through 4. Figures 3-3 and 3-4 contain complete information on words 0 through 4.

### Open

The open log record, shown in figure 3-5, is generated when an area is activated for processing. The log file is opened by the log processing module when any one of the areas associated with it is opened.

### Close

The close log record, shown in figure 3-6, is generated when an area is deactivated. The log file is closed when the user program terminates, not when its associated areas are closed.

### Transaction

The transaction log record is written in response to specific directives, depending on the data base management program calling the logging facility. Open, close, and checkpoint log records are generated as they are for before or after image logging. If transaction logging and before or after image logging are

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Unused | T | Unused | DC | Unused | | | L | | | Word 0 |

| Word | Character Positions | Description |
|---|---|---|
| 0 | 1 | Unused. |
| 0 | 2 | T: The following type codes in display code format are applicable.<br>1 Log record indicates that an open directive for an area has been issued.<br>2 Log record contains an entry for a transaction: the record key and related details about it.<br>3 Log record contains a before image of the record.<br>4 Log record contains an after image of the record.<br>5 Log record indicates that a close directive has been issued for an area or that a user program has terminated.<br>9 Log record indicates that a checkpoint has been taken. |
| 0 | 3 | Unused. |
| 0 | 4 | DC: The following directive codes in display code format are applicable.<br>A Read Random<br>B Read Sequential<br>C Write Random<br>D Write Sequential<br>E Rewrite<br>F Delete<br>H Open<br>I Close<br>N Checkpoint<br>P Relation Read Random<br>Q Relation Read Sequential |
| 0 | 5 | Unused. |
| 0 | 6-10 | L: The log record length in display code format (the number of characters minus 10). |

Figure 3-3. Word 0

both specified, a transaction log record is entered for those directives for which before or after image records are written. The format of the transaction log record is shown in figure 3-7.

### Before Image

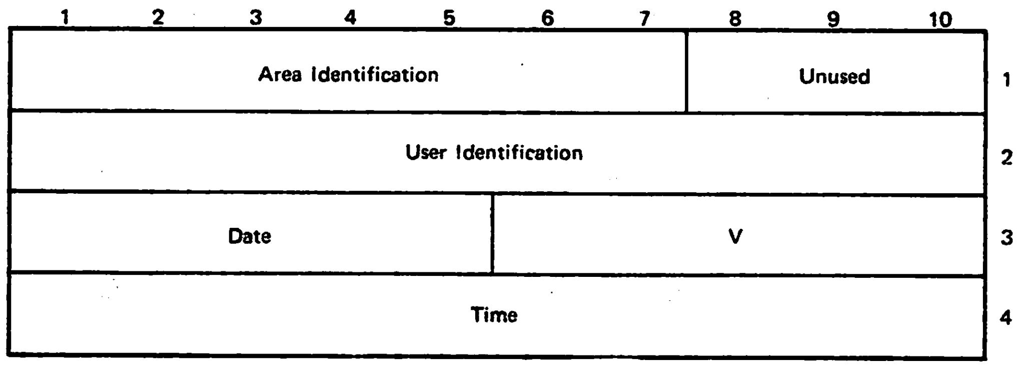
The before image log record is written in response to a directive that modifies the data base. The record includes the information shown in figure 3-8.

### After Image

The after image log record is written in response to a directive that modifies the data base. The record includes the information shown in figure 3-9.

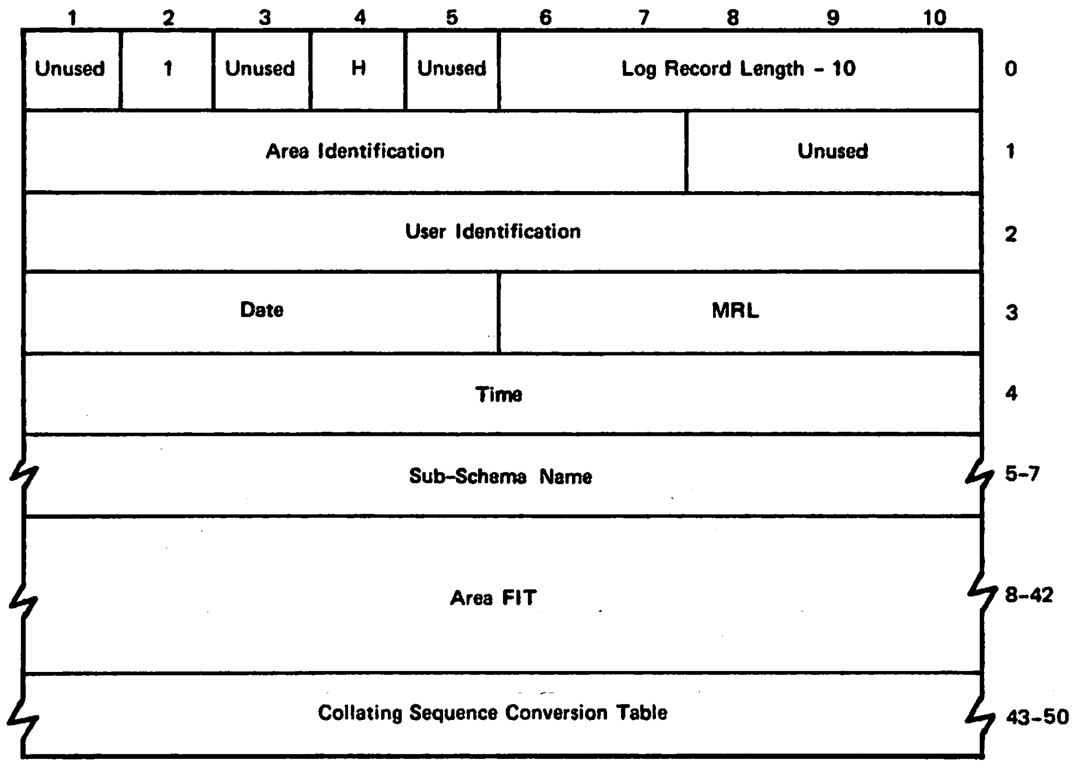
### Checkpoint

The checkpoint log record, shown in figure 3-10, is written when an applications program takes a checkpoint. The checkpoint log record provides the recover



| Word | Character Positions | Description |
|---|---|---|
| 1 | 1-7 | Area Identification: Holds the 7-character logical file name (LFN) of the area. This field is left-justified; unused positions are zero-filled. For record type 9 (checkpoint), this word is undefined. |
| 1 | 8-10 | Unused. |
| 2 | 1-10 | User Identification: Contains the 10-character user identification. This field is left-justified; unused positions are zero-filled. |
| 3 | 1-5 | Date: Contains the date in Julian format (yyddd). The first two digits are the year, the last three, the number of the day in the year. |
| 3 | 6-10 | V: The contents of this field vary in the open, transaction, before image, and after image log records. Refer to the specific log record types for a description of the contents. |
| 4 | 1-10 | Time: Contains the time of the log entry in hours, minutes, and seconds (Δhh.mm.ss.). Character position 1 contains a blank (Δ). |

Figure 3-4. Words 1-4



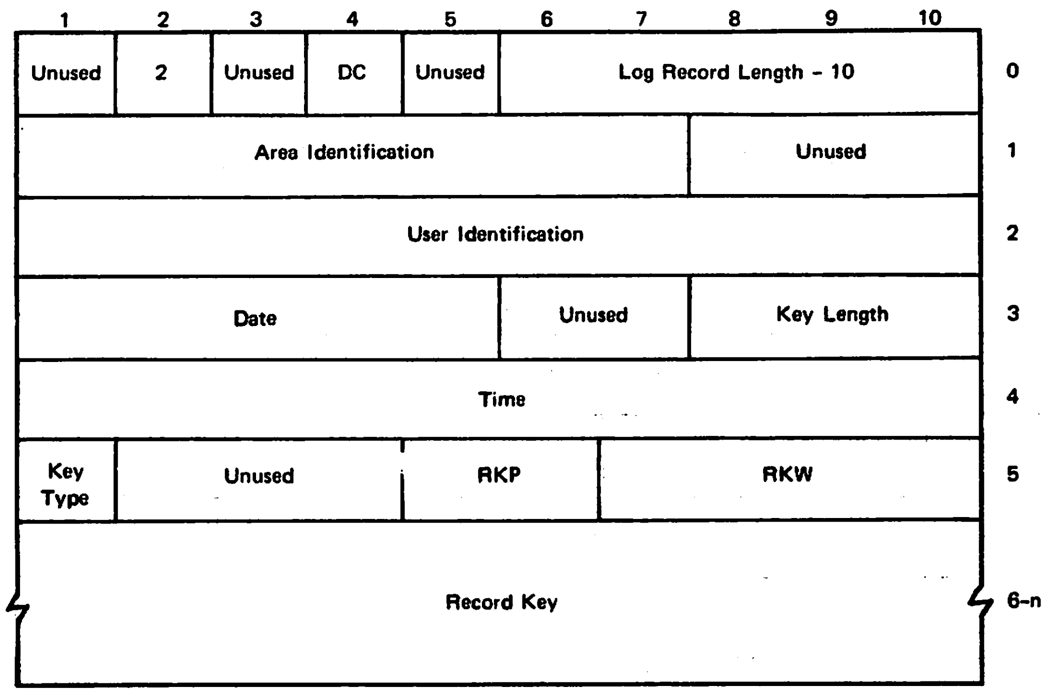
| Word | Character Positions | Description |
|---|---|---|
| 0 | — | Contains the record type code in position 2, the directive code in position 4, and the log record length (the number of characters minus 10) in positions 6 through 10. |
| 1-2 | — | Holds the area and user identifications. |
| 3 | 1-5 | Contains the date in Julian format (yyddd). |
| 3 | 6-10 | MRL: Holds the maximum record length of the records in the area. |
| 4 | 1-10 | Contains the time of the log entry (Δhh.mm.ss.). |
| 5-7 | — | Contains the sub-schema name (if applicable) as specified in the sub-schema; can be 1 to 30 characters in length. The field is left-justified; unused positions are zero-filled. |
| 8-42 | — | Contains the file information table (FIT) for the area being opened. |
| 43-50 | — | Contains the user-supplied display code to collating sequence conversion table if the system default is not used. This field is included only when a new indexed sequential file is opened. |

Figure 3-5. Open Log Record

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Unused | 5 | Unused | I | Unused | Log Record Length - 10 | | | | | 0 |
| Area Identification | | | | | | | Unused | | | 1 |
| User Identification | | | | | | | | | | 2 |
| Date | | | | | Unused | | | | | 3 |
| Time | | | | | | | | | | 4 |
| Number of Type 2 Log Records | | | | | | | | | | 5 |
| Number of Type 3 Log Records | | | | | | | | | | 6 |
| Number of Type 4 Log Records | | | | | | | | | | 7 |

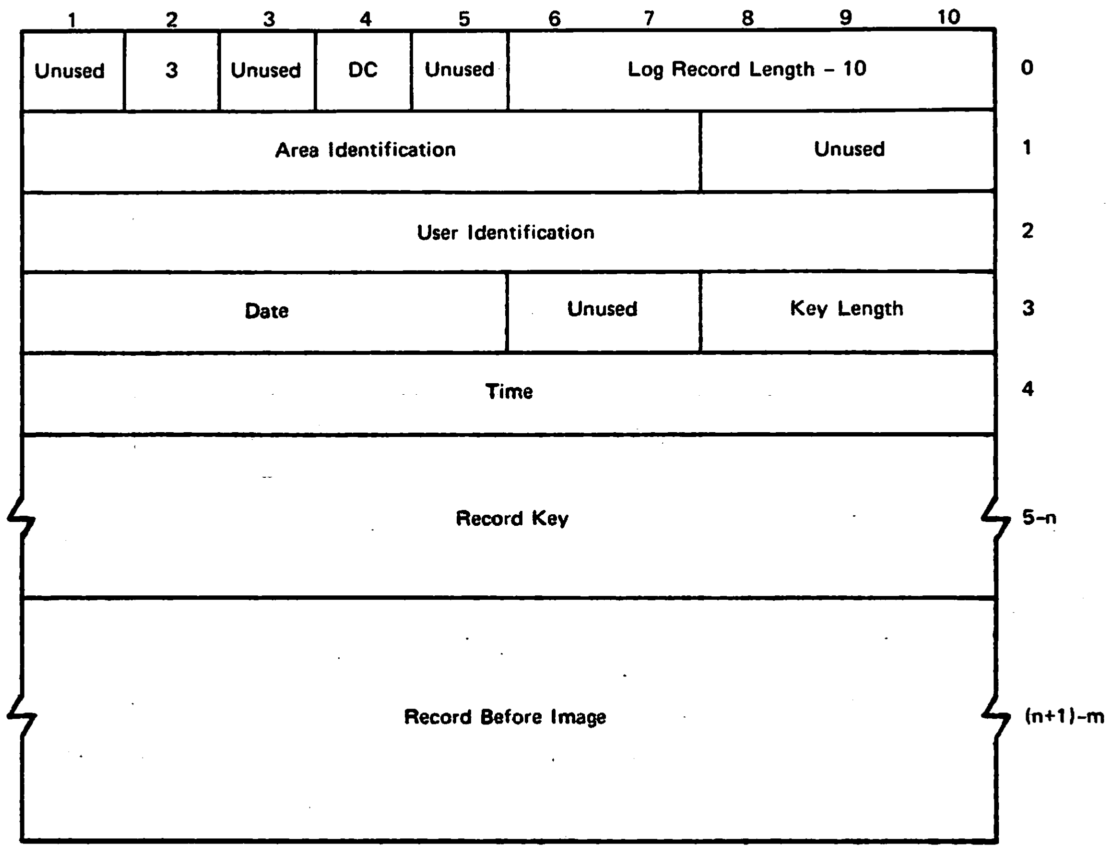
| Word | Character Positions | Description |
|---|---|---|
| 0 | – | Contains the record type code in position 2, the directive code in position 4, and the log record length (the number of characters minus 10) in positions 6 through 10. |
| 1-2 | – | Holds the area and user identifications. |
| 3 | 1-5 | Contains the date in Julian format (yyddd). |
| 4 | 1-10 | Contains the time of the log entry (Δhh.mm.ss.). |
| 5 | 1-10 | Contains the number of type 2 log records (transaction entries) for the area being closed. |
| 6 | 1-10 | Contains the number of type 3 log records (before image entries) for the area being closed. |
| 7 | 1-10 | Contains the number of type 4 log records (after image entries) for the area being closed. |

Figure 3-6. Close Log Record



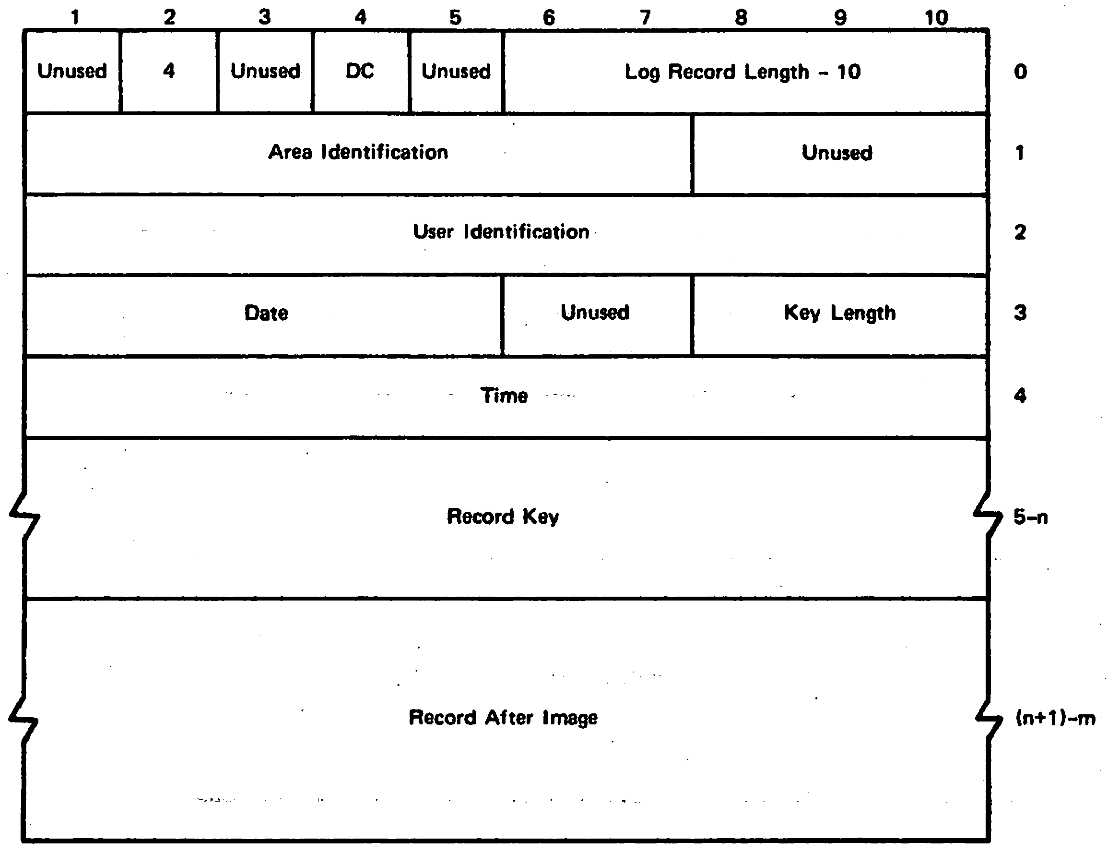
| Word | Character Positions | Description |
|---|---|---|
| 0 | – | Contains the record type code in position 2, one of the directive codes in position 4, and the log record length (the number of characters minus 10) in positions 6 through 10. |
| 1-2 | – | Holds the area and user identifications. |
| 3 | 1-5 | Contains the date in Julian format (yyddd). |
| 3 | 6-7 | Unused. |
| 3 | 8-10 | Key Length: Contains the length of the record key in characters. |
| 4 | 1-10 | Contains the time of the log entry (Δhh.mm.ss.). |
| 5 | 1 | Key Type: For a record in a CYBER Record Manager IS, DA, or AK file, contains one of the following key types:<br>A no key<br>B primary key<br>C alternate key |
| 5 | 2-4 | Unused. |
| 5 | 5-6 | RKP: Contains the relative character position of the key in the word in which the key begins. This field is used only for key type C. |
| 5 | 7-10 | RKW: Contains the relative word location of the key in the record. This field is used only for key type C. |
| 6-n | – | Record Key: Contains the value of the key. |

Figure 3-7. Transaction Log Record



| Word | Character Positions | Description |
|---|---|---|
| 0 | – | Contains the record type code in position 2, one of the directive codes in position 4, and the log record length (the number of characters minus 10) in positions 6 through 10. |
| 1-2 | – | Holds the area and user identifications. |
| 3 | 1-5 | Contains the date in Julian format (yyddd). |
| 3 | 6-7 | Unused. |
| 3 | 8-10 | Key Length: Contains the length of the record key in characters. |
| 4 | 1-10 | Contains the time of the log entry (Δhh.mm.ss.). |
| 5-n | – | Record Key: Contains the primary key of the record. A maximum of 240 characters is allowed for the key. |
| (n+1)-m | – | Record Before Image: Contains the before image of the data base record. If the directive code is C (write random), this portion of the record is not generated. |

Figure 3-8. Before Image Log Record



| Word | Character Positions | Description |
|---|---|---|
| 0 | – | Contains the record type code in position 2, one of the directive codes in position 4, and the log record length (the number of characters minus 10) in positions 6 through 10. |
| 1-2 | – | Holds the area and user identifications. |
| 3 | 1-5 | Contains the date in Julian format (yyddd). |
| 3 | 6-7 | Unused. |
| 3 | 8-10 | Key Length: Contains the length of the record key in characters. |
| 4 | 1-10 | Contains the time of the log entry (Δhh.mm.ss.). |
| 5-n | – | Record Key: Contains the primary key of the record. A maximum of 240 characters is allowed for the key. |
| (n+1)-m | – | Record After Image: Contains the after image of the data base record. If the directive code is F (delete), this portion of the log record is not generated. |

Figure 3-9. After Image Log Record

and restore utilities with supplementary information. Checkpoint log records are recorded automatically. If several log files are being utilized by an applications program, a checkpoint log record is recorded in each one. If a log file has not been defined, the Checkpoint/restart facility can be used in the usual manner by an applications program, but no checkpoint log records are written.

Checkpoint log records are entered if the log file is open; if it has not yet been opened, checkpoint information is saved each time a checkpoint is taken pending the opening of the log file. When this eventually occurs, the number of the last checkpoint taken by the program is recorded in a checkpoint log record generated for inclusion in the log file.

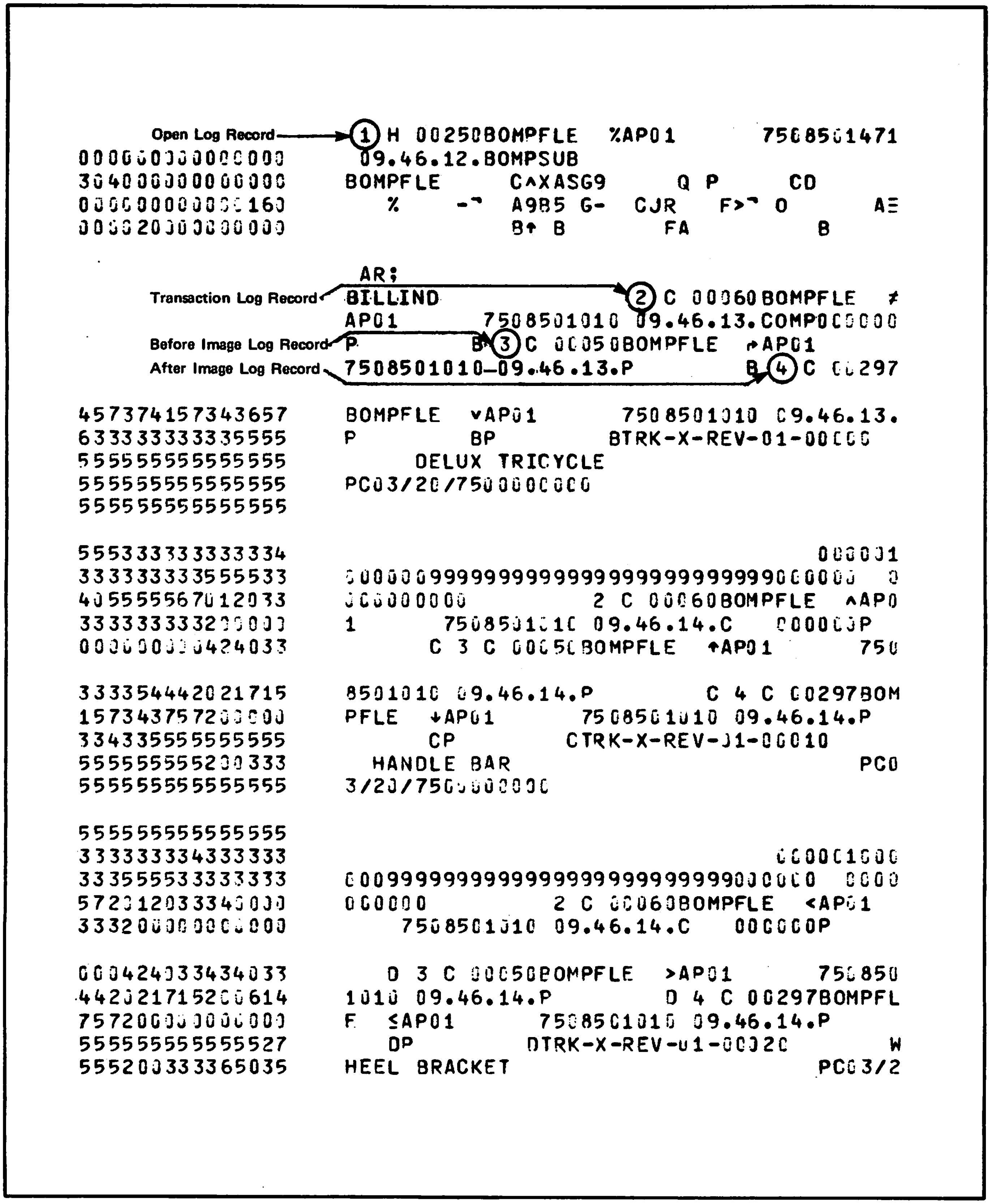
## SAMPLE LOG FILE

Figure 3-11 shows an excerpt from the log file BOMPLOG. BOMPLOG is assigned to BOMPFLE, a bill of materials area from a bicycle manufacturing data base. An open log record and several sequences of transaction, before image, and after image log records are apparent in this dump. The type codes contained in word 0 of the first few log records are circled for reference purposes.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Unused | 9 | Unused | N | Unused | 0 | 0 | 0 | 5 | 0 | 0 |
| Unused | | | | | | | | | | 1 |
| User Identification | | | | | | | | | | 2 |
| Date | | | | | Unused | | | | | 3 |
| Time | | | | | | | | | | 4 |
| Checkpoint Number | | | | Unused | | | | | | 5 |

| Word | Character Positions | Description |
|---|---|---|
| 0 | – | Contains the record type code in position 2, the directive code in position 4, and the log record length (the number of characters minus 10) in positions 6 through 10. |
| 1 | 1-10 | Unused. |
| 2 | 1-10 | Holds the user identification. |
| 3 | 1-5 | Contains the date in Julian format (yyddd). |
| 3 | 6-10 | Unused. |
| 4 | 1-10 | Contains the time of the log entry (Δhh.mm.ss.). |
| 5 | 1-4 | Checkpoint Number: Contains the 4-character checkpoint number. |
| 5 | 5-10 | Unused. |

Figure 3-10. Checkpoint Log Record



```
Open Log Record ──────> (1) H 00250BOMPFLE   %AP01        7508501471
000000000000000      09.46.12.BOMPSUB
304000000000000     BOMPFLE     C^XASG9      Q P      CD
000000000000160        %      -¬  A9B5 G-  CJR    F>¬ O        A=
000020000000000                   B↑ B        FA           B

                      AR;
Transaction Log Record  BILLIND ──────────────> (2) C 00060BOMPFLE  ≠
                     AP01         7508501010 09.46.13.COMP0C0000
Before Image Log Record P ──────────── B (3) C 00050BOMPFLE  ↱AP01
After Image Log Record 7508501010─09.46.13.P ──────── B (4) C 00297

457374157343657     BOMPFLE  vAP01       7508501010 09.46.13.
633333333335555     P          BP        BTRK-X-REV-01-00000
555555555555555          DELUX TRICYCLE
555555555555555     PC03/20/7500000000
555555555555555

555333333333334                                        000001
333333333555533     0000009999999999999999999999990000000  0
405555567012033     000000000          2 C 00060BOMPFLE   ^AP0
333333333200003     1         7508501010 09.46.14.C     C00000P
000000000424033           C 3 C 00050BOMPFLE   ↑AP01        750

333544420217 15     8501010 09.46.14.P        C 4 C 00297BOM
157343757200000     PFLE  ↓AP01        7508501010 09.46.14.P
334335555555555           CP          CTRK-X-REV-01-00010
555555555200333       HANDLE BAR                            PC0
555555555555555     3/20/7500000000

555555555555555
333333334333333                                      00001000
333555533333333     0009999999999999999999999999000000  0000
572012033340000     000000           2 C 00060BOMPFLE   <AP01
333200000000000         7508501010 09.46.14.C     000000P

000424033434033        D 3 C 00050BOMPFLE   >AP01        750850
442021715200614     1010 09.46.14.P          D 4 C 00297BOMPFL
757200000000000     F  ≤AP01        7508501010 09.46.14.P
555555555555527        DP         DTRK-X-REV-01-00020         W
555200333365035     HEEL BRACKET                          PC03/2
```

Figure 3-11. Sample Log File Dump

## CYBER RECORD MANAGER INTERFACE

All input/output for the log processing module is performed by CYBER Record Manager. The log processing module determines whether the log file is a permanent file or a magnetic tape file when the file is opened. If it is a disk file but not a permanent file, a fatal diagnostic is issued. A disk log file is opened and positioned at end-of-information (EOI); a tape log file is opened and positioned at the beginning-of-information (BOI) marker.

Log records are written sequentially until the user program finishes processing or terminates with an error, at which time the log file is closed. The log file remains open until the job terminates so that checkpoint entries can be recorded.

Input/output buffering is handled by CYBER Record Manager. Specification of a buffer size when the log file is assigned is optional. If no size is specified, the system default sizes for disk devices and magnetic tape are used.

If logging is being recorded on a permanent file under the NOS/BE 1 operating system, the log processing module issues a permanent file EXTEND to preserve the data integrity of the permanent file. An EXTEND is issued each time an area is closed, a checkpoint is recorded, and a job is terminated.

## ERROR CONDITIONS

The manner of handling an error encountered by the log processing module while it is processing a log file depends on the data base management program that calls the module. The log processing module issues a message on the job dayfile and returns control to the calling data base management program. Refer to the appropriate reference manual describing the data base management program for details. Appendix B contains a list of error diagnostics issues for logging and the error conditions associated with them.

## LOG FILE REPORT GENERATION

By utilizing FORM, QUERY UPDATE, or a COBOL program to read the log file and create a log report, the contents of the log file can be examined and the exact state of the data base can be determined before beginning a recovery or restoration sequence. Reports consisting of information by area, user, time, record type, or a combination of these can be generated.

If QUERY UPDATE is used to generate a report, the log file should be described with the QUERY UPDATE DESCRIBE directive. Then a formatted log report can be prepared from the data in the log file using QUERY UPDATE report preparation facilities. The prepared report can be examined at a terminal or it can be printed. The QUERY UPDATE reference manual contains specific information on describing files and generating reports.

FORM, a general-purpose file management utility for manipulating records and converting files, can be used to print the log file. Format options can be selected. The information needed to use this utility is found in the FORM reference manual.

A third method of producing a log report is through the use of COBOL. A COBOL program or several programs can be written to extract and format the pertinent information in the log file. Routines can be written to format report pages or the Report Writer statements can be employed to define a report format and to generate the log report. The COBOL reference manual gives a detailed description of both user-controlled report generation and report generation through the Report Writer.

When the log file has been used as a means of collecting information for statistical purposes, the methods described above can also be employed to generate statistical reports.

## CONTROL STATEMENTS

A log file must exist at the control point prior to the execution of the user program that utilizes it. A disk-resident log file must already be initialized as a permanent file. At execution time, the user must include the appropriate tape or permanent file control statements that are required to make each log file referenced available. Refer to the appropriate reference manual of the data base management program calling the log processing module for details.

# DATA BASE RECOVERY 4

Data base recovery is accomplished through the utilization of the recover and restore off-line utilities. Selected entries from a log file can be processed to recreate or restore one or more areas of a data base.

The recover utility applies after images recorded in the log file to a previously dumped backup copy of the data base; it handles recovery from a lost or destroyed data base. Working from the current state of the data base, the restore utility employs before images in the log file to reverse the results of invalid operations performed on the data base, or to restore the data base to a previous state.

Both recovery and restoration can be performed selectively by area, user, and date and time. Recovery or restoration to a specific checkpoint to allow a job to be restarted from that point is also provided by the utilities. During data base restoration, restore logging is available to log modifications made to the data base by the restore utility.

The recover and restore utilities are called by the DFRCV and DFRST control statements, respectively. In addition to the applicable utility control statement, appropriate control statements must be supplied to make available all areas involved in the recovery process.

## RECOVERY CONDITIONS

The recovery sequence chosen is dependent upon the type of failure that has occurred and the logging technique employed. Table 4-1 indicates the major conditions or cases in which the recover and restore utilities would be called. Column 3 gives the corrective action to be taken in each case.

The first case covers hardware problems such as major parity errors and physically damaged disk drives. The only recourse is to load a backup copy of all or part of the data base, identify the log files produced since the last dump, and invoke the recover utility. It may be necessary to call the recover utility several times to complete recovery for all damaged areas if more than one log file was produced.

The second condition involves an applications program that aborts after considerable processing. Reprieve processing provided by the system almost always ensures that all files are closed. Initially, the contents of the log file might have to be examined to determine which date and time values to specify in the control statement for the restore utility. The utility can then be called to reset the data base to a previous state that corresponds to the beginning of the job or to a previous checkpoint, and the job can be restarted. In the latter case, most of the processing time already spent by this job can be saved. Before the applications program is permitted to access the data base again, the offending program or data must be corrected.

The third case covers an applications program logic error or other problem that causes invalid data to be stored in the data base by an applications program. The error is detected at a later time, possibly after a second applications program has processed the invalid data. An examination of the log file can reveal whether subsequent applications programs have also processed the erroneous data. If this is not the case, the restore utility can be utilized to undo the effects of the offending program. If other applications programs are involved, it may be necessary to rerun one or more of them after restoration has been accomplished.

System failure is the last case shown in the table. Some system failures, such as power failure and operating system failure, often leave the data base in a recoverable state. If the log file has been closed, the contents of the log file can be used to restore the data base to a previous valid state.

In certain cases, however, the data base might be found to be unreadable when restoration is attempted; if so, that portion of the data base affected must be recovered utilizing a backup copy and the recover utility. If the log file has not been closed and is unreadable, loading a backup copy of the data base is the only recourse. Any data base transactions that were lost must be processed again.

TABLE 4-1. RECOVERY CONDITIONS

| Condition | Explanation | Action |
|---|---|---|
| Physical storage failure | All or part of data base is lost or otherwise unreadable. | Recover from backup copy. |
| Applications program failure | Applications program is aborted. Reprieve processing provided by the system almost always ensures that all files are closed. | Restore data base to previous checkpoint or beginning of job. Correct problem that caused the abort. Restart or rerun job. If all files are not closed, recover from backup copy, correct problem, and rerun job. |
| Applications program logic error | Invalid data is stored in data base. Applications program terminates normally. | Restore data base to previous valid state. |
| System failure | A power failure or operating system failure occurs. | If log file has been properly closed, restore the data base. If data base is found to be unreadable, recover from backup copy. |

## THE RECOVER UTILITY

Recovery of an area requires a recent backup copy of the area. A backup copy of all or selected portions of a data base can be made at predetermined intervals. Utility routines for dumping and loading permanent files are provided under each of the operating systems. The operating system reference manuals contain detailed information on these utilities.

After the backup copy has been loaded, the recover utility can be called. It selects log records from the specified log file according to the parameters specified in the DFRCV control statement. The pertinent after image entries from the log file are applied in chronological sequence beginning at the start position indicated in the utility control statement.

An area whose records have duplicate keys can be recovered only if both before and after image records are present in the log file. Duplicate key records are not guaranteed to be in the same order in the recovered file as they were in the original file.

## THE RESTORE UTILITY

An area of the data base can be restored to a previous state by utilizing the restore utility. The current invalid state of the data base is presented to the utility for restoration.

### RESTORATION

The restore utility selects log entries from the log file according to the parameters specified in the DFRST control statement. The applicable before image entries in the log file are processed in reverse chronological sequence.

An area whose records have duplicate keys can be restored only if both before and after image records are present in the log file. Duplicate key records are not guaranteed to be in the same order in the restored file as they were in the original file.

## RESTORE LOGGING

Restore logging can be selected during data base restoration to log modifications made to the data base by the restore utility. The restore logging facility provides for logging before and after images of records to the primary log file or to another independent log file. Unless provision is made to log modifications made by the restore utility, they would be lost to future recovery operations. Restore logging also protects the data base against a system or hardware failure during the restoration job.

The following example illustrates the need for restore logging. A data base contains the record occurrences A, B, and C. A program that contains an error rewrites record occurrence B, writes record occurrence D, and then aborts. The log file produced from the job would contain the following information:

A before image of B

An after image of B (called B1)

A null before image of D

An after image of D

Since the program has stored erroneous data in the data base, the restore utility is called to restore the data base to a previous valid state. The restore utility rewrites B over B1 and deletes D. At the end of the restoration the data base contains A, B, and C.

If a hardware storage failure occurs at some time after the restoration (before another complete backup dump of the data base can be made), the recover utility must be called to reconstruct the data base from an earlier state using the log file described above. The earlier state of the data base contains records A, B, and C. The effect of the recovery operation would be to rewrite B1 over B and to write D. The data base then would be identical to the data base following the execution of the erroneous program. If restore logging had been used, however, the log file would contain the following additions:

A before image of B1

An after image of B1 (same as B)

A before image of D

A null after image of D

Using this expanded log file, the recover utility would duplicate the operation of the previous restoration job, and the data base would be recovered to reflect the most recent version of the data base desired.

The restore utility is responsible for generating log records for the restore log file. Since the restore utility does not have access to the logging options selected by the data administrator, it unconditionally logs before and after images and open and close operations. This combination allows both recovery of modifications made by the restore utility and restoration of a previous operation performed by the restore utility. The user identification in the log records generated by the restore utility holds the identification DFRST.

The RL parameter in the DFRST control statement is used to specify restore logging. If the primary log file used as input to the restore utility is a tape file, it cannot be used for restore logging; an independent log file must be designated to hold restore logging records.

The method used for restore logging depends on the procedures set up for maintaining the data base. If, for example, the procedure is to keep all logging cumulative to the same log file, then the default option of logging to the primary log file can be selected. Alternatively, if an independent log file is chosen for restore logging, the primary log can become a backup copy and the log file used for restore logging can serve as the primary log file until the next restoration or recovery operation is performed. If a backup copy of the data base is made following every restoration, restore logging might not be deemed necessary. Only in the case of a system failure during a restoration job would the data base be unprotected.

## CONTROL STATEMENT FORMATS

The recover and restore utility options are specified in the DFRCV and DFRST control statements. All parameters, except the date and time parameters and the RL parameter in the DFRST control statement, have identical meanings for both control statements. In the formats shown in figures 4-1 and 4-2, constants are in uppercase and variables are in lowercase. The log file logical file name is the only required parameter. No parameter can be specified more than once.

No restriction is imposed on the ordering of the parameters; brackets enclose optional items within the optional parameters. Parameters can be continued to a second control statement but cannot be split across control statements. If the terminator period does not appear in the first statement, the next statement is assumed to be a continuation of the first. Commas, equals, and slashes must be used as indicated in the formats shown.

Additional control statements must be supplied to make the associated data base areas and index files available to the utility. The next subsection contains specific information on all necessary control statements.

```
DFRCV,L=log-file-lfn
      ,A=area-id
      ,U=user-id
      ,D=[start-date/]end-date
      ,T=[start-time/]end-time
      ,C=checkpoint-number
      ,H=hash-routine-entry-name/libname
      ,O=output-lfn
      .
```

Figure 4-1. Recover Control Statement Format

The required log file logical file name parameter is:

L The logical file name of the log file as it appears in the ATTACH or REQUEST (for tape) control statement used in the recover or restore control statement setup. A name having 1 to 7 characters is permissible. If no

```
DFRST,L=log-file-lfn
      ,A=area-id
      ,U=user-id
      ,D=start-date[/end-date]
      ,T=start-time[/end-time]
      ,C=checkpoint-number
      ,H=hash-routine-entry-name/libname
      ,O=output-lfn
      ,RL [=restore-log-file-lfn]
          [=0                   ]
      .
```

Figure 4-2. Restore Control Statement Format

other parameters are specified, the entire contents of the log file are used to perform the recovery or restoration operation. If multiple log files exist, the recover or restore job must be run for each log file.

The optional parameters follow.

A The 1 to 7 character area identification of the area to be recovered or restored as it appears in the log file. If no area identification is specified, all areas referenced in the log file are processed in the recovery or restoration operation.

U The 1 to 10 character user identification contained in the log file. If no user identification is specified, the default is all users. If the user identification contains a hyphen, the entire name must be enclosed by dollar signs ($). Both the U and A options serve to restrict the scope of recovery or restoration to specific log entries.

D Start-date and end-date are expressed in the format yyddd, where the first two digits are the year, the last three, the number of the day in the year. If the D parameter is specified, then the T parameter must also be included.

The range of records to be processed is determined by the range specified in the start-date and end-date values. If only one date is given in the DFRCV control statement, it is considered to be the end-date. Records on the log file containing a date later than end-date are not recovered. The default start-date is the beginning of the log file.

If only one date is given in the DFRST control statement, it is considered to be the start-date. Records on the log file containing a date earlier than start-date are not restored. The default end-date is the end of the log file.

The combination of start-date and start-time must always be earlier than the combination of end-date and end-time.

T Start-time and end-time are expressed in the format hhmmss, designating hours, minutes, and seconds. If the T parameter is specified, then the D parameter must also be included.

The start-date and start-time and end-date and end-time sequences specified in the D and T parameters determine the range of records to be processed. For the DFRCV control statement, the default start-time is the beginning of the log file. The default end-time is the end of the log file for the DFRST control statement.

The combination of start-date and start-time must be earlier than the combination of end-date and end-time.

C An integer 1 to 2047 corresponding to the applicable checkpoint number recorded in the log file. If the checkpoint parameter is specified, it designates a termination point for the restore or recover utility. This point must be within the start-date and start-time and end-date and end-time period if D and T are specified in the control statement.

H The 1 to 7 character name of the entry point of a hash routine. The H parameter applies to direct access (DA) files and is used only if a hash routine is supplied by the user. The hash routine is stored in a library; the library name must be included as shown in the format.

If no user hash routine is specified, the default is a hash routine supplied by the system. The hash routine specified applies to all DA files processed in the recover or restore operation. If the hash routine is stored in a user library, this library must be attached with a logical file name specified as the hash library name.

O The 1 to 7 character logical file name of the output file to which nonfatal error diagnostics are written. The default is OUTPUT.

RL The single specification RL or the omission of the parameter designates restore logging to the primary log file whose logical file name is specified in the L parameter. If the primary log file is a tape file, it cannot be used as the restore log file.

If RL=0 is specified, no restore logging is done. The restoration job field length and execution time are reduced when restore logging is not selected; recovery of the data base might be more difficult if a system failure occurs.

The logical file name of an independent log file can be specified with the RL parameter. It can be a disk or a tape file.

## ADDITIONAL CONTROL STATEMENTS

ATTACH or REQUEST control statements must be supplied for all areas involved in the recovery or restoration process. An ATTACH control statement or the appropriate tape processing control statements (VSN, LABEL, and REQUEST or ASSIGN) must be included for the input log file. If a special output file has been designated in the utility control statement, this file must also be made available. The log file is rewound by the utility before processing begins.

If a multiple-indexed area is to be recovered or restored, an ATTACH control statement must be supplied to attach the index file associated with the multiple-indexed area. Additionally, a FILE control statement must be included to provide the logical file name of the index file for the XN field in the FIT of the area being recovered or restored.

Appropriate tape or permanent file control statements must be included for the log file to be used for restore logging. A tape file must be mounted with a write-enable ring.

## SAMPLE DECK SETUPS

Typical deck setups for calling the recover and restore utilities under each of the operating systems appear in figure 4-3. The control statements included depend on the purpose of the job, the types of files, and the number of areas to be recovered.

The recover and restore utilitites generate statistics concerning the total number of writes, deletes, and rewrites processed during a recovery or restoration job. The following statistics appear in the job output and dayfile:

```
TOTAL WRITES PROCESSED = nnnnnn
TOTAL DELETES PROCESSED = nnnnnn
TOTAL REWRITES PROCESSED = nnnnnn
```

If the relevant total is not zero for the number of writes, deletes, or rewrites not processed during a recovery or restoration job, any of the following lines of statistics might be generated in the job output and dayfile:

```
TOTAL WRITES NOT PROCESSED = nnnnnn
TOTAL DELETES NOT PROCESSED = nnnnnn
TOTAL REWRITES NOT PROCESSED = nnnnnn
```

The success or failure of the recovery or restoration job can be determined immediately using these statistics.

## CHECKPOINT/RESTART INTERFACE

An applications program might be terminated abnormally at any time during execution as a result of system, operator, or programmer error. If a job has utilized the Checkpoint/restart system facility, it is usually possible to restart the job from a specific checkpoint rather than from the beginning of the job.

The restore utility can be used to reset the data base to the point at which a particular checkpoint was taken. This is accomplished by specifying the applicable checkpoint number in the C parameter in the DFRST control statement. After the restoration procedure the job can be restarted from its checkpoint tape in the usual manner. The recover utility can also be called with the checkpoint parameter. Checkpoint/restart is described fully in the reference manual for each of the operating systems.

## ERROR PROCESSING

Diagnostics are issued by the recover and restore utilities for fatal and nonfatal errors. Fatal errors are written to the job dayfile; nonfatal errors appear in the output file specified in the O parameter of the utility control statement. A complete list of diagnostics is contained in appendix B.

Example 1

1. Execute recover utility, selecting log records that apply to area CUSTOME, an IS file. The log file is a permanent file.

NOS/BE 1

```
Job Statement
ATTACH(CUSTOME,CUSTOME,PW=XYZ,...)
ATTACH(LOGA,CUSTLOG,PW=XYZ,...)
DFRCV,L=LOGA,A=CUSTOME.
6/7/8/9
```

NOS 1

```
Job Statement
ATTACH(CUSTOME=CUSTOME/PW=XYZ,...)
ATTACH(LOGA=CUSTLOG/PW=XYZ,...)
DFRCV,L=LOGA,A=CUSTOME.
6/7/8/9
```

Example 2

2. Execute restore utility, selecting log records for applications program UPDATE and area EMPLOYE, an IS file, for a specified date and time span. The log file is a permanent file.

NOS/BE 1

```
Job Statement
ATTACH(EMPLOYE,EMPLOYE,PW=XYZ,...)
ATTACH(LOGB,EMPLOG,PW=XYZ,...)
DFRST,L=LOGB,A=EMPLOYE,U=UPDATE,
D=75301/75301,T=120130/164501.
6/7/8/9
```

NOS 1

```
Job Statement
ATTACH(EMPLOYE=EMPLOYE/PW=XYZ,...)
ATTACH(LOGB=EMPLOG/PW=XYZ,...)
DFRST,L=LOGB,A=EMPLOYE,U=UPDATE,
D=75301/75301,T=120130/164501.
6/7/8/9
```

Example 3

3. Execute recover utility using the entire contents of the log file. Log entries for a DA file, PARTSFI, which uses a user hash routine, and an AK file, PRODUCT, are recorded on the log file. The log file is a magnetic tape file.

NOS/BE 1

```
Job Statement
ATTACH(PARTSFI,PARTSFI,PW=XYZ,...)
ATTACH(HASH,PW=XYZ,...)
ATTACH(PRODUCT,PW=XYZ,...)
VSN(TAPE1=1234)
REQUEST(TAPE1)
DFRCV,L=TAPE1,H=DBHASH/HASH.
6/7/8/9
```

NOS 1

```
Job Statement
ATTACH(PARTSFI=PARTSFI/PW=XYZ,...)
ATTACH(HASH/PW=XYZ,...)
ATTACH(PRODUCT/PW=XYZ,...)
VSN(TAPE1=1234)
REQUEST(TAPE1)
DFRCV,L=TAPE1,H=DBHASH/HASH.
6/7/8/9
```

Figure 4-3. Data Base Recovery: Sample Deck Setups

# STANDARD CHARACTER SETS A

CONTROL DATA operating systems offer the following variations of a basic character set:

CDC 64-character set

CDC 63-character set

ASCII 64-character set

ASCII 63-character set

The set in use at a particular installation was specified when the operating system was installed.

Depending on another installation option, the system assumes an input deck has been punched either in 026 or in 029 mode (regardless of the character set in use). Under NOS/BE 1, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of the job statement or any 7/8/9 card. The specified mode remains in effect through the end of the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card.

Under NOS 1, the alternate mode can be specified also by a 26 or 29 punched in columns 79 and 80 of any 6/7/9 card, as described above for a 7/8/9 card. In addition, 026 mode can be specified by a card with 5/7/9 multipunched in column 1, and 029 mode can be specified by a card with 5/7/9 multipunched in column 1 and a 9 punched in column 2.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal type. Characters shown in the CDC Graphic column of the standard character set table are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII-CRT and ASCII-TTY terminals.

TABLE A-1. STANDARD CHARACTER SETS

| Display Code (octal) | CDC Graphic | CDC Hollerith Punch (026) | CDC External BCD Code | ASCII Graphic Subset | ASCII Punch (029) | ASCII Code (octal) |
|---|---|---|---|---|---|---|
| 00† | : (colon)†† | 8-2 | 00 | : (colon)†† | 8-2 | 072 |
| 01 | A | 12-1 | 61 | A | 12-1 | 101 |
| 02 | B | 12-2 | 62 | B | 12-2 | 102 |
| 03 | C | 12-3 | 63 | C | 12-3 | 103 |
| 04 | D | 12-4 | 64 | D | 12-4 | 104 |
| 05 | E | 12-5 | 65 | E | 12-5 | 105 |
| 06 | F | 12-6 | 66 | F | 12-6 | 106 |
| 07 | G | 12-7 | 67 | G | 12-7 | 107 |
| 10 | H | 12-8 | 70 | H | 12-8 | 110 |
| 11 | I | 12-9 | 71 | I | 12-9 | 111 |
| 12 | J | 11-1 | 41 | J | 11-1 | 112 |
| 13 | K | 11-2 | 42 | K | 11-2 | 113 |
| 14 | L | 11-3 | 43 | L | 11-3 | 114 |
| 15 | M | 11-4 | 44 | M | 11-4 | 115 |
| 16 | N | 11-5 | 45 | N | 11-5 | 116 |
| 17 | O | 11-6 | 46 | O | 11-6 | 117 |
| 20 | P | 11-7 | 47 | P | 11-7 | 120 |
| 21 | Q | 11-8 | 50 | Q | 11-8 | 121 |
| 22 | R | 11-9 | 51 | R | 11-9 | 122 |
| 23 | S | 0-2 | 22 | S | 0-2 | 123 |
| 24 | T | 0-3 | 23 | T | 0-3 | 124 |
| 25 | U | 0-4 | 24 | U | 0-4 | 125 |
| 26 | V | 0-5 | 25 | V | 0-5 | 126 |
| 27 | W | 0-6 | 26 | W | 0-6 | 127 |
| 30 | X | 0-7 | 27 | X | 0-7 | 130 |
| 31 | Y | 0-8 | 30 | Y | 0-8 | 131 |
| 32 | Z | 0-9 | 31 | Z | 0-9 | 132 |
| 33 | 0 | 0 | 12 | 0 | 0 | 060 |
| 34 | 1 | 1 | 01 | 1 | 1 | 061 |
| 35 | 2 | 2 | 02 | 2 | 2 | 062 |
| 36 | 3 | 3 | 03 | 3 | 3 | 063 |
| 37 | 4 | 4 | 04 | 4 | 4 | 064 |
| 40 | 5 | 5 | 05 | 5 | 5 | 065 |
| 41 | 6 | 6 | 06 | 6 | 6 | 066 |
| 42 | 7 | 7 | 07 | 7 | 7 | 067 |
| 43 | 8 | 8 | 10 | 8 | 8 | 070 |
| 44 | 9 | 9 | 11 | 9 | 9 | 071 |
| 45 | + | 12 | 60 | + | 12-8-6 | 053 |
| 46 | - | 11 | 40 | - | 11 | 055 |
| 47 | * | 11-8-4 | 54 | * | 11-8-4 | 052 |
| 50 | / | 0-1 | 21 | / | 0-1 | 057 |
| 51 | ( | 0-8-4 | 34 | ( | 12-8-5 | 050 |
| 52 | ) | 12-8-4 | 74 | ) | 11-8-5 | 051 |
| 53 | $ | 11-8-3 | 53 | $ | 11-8-3 | 044 |
| 54 | = | 8-3 | 13 | = | 8-6 | 075 |
| 55 | blank | no punch | 20 | blank | no punch | 040 |
| 56 | , (comma) | 0-8-3 | 33 | , (comma) | 0-8-3 | 064 |
| 57 | . (period) | 12-8-3 | 73 | . (period) | 12-8-3 | 056 |
| 60 | ≡ | 0-8-6 | 36 | # | 8-3 | 043 |
| 61 | [ | 8-7 | 17 | [ | 12-8-2 | 133 |
| 62 | ] | 0-8-2 | 32 | ] | 11-8-2 | 135 |
| 63 | %†† | 8-6 | 16 | %†† | 0-8-4 | 045 |
| 64 | ≠ | 8-4 | 14 | " (quote) | 8-7 | 042 |
| 65 | ↱ | 0-8-5 | 35 | _ (underline) | 0-8-5 | 137 |
| 66 | ∨ | 11-0 or 11-8-2††† | 52 | ! | 12-8-7 or 11-0††† | 041 |
| 67 | ∧ | 0-8-7 | 37 | & | 12 | 046 |
| 70 | ↑ | 11-8-5 | 55 | ' (apostrophe) | 8-5 | 047 |
| 71 | ↓ | 11-8-6 | 56 | ? | 0-8-7 | 077 |
| 72 | < | 12-0 or 12-8-2††† | 72 | < | 12-8-4 or 12-0††† | 074 |
| 73 | > | 11-8-7 | 57 | > | 0-8-6 | 076 |
| 74 | ≤ | 8-5 | 15 | @ | 8-4 | 100 |
| 75 | ≥ | 12-8-5 | 75 | \ | 0-8-2 | 134 |
| 76 | ¬ | 12-8-6 | 76 | − (circumflex) | 11-8-7 | 136 |
| 77 | ; (semicolon) | 12-8-7 | 77 | ; (semicolon) | 11-8-6 | 073 |

†Twelve zero bits at the end of a 60-bit word in a zero byte record are an end of record mark rather than two colons.

††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations yield a blank ($55_8$).

†††The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.

## CDC CHARACTER SET COLLATING SEQUENCE

| Collating Sequence Decimal/Octal | | CDC Graphic | Display Code | External BCD | Collating Sequence Decimal/Octal | | CDC Graphic | Display Code | External BCD |
|---|---|---|---|---|---|---|---|---|---|
| 00 | 00 | blank | 55 | 20 | 32 | 40 | H | 10 | 70 |
| 01 | 01 | ≤ | 74 | 15 | 33 | 41 | I | 11 | 71 |
| 02 | 02 | % | 63 † | 16 † | 34 | 42 | ∨ | 66 | 52 |
| 03 | 03 | [ | 61 | 17 | 35 | 43 | J | 12 | 41 |
| 04 | 04 | → | 65 | 35 | 36 | 44 | K | 13 | 42 |
| 05 | 05 | ≡ | 60 | 36 | 37 | 45 | L | 14 | 43 |
| 06 | 06 | ∧ | 67 | 37 | 38 | 46 | M | 15 | 44 |
| 07 | 07 | ↑ | 70 | 55 | 39 | 47 | N | 16 | 45 |
| 08 | 10 | ↓ | 71 | 56 | 40 | 50 | O | 17 | 46 |
| 09 | 11 | > | 73 | 57 | 41 | 51 | P | 20 | 47 |
| 10 | 12 | ≥ | 75 | 75 | 42 | 52 | Q | 21 | 50 |
| 11 | 13 | ¬ | 76 | 76 | 43 | 53 | R | 22 | 51 |
| 12 | 14 | . | 57 | 73 | 44 | 54 | ] | 62 | 32 |
| 13 | 15 | ) | 52 | 74 | 45 | 55 | S | 23 | 22 |
| 14 | 16 | ; | 77 | 77 | 46 | 56 | T | 24 | 23 |
| 15 | 17 | + | 45 | 60 | 47 | 57 | U | 25 | 24 |
| 16 | 20 | $ | 53 | 53 | 48 | 60 | V | 26 | 25 |
| 17 | 21 | * | 47 | 54 | 49 | 61 | W | 27 | 26 |
| 18 | 22 | - | 46 | 40 | 50 | 62 | X | 30 | 27 |
| 19 | 23 | / | 50 | 21 | 51 | 63 | Y | 31 | 30 |
| 20 | 24 | , | 56 | 33 | 52 | 64 | Z | 32 | 31 |
| 21 | 25 | ( | 51 | 34 | 53 | 65 | : | 00 † | none† |
| 22 | 26 | = | 54 | 13 | 54 | 66 | 0 | 33 | 12 |
| 23 | 27 | ≠ | 64 | 14 | 55 | 67 | 1 | 34 | 01 |
| 24 | 30 | < | 72 | 72 | 56 | 70 | 2 | 35 | 02 |
| 25 | 31 | A | 01 | 61 | 57 | 71 | 3 | 36 | 03 |
| 26 | 32 | B | 02 | 62 | 58 | 72 | 4 | 37 | 04 |
| 27 | 33 | C | 03 | 63 | 59 | 73 | 5 | 40 | 05 |
| 28 | 34 | D | 04 | 64 | 60 | 74 | 6 | 41 | 06 |
| 29 | 35 | E | 05 | 65 | 61 | 75 | 7 | 42 | 07 |
| 30 | 36 | F | 06 | 66 | 62 | 76 | 8 | 43 | 10 |
| 31 | 37 | G | 07 | 67 | 63 | 77 | 9 | 44 | 11 |

†In installations using the 63-graphic set, the % graphic does not exist. The : graphic is display code 63, External BCD code 16.

ASCII CHARACTER SET
COLLATING SEQUENCE

| Collating Sequence Decimal/Octal | | ASCII Graphic Subset | Display Code | ASCII Code | Collating Sequence Decimal/Octal | | ASCII Graphic Subset | Display Code | ASCII Code |
|---|---|---|---|---|---|---|---|---|---|
| 00 | 00 | blank | 55 | 20 | 32 | 40 | @ | 74 | 40 |
| 01 | 01 | ! | 66 | 21 | 33 | 41 | A | 01 | 41 |
| 02 | 02 | " | 64 | 22 | 34 | 42 | B | 02 | 42 |
| 03 | 03 | # | 60 | 23 | 35 | 43 | C | 03 | 43 |
| 04 | 04 | $ | 53 | 24 | 36 | 44 | D | 04 | 44 |
| 05 | 05 | % | 63† | 25 | 37 | 45 | E | 05 | 45 |
| 06 | 06 | & | 67 | 26 | 38 | 46 | F | 06 | 46 |
| 07 | 07 | ' | 70 | 27 | 39 | 47 | G | 07 | 47 |
| 08 | 10 | ( | 51 | 28 | 40 | 50 | H | 10 | 48 |
| 09 | 11 | ) | 52 | 29 | 41 | 51 | I | 11 | 49 |
| 10 | 12 | * | 47 | 2A | 42 | 52 | J | 12 | 4A |
| 11 | 13 | + | 45 | 2B | 43 | 53 | K | 13 | 4B |
| 12 | 14 | , | 56 | 2C | 44 | 54 | L | 14 | 4C |
| 13 | 15 | - | 46 | 2D | 45 | 55 | M | 15 | 4D |
| 14 | 16 | . | 57 | 2E | 46 | 56 | N | 16 | 4E |
| 15 | 17 | / | 50 | 2F | 47 | 57 | O | 17 | 4F |
| 16 | 20 | 0 | 33 | 30 | 48 | 60 | P | 20 | 50 |
| 17 | 21 | 1 | 34 | 31 | 49 | 61 | Q | 21 | 51 |
| 18 | 22 | 2 | 35 | 32 | 50 | 62 | R | 22 | 52 |
| 19 | 23 | 3 | 36 | 33 | 51 | 63 | S | 23 | 53 |
| 20 | 24 | 4 | 37 | 34 | 52 | 64 | T | 24 | 54 |
| 21 | 25 | 5 | 40 | 35 | 53 | 65 | U | 25 | 55 |
| 22 | 26 | 6 | 41 | 36 | 54 | 66 | V | 26 | 56 |
| 23 | 27 | 7 | 42 | 37 | 55 | 67 | W | 27 | 57 |
| 24 | 30 | 8 | 43 | 38 | 56 | 70 | X | 30 | 58 |
| 25 | 31 | 9 | 44 | 39 | 57 | 71 | Y | 31 | 59 |
| 26 | 32 | : | 00† | 3A | 58 | 72 | Z | 32 | 5A |
| 27 | 33 | ; | 77 | 3B | 59 | 73 | [ | 61 | 5B |
| 28 | 34 | < | 72 | 3C | 60 | 74 | \ | 75 | 5C |
| 29 | 35 | = | 54 | 3D | 61 | 75 | ] | 62 | 5D |
| 30 | 36 | > | 73 | 3E | 62 | 76 | ^ | 76 | 5E |
| 31 | 37 | ? | 71 | 3F | 63 | 77 | _ | 65 | 5F |

†In installations using a 63-graphic set, the % graphic does not exist. The : graphic is display code 63.

# DIAGNOSTIC MESSAGES B

Appendix B is composed of messages issued by the log processing module and the recover and restore utilities. Diagnostics written by the log processing module are primarily for problems that relate to CYBER Record Manager and permanent file processing. Recover and restore errors indicate illegal parameters, problems in processing the log file, and CYBER Record Manager errors.

The following codes designate two categories of errors:

| | |
|---|---|
| F | Fatal error; the job is aborted |
| N | Nonfatal error; the job is permitted to continue processing |

Fatal errors encountered for logging or the recovery utilities produce diagnostics that appear in the job dayfile only. All logging errors are considered fatal. An output file designated in the utility control statement holds any diagnostics for nonfatal errors issued by the recovery utilities.

The manner of handling an error encountered during logging depends on the data base management program calling the log processing module. The log processing module issues a message on the job dayfile and returns control to the calling data base management program. Refer to the appropriate reference manual describing the data base management program for details.

Messages are shown in table B-1. One or more x's appearing in a message field signify that the field is replaced by applicable text when the diagnostic is issued at execution time. CYBER Record Manager error codes, issued during execution of the recover and restore utilities, and the log file names are indicated in this manner.

The logging and utility diagnostics are alphabetized and grouped in fatal and nonfatal error categories. A general interpretation of the diagnostic and the corrective action accompany each message listed in the table.

| Type | Utility | Message | Significance |
|---|---|---|---|
| F | Logging | CRM ERROR OPENING LOG FILE LOG FILE=xxxxxxx | Indicated log file cannot be opened by CYBER Record Manager because of the probable omission of required FIT field parameters or inconsistencies in parameters. Refer to the applicable CYBER Record Manager error diagnostic. |
| F | Logging | CRM ERROR WRITING LOG FILE LOG FILE=xxxxxxx | A CYBER Record Manager error occurred while writing a log record to the indicated log file. Refer to the applicable CYBER Record Manager error diagnostic. |
| F | Logging | ERROR EXTENDING LOG FILE LOG FILE=xxxxxxx | A permanent file error occurred while extending the indicated log file. Refer to the applicable permanent file error diagnostic. |
| F | Logging | LOG FILE HAS NOT BEEN OPENED LOG FILE=xxxxxxx | An attempt to write a log record to the indicated log file was not successful because the log file was not open. Notify the systems analyst. |

TABLE B-1. DATA BASE UTILITIES DIAGNOSTICS

| Type | Issued By | Message | Significance | Action |
|---|---|---|---|---|
| F | Logging | CRM ERROR OPENING LOG FILE LOG FILE=xxxxxxx | Indicated log file cannot be opened by CYBER Record Manager because of the probable omission of required FIT field parameters or inconsistencies in parameters. | Refer to the applicable CYBER Record Manager error diagnostic and correct as noted in the CYBER Record Manager reference manuals. |
| F | Logging | CRM ERROR WRITING LOG FILE LOG FILE=xxxxxxx | A CYBER Record Manager error occurred while writing a log record to the indicated log file. | Refer to the applicable CYBER Record Manager error diagnostic and correct as noted in the CYBER Record Manager reference manuals. |
| F | Logging | ERROR EXTENDING LOG FILE LOG FILE=xxxxxxx | A permanent file error occurred while extending the indicated log file. | Refer to the applicable permanent file error diagnostic in the operating system reference manuals. |
| F | Logging | LOG FILE HAS NOT BEEN OPENED LOG FILE=xxxxxxx | An attempt to write a log record to the indicated log file was not successful because the log file was not open. | Notify the systems analyst. |
| F | Logging | LOG FILE NOT A PERMANENT FILE LOG FILE=xxxxxxx | Indicated log file is not a permanent file. A disk file must be initialized as a permanent file before log records can be recorded. | Initialize a disk file as a permanent file for logging. |

TABLE B-1. DATA BASE UTILITIES DIAGNOSTICS (Cont'd)

| Type | Issued By | Message | Significance | Action |
|---|---|---|---|---|
| F | Logging | MAX NO. OF AREAS EXCEEDED LOG FILE=xxxxxxx | More data base areas than can be supported by the logging facility are assigned to the indicated log file. Error results when an attempt is made to write a log record to the log file. | Change the number of areas assigned to the indicated log file. |
| F | Recover Restore | CANNOT READ LOG FILE | A CYBER Record Manager error occurred while reading the log file. | Refer to the applicable CYBER Record Manager error diagnostic and correct as noted in the CYBER Record Manager reference manuals. |
| F | Recover Restore | CHECKPOINT NOT FOUND ON LOG FILE | The checkpoint number specified in the utility control statement is not found on the log file. This error also occurs if a check-point is not within the date and time range specified in the control statement. | Check contents of log file, correct control statement, and resubmit job. |
| F | Recover Restore | DUPLICATE PARAMETER | A utility control statement param-eter cannot appear more than once. | Correct control statement and resubmit job. |
| F | Recover Restore | ERROR OPENING DATA FILE | A CYBER Record Manager error occurred while opening a data base area associated with the log file. | Refer to the applicable CYBER Record Manager error diagnostic and correct as noted in the CYBER Record Manager reference manuals. |

TABLE B-1. DATA BASE UTILITIES DIAGNOSTICS (Cont'd)

| Type | Issued By | Message | Significance | Action |
|---|---|---|---|---|
| F | Recover Restore | ERROR OPENING LOG FILE | A CYBER Record Manager error occurred while opening the log file. | Refer to the applicable CYBER Record Manager error diagnostic and correct as noted in the CYBER Record Manager reference manuals. |
| F | Recover Restore | HASH ROUTINE NOT FOUND | Hash routine specified in the utility control statement cannot be found. | Check hash routine name, correct control statement, and resubmit job. |
| F | Recover Restore | HASH ROUTINE PARAMETER REQUIRES LIBRARY NAME | The hash routine specification in the utility control statement must contain the name of the library in which the hash routine resides. | Correct control statement and resubmit job. |
| F | Recover Restore | ILLEGAL CHECKPOINT NUMBER | Checkpoint number must be an integer in the range from 1 to 2047. | Correct control statement and resubmit job. |
| F | Recover Restore | ILLEGAL DATE | Date parameter included in the control statement is not specified in the correct format. | Correct control statement and resubmit job. |
| F | Recover Restore | ILLEGAL DELIMITER/PARAMETER | An illegal parameter or delimiter is included in the utility control statement. | Correct control statement and resubmit job. |
| F | Recover Restore | ILLEGAL DIRECTIVE CODE ON LOG FILE | The log file contains an unrecognizable directive code. The log record examined might be faulty. The areas to which the log file is assigned cannot be recovered or restored because the log file is unusable. | Load a backup copy of the data base and rerun applicable programs. |

TABLE B-1. DATA BASE UTILITIES DIAGNOSTICS (Cont'd)

| Type | Issued By | Message | Significance | Action |
|---|---|---|---|---|
| F | Recover Restore | ILLEGAL TIME | Time parameter included in the control statement is not specified in the correct format. | Correct control statement and resubmit job. |
| F | Recover Restore | INSUFFICIENT FL FOR EXECUTION | Insufficient field length is available for the recover or restore job to execute. | Request more field length and resubmit job. |
| F | Recover Restore | INTERNAL RECOVER/RESTORE ERROR xxx | An internal error exists within the recover or restore utility. | Notify the systems analyst. |
| F | Recover Restore | LOG FILE LFN MISSING | The log file logical file name must be specified in the utility control statement. | Correct control statement and resubmit job. |
| F | Recover Restore | NO INFORMATION ON LOG FILE | EOI is encountered on the log file before any information is read. | Correct control statement if in error; otherwise, ascertain status of log file. Resubmit job. |
| F | Recover Restore | TIME AND DATE PARAMETERS REQUIRE EACH OTHER | Both time and date parameters must be specified or none at all. | Correct control statement and resubmit job. |
| F | Restore | RESTORE LOGFILE NOT PF WITH EXTEND PERMISSION | Additions cannot be made to the restore log file because extend permission was not granted for the permanent file, or because the restore log file is not a permanent file. | Redefine the restore log file with extend permission, or define it as a permanent file, and resubmit job. |
| F | Restore | CRM ERROR OPENING RESTORE LOGFILE | The restore log file cannot be opened by CYBER Record Manager because of the probable omission of required FIT field parameters or inconsistencies in parameters. | Refer to the applicable CYBER Record Manager error diagnostic and correct as noted in the CYBER Record Manager reference manuals. |

TABLE B-1. DATA BASE UTILITIES DIAGNOSTICS (Cont'd)

| Type | Issued By | Message | Significance | Action |
|---|---|---|---|---|
| F | Restore | CANNOT RESTORE LOG TO INPUT TAPE LOGFILE | If the primary log file is a tape file, it cannot be used as the log file for restore logging. | Specify an independent restore log file in the RL parameter and resubmit job. |
| F | Restore | CRM ERROR WRITING TO LOGFILE | A CYBER Record Manager error occurred while writing a log record to the restore log file. | Refer to the applicable CYBER Record Manager error diagnostic and correct as noted in the CYBER Record Manager reference manuals. |
| F | Restore | ERROR EXTENDING RESTORE LOGFILE | A permanent file error occurred while extending the restore log file. | Refer to the applicable permanent file error diagnostic in the operating system reference manuals. |
| N | Restore | AFTER IMAGE NOT FOUND ON LOG FILE | An after image log record has not been entered for a record with duplicate keys. A record with duplicate keys can be restored only if both before and after image log records are present on the log file. This record is not restored. | If after image logging has not been specified, it must be defined so that any log file created thereafter will contain the required information. |
| N | Recover | BEFORE IMAGE NOT FOUND ON LOG FILE | A before image log record has not been entered for a record with duplicate keys. A record with duplicate keys can be recovered only if both before and after image log records are present on the log file. This record is not recovered. | If before image logging has not been specified, it must be defined so that any log file created thereafter will contain the required information. |

TABLE B-1. DATA BASE UTILITIES DIAGNOSTICS (Cont'd)

| Type | Issued By | Message | Significance | Action |
|---|---|---|---|---|
| N | Recover Restore | CRM ERROR xxx PROCESSING DELETE | A CYBER Record Manager error occurred while performing a DELETE operation on a record in the data base area. The record is not recovered or restored. | Refer to the applicable CYBER Record Manager error diagnostic and correct as noted in the CYBER Record Manager reference manuals. |
| N | Recover Restore | CRM ERROR xxx PROCESSING REWRITE | A CYBER Record Manager error occurred while performing a REWRITE operation on a record in the data base area. The record is not recovered or restored. | Refer to the applicable CYBER Record Manager error diagnostic and correct as noted in the CYBER Record Manager reference manuals. |
| N | Recover Restore | CRM ERROR xxx PROCESSING WRITE | A CYBER Record Manager error occurred while performing a WRITE operation to the data base area. The record is not recovered or restored. | Refer to the applicable CYBER Record Manager error diagnostic and correct as noted in the CYBER Record Manager reference manuals. |
| N | Recover Restore | RECOVER/RESTORE ABORT, JOB REPRIEVED | This informative message is always issued in conjunction with any of the utility diagnostics for fatal CYBER Record Manager errors or for internal recover or restore errors. It indicates that the job was successfully reprieved by the utility. | None. |

# GLOSSARY

ACTUAL KEY – A file organization in which records are identified by system-assigned keys.

ADVANCED ACCESS METHODS (AAM) – A file manager that processes indexed sequential, direct access, and actual key file organizations and supports the Multiple-Index Processor. See CYBER Record Manager.

AFTER IMAGE – A copy of a record in a data base after it has been modified.

AREA – A uniquely named schema data base subdivision that contains data records; identified in the sub-schema as a realm; a file.

ATTACH – The process of making a permanent file accessible to a job by specifying the proper permanent file identification and passwords.

BACKUP DUMP – A copy of all or selected portions of a data base, which is produced on a regularly scheduled basis for the explicit purpose of data base recovery.

BASIC ACCESS METHODS (BAM) – A file manager that processes sequential and word addressable file organizations. See CYBER Record Manager.

BEFORE IMAGE – A copy of a record in a data base after it has been modified.

BLOCKS – The term block has several meanings depending on context. On tape, a block is information between interrecord gaps on tape. CYBER Record Manager defines several blocks depending on organization:

| Organization | Blocks |
|---|---|
| Indexed sequential | Data block; index block |
| Direct access | Home block; overflow block |
| Actual key | Data block |
| Sequential | Block type I, C, K, E |

BOI (Beginning-of-Information) – CYBER Record Manager defines beginning-of-information as the start of the first user record in a file. System-supplied information, such as an index block or control word, does not affect beginning-of-information. Any label on a tape exists prior to beginning-of-information.

CATALOG – The process of making a file a permanent file.

CHECKPOINT/RESTART – A facility that captures the total environment of a job at specified intervals so that a job can be restarted should a system failure interrupt its processing. Records comprising a checkpoint are written to an independent magnetic tape file, not a data base area.

CONTROL WORD – A system-supplied word that precedes each W type record in storage.

CYBER DATABASE CONTROL SYSTEM (CDCS) – The DMS-170 controlling module that provides the interface between the applications program and the data base.

CYBER RECORD MANAGER – A generic term relating to the common products BAM and AAM, which run under the NOS 1 and NOS/BE 1 operating systems and allow a variety of record types, blocking types, and file organizations to be created and accessed. The execution time input/output of COBOL 4, COBOL 5, FORTRAN Extended 4, Sort/Merge 4, ALGOL 4, and the DMS-170 products is implemented through CYBER Record Manager. Neither the input/output of the NOS 1 and NOS/BE 1 operating systems themselves nor any of the system utilities such as COPY or SKIPF is implemented through CYBER Record Manager. All CYBER Record Manager file processing requests ultimately pass through the operating system input/output routines.

DATA ADMINISTRATOR – A person who defines the format and organization of the data base, creates the areas, and is responsible for maintaining and monitoring the data base.

DATA BASE – A systematically organized, central pool of information, whose organization is described by a schema.

DATA BASE PROCEDURE – A special-purpose subroutine written by the data administrator.

It performs an operation that is not part of the CDCS services; specified in the schema and called at execution time by CDCS.

DATA INTEGRITY – Protection of data items within the data base from accidental or deliberate invalidation.

DIRECT ACCESS – A file organization in which records are stored randomly by means of a hashing mechanism.

DIRECT ACCESS FILE – In the context of CYBER Record Manager, a direct access file is one of the five file organizations. It is characterized by the system hashing of the unique key within each file record to distribute records randomly in blocks called home blocks of the file.

In the context of NOS 1 permanent files, a direct access file is a file that is accessed and modified directly, as contrasted with an indirect access permanent file.

DIRECTORY – A file that contains area and record attributes of the data base; created when the schema is compiled; an object schema.

EOI (End-of-Information) – CYBER Record Manager defines end-of-information in terms of the file organization and file residence.

| File Organization | File Residence | Physical Position |
|---|---|---|
| Sequential | Mass storage | After last user record. |
| | Labeled tape in SI, I, S, L format | After last user record and before any file trailer labels. |
| | Unlabeled tape in SI, I format | After last user record and before any file trailer labels. |
| | Unlabeled tape in S or L format | Undefined. |
| Word addressable | Mass storage | After last word allocated to file, which might be beyond the last user record. |
| Indexed Sequential, Actual Key | Mass storage | After record with highest key value. |
| Direct Access | Mass storage | After last record in most recently created overflow block or home block with the highest relative address. |

FILE – A collection of records treated as a unit; called an area in the schema, a realm in the sub-schema.

FILE INFORMATION TABLE (FIT) – A table through which a user program communicates with CYBER Record Manager.

FORM – A general-purpose file management utility for manipulating records and creating and converting files.

HASHING – The process of mapping keys to produce a relative block address for records in a direct access file.

HOME BLOCK – Mass storage allocated for a file with direct access organization at the time the file is created.

INDEXED SEQUENTIAL – A file organization in which records are stored in ascending order by key.

LEVEL – For system-logical-records, an octal number 0 through 17 in the system-supplied 48-bit marker that terminates a short or zero-length PRU.

LOG FILE – An independent permanent file or magnetic tape file (not a data base area) assigned for the purpose of collecting designated information to be used to recover or restore a data base.

LOGICAL FILE NAME (LFN) – The 1 to 7 display code alphabetic or numeric characters by which the operating system recognizes a file. Every LFN in a job must be unique and begin with a letter.

LOGICAL RECORD – Under NOS 1, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. Equivalent to a system-logical-record under NOS/BE 1.

NOISE RECORD – Number of characters the tape drivers discard as being extraneous noise rather than a valid record. Value depends on installation settings.

OUTPUT – A logical file name assigned by the operating system to each job to receive information such as assembly listing, diagnostics, load map, dayfile and program output; printed at job termination unless otherwise disposed by the user. This is the default output file to which nonfatal error diagnostics are written by the recover and restore utilities.

OVERFLOW BLOCK – Mass storage the system adds to a file with direct access organization when records cannot be accommodated in the home block.

PARTITION – CYBER Record Manager defines a partition as a division within a file with sequential organization. Generally, a partition contains several records or sections. Implementation of a partition boundary is affected by file structure and residence.

| Device | RT | BT | Physical Boundary |
|---|---|---|---|
| PRU device | W | I | A short PRU of level 0 containing one-word deleted record pointing back to last I block boundary, followed by a control word with flag indicating partition boundary. |
| | W | C | A short PRU of level 0 containing a control word with a flag indicating partition boundary. |
| | D, F, R, T, U, Z | C | A short PRU of level 0 followed by a zero-length PRU of level 17. |
| S or L format type | W | I | Separate tape block containing as many deleted records of record length 0 as required to exceed noise record size, followed by a deleted one-word record pointing back to the last I block boundary, followed by a control word with flag indicating a partition boundary. |
| | W | C | Separate tape block containing as many deleted records or record length 0 as required to exceed noise record size, followed by a control word with a flag indicating a partition boundary. |
| | D, F, T, R, U, Z | C, K, E | Tapemark |
| | S | – | Zero-length PRU of level number 0. |
| Any other tape format | | | Undefined |

Notice that in a file with W type records, a short PRU of level 0 terminates both a section and a partition.

PERMANENT FILE – A file on a mass storage permanent file device that is protected against accidental destruction by the system and can be protected against unauthorized access or destruction.

PRU – Under NOS 1 and NOS/BE 1, the amount of information transmitted by a single physical operation of a specified device (see table below).

A PRU which is not full of user data is called a short PRU; a PRU that has a level terminator but no user data is called a zero-length PRU.

| Device | Size in Number of 60-Bit Words |
|---|---|
| Mass storage | 64 |
| Tape in SI format with coded data | 128 |
| Tape in SI format with binary data | 512 |
| Tape in I format | 512 |
| Tape in other format | Undefined |

PRU DEVICE – Under NOS 1 and NOS/BE 1, a mass storage device or a tape in SI or I format, so called because records on these devices are written in PRUs.

RANDOM FILE – In the context of CYBER Record Manager, a file with word addressable, indexed sequential, direct access, or actual key organization in which individual records can be accessed by the values of their keys.

In the context of the NOS 1 or NOS/BE 1 operating systems, a file with the random bit set in the file environment table in which individual records are accessed by their relative PRU numbers.

RECORD – CYBER Record Manager defines a record as a group of related characters. A record or a portion thereof is the smallest collection of information passed between CYBER Record Manager and a user program. Eight different record types exist, as defined by the RT field of the file information table.

Other parts of the operating systems and their products might have additional or different definition of records.

RECORD TYPE – The term record type can have one of several meanings, depending on the context. CYBER Record Manager defines eight record types established by an RT field in the file information table. Tables output by the loader are classified as record types such as text, relocatable, or absolute, depending on the first few words of the tables.

RECOVERY – Recreation of all or specified portions of a data base utilizing a backup dump of the data base and the after image entries from the log file.

RESTORATION – Resetting of a data base to a previous state by applying the before image entries from the log file to the data base in its current state.

SCHEMA – A detailed description of the internal structure of the complete data base.

SECTION – CYBER Record Manager defines a section as a division within a file with sequential organization. Generally, a section contains more than one record and is a division within a partition of a file. A section terminates with a physical representation of a section boundary:

| Device | RT | BT | Physical Representation |
|---|---|---|---|
| PRU device | W | I | Deleted one-word record pointing back to last I block boundary followed by a control word with flags indicating a section boundary. At least the control word is in a short PRU of level 0. |
| | W | C | Control word with flags indicating a section boundary. The control word is in a short PRU of level 0. |
| | D, F, R, T, U, Z | C | Short PRU with level less than $17_8$. |
| | S | Any | Undefined. |
| S or L format tape | W | I | A separate tape block containing as many deleted records of record length 0 required to exceed noise record size followed by a deleted one-word record pointing back to last I block boundary followed by a control word with flags indicating a section boundary. |

| Device | RT | BT | Physical Representation |
|---|---|---|---|
| | W | C | A separate tape block containing as many deleted records of record length 0 required to exceed noise record size followed by a control word with flags indicating a section boundary. |
| | D, F, R, T, U, Z | C, K, E | Undefined. |
| | S | Any | Undefined. |
| Any other tape format | | | Undefined. |

The NOS 1 and NOS/BE 1 operating systems equate a section with a system-logical-record of level 0 through $16_8$.

SEQUENTIAL – A file organization in which records are stored in the order in which they are generated.

SHORT PRU – A PRU that does not contain as much user data as the PRU can hold and that is terminated by a system terminator with a level number.

Under NOS 1, a short PRU defines EOR.

Under NOS/BE 1, a short PRU defines the end of a system-logical-record. In the CYBER Record Manager context, a short PRU can have several interpretations depending on the record and blocking types.

SUB-SCHEMA – A detailed description of the portion of the data base to be made available to one or more applications programs.

SYSTEM-LOGICAL-RECORD – Under NOS/BE 1, a data grouping that consists of one or more PRUs terminated by a short PRU or zero-length PRU. These records can be transferred between devices without loss of structure.

Equivalent to a logical record under NOS 1.

Equivalent to a CYBER Record Manager S type record.

TRANSACTION – A record of an input/output request for service from a data base management program providing data management services.

W TYPE RECORD – One of the eight record types supported by CYBER Record Manager. Such records appear in storage preceded by a system-supplied control word. The existence of the control word allows files with sequential organization to have both partition and section boundaries.

ZERO-BYTE TERMINATOR – 12 bits of zero in the low order position of a word that marks the end of the line to be displayed at a terminal or printed on a line printer. The image of cards input through the card reader or terminal also has such a terminator.

ZERO-LENGTH PRU – A PRU that contains system information, but no user data. Under CYBER Record Manager, a zero-length PRU of level 17 is a partition boundary. Under NOS 1, a zero-length PRU defines EOF.

# CONTROL STATEMENT SUMMARY D

The recover and restore utility control statements are summarized in this appendix. Parameter constants are in uppercase and variables are in lowercase; the variables are defined below the illustrated format. In the formats shown, required commas (,), equals (=), and slashes (/) are used to separate fields where applicable. A period terminates the control statement. Section 4 of this manual should be consulted for a thorough discussion of the utility control statements.

| Statement Format | Function |
|---|---|
| DFRCV,L=llfn,A=aid,U=uid,D=[sdate/]edate, T=[stime/]etime,C=ckpt,H=hashrtn/libname,O=olfn. | Calls the recover utility |

| | |
|---|---|
| llfn | Logical file name of log file |
| aid | Area identification contained in the log file |
| uid | User identification contained in the log file |
| sdate | Start-date in year and day format (yyddd) |
| edate | End-date in year and day format (yyddd) |
| stime | Start-time in hour, minute, and second format (hhmmss) |
| etime | End-time in hour, minute. and second format (hhmmss) |
| ckpt | Checkpoint number |
| hashrtn | Entry point name of a hash routinr |
| libname | Name of library containing hash routine |
| olfn | Logical file name of output file for error diagnostics |

| Statement Format | Function |
|---|---|
| DFRST,L=llfn,A=aid,U=uid,D=sdate [/edate], T=stime [/etime] ,C=ckpt,H=hashrtn/libname,O=olfn, RL [=rlfn / =0]. | Calls the restore utility |

| | |
|---|---|
| llfn | Logical file name of log file |
| aid | Area identification contained in the log file |
| uid | User identification contained in the log file |
| sdate | Start-date in year and day format (yyddd) |
| edate | End-date in year and day format (yyddd) |
| stime | Start-time in hour, minute, and second format (hhmmss) |
| etime | End-time in hour, minute, and second format (hhmmss) |
| ckpt | Checkpoint number |
| hashrtn | Entry point name of a hash routine |
| libname | Name of library containing hash routine |
| olfn | Logical file name of output file for error diagnostics |
| rlfn | Logical file name of restore log file |

# FIELD LENGTH REQUIREMENTS

The log processing module is loaded within the user's field length. It requires $2200_8$ words plus space for CYBER Record Manager buffers. Basic field length requirements for CYBER Record Manager are described in the CYBER Record Manager Basic Access Methods and the Advanced Access Methods reference manuals.

A minimum execution field length of $62000_8$ words is required to execute the recover or restore off-line utility. Typical recovery or restoration jobs require about $75000_8$ words. The restore logging facility adds $2000_8$ words to the field length for a data base restoration job using the restore utility.

# INDEX

COMMENT SHEET

CONTROL DATA CORPORATION

TITLE: Data Base Utilities Version 1 Reference Manual

PUBLICATION NO. 60498800 REVISION D

This form is not intended to be used as an order blank. Control Data Corporation solicits your comments about this manual with a view to improving its usefulness in later editions.

Applications for which you use this manual.

Do you find it adequate for your purpose?

What improvements to this manual do you recommend to better serve your purpose?

Note specific errors discovered (please include page number reference).

CUT ON THIS LINE

General comments:

FROM NAME: ______________________ POSITION: ______________________

COMPANY NAME: ______________________

ADDRESS: ______________________

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

**BUSINESS REPLY MAIL**
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

**CONTROL DATA CORPORATION**
*Publications and Graphics Division*
**215 Moffett Park Drive**
**Sunnyvale, California 94086**

CUT ON THIS LINE

FOLD FOLD

STAPLE STAPLE