CRAY RESEARCH, INC.

CRAY X-MP™ AND CRAY-1® COMPUTER SYSTEMS

COS VERSION 1
REFERENCE MANUAL

SR-0011

RECORD OF REVISION CRAY RESEARCH, INC. PUBLICATION NUMBER SR-0011

Each time this manual is revised and reprinted, all changes issued against the previous version are incorporated into the new version and the new version is assigned an alphabetic level.

Every page changed by a reprint with revision has the revision level in the lower righthand corner. Changes to part of a page are noted by a change bar in the margin directly opposite the change. A change bar in the margin opposite the page number indicates that the entire page is new. If the manual is rewritten, the revision level changes but the manual does not contain change bars.

Requests for copies of Cray Research, Inc. publications should be directed to the Distribution Center and comments about these publications should be directed to:

CRAY RESEARCH, INC.
1345 Northland Drive
Mendota Heights, Minnesota 55120

Revision	Description
	June 1976 - Original printing.
A	September 1976 - General technical changes; changes to JOB, MODE, RFL, and DMP statements; names of DS and RETURN changed to ASSIGN and RELEASE. STAGEI deleted, STAGEO replaced by DISPOSE. RECALL macro added and expansions provided for all logical I/O macros. RELEASE, DUMPDS, and LOADPDS renamed to DELETE, PDSDUMP, and PDSLOAD. Detailed description of BUILD added (formerly LIB). EDIT renamed to UPDATE.
B	February 1977 - Addition of Overlay Loader; deletion of Loader Tables (information now documented in CRI publication SR-0012); deletion of UPDATE (information now documented in CRI publication SR-0013); changes to reflect current implementation.
C	July 1977 - Addition of BKSPF, GETPOS, and POSITION logical I/O macros and $BKSPF, $GPOS, and $SPOS routines. Addition of random I/O. Changes to dataset structure, JOB, ASSIGN, MODE, and DUMP statements; BUILD; logical I/O and system action macro expansions. General technical changes to reflect current implementation.

C-01 January 1978 - Correction to DISPOSE and LDR control statement documentation, addition of description of $WWDS write routine, miscellaneous changes to bring documentation into agreement with January 1978 released version of the operating system.

D February 1978 - Reprint with revision. This printing is exactly the same as revision C with the C-01 change packet added.

D-01 April 1978 - Change packet includes the addition of the ADJUST control statement; MODE and SWITCH macros; and PDD, ACCESS, SAVE DELETE, and ADJUST permanent dataset macros. Miscellaneous changes to bring documentation into agreement with released system, version 1.01.

E July 1978 - Complete rewrite. Changes are not marked by change bars. New features for version 1.02 of the operating system that are documented in this revision include: addition of the MODIFY control statement and the DSP, SYSID, and DISPOSE macros; the addition of parameters to some control statements, the implementation of BUILD. The POSITION macro has been renamed SETPOS. Other changes to bring documentation into agreement with released version 1.02 of the operating system.

E-01 October 1978 - Change packet includes the implementation of ACQUIRE and COMPARE control statements; changes to the AUDIT and LDR control statements; changes to the MODE control statement and macro; the addition of control statement continuation, GETPARAM, and the GETMODE macro; and other minor changes to bring documentation into agreement with the released version 1.03 of the operating system.

F December 1978 - Revision F is the same as revision E with change packet E-01 added. No additional changes have been made.

F-01 January 1979 - Change packet includes implementation of some features of BUILD; the addition of the BUFIN, BUFINP, BUFOUT, BUFOUTP, BUFEOF, and BUFEOD macros and other minor changes to bring documentation into agreement with the released version 1.04 of the operating system.

F-02 April 1979 - Change packet includes the implementation of the DEBUG, RERUN, and NORERUN control statements, the RERUN, NORERUN, and BUFCHECK macros; changes to DUMP, DSDUMP, AUDIT, and ASSIGN control statements; implementation of job rerun and memory-resident datasets. Other minor changes were made to bring documentation into agreement with the released version 1.05 of the operating system.

G July 1979 - Reprint with revision. Changes are marked with change bars. The changes bring this documentation into agreement with the released version 1.06 of the operating system. This printing obsoletes all previous versions.

G-01 December 1979 - Change packet includes the implementation of the WAIT and NOWAIT options on the DISPOSE control statement; the addition of a new DUMP format and CFT Linkage Macros; and other minor changes to bring documentation into agreement with the released version 1.07 of the operating system.

H January 1980 - Revision H is the same as revision G with change packet G-01 added. No additional changes have been made.

I April 1980 - Revision I is a complete reprint of this manual. All changes are marked by change bars. New features for version 1.08 of the operating system that are documented in this revision include: the addition of the CALL and RETURN control statements, job classes, the NA parameter on permanent dataset management control statements, the NRLS parameter on the DISPOSE control statement and PDD macro, and the CW parameter on the COMPARE control statement. Changes to the LDR control statement include the addition of the LLD, NA, USA, and I parameters and the new selective load directives. New documentation has been added for unblocked I/O, including descriptions of the READU and WRITEU macros. Other new macros include SETRPV, ENDRPV, DUMPJOB, and the debugging aids SNAP, DUMP, INPUT, OUTPUT, FREAD, FWRITE, UFREAD, UFWRITE, SAVEREGS, and LOADREGS. Documentation on CRAY-1 interactive capabilities and changes to reflect the CRAY-1 S series have also been added. Other changes were made to bring documentation into agreement with released version 1.08 of the operating system.

With this revision, the publication number has been changed from 2240011 to SR-0011.

I-01 October 1980 - Change packet includes the implementation of the IOAREA, SETRPV, ROLL, and INSFUN macros and the IOAREA control statement; the addition of execute-only datasets including adding the EXO parameter to the SAVE and MODIFY control statements and the PDD macro; the lengthening of the TEXT parameter field; the addition of the DEB parameter to the LDR control statement; and a change to the formats of the UFREAD and UFWRITE macros. The DEBUG option allowing conditional execution of the SNAP, DUMP, INPUT, and OUTPUT macros has been implemented. Other minor changes were made to bring documentation into agreement with the released version 1.09 of the operating system.

I-02 July 1981 - This change packet includes changes to Job Control Language syntax; the addition of JCL block control statements for procedure definition (PROC, ENDPROC, &DATA, and prototype statement), conditional processing (IF, ELSE, ELSEIF, and ENDIF), and iterative processing (LOOP, EXITLOOP, and ENDLOOP); the addition of ROLLJOB, SET, LIBRARY, ECHO, PRINT, FLODUMP, and SYSREF control statements; the addition of CSECHO macro; the addition of CNS parameter to CALL statement, REPLACE parameter to BUILD statement, ARGSIZE parameter to ENTER macro, KEEP parameter to EXIT macro, USE parameter to ARGADD macro; the addition of the two JCL tables JBI and JST. Other minor changes were made to bring the documentation into agreement with the released version of 1.10 of the operating system.

J February 1982 - Reprint. This reprint incorporates revision I with change packets I-01 and I-02. No other changes have been made.

J-01 June 1982 - This change packet includes the following additions: magnetic tape characteristics, temporary and local dataset clarification, mass storage permanent datasets, magnetic tape permanent datasets, tape I/O formats, interchange format, transparent format, new accounting information, **gn=nr* parameter, several CHARGES parameters, the OPTION control statement, procedure definition, HOLD parameter, new information to the ACCESS control statement, new tape dataset parameters, tape dataset conversion parameters, SUBMIT job control statement, PDSDUMP and PDSLOAD sample listings, SID parameter on the LDR control statement, new loader errors, relocatable overlays, CONTRPV macro, SUBMIT macro, unrecovered data error information, POSITION macro, new PDD macro parameters, the LDT macro, and new glossary terms. The information formerly in appendix C is now in the COS EXEC/STP/CSP Internal Reference Manual, publication SM-0040. Other miscellaneous technical and editorial changes were made to bring the documentation into agreement with version 1.11 of the operating system.

K July 1982 - Reprint. This reprint incorporates revision J with change packet J-01. No other changes have been made.

L July 1983 - Rewrite. Extensive editorial changes have been made, including moving macro information which was in part 3 to Macro and Opdefs Reference Manual, CRI publication SR-0012. Other major reorganization has occurred. Part 3 now contains job control language structures. Information has been added on interactive job processing and job step abort processing. Major new features documented include enhanced support of tape datasets, the FETCH control statement, memory management, enhancements to COS security, permanent dataset privacy, and support of the CRAY X-MP computer system. Miscellaneous editorial and technical changes have been made to bring the documentation into agreement with version 1.12 of the operating system. This printing obsoletes all previous versions.

L-01 October 1983 - This change packet describes two new ACCOUNT control statement parameters: APW and NAPW. The use of APW and NAPW, and their interrelationship with existing parameters on ACCOUNT, are also explained. A new parameter on the AUDIT control statement, ACC, is described. In addition, illustrative information is provided on how the OWN parameter of the AUDIT utility affects output listings.

L-02 February 1984 - This change packet supports the COS 1.13 release. It includes editorial and technical amendments to information that had been included in previous versions of this manual. The contents reflect new multitasking capabilities. Additional information has been included for coding the CALL statement. New parameters have also been documented in this manual for foreign dataset processing, particularly on the ASSIGN and ACCESS control statements. The LDR statement has been modified considerably; RELEASE, SAVE, MODIFY, DELETE, PERMIT, ACQUIRE, and PDSLOAD also have new parameters. Furthermore, new information is included for managed memory capabilities, the EXITIF control statement block identifier, the COPYU utility for unblocked datasets, and new error codes for reprieve processing.

M December 1984 - This reprint with revision describes many technical changes to COS for the 1.14 release, including contiguous disk allocation and the tape features multitape mark, on-line tape ring processing, partial IBM multifile, special end-of-volume processing, and superblock size. The revision describes software to support four-processor CRAY X-MP computer systems and systems with up to 8 Mwords of memory. Appendix B provides instructions for Subsystem Support: interjob communication, user channel access, and event recall. This revision also documents the Integrated Support Processor (ISP). Note that ISP code will be released later.

This revision contains several format changes. To increase the accuracy of the tables and related information in appendix A, the section is printed as generated by the system. In the body of the manual the "parts" have been removed and the sections numbered consecutively. Material in the four sections of part 3 has been consolidated into one section, 16. This reprint obsoletes all previous editions.

N January 1986 - This reprint with revision brings the manual into agreement with the COS 1.15 release. Technical information added includes documentation of permanent dataset archiving, the HOLD and NOHOLD commands to control an allocated generic resource, changes to resource accounting, and partial support for the IBM 3480 tape subsystem.

There is one significant editorial change: To make information more retrievable, the control statements in sections 7 through 13 now appear in alphabetical order by verb within each section. This reprint obsoletes all previous editions.

O May 1987 - This reprint with revision brings the manual into agreement with the COS 1.16 release. Technical information added includes access of SEGLDR with the new LD2 control statement, the BLOCK and QUERY control statements, the FETCH SF parameter, the RESTORE type parameter, the ASSIGN SPD parameter, and new options for VMS tape files in the ASSIGN and ACCESS RF parameter. Concatenated dataset information has also been added. System error codes have been removed from appendix E. Refer to the COS Message Manual, publication SR-0039, for these messages. Appendix F, which lets you record site-specific information, has been added.

PREFACE

This manual describes the external features of the Cray operating system COS and is intended as a reference document for all users of COS. It deals with three aspects of COS:

- Job processing. Sections 1 through 5 discuss the fundamentals of creating and running jobs on a Cray computer system. These sections describe the system components, storage of information on a Cray computer system, and job processing. They also introduce COS job control and describe the use of libraries.

- Job control statements. Sections 6 through 15 describe each COS job control statement and give the format of each with an explanation of its function.

- Control statement structures. Section 16 describes the control statement block structures available with COS. Examples at the end of the section demonstrate the COS control statement procedure substitution process.

OTHER PUBLICATIONS

Other Cray Research, Inc. (CRI) publications that may be of interest to the reader include the following:

- Products and Utilities

SR-0010	Software Tools Reference Manual
SR-0013	UPDATE Reference Manual
SG-0055	Text Editor (TEDI) User's Guide
SG-0056	Symbolic Interactive Debugger (SID) User's Guide
SR-0066	Segment Loader (SEGLDR) Reference Manual
SR-0073	Cray Simulator (CSIM) Reference Manual
SR-0074	SORT Reference Manual
SR-0112	Symbolic Debugging Package Reference Manual
SR-0146	COS Performance Utilities Reference Manual
SN-0236	Foreign Dataset Conversion on CRAY-1 and CRAY X-MP Computer Systems

- Languages

SR-0000	CAL Assembler Version 1 Reference Manual
SR-0009	Fortran (CFT) Reference Manual
SR-0012	Macros and Opdefs Reference Manual
SR-0018	CFT77 Reference Manual
SR-0060	Pascal Reference Manual
SR-0113	Programmer's Library Reference Manual
SR-2003	CAL Assembler Version 2 Reference Manual
SR-2024	Cray C Reference Manual

- Miscellaneous

SR-0039	COS Message Manual
SI-0154	SUPERLINK/ISP General Information Manual
SI-0178	SUPERLINK/MVS User Guide
SR-0222	CRAY X-MP Multitasking Programmer's Manual

CONVENTIONS

This manual uses the following conventions in presenting control statements:

Convention	Description
italics	Define generic terms representing the words or symbols you supply
[ ] Brackets	Enclose optional portions of a command format
Choice 1 Choice 2	Stacked items indicate two or more literal parameters when only one choice can be used

Numbers are decimal unless otherwise indicated.

CONTENTS

APPENDIX SECTION

FIGURES

FIGURES (continued)

FIGURES (continued)

TABLES

COS 1.16 NEW FEATURES

The 1.16 release of COS includes numerous enhancements of and additions to previous versions of the operating system.

New features include:

- Access of SEGLDR with the new LD2 control statement. LD2 is a new product that has the same interface as does LDR, but it invokes SEGLDR. The purpose of LD2 is to assist users in migrating from LDR to SEGLDR.

- The BLOCK/UNBLOCK control statements. BLOCK and UNBLOCK convert between COS blocked and unblocked dataset formats. In addition to converting datasets containing native Cray data, these utilities interpret and convert between Cray and front-end record structures.

- The QUERY control statement. QUERY returns local mass storage dataset status and position information.

- The TYPE parameter on the RESTORE control statement enables you to select retired and/or migrated datasets.

- The SF parameter for FETCH has been added to $SYSLIB.

- The SPD parameter on the ASSIGN control statement allows striping without system stripe devices.

- The RF parameter on the ACCESS and ASSIGN control statements offers new options for VMS tape files.

- Concatenated datasets. The concatenated dataset feature lets you view logically connected tape datasets as one dataset for the duration of a job step. This feature also provides positioning and rewinding within the same dataset.

INTRODUCTION TO JOB PROCESSING 1

COS is a multiprogramming, multiprocessing, and multitasking operating system for Cray computer systems. It makes efficient use of system resources by monitoring and controlling work presented to the system in the form of jobs. COS optimizes the use of system resources and resolves conflicts when jobs compete for resources.

COS is a collection of programs that reside in either Cray mainframe Central Memory or on system mass storage following startup of the system. (Startup is the process of bringing the Cray computer system and the operating system to an operational state.)

Jobs are submitted to the Cray computer system from one or more front-end computers (also referred to as stations in CRI manuals). Front-end computers can be any of a variety of computer systems. (Software executing on the front-end computer system is beyond the scope of this manual.)

COS provides for the initiation and control of interactive jobs and data transfers between the Cray computer system and users on the front-end system. These features are available only where supported by the front-end system.

1.1 HARDWARE REQUIREMENTS

COS executes on the basic configuration of any CRAY X-MP or CRAY-1 computer system. Each computer system contains the following components:

- One or more central processing units (CPUs)
- Central Memory
- An I/O Subsystem (IOS) or a minicomputer-based maintenance control unit (MCU). The IOS performs all required MCU functions.
- A mass storage subsystem. The mass storage subsystem consists of disk drives, an optional SSD solid-state storage device, and IOS Buffer Memory (BMR).
- An optional IBM-compatible tape subsystem. The tape subsystem requires that an IOS be present.

The IOS consists of from two to four I/O processors (IOPs) and 1/2-, 1-, 2-, 4-, or 8-Mwords of shared BMR. The optional tape subsystem is composed of at least one block multiplexer channel, one tape controller, and two tape units. The tape units supported are IBM-compatible 9-track, 200 ips, 1600 or 6250 bpi devices, and IBM 3480 cartridge drives.

Figure 1-1 shows a basic system configuration.

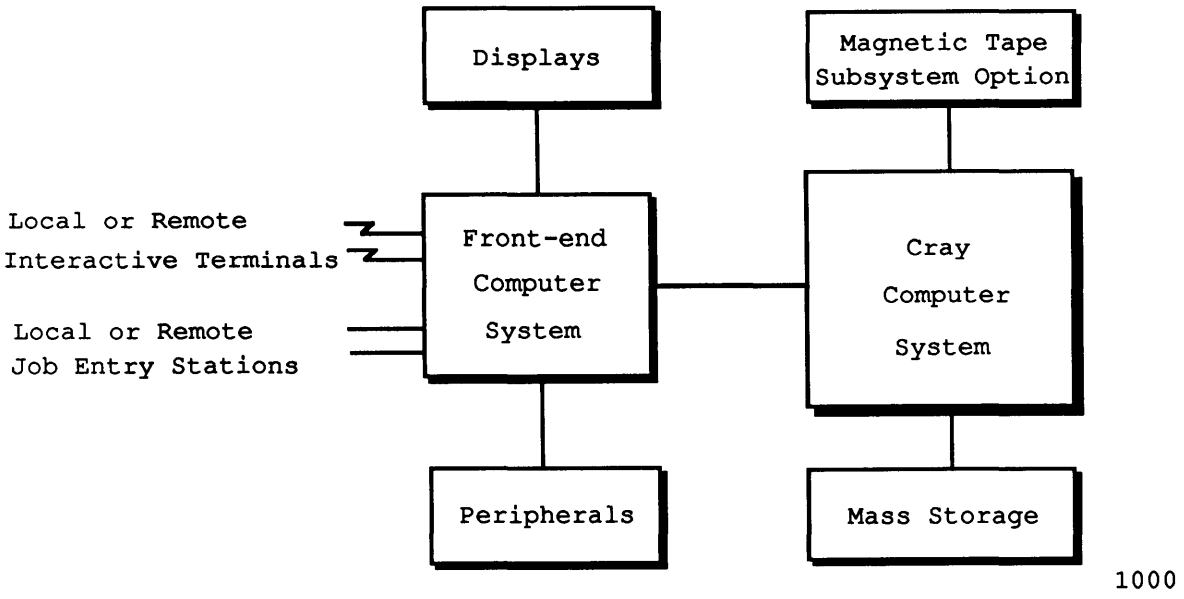


Figure 1-1. Cray Computer System Configuration

1.2 COS STARTUP

COS is loaded into Central Memory and initiated through a system startup procedure performed at the IOS or MCU. At startup, linkage to the Dataset Catalog (DSC) is reestablished on mass storage. All permanent mass storage datasets are recorded in the DSC; thus, permanent datasets survive startup and the user can always assume that they are present. Refer to section 2 for more information on datasets.

1.3 CENTRAL MEMORY ASSIGNMENT AND CHARACTERISTICS

Central Memory is shared by COS, jobs running on the Cray mainframe, dataset I/O buffers, and system tables associated with the jobs. COS allocates the required resources to each job as these resources become available. As a job progresses, information is transferred between Central Memory and mass storage. These transfers can be initiated by either the job or COS.

Figure 1-2 shows the assignment of memory to COS and to jobs.

1.3.1 MEMORY-RESIDENT COS

COS occupies two areas of Central Memory. The memory-resident portion of COS occupying lower memory consists of Exchange Packages, the System Executive (EXEC), the System Task Processor (STP), and optionally the Control Statement Processor (CSP). The memory-resident portion of COS occupying extreme upper memory contains station I/O buffers, space for the system log buffer, and DSC information and buffers.

1.3.2 USER AREA OF MEMORY

COS assigns every job a user area in Central Memory. The user area consists of a Job Table Area (JTA) and a user field.

1.3.2.1 Job Table Area (JTA)

The JTA of each job contains the parameters and information required for monitoring and managing that job. You cannot access the JTA, but it can be dumped for analysis (refer to section 13, Analytical Aids).

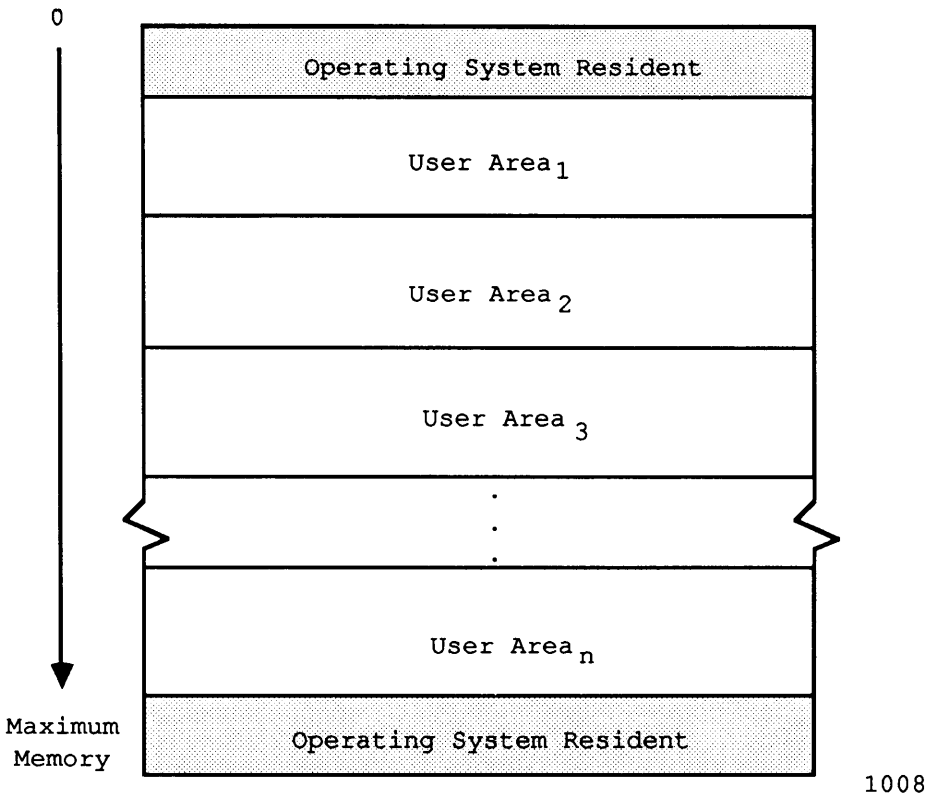


Figure 1-2. Central Memory Assignment

1.3.2.2 User field

The job's user field is a block of memory immediately following the job's JTA and is always a multiple of 512 words. The beginning address [Base Address (BA)] and the end address [Limit Address (LA)] are set by COS. The maximum user field size is specified by a parameter on one of the job control statements (refer to section 6) or by installation-defined default. You can request changes in your user field size while the job is running.

Compilers, assemblers, system utility programs, and user programs are loaded from mass storage into the user field and are executed in response to control statements in the job control statement file. Each load and execution of a program is referred to as a *job step*.

Section 3, COS Job Processing, gives a detailed description of the contents of the user field. Briefly, however, the first 200_8 words of the user field are reserved for an operating system/job communication area known as the Job Communication Block (JCB). Programs are loaded starting at BA+200_8 and reside in the lower portion of the user field. The upper portion of the user field contains tables and dataset I/O buffers. The user field addressing limit is equal to LA-1.

All memory addresses for instructions and operands are relative to BA. The Cray mainframe adds the contents of BA to the address specified by a memory reference instruction to form an absolute address. A user cannot reference memory outside of the user field as defined by the BA and LA register contents; LA-1 is the user limit.

1.4 MASS STORAGE CHARACTERISTICS

All information maintained on mass storage by COS (except specific preallocated areas such as the Device Label) is organized into quantities of information known as datasets. You do not need to concern yourself with the physical transfer of data between disks and memory or with the exact location and physical form in which datasets are maintained on mass storage. COS translates your logical requests for data input and output into disk controller functions automatically.

Each disk storage unit (DSU) contains a Device Label, datasets, and unused space to be allocated to datasets. The Device Label lists unusable (flawed) space on the DSU and indicates which DSU is the Master Device. The Master Device is the DSU that contains the DSC table. The DSC table contains information needed to maintain permanent datasets.

Mass storage permanent datasets are always present and available. This permanence is achieved with techniques that permit the datasets listed in the DSC to be recovered or reestablished if a system failure occurs. Portions of COS (such as loaders, utility programs, compilers, assemblers, and library maintenance and generation routines) reside on mass storage devices as permanent datasets accessible by user jobs at any time.

Job input and output datasets also reside on mass storage and are listed in the DSC. Because they are listed in the DSC, they are also regarded as permanent. This designation is somewhat misleading because their permanence is by definition not status. The input dataset is "permanent" from the time it is staged from the front-end system to the Cray computer

system until the job terminates. Output datasets being disposed to a front end are "permanent" from job termination (or whenever the disposition was initiated) until the disposition is complete. The "permanent" status of these system-defined datasets allows them to be recovered (along with other permanent datasets) after a system failure.

Any job can create a mass storage permanent dataset that can be subsequently accessed, modified, or deleted by any other job that has the correct access privileges and produces the correct permission control words. Permission control words are defined at the time the dataset is designated as permanent (that is, saved).

A permanent dataset can be deleted by any user with the correct permission control word. Deleting a dataset notifies COS that the space occupied by the dataset is no longer permanent. However, the space is still reserved by the dataset until you release it. (Refer to sections 8 and 10, respectively, for information on the RELEASE and DISPOSE control statements.)

In addition to permanent datasets, mass storage is used for *temporary datasets*. Temporary datasets are created by a job and remain temporary unless designated permanent, released, or disposed to a front end by the job. A temporary dataset that is not saved or disposed is termed a *scratch dataset* and is deleted when the job releases it or when the job terminates.

COS allocates space to a mass storage dataset by disk tracks. The space assigned to a single dataset can be noncontiguous and can even be on several different disk units. Both default and maximum size limits for datasets are defined by system parameters. Using the ASSIGN control statement, you have limited control of how mass storage is allocated to a dataset.

1.5 MAGNETIC TAPE CHARACTERISTICS

An IOS can include an Auxiliary I/O Processor (XIOP) with the capability of addressing up to 16 block multiplexer channels of tape units. Each block multiplexer channel can be attached to IBM-compatible control units and tape units in a variety of configurations. The block multiplexer channels communicate with the control units and tape units to allow reading and writing data that can also be read and written by IBM-compatible CPUs. Table 1-1 summarizes the physical characteristics of 200 ips, 9-track tape drives, and IBM 3480 cartridge drives. The block sizes in this table are used by the COS tape system for transparent-format tape datasets (described in section 2).

Table 1-1. Physical Characteristics of Tape Devices

Device	Density (Bits/In)	Transfer Rate (Kbytes/s)	Data/2400-ft Reel Equiv. (Mbytes)	Block Size (Bytes)
Reel-to-reel	6250	1170	168	32768
Reel-to-reel	1600	300	43	16384
Cartridge	N/A	2700†	200	32768

† Data-streaming mode

DATASETS 2

Nearly all information maintained by COS is organized as *datasets*. COS supports blocked and unblocked, interactive and tape (interchange and transparent) format dataset structures. Some important factors to remember about datasets are the following:

- Dataset *medium* is the type of physical device on which the dataset resides.
- Dataset *structure* is the logical organization of the dataset.
- Dataset *longevity* is the retention period for the dataset.
- Datasets must be *local* to the job to be usable.
- The dataset *disposition code* tells the operating system what action to take when the dataset is no longer local.
- Each dataset is known by its *dataset name*.
- Datasets are read and written using operating system requests (user I/O interfaces).

2.1 DATASET MEDIA

Datasets are often classified by medium. COS uses the classifications to identify the various types of datasets.

- Mass storage datasets
- Memory-resident datasets
- Interactive datasets
- Magnetic tape datasets
- Integrated Support Processor (ISP) datasets

2.1.1 MASS STORAGE DATASETS

Mass storage datasets are those that reside on Cray mass storage devices; that is, on mass storage devices attached directly to the mainframe or to the I/O Subsystem (IOS).

2.1.2 MEMORY-RESIDENT DATASETS

Datasets classified as memory-resident are those you specify to be kept in memory and are typically temporary datasets. A *memory-resident dataset* is wholly contained within one buffer (refer to the BS parameter on the ASSIGN control statement in section 8) and remains in memory at all times. Such a dataset ordinarily occupies no mass storage. A memory-resident dataset is normally a temporary dataset; however, a mass storage permanent dataset can be declared memory resident.

A memory-resident dataset is defined through an ASSIGN control statement containing the MR parameter or through an F$DNT (described later) call to the system. If the F$DNT call is used, the Dataset Definition List (DDL) supplied should specify DDMR=1. (Refer to the description of the ASSIGN control statement in section 8.) In addition, the buffer size parameter on the ASSIGN control statement should specify a buffer large enough to contain the entire dataset plus one block.

A dataset can be declared memory resident to reduce the number of I/O requests and disk blocks transferred. Memory residence is particularly useful for intermediate datasets not intended to be saved or disposed to another mainframe. All I/O performed on a memory-resident dataset occurs in the dataset buffers in memory and the contents of the buffers are not ordinarily written to mass storage. Such a dataset can neither be made permanent, nor may it be disposed to another mainframe, unless copied to mass storage.

If at any time the system I/O routines are called to write to the dataset and the buffer appears to be full, the dataset ceases to be treated as memory resident, the buffer is flushed to mass storage, and all memory-resident indicators for the dataset are cleared.

Normally, a memory-resident dataset is empty until written on. If an existing dataset is declared memory resident, it is loaded when the first read occurs. A user attempting to write to a memory-resident dataset must have write permission. As long as the buffer does not appear full, however, no actual write to mass storage ever occurs. Therefore, changes made to an existing dataset declared memory resident are not reflected on the mass storage copy of the dataset.

Magnetic-tape datasets, mass storage execute-only datasets, and interactive datasets cannot be declared memory resident.

2.1.3 INTERACTIVE DATASETS

Interactive datasets are those specified as such by interactive jobs. Interactive datasets are supported by the front-end station. Batch users cannot create interactive datasets.

An interactive dataset differs from other datasets in that a physical image of the dataset is not maintained. Instead, records are transmitted to and from your terminal attached to the front-end station. Record positioning (for example, REWIND or BACKSPACE) is not possible.

Interactive datasets are created by interactive jobs through the use of the ASSIGN control statement or F$DNT system call.

2.1.4 MAGNETIC TAPE DATASETS

Magnetic tape datasets are available to any job that declares tape resource requirements on the JOB control statement and specifies the appropriate information on its ACCESS control statement. Refer to the ACCESS control statement description in section 9 for more details.

COS automatically switches volumes† during dataset processing unless user end-of-volume (EOV) processing (defined later) is requested, and returns to the first volume of a multivolume dataset in response to a REWIND control statement. If a permanent write error occurs when trying to write a tape block for the user, COS automatically attempts to close the current volume. If the attempt succeeds, the system continues to the next volume.

The COS tape system uses Buffer Memory (BMR) as a tape block buffering area so that the job's I/O buffer need not be as large as the tape block. This technique results in significant memory savings whenever large tape blocks are processed and increases transfer rates when smaller blocks are processed. The advantage in having a large I/O buffer is a reduction in the overhead in the tape subsystem.

This subsection discusses the following aspects of using tape datasets:

- Gaining access to a tape dataset
- Bypass label processing
- User tape end-of-volume processing
- Tape mark processing
- Multidataset access

2.1.4.1 Gaining access to a tape dataset

To gain access to an existing permanent tape dataset to read or rewrite or both, you must specify the file identifier (permanent dataset name), the desired device type, and, optionally, a volume identifier (VOL) list. The volume identifier list can consist of from 1 to 255 volume

† In this context, the term "volume" means a reel of magnetic tape.

identifiers. If the permanent dataset name (PDN) is omitted from the ACCESS control statement, the local dataset name is used as the file identifier.

To create a tape dataset, the file identifier, the desired device type, and the NEW parameter option must be specified on the ACCESS control statement. If no file identifier is present, the local dataset name is used. If a volume identifier list is not specified on the ACCESS control statement, it is a *nonspecific volume allocation* (scratch tape). A *specific volume allocation* occurs when a volume identifier list is specified on the ACCESS request. COS records the volume label on the tape. Like all other physical datasets, new tape datasets must be written to before a read is allowed.

More than one tape ACCESS control statement with the same dataset name, but a different permanent dataset name, will activate concatenation. Refer to the Concatenated Datasets subsection for more information on concatenated datasets.

2.1.4.2 Bypass label processing

Bypass label processing is a COS option controlled by the installation parameter I@BPL that lets you bypass a tape's label by declaring BP as a label type on the ACCESS control statement. Bypass label processing is not supported for transparent datasets.

Normally, tape labels are scanned during the beginning of tape processing and at the end-of-data (EOD) and volume processing. This label processing is not performed when bypass label processing is operative. When the tape is mounted, the tape subsystem positions it at the beginning-of-tape (BOT). The first I/O request (read or write) begins at this point. If tape labels are present, you must take them into consideration. Your job can read an existing label, overwrite it, or position past it. A tape is treated as a nonlabeled tape with embedded tape marks while bypass label processing is in effect if BP is the label type specified for the LB parameter on the ACCESS control statement.

If system security (I@SLVL) is in warning or full mode, bypass label processing is a privileged operation; otherwise, any user may request it.

2.1.4.3 User tape end-of-volume processing

The tape end-of-volume (EOV) feature, which may be used only by interchange format tapes, uses special processing system macros to allow you to gain control at tape EOV and perform special EOV and beginning-of-volume (BOV) processing. The special processing macros used, SETSP, STARTSP, ENDSP, TAPESTAT, and CLOSEV, affect individual datasets. If EOV processing is needed for more than one dataset, the macros must be issued for each tape dataset. Refer to the Macros and Opdefs Reference Manual, CRI publication SR-0012, for more information.

You instruct the system to perform EOV processing by issuing the SETSP macro (with the ON option) after a tape dataset is opened. Using SETSP with the OFF option informs the system that EOV processing is no longer needed. The CLOSE macro also terminates EOV special processing.

To test whether the tape dataset is at EOV, you must use the TAPESTAT macro after every READ, WRITE, and SYNCH macro. Not all macros that result in I/O operations return EOV status; for example, the CLOSE, POSITION, and REWIND macros do not return EOV status. For output datasets, you should use the SYNCH macro to flush the buffers and determine if EOV has been encountered before using such macros.

After EOV is encountered, you can start EOV processing by issuing the STARTSP macro. During EOV processing, you can execute read, write, and position operations. Volume switching is done by issuing the CLOSEV macro. When EOV processing is complete, the ENDSP macro notifies the system to return to normal processing.

During EOV processing, no read ahead is performed. Data blocks are read one at a time. Also, any position request with a relative block number is positioned from the current physical tape position. For output datasets, the physical and logical tape positions will differ because the last few blocks written will still be in the IOP buffer. The TAPEPOS macro lets your program determine how many blocks are buffered in the IOP.

For an output dataset, the data in the IOP buffer when EOV is encountered is considered part of the dataset and may be read during EOV processing. Once any of this data is read, it is no longer part of output data. Because no read ahead is performed during EOV processing, the program may position backwards and read *only* the blocks on the tape. If this is done, the data in the IOP buffer is kept intact, and it will be written to tape when the ENDSP macro is issued.

The use of the CLOSEV macro is not restricted to the EOV routine. You can issue the CLOSEV macro anytime during dataset processing. This macro lets you terminate an output tape anywhere and continue the dataset on the next tape. It also lets you read part of a tape and switch to the following tape.

2.1.4.4 Tape mark processing

Three label types are available that allow tape marks to be embedded in the data. These "field" formats are field ANSI labels (FAL), field standard IBM labels (FSL), and field nonlabeled (FNL). On output, a tape mark is created by a write EOF operation. On input, a tape mark is translated to an EOF.

Field format tapes cannot be used with the transparent recording format.

With these label types, when COS recognizes a tape mark, it translates it to an end-of-file (EOF) record control word and puts it in the data. You are responsible for recognizing EOF conditions.

An attempt to position past a tape mark (using the POSITION macro) results in the following actions: The tape moves forward until the tape mark is encountered. At that point, tape movement stops and you get control. A residual record count is returned to find the position on tape and the tape is physically positioned after the tape mark just encountered.

For input, all field format tapes (FAL/FNL/FSL) are processed for labels in the same way. If a label is encountered at BOT, it is validated based on its type. If no label is found, there is no validation. When a tapemark is detected, COS checks the next record for an EOV1 or EOF1 trailer label. If EOV1 is found, COS performs an automatic volume switch. If EOF1 is found, COS performs EOD processing. If neither EOV1 or EOF1 is encountered, the tape is left positioned immediately following the tape mark ready for the next read. Labeled tapes not terminated with either SL or AL standard labels must be terminated by the program using CLOSE or CLOSEV system calls.

For output, field format tapes are labeled based on the LB parameter on the ACCESS control statement. EOV labels are processed when either the EOT reflective marker is sensed or when the user program calls CLOSEV. EOF labels are written when the dataset is closed, rewound, or released.

2.1.4.5 Multidataset access

The user job can access more than one dataset on a tape labeled AL, SL, or NL. The FSEQ parameter on the ACCESS control statement identifies the accessible dataset. FSEQ=1 accesses the first dataset, FSEQ=2 accesses the second, and so on. Table 2-1 details the tape formats.

During ACCESS processing, the system requests a volume mount if the volume needed is not currently mounted. Only one dataset can be opened at a time on a volume.

During CLOSE processing, a volume remains loaded if it is the first volume of another dataset in the same job. If it is not the first volume of another dataset, the volume is unloaded.

During RELEASE processing, if the volume has not been unloaded, it remains loaded until no more datasets require the volume.

The examples that follow show possible arrangements of ACCESS, OPEN, CLOSE, and RELEASE processing.

Table 2-1. Tape Formats for Multidataset Access

AL and SL Single Volume Tapes	AL and SL Multivolume Tapes		NL Single Volume and Multivolume Tapes
	First Volume	Subsequent Volumes	
VOL1	VOL1	VOL1	DATA BLOCKS
HDR1	HDR1	HDR1	.
HDR2†	HDR2†	HDR2†	.
*	*	*	.
DATA BLOCKS	DATA BLOCKS	DATA BLOCKS	*
*	*	*	DATA BLOCKS
EOF1	EOF1	EOF1	.
EOF2†	EOF2†	EOF2†	.
*	*	*	.
HDR1	HDR1	HDR1	*
HDR2†	HDR2†	HDR2†	DATA BLOCKS
*	*	*	.
DATA BLOCKS	DATA BLOCKS	DATA BLOCKS	.
*	*	*	.
EOF1	EOF1	EOF1	*
EOF2†	EOF2†	EOF2†	*
*	*	*	
.	.	*	
.	.		
.	.		
*	*		
HDR1	HDR1		
HDR2†	HDR2†		
*	*		
DATA BLOCKS	DATA BLOCKS		
*	*		
EOF1	EOV1		
EOF2†	EOV2†		
*	*		
*	*		

† HDR2, EOF2, and EOV2 are written by COS, however, their presence is optional on tapes created by other computer systems.

* = Tapemark

Example 1:

This job uses two datasets on volume TAPE1 and reserves one tape drive. The order of processing does not have to be the same as the order of access.

```
JOB,JN=...,*TAPE=1.
ACCESS,DN=A,FSEQ=2,VOL=TAPE1.
ACCESS,DN=B,FSEQ=1,VOL=TAPE1.
  .
  .
  .
RELEASE,DN=A.
RELEASE,DN=B.
/EOF
                In the user program . . .
                  Open B, process B, close B
                  Open A, process A, close A
```

Example 2:

This job uses two datasets which are contained on three volumes and reserves one tape drive. The order of processing does not have to be the same as the order of access.

```
JOB,JN=...,*TAPE=1.
ACCESS,DN=A,FSEQ=2,VOL=TAPE1:TAPE2:TAPE3
ACCESS,DN=B,FSEQ=1,VOL=TAPE1:TAPE2:TAPE3.
  .
  .
  .
RELEASE,DN=A.
RELEASE,DN=B.
/EOF
                In the user program . . .
                  Open A, process A, close A
                  Open B, process B, close B
```

2.1.4.6 Concatenated datasets

The concatenated dataset feature lets your job logically connect a group of tape datasets for the duration of your job. The job treats the connected datasets as one. Concatenation is activated when more than one tape dataset with the same local dataset name (DN= parameter on the ACCESS control statement) is encountered. Each dataset must have its own ACCESS control statement. This example is for tapes with like blocksize and recordsize.

Examples:

```
ACCESS,DN=F1,PDN=ABC,VOL=T03461.
ACCESS,DN=F1,PDN=DEF,VOL=T03462.
```

Datasets with different record sizes but the same blocksize can be specified as follows:

```
ACCESS,DN=F1,PDN=ABC,RS=80,VOL=T03461.
ACCESS,DN=F1,PDN=DEF,RS=100,VOL=T03462.
```

The Front End Tape Management Catalog cannot be used with concatenated datasets.

A mixture of tapes ending with EOV or EOF is allowed. End-of-information is not returned to your program until all of the tapes accessed with the same local dataset name (DN=) have been read.

2.1.5 INTEGRATED SUPPORT PROCESSOR (ISP) DATASETS

An ISP dataset resides on another mainframe that communicates with COS using ISP software. COS and the ISP software function together to give the COS user access to the remote dataset as if it resides on a device directly attached to the Cray computer system. ISP datasets are accessible through the ISP and CONNECT control statements. Refer to the SUPERLINK/ISP General Information Manual, CRI publication SI-0154, or the SUPERLINK/MVS User Guide, CRI publication SI-0178, for further information on the ISP.

2.2 DATASET FORMATS

Dataset formats include blocked, unblocked, interactive, and tape. These are described in the sections that follow.

2.2.1 BLOCKED FORMAT

Blocked format is the default format for external types of datasets such as user input and output datasets. A blocked dataset is usually composed of one or more files, which are, in turn, composed of one or more records. Figure 2-1 shows the data hierarchy within a blocked dataset.

Data in a blocked dataset can be ASCII character, binary, or both. Blanks are normally compressed in blocked coded datasets. Each block consists of 512 words. Blocked datasets use two types of control words: block and record.

Record positioning requires a blocked format. The blocked format adds control words to the data to allow for processing of variable-length records and to allow for delimiting of levels of data within a dataset.

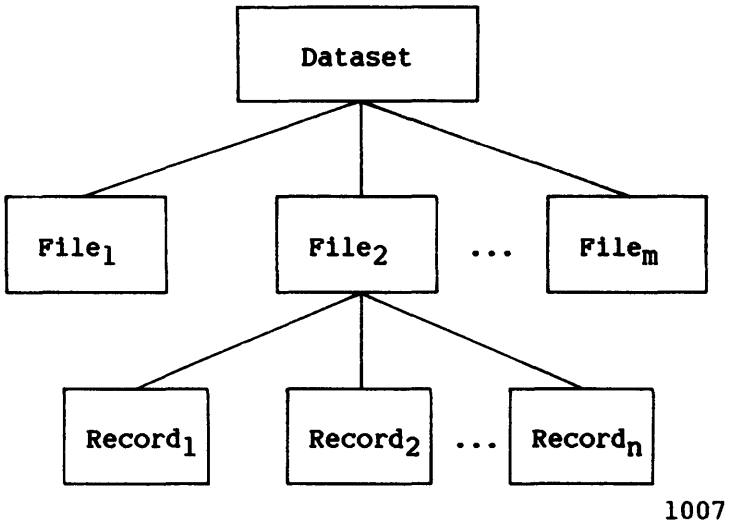


Figure 2-1. Data Hierarchy Within a Blocked Dataset

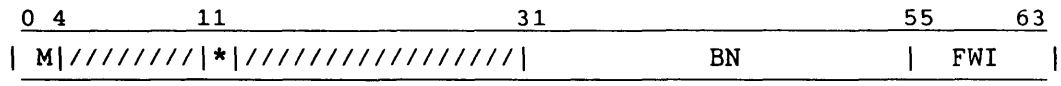
2.2.1.1 Blank compression

Blank fields can be compressed in files containing only ASCII characters. Blank field compression is indicated by a blank-field initiator code followed by a count. The default blank-field initiator code is defined by the installation parameter I@BFI, which is either an ASCII code or 777_8 indicating that blank compression will not be done. Blank compression can be inhibited using an ASSIGN statement parameter or an F$DNT system call. A blank field (3 to 96 characters) is compressed to a 2-character field. The count is biased by 36_8; the actual character count is limited to $41_8 \leq$ *character count* $\leq 176_8$ (the ASCII graphics).

2.2.1.2 Block control word

The block control word (BCW) is the first word of every 512-word block.

Format:

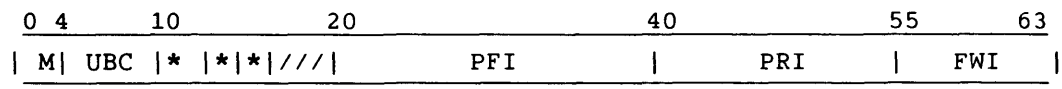


Field	Bits	Description
M	0-3	Type of control word (for BCW, M=0)
BDF	11	Bad Data flag; indicates that the following data, up to the next control word, is bad. This flag is set by the IOS for magnetic tape datasets in interchange format.
BN	31-54	Block number; designates the number of the current data block. The first block in a dataset is block 0.
FWI	55-63	Forward index; designates the number of words (starting with 0) to the next record control word (RCW) or BCW.

2.2.1.3 Record control word

A record control word (RCW) occurs at the end of each record, file, or dataset.

Format:



Field	Bits	Description
M	0-3	Type of control word: 10_8 End-of-record (EOR) 16_8 End-of-file (EOF) 17_8 End-of-data (EOD)
UBC	4-9	Unused bit count. For EOR, UBC designates the number of unused low-order bits in the last data word of the record terminated by the EOR. For EOF and EOD RCWs, this field is 0. The data area protected by UBC must be zero-filled.

Field	Bits	Description
TRAN	10	Transparent record field; used for an interactive output dataset only. If set, substitution of EOR RCWs is suppressed.
BDF	11	Bad Data flag; indicates the following data, up to the next control word, is bad. This flag is set by the IOS for magnetic tape datasets in interchange format. If flag is set, an irrecoverable error was encountered in the following data.
SRS	12	Skip remainder of sector; indicates that the next control word to follow is a BCW and the data after this RCW is not to be processed. This is used only in tape dataset processing.
PFI	20-39	Previous file index; this field contains an index modulo 2^{20} ($20,000,000_8$) to the beginning of the file; the index is relative to the current block such that if the beginning of the file is in the same block as this RCW, the PFI is 0.
PRI	40-54	Previous record index; this field contains an index modulo 2^{15} ($100,000_8$) to the block where the current record starts. The index is relative to the current block such that if the first word of data in this record is in the same block as this RCW, PRI is 0.
FWI	55-63	Forward word index (FWI); this field points to the next control word (RCW or BCW) and consists of a count of the number of data words up to the control word (that is, if the next word is an RCW or BCW, FWI is 0).

Disregarding BCWs occurring at 512-word intervals in a dataset, RCWs have the following logical relationship in a dataset.

An EOR RCW immediately follows the data for the record it terminates. If the record is null (contains no data), an EOR RCW can immediately follow an EOR or EOF RCW, or it can be the first word of the dataset.

An EOF RCW immediately follows the EOR RCW for the final record in a file. If the file is null (contains no records), the EOF RCW can immediately follow an EOF RCW, or it can be the first word of the dataset.

An EOD RCW immediately follows the EOF RCW for the final file in the dataset. If the dataset is null, the EOD RCW can be the first word on the dataset.

A typical dataset has many EOR RCWs per block. Figure 2-2 shows an example of dataset control words. In this example, a dataset is contained on four physical disk sectors, each beginning with a BCW (thus the four BCWs in this example are numbered 0, 1, 2, and 3). The dataset contains four files shown as F1, F2, F3, and F4. F1 contains the four records shown as R1 through R4, F2 contains records R5 through R7, F3 contains no records at all, and F4 contains record R8.

2.2.2 UNBLOCKED FORMAT

Dataset I/O can also be performed using unblocked datasets. The data stream for unblocked datasets does not contain COS RCWs or BCWs.

COS does not allocate buffers for unblocked datasets in the job's I/O buffer area. You must specify an area for data transfer. When a read or write is performed on an unblocked dataset, the data goes directly to or from the user data area without passing through I/O buffers. The word count for data to be transferred must be a multiple of 512.

Unblocked I/O cannot be performed on an interchange format tape dataset.

2.2.3 INTERACTIVE FORMAT

Interactive format closely resembles blocked format; however, each buffer begins with a block 0 BCW. Each record transmitted to or from COS by an F$RDC or an F$WDC call must contain a single record consisting of a BCW, data, and an EOR RCW.

Either of two formats for interactive output can be assigned when the dataset is created: character blocked or transparent. Character blocked mode is the default. In character blocked mode, an EOR RCW is interpreted as a line feed or a carriage return. In transparent mode, the EOR RCW is ignored and you supply carriage control characters.

2.2.4 TAPE FORMAT

Tape datasets are written to and read from tape volumes. A *tape volume* is a reel of tape. A tape volume is also known as a *dataset section* (for example, in FSEC= on the ACCESS control statement).

Data is read or written in tape blocks. A *tape block* is a unit of data recorded on magnetic tape between two consecutive interblock gaps.

Tape datasets can be read or written using two different formats: *interchange* or *transparent*. Tape datasets can also be labeled or unlabeled.

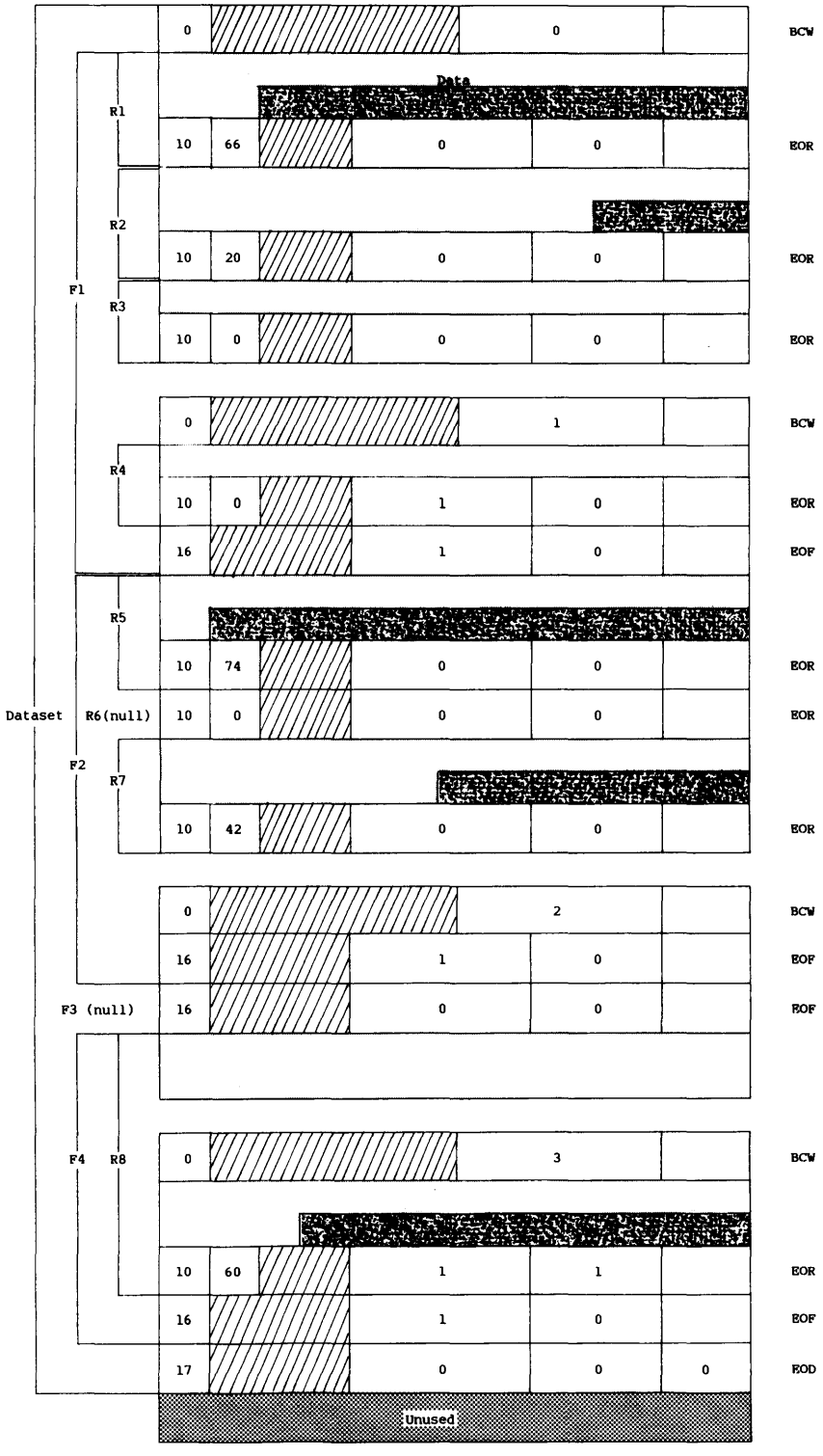


Figure 2-2. Example of Dataset Control Words
(Octal values shown)

2.2.4.1 Interchange format

Interchange format is useful for reading and writing tapes that are also to be read or written on other vendors' systems. In *interchange format*, each tape block corresponds to a single logical record in COS blocked format (that is, the data between RCWs).

In interchange format, tape block lengths can vary from one byte up to an installation-defined maximum, which cannot exceed 1,048,576 bytes (131,072 64-bit words). In general, the maximum block size should not exceed 200 kilobytes. Blocks exceeding this size may require special operational procedures, such as the use of specially prepared tape volumes having an extended length of tape following the end-of-tape (EOT) reflective marker and yield little increase in transfer rates or storage capacity.

When a tape dataset is read in interchange mode, physical tape blocks are represented in the user's I/O buffer with BCWs and RCWs added by COS. The data in each tape block is terminated by an RCW. The unused bit count field in the RCW indicates the amount of data in the last word of the tape block that is not valid data. A BCW is inserted before every 511 words of data, including the RCWs. The formats for RCWs and BCWs were described earlier.

Figure 2-3 shows a tape dataset in interchange format. Tape blocks within tape label groups are not included in this format. The end of the dataset is represented by an EOF RCW followed by an EOD RCW.

Multifile datasets are supported in interchange format by field label (FAL, FSL, and FNL) and BP label tapes.

2.2.4.2 Transparent format

In *transparent format* (disk image), each tape block is a fixed multiple of 512 words, generally based on the dataset density (that is, 16,384 bytes at 1600 b/i and 32,768 bytes at 6250 b/i). The data in the tape block is transferred unaltered between the tape and the I/O buffer in the user field; no control words are added on reading or discarded on writing. In transparent mode, the data can be in COS blocked or unblocked format. Transparent format tapes are not generally read or written by other vendors' equipment.

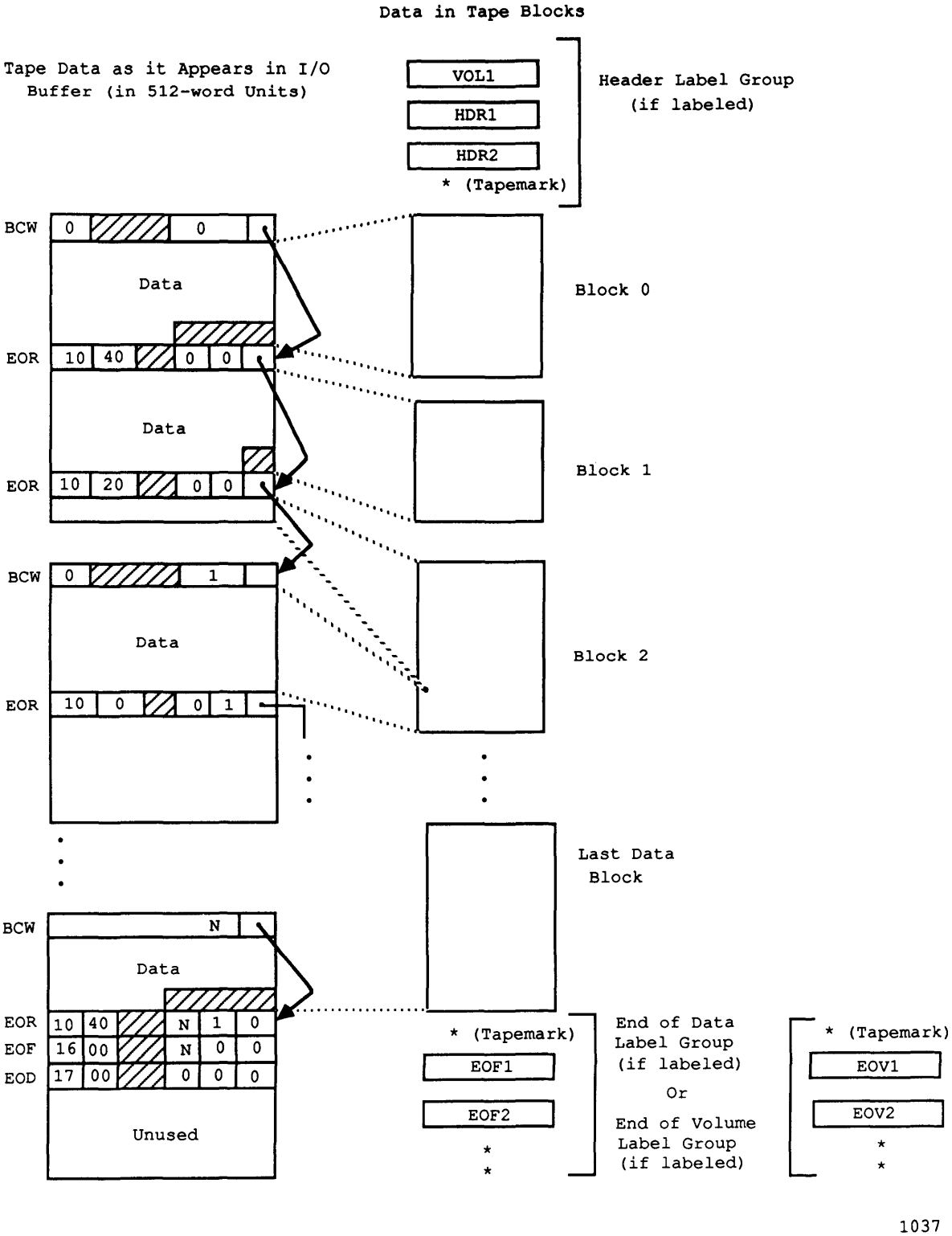


Figure 2-3. Interchange-format Tape Dataset (Octal values shown)

2.3 DATASET LONGEVITY

Permanent datasets are retained by COS until instructed otherwise. All other datasets are considered temporary, and are deleted when the job completes.

2.3.1 TEMPORARY DATASETS

A *temporary dataset* is available only to the job that created it. You can create temporary datasets explicitly by use of the ASSIGN control statement, or implicitly upon first reference to a dataset by name or unit number on an I/O request or an OPEN macro call.

A temporary mass storage dataset is empty until written on. Rewinding or backspacing of a dataset is necessary before it can be read.

To make a temporary dataset permanent, use the SAVE control statement. If the temporary dataset is not made permanent, it is released when the job terminates. A temporary dataset may also be released with the RELEASE control statement. When a temporary dataset is released, its mass storage (if used) is made available to the system.

2.3.2 PERMANENT DATASETS

Only mass storage or magnetic tape datasets can be permanent.

2.3.2.1 Magnetic tape permanent datasets

The subsection on dataset media earlier in this section discusses tape datasets.

2.3.2.2 Mass storage permanent datasets

A *mass storage permanent dataset* is maintained across system startups. Mass storage permanent datasets are of two types:

- Those created by SAVE control statements made by the user or as the result of a front-end system SAVE command (user permanent datasets)
- Input or output datasets

User permanent datasets are maintained for as long as the user or installation desires. They can be protected from unauthorized access using permission control words and ownership values on the SAVE control statements.

When a user permanent dataset is accessed through an ACCESS control statement (refer to section 9), it is copied to the job as a local dataset by the job requesting access. It still exists, however, as a permanent dataset on the system and can be used by other jobs unless unique access to that dataset was granted. You must have write permission to write to a permanent dataset. If any information in an existing permanent dataset is overwritten or if the size of a permanent dataset is changed, an ADJUST should be performed on that dataset (refer to section 9). When a permanent dataset is released or closed, an ADJUST is performed automatically if the size of the dataset changes.

System permanent datasets relate to particular jobs or reflect the current operational state of COS. A job's *input dataset* is made permanent when the job is received by the Cray computer system and is deleted when the job terminates. *Output datasets* local to the job can be disposed of while the job is running or can be automatically made permanent when the job terminates and are then deleted from the Cray computer system after being sent to the front-end system for processing.

An *execute-only dataset* is a user permanent dataset for which all forms of examination and modification by users are prohibited. An execute-only dataset is loaded by the COS Control Statement Processor (CSP) for execution. It differs from other user permanent datasets in several ways:

- The dataset can be accessed, but it cannot be opened for reading or writing.
- While an execute-only dataset is loaded in memory, DUMPJOB requests are not honored.
- The execute-only dataset cannot be staged to a front-end by a DISPOSE request.
- The execute-only dataset must be loaded by a dataset name call rather than by a load-and-go request by LDR or SEGLDR.

Because execute-only is a dataset state rather than a permission mode, it is advisable to set, at minimum, a maintenance permission control word to disallow modification or deletion of the dataset.

2.4 LOCAL DATASETS

A dataset to which a job has access is a *local dataset*. A local dataset can be a temporary or a permanent dataset. Permanent datasets are made local with the ACCESS control statement or the ACCESS library subroutine (described in the Programmer's Library Reference Manual, CRI publication SR-0113). If the dataset referenced is a tape dataset, the device resource must also be specified on the JOB control statement (refer to section 7).

2.5 DATASET DISPOSITION CODES

Each dataset is assigned a *disposition code* that tells COS what to do with the dataset when the job ends or the dataset is released. The disposition code is one of the parameters of the DISPOSE and ASSIGN control statements (refer to section 8).

Each disposition code is a 2-character alphabetic code describing the destination of the dataset. The default disposition code for a dataset is SC (scratch) when a dataset is opened, unless the dataset named is one of a group of special names such as $PLOT, $PUNCH, and $OUT. By default, COS assigns the disposition code PR (print) to $OUT when the dataset is created. No DISPOSE statement is required for $OUT; a PR disposition automatically routes it to the station and terminal from which the job was submitted unless a DISPOSE statement changes either the disposition code or destination station or terminal.

2.6 USER DATASET NAMING CONVENTIONS

There are two types of naming conventions for user datasets; one for local datasets and a different one for permanent datasets. Each type requires an assigned symbolic name.

A *local dataset* name consists of 1 to 7 characters: the first character must be an uppercase A through Z, $, @, or %; the remaining characters may be any alphanumeric character. If you specify a lowercase name, COS interprets the characters as uppercase. COS does not accept a lowercase local dataset name that is within double quotes. For example:

JCL Name Assignment	COS Interpretation
ASSIGN, DN = NAME.	NAME
ASSIGN, DN = name.	NAME

JCL Name Assignment	COS Interpretation
ASSIGN, DN = 'NAME'.	NAME
ASSIGN, DN = 'name'.	NAME
ASSIGN, DN = "name".	*Error*

A *permanent dataset* name is less restrictive; it can contain upper and lowercase alphanumeric characters, $, @, or %. If a lowercase name is specified, COS interprets the characters as uppercase. If a lowercase name within double quotes is specified, COS accepts the name as lowercase. For example:

JCL Name Assignment	COS Interpretation
SAVE, DN=X, PDN = NAME.	NAME
SAVE, DN=X, PDN = name.	NAME
SAVE, DN=X, PDN = 'NAME'.	NAME
SAVE, DN=X, PDN = 'name'.	NAME
SAVE, DN=X, PDN = "name".	name

Other considerations:

- Do not use characters with the octal codes 000 through 037 or 177 through 377. These are unprintable characters. Refer to the ASCII character set in appendix C for details.
- Certain language processors place further restrictions on dataset names.
- Most datasets defined by COS are assigned names of the form *$dn*. Because datasets whose names begin with a $ may receive special handling by the system, refrain from using this format when naming datasets.

2.7 USER I/O INTERFACES

When using logical I/O, you are never directly concerned with the actual transfer of data between the devices and the system buffers. Figure 2-4 shows the relationship of different levels of user logical I/O interfaces and routines. In this figure, the request levels and routine calls are summarized without going into detail about the movement of data between the system buffers and user program areas. Refer to the Macros and Opdefs Reference Manual, CRI publication SR-0012, for details on logical I/O.

The highest level of user interface is I/O statements used by programming languages such as Fortran and Pascal; the lowest level is in the form of specially formatted requests called Exchange Processor requests.

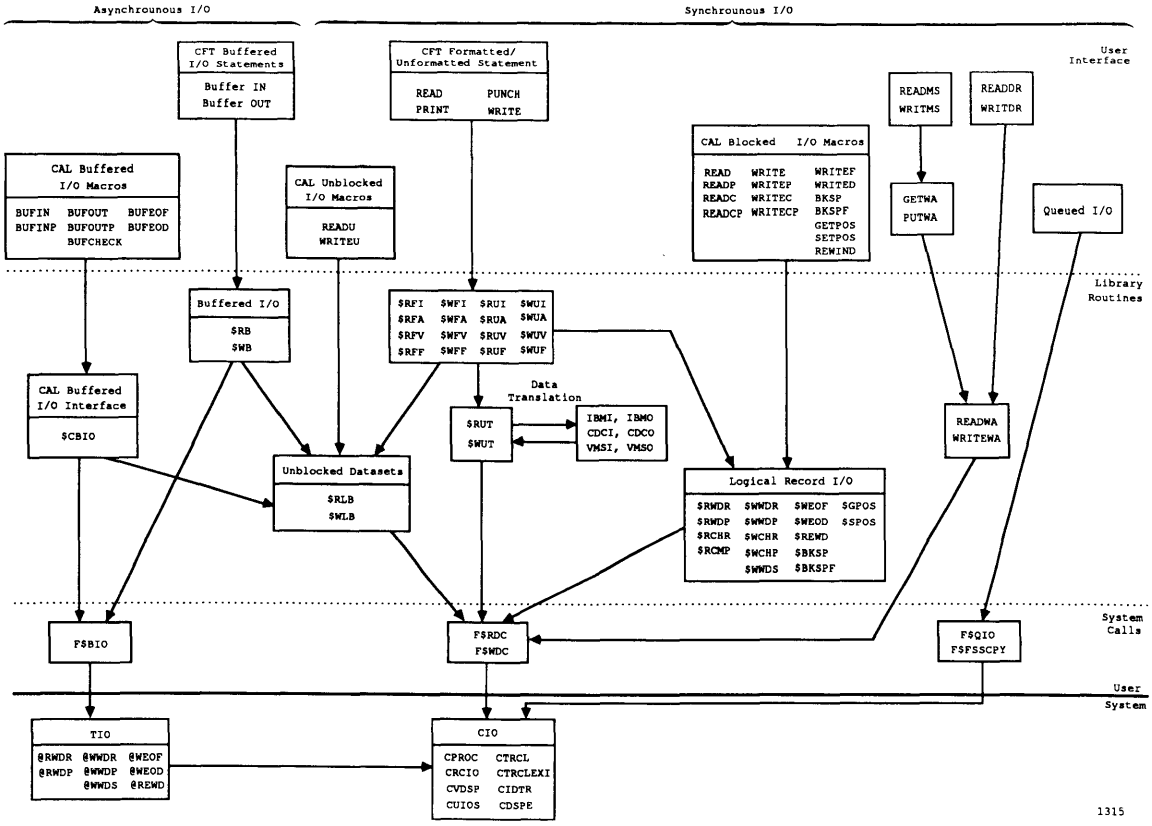


Figure 2-4. Relationship of Levels of User I/O

Fortran statements fall into two categories: formatted/unformatted and buffered. The formatted/unformatted statements result in calls to library routines $RFI through $WUF. If the dataset is blocked, these routines call the logical record I/O routines. The logical record I/O routines perform blocking and deblocking. The logical record I/O routines communicate with COS through the Exchange Processor requests, F$RDC and F$WDC.

If the dataset is unblocked, $RUA or $WUA calls the unblocked dataset routine $RLB or $WLB. These routines do no blocking or unblocking of data. The unblocked I/O routines communicate with the system through the F$RDC and F$WDC Exchange Processor calls.

Buffered I/O takes a different path from formatted/unformatted I/O. These routines interface (through an F$BIO Exchange Processor request) to routines in COS that normally perform logical I/O for system tasks. These routines, called Task I/O (TIO), closely resemble the logical record I/O routines. TIO and the logical record I/O routines make similar requests of circular I/O routines in COS although the mechanism for making these requests is different.

Circular I/O (CIO) routines are the focal point for all logical I/O generated by COS. CIO communicates its needs for physical I/O to the Disk Queue Manager (DQM) or Tape Queue Manager (TQM).

All I/O on the lowest levels from DQM and TQM is asynchronous; meaning that when you do a write, the information is passed to COS, but the actual transfer to disk or tape is performed later. This method of I/O is a performance feature termed *write-behind* (which for disk is controlled by the COS installation parameter I@DTDREP). On a rare occasion with write-behind enabled, a job can complete before the physical transfer of data actually occurs, and if an error is found after job completion, there is no mechanism for reporting it.

A Fortran buffered I/O request issued for an unblocked dataset results in the buffered I/O routines calling the unblocked dataset routines $RLB and $WLB, which then process these requests. These requests are processed the same as formatted/unformatted requests except that buffered I/O requests return control to you after initiating I/O rather than waiting for completion of the I/O request. For a Cray Assembly Language (CAL) buffered I/O request, $CBIO is called to route the request to either the blocked or unblocked I/O processing routines.

The Macros and Opdefs Reference Manual, CRI publication SR-0012, describes the CAL I/O macros. The Programmer's Library Reference Manual, CRI publication SR-0113, describes the logical record I/O routines and Fortran I/O routines. Refer to the Fortran (CFT) Reference Manual, CRI publication SR-0009, or the CFT77 Reference Manual, CRI publication SR-0018, for a description of Fortran statements.

COS JOB PROCESSING 3

A *job* is a unit of work submitted to COS. It consists of one or more files of statements, which may be control statements or input to a processing routine. The files form a *job dataset*. Each job passes through several stages from the time the job is entered until the job terminates.

3.1 JOB DATASET STRUCTURE

A job originates as a dataset at a front-end computer system. Control statements and data in the job dataset are organized into one or more files. The following example represents a typical job dataset consisting of a control statement file, a source file, and a data file. (The statement formats for *end-of-file* and *end-of-data* are defined by the front-end system.)

Example:

```
JOB,JN= . . .
Control statements
   .
   .
   .
<eof>
Source file
   .
   .
   .
<eof>
Data file
   .
   .
   .
<eof>
```

The first (or only) file of the job dataset must contain the job control language (JCL) statements that specify the processing requirements for the job (section 4 describes JCL). Each job begins with a JOB control statement that identifies the job to COS. If accounting is mandatory in

your system, the ACCOUNT control statement must immediately follow the control JOB statement. All other control statements follow. Control statements can be grouped into control statement blocks as described in section 16.

At the end of the JCL file is an end-of-file (EOF) record (or an end-of-data (EOD) record if the job consists of a control statement file only).

Files following the control statement file can contain source code or data. These files are handled according to instructions given in the JCL file.

3.2 JOB FLOW

A job passes through the following stages from the time it is read by the front-end computer system until it completes:

- Entry into COS
- Initiation on the system
- Advancement through the system
- Termination

3.2.1 JOB ENTRY

A job enters the system in the form of a dataset submitted from a front-end computer system or by a JCL SUBMIT control statement and a job already executing (described in section 10). The job is transferred to Cray computer system mass storage, where it resides until it is scheduled to begin processing. The job input dataset ($IN) is made permanent until it is deleted at the completion of the job.

3.2.2 JOB INITIATION

COS examines the parameters on the JOB control statement to determine the resources needed. When the system resources required to begin are available, the job is scheduled to begin processing (initiated).

Initiation of a job includes preparing a Job Table Area (JTA) and user field in memory, positioning the input dataset for the first job step, and placing the job in a queue for the CPU.

When COS schedules the job for processing, it creates four datasets: $CS, $IN, $OUT, and $LOG.

$CS is the job's control statement file from $IN and is used only by the system; you cannot access $CS by name. This dataset is used to read job control statements, and its disposition code is SC (scratch).

$IN is the job input dataset. The job itself can access the input dataset, with read-only permission, by its local name, $IN, or as Fortran unit 5. The disposition code for $IN is SC (scratch).

$OUT is the job output dataset. The job can access this dataset by its local name or as Fortran unit 6. The disposition code for $OUT is PR (print).

$LOG is the job's logfile and contains a history of the job. This dataset is known only to COS; you cannot access $LOG by name. User messages can be added to the job's logfile with the MESSAGE system action request macro (refer to the Macros and Opdefs Reference Manual, CRI publication SR-0012,) or the REMARK, REMARK2, or REMARKF subroutines (refer to the Programmer's Library Reference Manual, CRI publication SR-0113).

3.2.3 JOB ADVANCEMENT

Job advancement is the processing of a job according to the instructions in a control statement file. Advancement occurs as a normal advance or as an abort advance.

A normal advance causes COS to interpret the next control statement in the job's control statement file. When a job step is multitasked, a job advance deletes all user tasks except the one that causes the advance.

An abort advance occurs if COS detects an error or if you request that the job abort. The Exit Processing subsection describes abort advances.

3.2.4 JOB TERMINATION

Output from a job is placed on system mass storage. At completion of a job, COS appends $LOG to $OUT and returns $OUT to its originating station. $IN, $CS, and $LOG are released. $OUT is renamed *jn* (from the JN parameter value of the JOB control statement described in section 7) and is directed to the output queue for staging to the originating front-end computer system. When the front end receives the entire contents of $OUT, the output dataset is deleted from COS mass storage.

The front-end computer processes $OUT as specified by the dataset disposition code. If, for any reason, $OUT does not exist, $LOG is the only output returned at job termination.

If COS encounters an error as it attempts to copy $LOG to $OUT, $LOG is disposed as a separate dataset.

3.3 JOB MEMORY MANAGEMENT

Central Memory is one of the resources allocated to a job by COS. A job's memory is composed of several distinct areas. Some of these areas are managed exclusively by COS; others are managed by both you and COS.

Figure 3-1 shows a job in memory. The total job size equals the length of the job's JTA plus the user field length. The lined area between WJCHLM and WJCLFT is unused space within the job and contains enough memory to guarantee that the user area is always a multiple of 512 words.

3.3.1 INITIAL MEMORY ALLOCATION

When the job initiates, it is given sufficient memory for the Control Statement Processor (CSP) to execute. Once the JOB statement is processed, the job is allowed a user field length no larger than the amount specified by the MFL parameter on the JOB control statement (refer to section 7).

3.3.2 FIELD LENGTH REDUCTION

There are two modes of user field length reduction: automatic and user managed. A job initiates in automatic field length reduction mode, and the system automatically increases and decreases the job's field length as the areas within the job increase and decrease.

When a job is in user-managed field length reduction mode, the system continues to increase the job's field length as before, but never automatically decreases it. The job's field length can be decreased only by the user until the job is returned to automatic field length reduction mode.

Increases in field length can result in the job requiring more memory than can be immediately supplied, which causes the job to be delayed until sufficient memory can be given to it. Therefore, you may want to manage the job's field length when it is known that the job will undergo frequent short-lived fluctuations in size. The field length can be reduced at the beginning of each job step and during each job step if the job is in automatic field length reduction mode and any area of the job decreases.

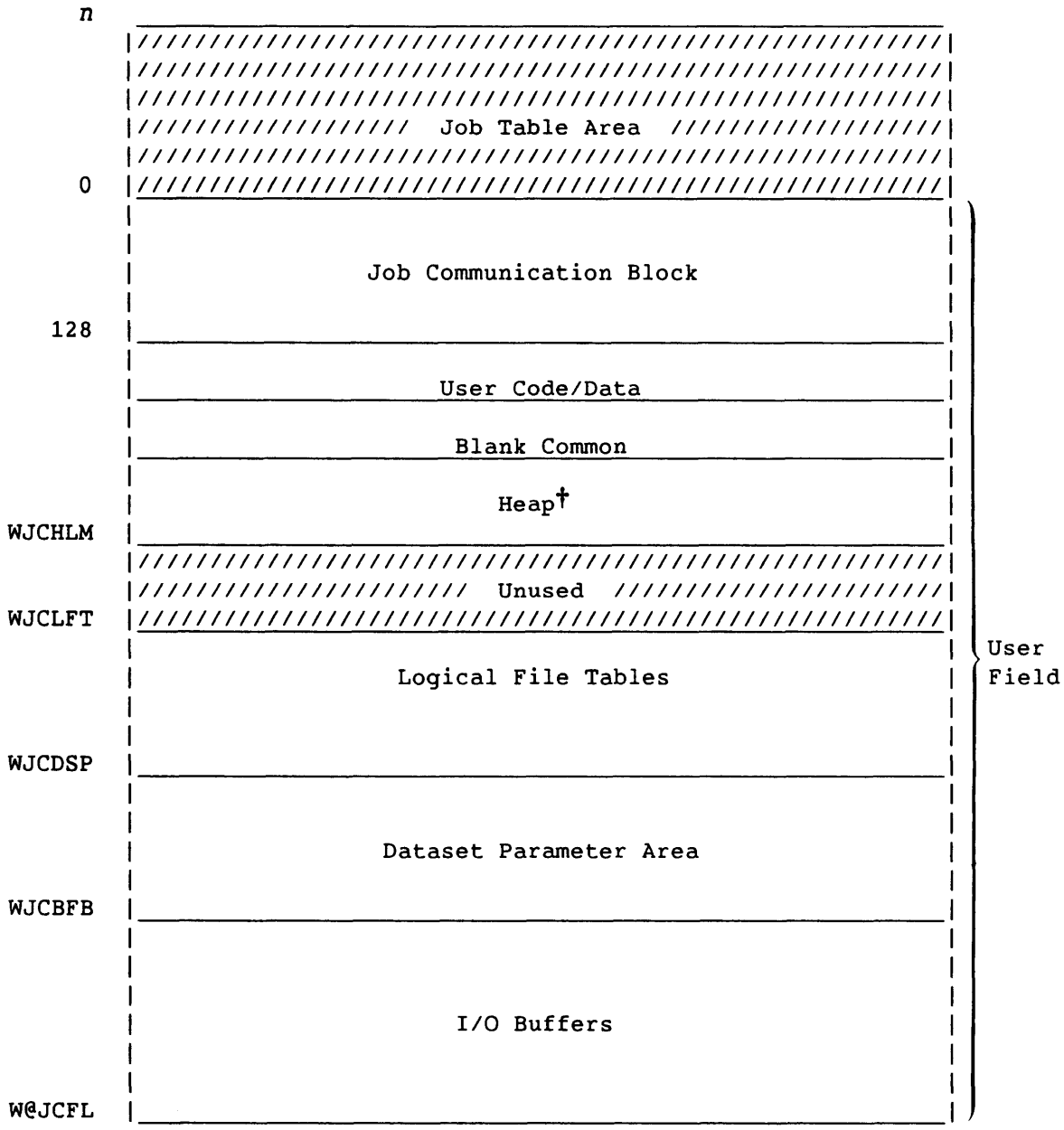


Figure 3-1. User Area of Memory for a Job

† Although the heap follows blank common in the figure, it can optionally precede blank common.

3.3.3 USER MANAGEMENT OF MEMORY

A user can dynamically manage the user code/data area of the job by requesting an increase or decrease of memory at the end of the user code/data area.

A user can manage the field length of the job by requesting a specific field length.

When the user manages the field length of the job, the job is placed in user-managed field length reduction mode for the duration of the job step (the next job step when using the MEMORY control statement described in section 7).

A user can place the job in user-managed field length reduction mode across job steps by explicitly requesting that mode. The job remains in user-managed field length reduction mode until the user explicitly requests automatic field length reduction mode.

3.3.3.1 Management by control statement from the run stream

A user can use the MEMORY control statement to manage the job's field length. When the user manages the job's field length, the job will be placed in user-managed field length reduction mode for the duration of the next job step. The MEMORY control statement may also place the job in user-managed field length reduction mode across job steps or return the job to automatic mode.

3.3.3.2 Management from within a program

From within a program, the MEMORY macro or MEMORY routine requests user management of the job's user code/data area and field length. When the user manages the job's field length, the job is placed in user-managed field length reduction mode for the duration of the job step. The MEMORY macro or MEMORY routine may also place the job in user-managed field length reduction mode across job steps or return the job to automatic mode.

3.3.3.3 Management associated with a program

Use of the SEGLDR directives BCINC, PADINC, and NORED, and the LDR control statement parameters BC, PAD, and NORED causes certain types of memory management to be associated with the binary being loaded. (Refer to the Segment Loader (SEGLDR) Reference Manual, CRI publication SR-0066, for more information on SEGLDR and section 14 of this manual for more information on LDR.) This association is stored with the binary if the

binary is saved on a dataset. The management associated can be user code/data area management or field length management and occurs when the binary is loaded for execution. If the field length is being managed, the job is placed in user-managed field length reduction mode for the duration of program execution.

3.3.4 SYSTEM MANAGEMENT OF MEMORY

The system changes appropriate areas of the job's memory when a job initiates certain system actions (that is, advances to the next job step, does I/O, and so on). The JTA, Logical File Tables (LFTs), and Dataset Parameter Area (DSP) pictured in figure 3-1 can increase but will never decrease. The user code/data and buffer areas may both increase and decrease in size. If the job is in automatic field length reduction mode, the system automatically increases and decreases the job's field length when any area in the job increases or decreases. If the job is in user-managed field length reduction mode, the system continues to increase the field length when it needs to, but never automatically decreases the field length.

3.4 JOB RERUN

Under certain circumstances, you may want to rerun a job from the beginning. Conditions that cause the system to attempt to rerun a job are as follows:

- An operator command
- An uncorrectable memory error
- An uncorrectable error reading the mass storage image of a job
- A system restart

A user job may perform certain functions that make it impossible to rerun. The functions render a job nonrerunnable because they produce results that might cause the job to run differently if it were rerun. These functions include the following:

- Writing to a permanent dataset
- Saving, deleting, adjusting, or modifying a permanent dataset
- Acquiring a dataset from a front-end system

Ordinarily, when a job becomes nonrerunnable, it remains so; however, you may declare that the job is rerunnable. You should do this only when changes in job results due to execution of nonrerunnable functions are acceptable. COS never makes a job rerunnable automatically.

You can also override system monitoring of job rerunnability, regardless of what functions the job performs. This ordinarily is done only if the job is structured to run correctly regardless of the functions performed.

3.5 EXIT PROCESSING

When COS detects an error condition or when you request a job step abort, COS checks to see if the condition is to be reprieved. (The next subsection describes reprieve processing.) If no reprieve occurs, exit processing occurs.

Generally, when a job step abort occurs, the current job step is immediately abandoned and control statements are skipped until the next eligible EXIT statement is encountered (section 7 describes EXIT). Normal job advancement occurs with the EXIT statement that is found. If no eligible EXIT statement is found, the job is terminated. EXIT statements within control statement blocks (iterative, conditional, or in-line procedure) that have not yet been invoked are ignored during the search for the next eligible EXIT statement.

If the block currently being processed is a conditional block (refer to section 16), and the system encounters an abort condition, COS suspends execution until it reaches the first EXIT statement at the same conditional level. If there is no EXIT within the block, COS suspends execution until the first EXIT statement after the conditional block.

COS ignores all statements including EXITs within any unexecuted blocks and, if no EXIT statement is at the same conditional level, also ignores statements between that block and the first EXIT following it. For example, in the following control statement sequence, an abort advance occurs at the control statement THIS IS A JOB STEP ABORT CONDITION because it does not begin with a valid verb. Control statement interpretation resumes with the control statement *. RESUME HERE.

Exit processing is not performed for interactive jobs except inside an invoked procedure. After a job step abort occurs, you are simply prompted for the next control statement.

Example:

```
.
.
.
SET,J1=0.
IF(J1.EQ.0)
  .
  .
  .
  THIS IS A JOB STEP ABORT CONDITION.
ELSEIF (J1.EQ.1)
    .
    .
    .
    EXIT.
ELSE.
    EXIT.
ENDIF.
    .
    .
    .
EXIT.
*.  RESUME HERE
    .
    .
```

3.6 REPRIEVE PROCESSING

Normally, when a job step abort condition occurs, exit processing begins. Reprieve processing, however, lets a user program attempt recovery from many of the job step abort conditions or perform clean-up functions before continuing with the abort.

Reprieve processing can also be invoked during the normal termination of a job step. In this case, control transfers to the user's reprieve code instead of to the next normal job step.

Two types of error conditions are related to a job step abort condition: nonfatal and fatal. They are as follows:

- Nonfatal error conditions are those that you can reprieve any number of times per job step.
- Fatal error conditions can be reprieved only once for each type per job step.

When requesting reprieve processing, you select the conditions to be reprieved by setting a mask in the SETRPV subroutine or macro call. If a selected condition occurs during job processing, your current job step maintains control. The user's Exchange Package, vector mask register, error code, and error class are saved, and control passes to the user's reprieve code.

3.7 INTERACTIVE JOB PROCESSING

An interactive job dataset has interleaved control statements, program or utility input, and program or utility output. In an interactive job, the control statement file ($CS), standard input dataset ($IN), standard output dataset ($OUT), and logfile ($LOG) are all defined by the system to be interactive datasets. Refer to section 2 for more information on interactive datasets.

Each job step of an interactive job is initiated with a control statement. Control statements can be either part of a procedure invocation or entered directly from the interactive terminal. After each control statement is received by COS, input to the job step can be entered from the terminal, and output and logfile information is returned to the terminal. When the current job step is complete, normal job advancement occurs, and COS prompts for the next control statement or reads it from the invoked procedure file. Exit processing (refer to section 3) is never performed on an interactive job except within a procedure invocation.

Whenever a program or utility executing as part of an interactive job requests to read from the standard input dataset, the interactive user is prompted to enter data one record at a time. Likewise, any data written to $OUT, the standard output dataset, is sent to the interactive terminal. User logfile messages are also sent to the interactive terminal.

3.8 JOB LOGFILE AND ACCOUNTING INFORMATION

For each job that runs, COS produces a logfile, which is an abbreviated history of the progress of the job through the system. The logfile for a noninteractive job appears at the end of the job output. Each job control statement is listed sequentially, followed by any messages associated with the job step. Clock time, accumulated CPU time, and COS information are also given for each job step. Figure 3-2 shows the items usually contained in a logfile. Item 6 illustrates the accounting information given to the user.

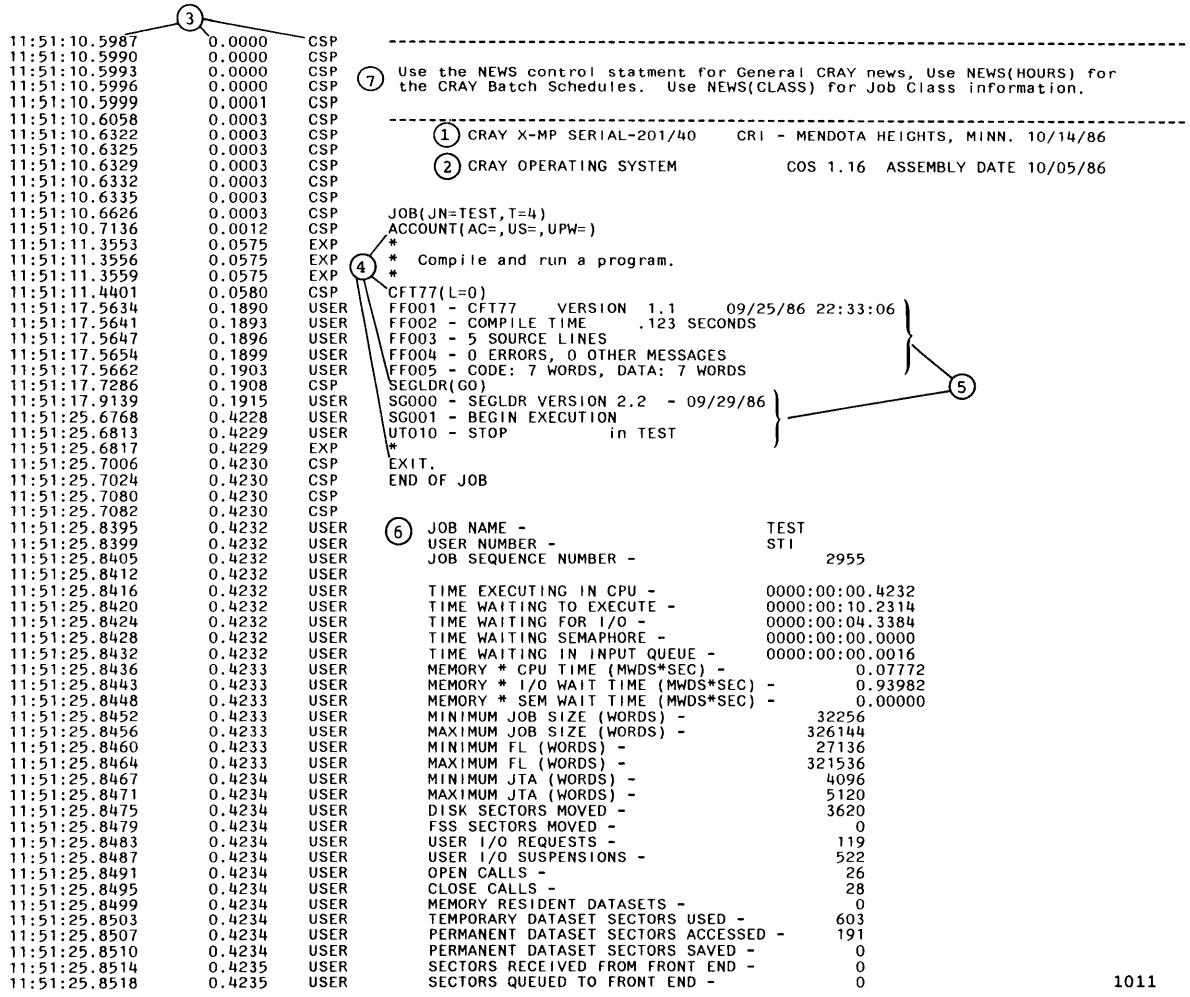


Figure 3-2. Example of a Job Logfile

(1) First header line: Installation-defined message, usually identifying the site and date the job was run.

(2) Second header line: Installation-defined message, usually identifying the operating system, its current revision level, and the date of the last revision.

(3) Columns: The leftmost column identifies the wallclock time for each job step and the middle column identifies the accumulated CPU time for the job. The rightmost column identifies a system module or the user as the originator of the message on that line. All times are in decimal. Entries commonly noted include the following:

Entry	Meaning
CSP	Control Statement Processor
PDM	Permanent Dataset Manager
EXP	Exchange Processor
ABORT	Abort Message
USER	Program in user field

(4) Control statements: The logfile lists every control statement processed.

(5) Logfile messages: Any messages related to control statement processing are shown below the statement.

(6) Accounting information: When a job reaches completion, COS writes a summary of basic accounting data onto the logfile for the job. All times given are in hours, minutes, and seconds (to the nearest ten-thousandth of a second). The following accounting information is provided (in decimal):

- Jobname and user number
- CPU time used by the job and by each job task in a multitasked job step
- Time waiting to execute for the job and each job task in a multitasked job step; includes time waiting for the CPU, memory, operator suspension, and recovery.
- Time waiting for I/O for the job and each job task in a multitasked job step
- Time waiting in input queue
- Memory usage based on the execution and I/O wait time in million word-seconds
- Minimum and maximum job size including JTA (words)

- Minimum and maximum field length used (words)
- Minimum and maximum JTA used (words)
- Number of 512-word disk blocks (sectors) moved
- Number of fast secondary storage (FSS) sectors moved to either the SSD solid-state storage device or Buffer Memory (BMR)
- Number of user I/O requests made by the job
- Open and close calls
- Memory-resident datasets
- Number of 512-word disk blocks (sectors) used for temporary datasets
- Number of 512-word disk blocks (sectors) accessed and saved for permanent datasets
- Number of 512-word disk blocks (sectors) received from and queued to the front end
- For each generic resource specified on the JOB control statement, the accounting information includes a report describing the device type (tape, disk, or ISP), number of units reserved from the JOB control statement, number of sectors transferred, largest number of units allocated concurrently during job execution, and resource allocation integral. If the resource consists of tape devices, the report includes the number of tape volumes mounted and number of tape blocks transferred.
- For each FSS device not configured as a generic resource, the accounting information includes a report describing the logical device name, number of sectors transferred, maximum number of sectors allocated concurrently during job execution, and resource allocation integral.

(7) System bulletin: The system bulletin allows the installation to print messages in the logfile, usually about the status of the system environment. It is an installation-maintained message dataset and may not be present.

JOB CONTROL LANGUAGE 4

The job control language (JCL) for COS lets you present a job to the Cray computer system, define and control execution of programs, and manipulate datasets.

The JCL is composed of *control statements*. Each control statement contains information for a job step. COS initially creates a *control statement dataset*, $CS, to hold job control statements. Additional control statement datasets can be created through procedure definition or the CALL control statement (refer to section 7).

The syntax of a control statement is as follows:

verb	sep_1	$param_1$	sep_2	$param_2$	...	sep_n	$param_n$	*term*	*comments*

All control statements must adhere to a set of general syntax rules. Every control statement must have a *verb* and a terminator (*term*) as a minimum, except for a comment control statement (introduced by an asterisk *) which does not require a terminator. Most control statements also require parameters (*param*) and separators (*sep*) between the verb and its parameters. The maximum number of parameters depends on the verb.

Lowercase letters are converted to uppercase letters unless they are used in a literal string.

The continuation separator (the caret symbol ^) allows a control statement to consist of more than one record (80 characters). The JOB, DUMPJOB, EXIT, and * (comment) control statements cannot be continued. All other control statements can have any number of continuation lines, subject to restriction of the verb. (A caret occurring within a literal string has no special significance. Refer to section 16 for more information about literal strings.)

A *comment* is an optional annotation to a control statement and can be a string of any ASCII graphic characters. The comment follows the statement terminator. The control statement interpreter ignores comments. All comments appear in the logfile unless suppressed by the ECHO control statement.

Blanks are ignored unless they are embedded in a literal string. Blanks cannot precede the verb on the JOB control statement.

4.1 SYNTAX VIOLATIONS

COS notes syntax violations in the system and user logfiles. If the JOB control statement is in error, processing of the job terminates immediately. If accounting is mandatory, ACCOUNT statement errors also cause job termination. All other syntax errors cause a *job step abort* condition, which causes the system to search for an EXIT control statement. A successful search resumes control statement processing with the job step following EXIT. If no such job step exists or if an EXIT statement is not found, the job is terminated. Job step abort can also direct control to a user-specified routine (refer to exit processing and reprieve processing in section 3).

4.2 CONTROL STATEMENT VERBS

A *control statement verb* is the first nonblank field of a control statement. It specifies what action COS will perform for that statement. COS recognizes three types of control statement verbs: *system verbs, dataset name verbs* (*local* and *system*), and *library-defined verbs*. A control statement verb cannot be continued to a second record.

When COS encounters a verb in a control statement file, it searches for a match to that verb. First, it searches the list of system verbs for a match. If the verb is not a system verb, COS searches first for a local dataset, next for a matching program name in the datasets in the library searchlist, and then for a matching system dataset name in the System Directory Table (SDR). If a match for the verb is not found under any of these categories, COS issues a control statement error and aborts the job step.

4.2.1 SYSTEM VERBS

A system verb consists of an alphabetic character that can be followed by 1 to 7 alphanumeric characters.† The system verb requests that COS perform a function. The system verbs are as follows:

*	ACCESS	ACQUIRE	ADJUST	ASSIGN	CALL
&DATA	DELETE	DISPOSE	DUMPJOB	ECHO	ELSE
ELSEIF	ENDIF	ENDLOOP	ENDPROC	EXIT	EXITIF
EXITLOOP	FETCH	HOLD	IF	IOAREA	JOB
LIBRARY	LOOP	MEMORY	MODE	MODIFY	NOHOLD
NORERUN	OPTION	PERMIT	PRINT	PROC	RELEASE
RERUN	RESTORE	RETIRE	RETURN	REWIND	ROLLJOB
SAVE	SET	SUBMIT	SWITCH	TARGET	

The Cray Simulator (CSIM) Reference Manual, publication SR-0073, describes the SIMABORT control statement.

4.2.2 LOCAL DATASET NAME VERBS

Local dataset name verbs begin with an alphabetic character followed by 1 to 6 alphanumeric characters.† Local dataset name verbs request that COS load and execute an absolute binary program from the first record of the named dataset. If the user job has a dataset with the indicated name, COS loads and executes the program from that dataset.

4.2.3 LIBRARY-DEFINED VERBS

Library-defined verbs consist of 1 to 8 characters. The library-defined verb is either a program or procedure definition residing in a library that is a part of the current *library searchlist*. (The library searchlist defines the library and the order in which the libraries are searched by COS. This order can be specified with the LIBRARY statement described in section 7.) A program in a library is an absolute binary program to be loaded and executed. A procedure definition is a group of control statements or data or both to be processed (refer to section 16).

† Alphabetic characters include $, %, @, and the letters A through Z (uppercase and lowercase). Alphanumeric characters include all the alphabetic characters and the digits 0 through 9.

4.2.4 SYSTEM DATASET NAME VERBS

COS searches for a verb that is the name of a system-defined dataset in the SDR. A system-defined dataset name verb begins with an alphabetic character followed by 1 to 6 alphanumeric characters. The SDR is a list of common language processors and utilities known to the system and made available to users at startup. The name of the program (for example, CAL, CFT, or DUMP) is also the name of the dataset containing the absolute binary of the program. The exact list of system dataset name verbs is site-dependent.

4.3 SEPARATORS

A *separator* is a character used as a delimiter in a control statement. It separates the verb from the first parameter, separates parameters from one another, delimits subparameters, terminates verbs and parameters, and separates a keyword from its value in parameters having keyword form.

Table 4-1 shows the control statement separators allowed by COS.

4.4 PARAMETERS

A *parameter* is a control statement argument whose exact requirements are defined by the control statement verb. Parameters are used in control statements to specify information to be used by the verb-defined process. Parameters that can be used with COS control statements are either *positional* or *keyword*. For certain verbs, a parameter value can be an expression. Detailed information on the use of expressions is presented later in this section. Parameters are separated by commas.

4.4.1 POSITIONAL PARAMETERS

A positional parameter has a precise position relative to the separators in the control statement. Even a null positional parameter must be delimited from the control statement verb or other parameters by a separator.

Table 4-1. Control Statement Separators

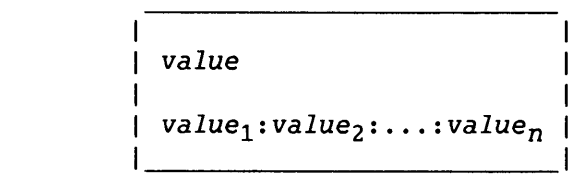
Function	Character	Examples
Initial separator (comma or open parenthesis)† - Separates the verb from the first parameter	, (	*VERB,parameter.* *VERB(parameter)*
Statement terminator (period if initial separator is comma, close parenthesis if initial separator is open parenthesis)† - Signifies end of control statement	.)	*VERB.* *VERB,parameter.* *VERB(parameter)*
Parameter separator (comma) - Indicates the end of one parameter and the beginning of the next	,	*VERB(parameter,parameter)*
Equivalence separator (equal sign) - Delimits a parameter keyword from the first parameter value for that keyword. Adjacent equivalence separators are illegal.	=	*VERB(keyword=value)*
Concatenation separator (colon) - Separates multiple parameter values from each other	:	*VERB(keyword=value*$_1$*:value*$_2$*)*
Continuation character (caret) - Indicates that the control statement consists of more than one 80-character card; may appear anywhere after the initial separator.	^	*VERB(...parameters...^ parameters)*
Literal string delimiters (apostrophes)†† - Identifies the beginning and end of a literal string	'...'	*VERB(keyword='string')*
Parenthesis delimiters (open and close parentheses) - Indicates a group of characters to be treated as one value	(...)	*VERB(keyword=(value:value)*

† By convention, the comma and period are used as initial and terminator separators for all control statements except on the JCL block control statements (procedure definition, iterative, and conditional), where paired parentheses are conventional.

†† Refer to section 16 for additional information on strings and string delimiters.

1003

The formats for a positional parameter follow:



Each $value_i$ is a string of alphanumeric characters, a literal string, or a null string. Positional parameters are represented by at least one *value*, unless the value is null. To represent null values, use only the closing comma.

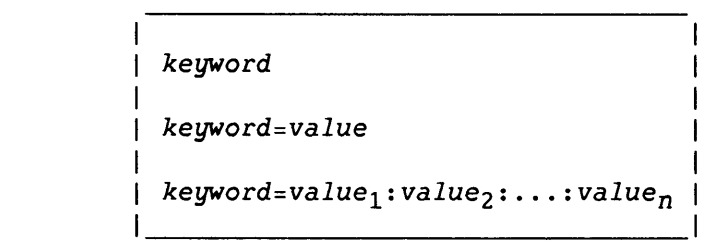
Examples of positional parameters:

...,ABCDE,...	The parameter value is ABCDE.
...,,...	The adjacent parameter separators indicate a null positional parameter.
...,P1:P2:P3,...	The parameter consists of multiple values.
VERB() or VERB,. or VERB.	The positional parameter 1 is null.

4.4.2 KEYWORD PARAMETERS

A *keyword parameter* is identified by its form rather than by its position in the control statement. The keyword is a string of 1 to 8 alphanumeric characters uniquely identifying the parameter. Parameters of this type can occur in any order but must be placed after all of the positional parameters for the control statement, or they can sometimes be omitted.

The formats of keyword parameters are as follows:



keyword is an alphanumeric string that depends on the requirements of the verb. $Value_i$ is the value associated with the keyword. A keyword parameter can occur anywhere in the control statement after all positional

parameters are specified. Whether a keyword parameter is required depends on the verb's requirements. If the keyword is not included in the control statement, a default value can be assigned.

Examples of keyword parameters:

...,DN=FILE1,...	The parameter consists of the keyword and a value.
...,UQ,...	The parameter consists of the keyword only.
...,DN=FILE1:FILE2:FILE3,...	The parameter consists of the keyword and a list of values.
...,DN=,...	The parameter contains a null value. (The value is omitted from the statement.)
...,DN=A:::B,...	The parameter value contains A, two null parameters values, and B.

The parameter associated with a keyword may be defined as a secure parameter. Every secure parameter is edited out of the statement before it is echoed to the user logfile. When a keyword is secure, all that appears in the user's logfile is the keyword and the = sign, followed by the next delimiter. Secure parameters are defined when calling GETPARAM as described in the Programmer's Library Reference Manual, CRI publication SR-0113.

4.4.3 PARAMETER INTERPRETATION

The decoding (parsing) of control statement parameters is normally performed by the routines $CCS and GETPARAM, as described in the Programmer's Library Reference Manual, CRI publication SR-0113. Parameter interpretation is performed by the particular program or utility that calls $CCS or GETPARAM.

LIBRARIES 5

Job control statements, programs, and compiled subprograms are maintained in libraries. The following types of libraries are available on COS:

- Procedure libraries
- Program libraries (PLs)
- Object code libraries

The CALL and LIBRARY control statements (refer to section 7) refer to procedure libraries; UPDATE (refer to the UPDATE Reference Manual, CRI publication SR-0013) maintains program libraries.

5.1 PROCEDURE LIBRARY

A *procedure library* is made up of procedures that consist of a sequence of control statements or data (or both) saved for processing at a later time.

A procedure library is created by the in-line procedure definition process described in section 16. After it is created, a procedure library is made available for using the LIBRARY control statement (refer to section 7).

5.2 PROGRAM LIBRARY

A *program library* (PL) is a means of maintaining programs and other data on datasets. These datasets are created and maintained by the UPDATE utility described in the UPDATE Reference Manual, CRI publication SR-0013. A PL contains one or more specially formatted files consisting of records of ASCII characters. The files are separated by end-of-file (EOF) records. The decks can be programs, portions of programs, input data for programs, or even job control statements. Refer to the UPDATE Reference Manual for full information on using PLs.

5.3 OBJECT CODE LIBRARIES

Object code libraries are termed *library datasets* or simply *libraries*. A *library dataset* is a dataset containing a program file followed by a directory file. Within the category of object code libraries are relocatable libraries and absolute libraries. Relocatable libraries are designed to provide the loader with a means of rapidly locating and accessing program modules. Relocatable library datasets are created and maintained by the BUILD utility as described in section 15. Any library dataset can be inspected and described by ITEMIZE. Refer to section 13 for more information on ITEMIZE.

Absolute binaries are created by LDR or SEGLDR. From them, BUILD produces a collection of absolute binaries called an absolute library. The absolute libraries are searched for system verbs when the object library's dataset name is in a search list specified by the LIBRARY control statement. For information on library-defined verbs, refer to section 4.

JOB CONTROL STATEMENTS 6

Job control statements perform the following functions:

- Identify a job to the system
- Define operating characteristics for the job
- Manipulate datasets
- Call for the loading and execution of user programs
- Call COS programs that perform utility functions for the user
- Define and manipulate other control statements

The first file of a job dataset contains control statements that are read, interpreted, and processed one at a time. The sequential processing of control statements determines the *job flow* through the operating system. Refer to section 3 for a general description of job flow. Sequential processing of control statements can be altered by exit or reprieve processing, or by control statement structures described in section 16.

Section 4 presented information on the general syntax rules and conventions for control statements. Sections 6 through 15 describe COS control statements and give examples in some cases. The control statements are described in the following categories:

- Job definition and control
- Dataset definition and control
- Permanent dataset management
- Dataset staging control
- Permanent dataset utilities
- Local dataset utilities
- Analytical aids
- Executable program creation
- Object library management

6.1 JOB DEFINITION AND CONTROL

Several control statements let you specify job processing requirements. Control statements defining a job and its operating characteristics to the operating system include the following:

Verb	Function
*	Annotates control statements with comments
ACCOUNT	Validates the job's account number, user number, and optional passwords
CALL, RETURN	Allows the use of alternate control statement files
CHARGES	Obtains partial or total resource reporting for a job
ECHO	Controls types of messages written to the job's logfile
EXIT	Indicates the point in a series of control statements at which processing of control statements resumes following a job step abort from a program, or indicates the end of control statement processing
IOAREA	Denies or allows access to the job's I/O area, the upper (high-address) portion of user memory that contains tables and buffers managed by the system I/O library routines
JOB	Introduces the job to the operating system and defines characteristics such as size, time limit, and priority levels
LIBRARY	Specifies the datasets to be searched when looking for defined procedures during job processing. LIBRARY also specifies the order in which to perform the search.
MEMORY	Requests a new field length and/or mode of field length reduction
MODE	Sets or clears mode bits in the job's Exchange Package
OPTION	Specifies user-defined options, such as the format of the job's listing and the amount of dataset accounting statistics produced
RERUN, NORERUN	Control job rerunnability
ROLLJOB	Protects a job by writing it to disk
SET	Changes the value of a job control language (JCL) symbolic variable

Verb	Function
SWITCH	Turns on or turns off pseudo sense switches
TARGET	Sets CPU characteristics

Section 7 fully describes job definition and control statements.

6.2 DATASET DEFINITION AND CONTROL

You can define and manage datasets using the following dataset control statements:

Verb	Function
ACCESS	Makes a permanent dataset local to a job. ACCESS can cause the creation of a tape dataset. If both are used, ACCESS must precede the ASSIGN control statement.
ASSIGN	Defines characteristics for datasets, such as the amount of user memory to allocate for the dataset's I/O buffer. ASSIGN also can be used to create a mass storage dataset. The ACCESS control statement must precede ASSIGN when creating a tape dataset.
HOLD	Declares that dataset release occurs with implicit HOLD
NOHOLD	Rescinds the effect of the HOLD control statement
RELEASE	Relinquishes access to the named dataset for the job

Section 8 describes ASSIGN, HOLD, NOHOLD, and RELEASE. Section 9 describes ACCESS.

6.3 PERMANENT DATASET MANAGEMENT

Control statements for managing permanent datasets provide for creating, protecting, and accessing datasets assigned permanently to mass storage or magnetic tape. Such datasets cannot be destroyed by normal system activity or engineering maintenance.

Front-end computer systems cannot directly affect Cray-resident permanent datasets, because permanent dataset management is handled entirely by COS; however, permanent magnetic tape dataset management can be optionally coordinated with a front-end computer system.

Users can manage user permanent datasets only; system permanent datasets cannot be managed (modified or deleted) by the user. (Refer to section 2 for a description of the types of datasets.)

Table 6-1 shows the control statements available for user permanent mass storage and magnetic tape dataset management. Actual processing of these requests depends upon the medium on which the dataset resides. Mass storage datasets are controlled by the COS system task called the Permanent Dataset Manager (PDM). Magnetic tape datasets are controlled by a system task called the Tape Queue Manager (TQM). Both of these system tasks (PDM and TQM) have mechanisms for retaining the characteristic information about the dataset. Information for mass storage datasets is retained in the Central Memory-resident Dataset Catalog (DSC). Magnetic tape datasets can have characteristic information retained on a front-end computer system.

Section 9 fully describes the permanent dataset management control statements.

6.3.1 MASS STORAGE DATASET ATTRIBUTES

Every mass storage permanent dataset has several *attributes* associated with it. These attributes are as follows:

- Read, write, and maintenance permission control words
- Public access mode
- Public access tracking
- Permits
- Text
- Notes

6.3.1.1 Permission control words

A *permission control word* is a password that must be supplied to gain access to a particular permanent dataset. Permanent datasets are not required to have a permission control word, but if a permission control word is specified for the mode of dataset access desired (read, write, or maintenance), the control word must be specified to gain access to the named dataset. If more than one mode of access is desired (for example, both read and write), all appropriate control words must be supplied.

Table 6-1. Permanent Dataset Management Control Statements for Each Medium

Verb	Mass Storage	Magnetic Tape
ACCESS	Makes a user permanent dataset local to the requesting job with the requested and/or allowable modes (execute, read, write, or maintenance)	Makes an existing tape dataset available to the job or defines a NEW-type tape dataset that will be created by the job. Also optionally defines the front-end computer system that will be the central point for servicing that dataset.
ADJUST	Records the change in any of the size or allocation information for a dataset that might have contracted or expanded	Not applicable
DELETE	Removes the definition of a user permanent dataset from the DSC. It is possible to delete a dataset's contents and have its attributes retained by the system.	Requests the front-end computer system servicing the dataset to remove (delete) any information concerning the dataset
MODIFY	Changes the characteristic information for an existing user permanent dataset	Not applicable
PERMIT	Explicitly grants or denies specified users or groups of users access to a permanent dataset	Not applicable
SAVE	Enters a dataset's identification and location in a system-maintained DSC. Datasets recorded in the DSC using a user SAVE request are user permanent datasets and are recoverable at deadstart.	Supplies to a front-end computer system the characteristic information about a dataset for its retention

6.3.1.2 Public access mode attribute

If all users are to be allowed some kind of access to a permanent dataset, that dataset must have a *public access mode* defined. The public access mode is the type of access, as a minimum, all users can have to the permanent dataset. Users can be allowed read, write, and/or maintenance mode access to the dataset. Users can be restricted to only executing the dataset; the public access mode can alternatively be NONE, signifying that public access is not permitted. When public access to a dataset is granted, any required permission control words must still be supplied in order to gain access to the dataset.

6.3.1.3 Public access tracking attribute

Public access tracking is a facility that can be turned on or off. A record can be kept of every user who accesses a public dataset. Refer to the Dataset Use Tracking subsection for more information on the public access tracking mechanism.

6.3.1.4 Permits attribute

User permanent mass storage datasets can have a list of alternate users of the dataset and in what mode or modes each alternate user can access the dataset. Each element of the list is known as a *permit* and names a specific alternate user and that user's allowed mode of dataset access. Refer to the Access Mode subsection for more information on permits.

6.3.1.5 Text attribute

Text is a character string to be passed to a front-end computer system when requesting transfer of the dataset to or from Cray mass storage. Refer to the Dataset Staging Control subsection for more information on text.

6.3.1.6 Notes attribute

Notes is a string of up to 480 characters associated with a permanent dataset. There is no restriction on what *notes* contains. When *notes* is listed using the AUDIT utility (refer to the Permanent Dataset Utilities subsection), the caret symbol is interpreted as an end-of-line signal and AUDIT advances to a new line when listing the dataset *notes*. *Notes* can contain such information as dataset structure, usage instructions, or history. For example, if several versions of a program exist as different permanent datasets, the *notes* could identify the purpose, difference, and origin of each dataset.

6.3.2 ESTABLISHING ATTRIBUTES FOR MASS STORAGE DATASETS

Mass storage permanent dataset attributes are established at dataset creation time, though they can be later modified (or added to, in the case of permits). Attribute establishment depends on whether a dataset with the same Permanent Dataset Name (PDN), additional identification (ID), and ownership already exists.

Supplying the entire set of attributes every time a new permanent dataset is created, that is, when no permanent dataset with the same PDN, ID, and ownership currently exists, can become quite tedious, especially if a long list of permits must be established. Instead, the dataset creator can supply an *attributes dataset*.

6.3.2.1 Existing permanent dataset

If a permanent dataset with the requested PDN, ID, and ownership already exists, the current dataset's permission control words, public access mode, public access tracking, and permit list are set to the corresponding attributes of the permanent dataset with the highest existing edition number (ED) and identical PDN, ID, and ownership.

The text attribute is also copied from the highest existing edition unless otherwise specified; the notes attribute is not copied.

The discussion of creating a new edition of an existing permanent dataset applies to datasets created by SAVE or PDSLOAD (refer to the Permanent Dataset Utilities subsection for information on PDSLOAD). If you use MODIFY to create a new edition of an existing dataset (by changing the PDN or ID), any dataset attributes not explicitly modified remain unchanged. Thus, it is possible, though not recommended, for different permanent datasets with the same PDN, ID, and ownership to have different attributes.

6.3.2.2 New permanent dataset

Using SAVE or ACQUIRE when no permanent dataset currently exists with the same PDN, ID, and ownership causes a new permanent dataset to be created.

All permanent dataset attributes can be established for a new permanent dataset; no attribute is associated with any other dataset. For example, if the new permanent dataset is to have a read permission control word, the control word must be supplied. If a list of permits is needed, the list must be supplied. Establishing an attributes dataset (described in the next subsection) provides a convenient way of supplying a list of permits.

6.3.2.3 Attributes dataset

An *attributes dataset* is an existing permanent mass storage dataset from which any (or all) permanent dataset attributes can be copied. The actual dataset content is ignored; the attributes are copied from the dataset's catalog entry. The attributes dataset can even be partially deleted (refer to the Dataset Staging Control subsection for a discussion of partial dataset deletion). The attributes dataset must be local to the job referencing it.

The attributes dataset is referenced with the ADN parameter on the SAVE or ACQUIRE control statement. When the attributes dataset is referenced, all desired attributes (such as permission control words and the public access mode) are copied from the attributes dataset and used in establishing the attributes of the current dataset. Any attribute explicitly specified on the SAVE or ACQUIRE control statement is used instead of the attributes dataset's attribute. The end of section 9 includes examples of attribute dataset use.

An attributes dataset can also be used with the PERMIT control statement, although it is used slightly differently. When an attributes dataset is used with PERMIT, the entire permit list (but no other attribute) is copied from the attributes dataset and added to the permit list established (or being established) for the current dataset.

For example, suppose the same permit list is being used for several different datasets. A single permanent dataset can be created and the list of permits established. Then whenever a new dataset is created, the original dataset can be accessed and used as an attributes dataset. The new dataset creator need not even know what permits are being established.

6.3.3 PROTECTING AND ACCESSING MASS STORAGE DATASETS

Access of mass storage datasets can be restricted on two levels:

- Which users can access the dataset (privacy)
- What type of access is allowed (access mode)

The mass storage dataset protection system has two other dataset management aspects:

- Dataset use tracking
- Attribute association

6.3.3.1 Privacy

Mass storage permanent datasets fall into three categories, depending on which users can access the permanent dataset:

- *Private* datasets are accessible only to the dataset owner.
- *Semiprivate* datasets are accessible to the dataset owner and to a specific group of other users.
- *Public* datasets are accessible to all users.

New mass storage datasets are either public or private (not semiprivate) by default. Contact your CRI site analyst for the default value at your site. A new dataset can be explicitly declared as either public or private with the public access mode (PAM) parameter on the SAVE control statement. (Refer to section 9.)

6.3.3.2 Access mode

In addition to establishing which users may access a dataset, the owner must establish what mode of access alternate users are allowed; that is, whether users other than the dataset owner may execute, read, write, or maintain the permanent dataset. Specifying the mode of alternate access depends upon what category of user is being granted the access. The three categories of users are as follows:

- The dataset owner who is allowed all modes of access.
- Specific alternate users who are named with the USER parameter of the PERMIT control statement (refer to section 9); the alternate user's allowed mode of access is declared with the access mode (AM) parameter of the same PERMIT control statement. Multiple PERMIT statements can be issued for the same permanent dataset to provide a list of alternate users. PERMIT can also be used to change or remove the allowed mode of access for an alternate user of the dataset. The allowed access mode for a specific user is known as a *permit*.
- All other users (the public). All users of a dataset not in the preceding two categories can be allowed (or denied) access to the dataset by using the PAM parameter on the ACQUIRE (section 10), SAVE, or MODIFY control statement (section 9). The mode of public access to a dataset can be changed at any time with the MODIFY control statement.

Any mass storage permanent dataset can have a public access mode with any combination of permits. If an alternate user desiring access to a permanent dataset is allowed public access and is named in a permit, the alternate user is allowed the access named in the permit. The permit takes precedence over the public access mode.

Such a combination of public and permitted access is often desirable. For example, suppose dataset FROG is to be used (executed as a program) by many groups of users, maintained by the dataset owner, and backed up or restored as needed by another user. Then, the dataset should have a public access mode of execute only and a permit of maintenance mode access for the alternate user who does dataset backup and restoration.

All users, including the owner, must correctly specify any existing permission control words corresponding to the mode of access desired. For example, suppose dataset BIG has a public access mode of READ and a read password of README. Any user desiring to read the dataset must supply the read password (README) to gain access to the dataset. An exception occurs if the permanent dataset utilities are used. Refer to section 11 for more information.

6.3.3.3 Dataset use tracking

The total access count and date/time of last access are recorded for each dataset in the DSC. Access tracking capabilities include recording who accessed the dataset, how many times, and the date/time of last access. The permit mechanism described earlier in this section provides access tracking whenever a permit is issued for a user. A dataset that allows public access can also be tracked. The owner must explicitly state, however, that public access tracking is required with the track accesses (TA) parameter on the ACQUIRE, SAVE, or MODIFY control statement; the system does not normally provide it.

6.3.3.4 Attribute association

The system allows permanent datasets having the same PDN and additional ID to be distinguished by an ED. That is, there can be several datasets with different edition numbers that have the same PDN, ID, and ownership value.

A user permanent dataset is uniquely identified by the PDN, ID, ED, and *ownership value*. The ownership value recorded in the DSC when a dataset is made permanent is normally equal to the user number as specified on the ACCOUNT or JOB control statement. Specific installations can choose to define dataset ownership as the account number rather than the user number. Contact your CRI site analyst to find out which type of ownership value is used.

Permanent mass storage datasets with the same PDN, ID, and ownership are assumed to be closely related. Therefore, most permanent dataset attributes are the same for all editions of the permanent dataset. The read, write, and maintenance permission control words, public access mode, public access tracking, and permits are the same for all datasets with the same PDN, ID, and ownership.

The text attribute is treated slightly differently. Any *text* supplied when the dataset is created is kept as a dataset attribute; if no *text* is supplied, the text attribute from the highest existing edition of the permanent dataset, if any, is used.

The notes attribute is treated similarly to text except that *notes* are assumed to be different for each dataset edition. *Notes* supplied at dataset creation time are used; if no *notes* are supplied, none are used.

Deleting the data in a permanent dataset while leaving the dataset's name and attributes recorded in the DSC is possible. Such a dataset is referred to as a *partially deleted* dataset. The subsection on Dataset Staging Control describes partial dataset deletion.

6.4 DATASET STAGING CONTROL

Staging is the process of transferring jobs and data in the form of COS datasets from a front-end computer system to Cray mass storage or of transferring datasets from Cray mass storage to a front-end computer system. Three control statements support staging datasets between COS and a front-end system: ACQUIRE, DISPOSE, and FETCH. Another control statement, SUBMIT, directs datasets to the COS input queue. Section 10 fully defines the following control statements:

Verb	Function
ACQUIRE	Checks to see if the requested dataset is currently permanent on mass storage. If the dataset is already permanent, ACQUIRE works exactly like ACCESS (described earlier in this section) and allows dataset access to the job making the request. Alternatively, if the dataset is not mass storage resident, ACQUIRE obtains a front-end resident dataset, stages it to Cray mass storage, and makes it permanent and accessible to the job making the request. The dataset is staged from the front end only if it is not already permanent.
DISPOSE	Directs a dataset to the specified queue for staging to a front-end system. DISPOSE can also be used to release a local dataset or to change dataset disposition characteristics.
FETCH	Obtains a front-end resident dataset and makes it local to the requesting job
SUBMIT	Directs a dataset on Cray mass storage local to the submitting job to the COS input queue

Dataset control information such as save or access codes is usually required by a front-end system for management of its own files. Such control information can be sent by the Cray system user to the front-end system through the use of the text parameter (expressed as TEXT=*text*), which is a special parameter of the SAVE, MODIFY, ACQUIRE, FETCH, and DISPOSE statements. The contents of the character string provided with the TEXT parameter are defined by the front-end system (refer to the appropriate station reference manual for the use of the TEXT parameter on your front-end system).

The *text* information not only provides most of the directives for obtaining the dataset from the front-end computer system but can contain sensitive or secure information as well. When using the ACQUIRE control statement, the staged dataset is recorded in the DSC and thus made permanent. Like any other mass storage permanent dataset, the staged dataset's attributes are recorded and protected as described under the Protecting and Accessing Mass Storage Datasets subsection earlier in this section.

The owner of an acquired dataset can provide permission to acquire the dataset to other users by specifying a public access mode or by issuing permits. The actual dataset (that is, the data) need not reside on mass storage for the permissions to be issued. For this reason the *text*, as specified by the owner when the dataset was initially acquired, is retained by the system as an attribute. The owner can, at a later date, delete the data while still retaining all of the permanent dataset attributes. A dataset registered in the DSC in this manner is referred to as a *partially deleted* dataset.

When an authorized user acquires a partially deleted dataset, the text required to obtain the dataset from the front-end computer system is retrieved from the DSC and sent along with the request. Therefore, the user need not specify the *text* in the ACQUIRE request. In fact, if the ACQUIRE is being issued by an alternate user as opposed to the owner, any *text* in the request is ignored. In this manner, the owner does not have to disclose the *text* information to other users.

The owner can at any time replace the *text* using the MODIFY command. After a partially deleted permanent dataset has been successfully acquired, the data is once again made permanent and is considered completely Cray mass storage resident. Because the dataset is mass storage resident, a subsequent ACQUIRE request is treated as an ACCESS request. The ACQUIRE request stages a dataset only if it is not already permanent on Cray mass storage.

6.5 PERMANENT DATASET UTILITIES

Three utilities (AUDIT, PDSDUMP, and PDSLOAD) can be used with any mass storage permanent datasets available to the user. Datasets processed by these utilities need not be local to the user job. The following utility routines are provided for mass storage permanent datasets:

Verb	Function
AUDIT	Produces a report containing status information for each permanent dataset. AUDIT does not include system input or output datasets.
PDSDUMP	Dumps all specified permanent datasets to a user-specified dataset. Input and output datasets managed by the operating system can be included in the dump.
PDSLOAD	Loads permanent datasets that have been dumped by PDSDUMP and updates or regenerates the DSC. Input and output datasets managed by the operating system can also be loaded with PDSLOAD.
RESTORE	Recalls a retired dataset to on-line disk
RETIRE	Declares a dataset retired

These utilities are defined in Section 11.

6.6 LOCAL DATASET UTILITIES

Utility control statements provide the user with a convenient means of copying, positioning, or initializing local datasets. The following utilities are available to the user:

Utility	Function
BLOCK	Converts an unblocked dataset to a blocked dataset
COPYD	Copies blocked datasets
COPYF	Copies files of blocked datasets
COPYR	Copies records of blocked datasets
COPYU	Copies unblocked datasets or sectors of unblocked datasets

Utility	Function
NOTE	Writes text to a dataset
QUERY	Returns local mass storage dataset status and position information
REWIND	Positions a blocked or unblocked dataset at beginning-of-data, that is, before the first word of the dataset
SKIPD	Skips blocked datasets
SKIPF	Skips files of blocked datasets
SKIPR	Skips records of blocked datasets
SKIPU	Skips sectors of unblocked datasets
UNBLOCK	Converts a blocked dataset to an unblocked dataset
WRITEDS	Initializes a blocked random or sequential dataset

Section 12 describes these utilities.

6.7 ANALYTICAL AIDS

The following control statements provide analytical aids to the programmer.

Verb	Function
COMPARE	Compares two blocked datasets and lists all differences
DSDUMP	Dumps all or part of a blocked or unblocked dataset
DUMPJOB DUMP	DUMPJOB and DUMP are generally used together to examine the contents of registers and memory as they were at a specific time during job processing. DUMPJOB captures the information so that DUMP can later format selected parts of it.
FLODUMP	Dumps flowtrace tables when a program aborts with flowtrace active
FTREF	Generates information about a Fortran application

Verb	Function
ITEMIZE	Inspects and generates statistics about library datasets. Section 5 describes libraries; the Object Library Management subsection that follows describes dataset management.
PRINT	Writes the value of a JCL expression (as defined in section 16) to the logfile
SYSREF	Generates a global cross-reference listing for one or more CAL or APML programs

Section 13 describes these control statements.

6.8 EXECUTABLE PROGRAM CREATION

Two utilities are available under COS to prepare programs for execution. The segment loader (SEGLDR) is described in the Segment Loader (SEGLDR) Reference Manual, CRI publication SR-0066; the COS relocatable loader (LDR) is described in section 14. These utilities prepare programs for execution from *relocatable modules*. A series of *relocatable modules* is normally created when a program is compiled or assembled. Each relocatable module normally represents one subroutine of the whole program, or the main program itself. Each relocatable module (also known as a *module*, an *object module*, a *relocatable*, or a *binary*) consists of a series of tables. The tables contain such information as executable machine (program) instructions, references to other modules (such as when one subroutine calls another), and the location of where the main program is to start execution.

Before a collection of relocatable modules (the program) can be executed, the collection of modules must be linked together into a single module. This single module, the *absolute load module*, contains the main program and a copy of every subroutine called, including ones found in the various system libraries. An absolute load module can be executed any time without having to be reprocessed by SEGLDR or LDR. The loaders execute as utility programs within the user field and provide the loading and linking in memory of relocatable modules from datasets on mass storage.

Very large programs might not fit in the available user memory space or might not use large portions of memory while other parts of the program are in execution. For such programs, both loaders provide overlay capabilities. With SEGLDR, these are called segments; with LDR, overlays. Creating and using segments requires no source code changes; creating and using overlays requires source code to be changed to invoke the overlay processor.

In general, the capabilities (except overlays) that are available with LDR are available with SEGLDR. Most applications that use more than 4 Mwords of Central Memory, however, cannot be loaded by LDR because of internal limitations of its memory allocation algorithm. Such programs must use SEGLDR. SEGLDR also provides additional features not available with LDR. The LD2 utility assists in conversion from LDR to SEGLDR; LD2 is described in section 14.

6.9 OBJECT LIBRARY MANAGEMENT

BUILD, a utility called through the BUILD control statement, creates and maintains object libraries.

Compiled subroutines (relocatable modules) can be collected into libraries that can be referred to later when creating a new program. COS provides several standard object libraries (refer to the Programmer's Library Reference Manual, CRI publication SR-0113, for a description of the standard library routines available).

Any number of object libraries can be created, however, in addition to the ones supplied with COS.

Library datasets are designed primarily to provide the Relocatable Loader (refer to previous subsection) with a means of rapidly locating and accessing program modules. A *library dataset* is a dataset containing a program file followed by a directory file. The program file is composed of loader tables for one or more absolute or relocatable program modules. The directory file contains an entry for each program module.

Section 15 describes BUILD.

JOB DEFINITION AND CONTROL 7

Several control statements let you specify job processing requirements. This section contains the specifications for the following control statements used in defining a job and its operating characteristics to the operating system:

Control Statement	Function
* (Comment)	Allows the annotation of job control statements
ACCOUNT	Provides privacy and security; also provides accounting information for the installation.
CALL	Instructs COS to begin reading control statements from an alternate dataset
CHARGES	Monitors a job's usage of computer resources
ECHO	Controls the message classes to be written to your logfile
EXIT	Indicates the end of the control statement processing or the point in the control statement file where processing of control statements resumes following a job step abort
IOAREA	Controls access to your Dataset Parameter Area (DSP) and I/O buffers
JOB	Defines the job to COS
LIBRARY	Specifies the library datasets to be searched during the processing of control statement verbs
MEMORY	Requests a new field length, mode of field length reduction, or both
MODE	Sets or clears mode flags in the Exchange Package
NORERUN	Permits COS to recognize functions that would make a job rerunnable
OPTION	Specifies the format of the job's listing, selects the processor to be used, and specifies the level of statistics to gather on datasets

Control Statement	Function
RERUN	Declares a job to be rerunnable or nonrerunnable
RETURN	Returns control to the caller
ROLLJOB	Protects a job by writing it to disk
SET	Changes the value of a job control language (JCL) symbol
SWITCH	Sets or clears sense switches
TARGET	Sets CPU characteristics

7.1 * - COMMENT STATEMENT

The comment control statement is a system verb that you can use to annotate a job with comments. A terminator is not required for such statements. There are no parameters.

Format:

```
 _____________________
|                     |
| *  comment text     |
|_____________________|
```

7.2 ACCOUNT - VALIDATE USER NUMBER AND ACCOUNT

The ACCOUNT control statement validates the job's user number, user password, account number, and account password. A job is processed only if the user number/password pair and the account number/password pair (if specified) are valid. As implied by its name, the ACCOUNT control statement provides accounting data for the installation. In addition, privacy and security are ensured through the use of ACCOUNT parameters.

The ACCOUNT statement declares the user's account and charge numbers to COS. It must immediately follow the JOB control statement if the installation has defined accounting or security as mandatory. Only one ACCOUNT statement is allowed per job.

If the job is interactive and accounting is mandatory, the ACCOUNT statement must be the first statement entered in a session. If accounting is not mandatory, the first statement entered, a prompt is issued requesting the ACCOUNT statement. A similar prompt is issued if syntax errors are made on the ACCOUNT statement.

NOTE

ACCOUNT control statement parameters do not appear with the ACCOUNT control statement in the job logfile.

The installation generally sets up AC, APW, US, and UPW parameters. The user, however, specifies NAPW and NUPW. Including a new account password provides the user accounting protection, because only the person who knows the NAPW can run a job under a given user's account number. NUPW is available as an additional security check. Therefore, NAPW and NUPW values should be known only to the individual user who specifies them.

Format:

```
ACCOUNT,AC=ac,APW=apw,NAPW=napw,US=us,UPW=upw,NUPW=nupw.
```

AC=*ac* Account number. 1 to 15 alphanumeric characters assigned to the user. This number identifies the user for accounting purposes and is a required parameter. The account number is not the same as the user number on the JOB control statement unless the site chooses to use the same number.

APW=*apw* Account password. 1 to 15 alphanumeric characters or null. A password must be specified if the installation has made the password mandatory.

NAPW=*napw* New account password. 1 to 15 alphanumeric characters or null. This new password replaces the old account password if the account number/password pair given by the AC and APW parameters is valid. NAPW may be specified without a value to change the account password to null. To change an account password, you must specify the keyword APW with the old password and NAPW with the new password.

US=*us* User number. 1 to 15 alphanumeric characters assigned to the user. This number identifies the user for system access purposes and is a site-optional parameter. The user number is not the same as the account number unless the site chooses to use the same number for both. This parameter, if specified, overrides the user number on the JOB control statement. If US is not specified on the ACCOUNT control statement, the user number on the JOB statement is used by COS.

UPW=*upw* User password. 1 to 15 alphanumeric characters. A password must be specified if your site has made security checking mandatory.

NUPW=*nupw* New user password. 1 to 15 alphanumeric characters. This new password replaces the old user password *upw* if the user number/password pair given by the US and UPW parameters is valid.

7.3 CALL - READ CONTROL STATEMENTS FROM ALTERNATE DATASET

The CALL control statement tells COS to begin reading control statements from the first file of the dataset specified as a parameter to CALL. CALL can appear anywhere in the control statement file. Nesting of CALL statements to seven levels is allowed. COS reads and processes the control statements from the specified dataset until it encounters an end-of-file (EOF) or a RETURN statement. Control then reverts to the dataset that contained the CALL control statement. CALL rewinds the dataset before reading it.

The dataset that is called can contain either simple control statements or a procedure definition. Simple control statements are executed without any parameter substitution. On the other hand, parameter substitution is possible when the dataset that is called contains a procedure definition. The optional CNS parameter on the CALL statement allows COS to determine the form of control statements used. If CNS is not present, the statements on the dataset are assumed to be simple control statements and they are executed exactly as read from the dataset, beginning with the first statement.

If CNS is present on the CALL statement, the control statements on the dataset are treated as a procedure definition. This means that parameter substitution can be performed before executing the statements. In this case, the first statement is assumed to be a prototype statement and subsequent statements are the procedure body definition. If the dataset contains a procedure definition, the dataset is closed after parameter substitution and before invocation of the procedure.

If the dataset contains a procedure definition, the PROC and ENDPROC statements must not enclose the definition, unlike a procedure defined in-line within a control statement file. The PROC and ENDPROC statements may appear within the definition. Any statement enclosed by PROC and ENDPROC becomes a procedure definition that is included in the $PROC system procedure dataset when the enclosing procedure is invoked by a CALL statement. The enclosing procedure is not added to the $PROC dataset.

When the CNS option is used and the procedure definition contains a nested PROC/ENDPROC sequence, the parameter substitution performed according to the prototype statement for the outermost procedure definition (the first statement of the dataset) is also performed on all nested definitions. This can produce warning messages if the inner definitions use keywords or positional parameters different from those specified for the outer definition. The nested definitions are written to $PROC with all matching substitutions performed and all nonmatching substitutions retained in the original form.

CALL is a system verb.

Format:

```
CALL,DN=dn[,CNS].
```

DN=*dn* Begin reading control statements from this dataset. This is a required parameter.

CNS Crack next statement. This is an optional parameter. If present, the first statement on the dataset named by DN is treated as the prototype statement for the procedure whose body is defined by the remaining statements in the first file of the dataset, and the next statement in the control statement dataset containing the CALL statement is read by COS and treated as an invocation of the procedure. Parameters supplied on that statement are substituted according to the rules of parameter substitution described in section 16.

Example 1:

Use of CALL without CNS

Assume that dataset X contains the following control statements:

```
ACCESS,DN=A,PDN=B,UQ.
DELETE,DN=A.
RELEASE,DN=A.
```

If dataset B has been saved, the result of the statement

```
CALL,DN=X.
```

would be

```
ACCESS,DN=A,PDN=B,UQ.
PD000 - PDN = B            ID =      ED =  1   OWN = ABC
PD001 - ACCESS COMPLETE
DELETE,DN=A.
PD000 - PDN = B            ID =      ED =  1   OWN = ABC
PD001 - DELETE COMPLETE
RELEASE,DN=A.
```

Example 2:

Use of CALL with CNS

Assume the contents for dataset X are the same as in example 1. The result of the statement

```
CALL,DN=X,CNS.
```

would be

```
ACCESS,DN=A,PDN=B,UQ.
CS109 - POSITIONAL PARAM. AFTER KEYWORDS IN PROTOTYPE: UQ
*,DN=A.
CS122 - NO VALUE WAS ASSIGNED TO UQ
AB025 - USER PROGRAM REQUESTED ABORT
AB000 - JOB STEP ABORTED.    P = 00000743b
```

In this case, the CNS parameter causes COS to consider the ACCESS statement to be a prototype statement; the DN, PDN, and UQ keywords are assumed to be the identifiers of substitutable parameters.

Example 3:

Valid CALL with CNS without nested definitions

Assume that the contents of dataset X are the following:

```
D,A,B.
ACCESS,DN=&A,PDN=&B,UQ.
DELETE,DN=&A.
RELEASE,DN=&A.
```

If the permanent dataset EXAMPLE exists, the result of the statements

```
CALL,DN=X,CNS.
*,DS,EXAMPLE.
```

would be

```
ACCESS,DN=DS,PDN=EXAMPLE,UQ.
PD000 - PDN = EXAMPLE      ID =      ED =  1   OWN = ABC
PD001 - ACCESS COMPLETE
DELETE,DN=DS.
PD000 - PDN = EXAMPLE      ID =      ED =  1   OWN = ABC
PD001 - DELETE COMPLETE
RELEASE,DN=DS.
```

Example 4:

CALL with a nested PROC/ENDPROC definition

Assume that dataset X contains the following statements:

```
D,A,B.
PROC.
A,Q,B.
ACCESS,DN=&Q,ID=&B.
ENDPROC.
ACCESS,DN=&A,ID=&B,UQ.
DELETE,DN=&A.
RELEASE,DN=&A.
```

If permanent dataset Z with ID D exists, the result of the statements

```
CALL,DN=X,CNS.
*,Z,D.
```

would be

```
CS125 - NO SUCH FORMAL PARAMETER: Q
<DEFINITION>    PROC.
<DEFINITION>    A,Q,B.
<DEFINITION>    ACCESS,DN=&Q,ID=D.
<DEFINITION>    ENDPROC.
ACCESS,DN=Z,ID=D,UQ.
PD000 - PDN = Z            ID =      ED =  1   OWN = ABC
PD001 ACCESS COMPLETE
DELETE,DN=Z.
PD000 - PDN = Z            ID =      ED =  1   OWN = ABC
PD001 - DELETE COMPLETE
RELEASE,DN=Z.
```

The $PROC dataset would contain a procedure with the following definition:

```
A,Q,B.
ACCESS,DN=&Q,ID=D.
```

The &B in the original definition was replaced by the value that was specified for the corresponding parameter B in the outermost procedure. The &Q was retained, because there was no corresponding replacement in the outermost procedure.

7.4 CHARGES - JOB STEP ACCOUNTING

The CHARGES control statement lets you monitor the computer resources used by your job up to a specific point in the job. Hence, CHARGES can be used for either partial or total resource reporting.

Partial reporting occurs when parameters are specified on the CHARGES control statement and usage statistics for the computer resources specified on the CHARGES statement are given for the job steps preceding the CHARGES statement. The statistics are placed in the user log and the system log.

Total reporting occurs when usage statistics are obtained for all the resources in all the available resource groups. The summary is placed in the user log and the system log.

CHARGES is automatically invoked when a job terminates so that usage statistics of the entire job are reported.

Format:

```
CHARGES,SR=options.
```

SR=*options*

System resources used. Any one or more of the following groups of resources can be specified. Options are separated by colons. The default is a listing of the job's usage of resources in all of the following groups:

CPU Time executing in CPU, I/O waiting time, and time waiting for CPU. CPU gives the totals for the entire job.

DS Permanent dataset space accessed, permanent dataset space saved, temporary dataset space used, disk sectors moved, fast secondary storage (FSS) sectors moved (SSD or Buffer Memory), user I/O requests, memory-resident datasets used, number of OPEN calls, and number of CLOSE calls

FSU FSS usage. An FSS device is either an SSD or the Buffer Memory in the IOS. When a job uses an FSS device not configured as a generic resource, the FSU option reports device usage. The option reports the following information in the user log and system log:

Device name
Maximum concurrent allocation
Unit allocation integral (sector*sec)
Number of sectors transferred

GRU Generic resource usage. For each generic resource named on the JOB control statement, the following information appears in the user log and system log:

Generic resource name
Device type (tape, disk, or ISP)
Job limit
Maximum concurrent allocation
Unit allocation integral (tape unit*sec or sector*sec)
Number of sectors transferred
Number of tape blocks transferred
Number of tape volumes mounted

JNU Jobname and user number

JSQ Job sequence number

MM Minimum job size (words), maximum job size (words), execution-time memory usage in million words/second, I/O wait-time memory usage in million words/second, maximum field length used (words), minimum field length used (words), maximum JTA used (words), and minimum JTA used (words)

NBF Number of 512-word blocks (sectors) received from a front end and number of 512-word blocks (sectors) queued to a front end

TASK Time executing in CPU, I/O wait time, and time waiting for CPU. The TASK option breaks down the time information according to user task number, and provides a total for the entire job.

WT Time waiting in the input queue before beginning execution

7.5 ECHO - ENABLE OR SUPPRESS LOGFILE MESSAGES

The ECHO control statement controls the message classes written to your logfile by turning them ON or OFF. ECHO may be used more than once during a job to toggle the printing or suppression of message classes. ECHO is a system verb. ON is the default at the start of a job.

The keywords ON and OFF may be used in any combination. Ensure that the classes specified do not overlap between the keywords, however, and that both defaults are not included.

Format:

ECHO,ON=$class_1$:...:$class_n$,OFF=$class_1$:...:$class_n$.

ON=$class_i$ When a COS or a program issues messages, they are written to your logfile in the classes specified. If any other classes were specified but not turned off by this statement, the union of the two sets of classes is enabled. If the ECHO control statement contains only the keyword ON or ON=ALL, all messages are written to the logfile.

OFF=$class_i$
Messages in the classes specified are not written to the job's logfile. If any other classes were specified but not turned on by this statement, the union of the two sets of classes is suppressed. If the ECHO control statement contains only the keyword OFF or OFF=ALL, all messages in defined classes are suppressed.

Messages that are not classified may not be turned off.

The operating system recognizes the following classes:

Class	Description
ABORT	ABxxx and system traceback messages that COS issues when a job fails
JCL	Messages that originate in the job's JCL input file
PDMERR	Error messages produced by PDM
PDMINF	Dataset information messages produced by PDM

When a job calls a procedure, the echo state of the job is the same upon return from the procedure as before, even though the procedure may use a different echo state. The following occurs when ECHO is used with CALL and procedure invocations:

- The echo state of the caller is saved so that on return to the caller the same state is in effect as before the call.
- When the procedure includes an ECHO statement, the new echo state is in effect only for the duration of the procedure. If the procedure does not include an ECHO statement, the echo state of the caller is in effect.

7.6 EXIT - EXIT PROCESSING

An EXIT control statement points to the place in the control statement file where processing of control statements resumes following a job step abort from a program. If no job step abort occurs, the EXIT control statement indicates the end of control statement processing. EXIT is a system verb. It has no parameters.

Format:

```
 _______
|       |
| EXIT. |
|_______|
```

7.7 IOAREA - CONTROL USER'S ACCESS TO I/O AREA

The IOAREA control statement locks or unlocks that portion of the user field containing the user's DSP and I/O buffers. Locking denies the user access, unlocking allows the user access. This area follows the High Limit Memory (HLM) address of the user field. The user of the stack version of the COS libraries needs to note that IOAREA does not protect I/O buffers or DSPs that have been allocated within the user's stack space. IOAREA is a system verb.

Format:

```
IOAREA,LOCK   .
       UNLOCK
```

LOCK
UNLOCK

The keywords LOCK and UNLOCK are mutually exclusive. A parameter must be specified on the control statement. When the control statement is not used, the user's I/O area is assumed to be unlocked.

If LOCK is selected, the system sets the limit address to the base of the DSPs, thereby denying direct access to the user's DSP area and I/O buffers. When the I/O area is locked, the library I/O routines make a system request to gain access to the I/O area. Although the system request introduces additional overhead in job processing, it should prevent accidental destruction of the I/O area.

If UNLOCK is selected, the system sets the limit address to the value specified in JCFL, allowing access to the user's DSP area and I/O buffers.

7.8 JOB - JOB IDENTIFICATION

The JOB control statement defines the job to COS and must be the first statement in a control statement file. The JOB control statement cannot be continued, and no leading blanks are allowed. JOB is a system verb.

Format:

```
JOB,JN=jn,MFL=fl,T=tl,P=p,US=us,OLM=olm,CL=jcn,gn=nr,S.
```

JN=*jn* Job name; 1 to 7 alphanumeric characters. This name identifies the job and its subsequent output. JN is a required parameter.

MFL=*fl*† Maximum field length allowed the job, in 64-bit words. The job's maximum field length is set to the greater of *fl*, rounded up to the nearest multiple of 512 words, or the amount needed to load the Control Statement Processor (CSP). The job is aborted if the maximum field length is greater than the system maximum.

If this parameter is omitted, the maximum field length is set by the site parameter.

If MFL is present without a value, the field length is the system maximum. The system maximum is the smaller of the total amount of memory available after COS is initialized minus the job's JTA size (refer to section 1) or an installation-defined maximum job field length.

T=*tl* Time limit in seconds that the job may run. If this parameter is omitted, the time limit is set to a value determined by an installation parameter. If T is present without a value, a maximum of 16,777,215 (approximately 194 days) is allowed.

P=*p* Priority level 0 to 15 at which the job enters the system. If P is 0, the job is not initiated. If omitted, a value specified by the installation is assumed.

US=*us* User number; 1 to 15 alphanumeric characters. The default is no user number. This parameter identifies the user submitting the job. Specific usage is installation defined.

OLM=*olm* Maximum size of $OUT. *olm* is a count of 512-word blocks. A block holds about 45 print lines. The installation defines the default and maximum values for *olm*.

CL=*jcn* Name of the installation-defined job class that this job fits in; 1 to 7 alphanumeric characters. The job is aborted if it does not fit the requirements of its class or if the class does not exist. The default is no class name.

gn=nr Type and number of dedicated resources required by a job. *gn* is a generic resource name of 1 to 7 alphanumeric characters. A generic resource name corresponds to a device type. For example, a generic name of SSD could be given to

† The *fl* parameter on the JOB statement excludes the job's Job Table Area (JTA); space for the JTA is added by the system.

gn=nr (continued) an SSD. Site administration defines generic names. COS provides one generic name (*TAPE, which refers to a dual density tape unit capable of 1600 or 6250 b/i), but sites may define up to 16 generic names. Contact your CRI site analyst for the generic names used at your site.

nr is the decimal number of units of the specified resource type. If *gn* refers to a tape device type, *nr* is the number of tape units to be used concurrently. If *gn* refers to a disk device type, *nr* is the decimal number of sectors required. The default is 0. A job is initiated only when the amount of each resource reserved is eligible for use. The job is aborted if it attempts to access more resources than are reserved with the JOB control statement.

S System job. This is a privileged parameter that designates the job as a system job. Privileges are verified during account processing.

7.9 LIBRARY - LIST AND/OR CHANGE LIBRARY SEARCHLIST

The LIBRARY control statement lets you specify the library datasets that will be searched during the processing of control statement verbs. LIBRARY may also be used to list the current or new searchlist to the logfile for verification.

When modifying the searchlist, the current members of the searchlist can be retained in the new searchlist by including an asterisk in the LIBRARY control statement. The asterisk corresponds to all members of the current searchlist in their present order. If the asterisk is omitted, the new searchlist contains only the library dataset names identified on the LIBRARY control statement. LIBRARY is a system verb.

When a job initiates, the default library searchlist consists of the library dataset.

Format:

```
LIBRARY,DN=dn1:dn2...:dnn,V.
```

DN=dn_i Library dataset names that will be part of the new library searchlist. A maximum of 64 names can be given. The order in which they appear on the control statement is

DN=dn_i (continued) the order in which they are searched. An asterisk included in the list signifies the current searchlist members are to be part of the new searchlist in their current order.

V For verification, list the current library searchlist on the logfile. When specified along with the new searchlist, the new searchlist is listed.

7.10 MEMORY - REQUEST MEMORY CHANGE

The MEMORY control statement lets you request a new field length, change the mode of field length reduction, or both. Section 3 discusses job memory management. MEMORY is a system verb.

You must specify at least one parameter for the MEMORY control statement.

Format:

```
MEMORY[,FL=fl][,USER].
              [,AUTO]
```

FL=*fl* Field length. *fl* specifies the number of words to be allocated to the job. If FL is specified without a value, the new field length is set to the maximum allowed the job.

USER Field length reduction is managed by the user (user mode)

AUTO Field length reduction is managed by the system (automatic mode)

The field length is set to the larger of the requested amount rounded up to the nearest multiple of 512 words or the smallest multiple of 512 decimal words large enough to contain the user code/data, LFT, DSP, and buffer areas. Field length management is in user mode for the duration of the next job step.

When the USER parameter is specified, the job is placed in user mode until a subsequent request is made to return it to automatic mode. When the AUTO parameter is specified, the job is placed in automatic mode.

The job step is aborted if completing the request results in a field length greater than the maximum allowed the job. The maximum is the smaller of the total number of words available to user jobs minus the job's JTA or the amount determined by the MFL parameter on the JOB control statement.

Examples:

```
MEMORY,FL,USER.
```

The job's field length is set to the maximum allowed and the job is placed in user mode until an explicit request is made to return it to automatic mode.

```
MEMORY,AUTO.
```

The job is returned to automatic mode. Its field length is reduced at the next job step.

```
MEMORY,FL=28988.
```

The field length is adjusted. If the job is in user mode by explicit user request, no change in mode occurs; otherwise, the job is placed in user mode for the duration of the next job step.

```
MEMORY,FL=28988,AUTO.
```

The field length is adjusted and the job is placed in user mode for the duration of the next job step. After the next job step, the job is put in automatic mode.

7.11 MODE - SET OPERATING MODE

The MODE control statement sets or clears mode flags in the Exchange Package for the job. MODE is a system verb.

Format:

```
MODE,FI=option,BT=option,EMA=option,AVL=option,ORI=option.
```

FI=*option* Floating-point interrupt mode. *option* can be either of the following:

ENABLE	Enables floating-point error interrupts; default.
DISABLE	Disables floating-point error interrupts; floating-point errors are ignored.

BT=*option* Bidirectional transfer mode. The BT parameter is used on CRAY X-MP series computer systems only. *option* can be either of the following:

ENABLE	Enable bidirectional memory transfers; default.
DISABLE	Disable bidirectional memory transfers; block reads and writes are not performed concurrently.

EMA=*option*

Extended memory addressing mode. The EMA parameter is used on CRAY X-MP† series computer systems only; it causes an abort on CRAY-1 systems. *option* can be either of the following:

ENABLE	Enables extended memory addressing
DISABLE	Disables extended memory addressing; the default is an installation option released as EMA=DISABLE. On the CRAY X-MP model 48, the default is released as EMA=ENABLE.

AVL=*option*

Second vector logical functional unit mode. The AVL parameter is used on CRAY X-MP† series computer systems only; it causes an abort on CRAY-1 systems. *option* can be either of the following:

ENABLE	Makes two logical functional units available, the first of which shares reservation logic with the vector floating multiply unit.
DISABLE	Makes only one vector logical unit available. The vector multiply reservation path is not shared; default is an installation parameter released as AVL=DISABLE.

ORI=*option*

Operand range error interrupt mode. The ORI parameter is used on CRAY X-MP series computer systems only; *option* can be either of the following:

ENABLE	Enables interrupts on operand range errors; default.
DISABLE	Disables interrupts on operand range errors

† Not available on all CRAY X-MP systems. Check with a CRI site analyst to determine if this feature is available.

7.12 NORERUN - CONTROL DETECTION OF NONRERUNNABLE FUNCTIONS

The NORERUN control statement specifies whether COS is to recognize functions that would make a job rerunnable. The current rerunnability of the job is not affected. NORERUN is a system verb.

Format:

```
NORERUN,ENABLE .
        DISABLE
```

ENABLE
DISABLE
: The keywords ENABLE and DISABLE are mutually exclusive. The default for the system as released is NORERUN,ENABLE; however, this is an installation option.

  ENABLE instructs the system to begin monitoring functions performed by the job and to declare the job nonrerunnable if any of the nonrerunnable functions are performed.

  DISABLE instructs the system to stop monitoring functions for nonrerunnable operations. If a job has already been declared to be nonrerunnable, specifying DISABLE does not make the job rerunnable again.

7.13 OPTION - SET USER-DEFINED OPTIONS

The OPTION control statement specifies user-defined options, such as the format of the job's listing. OPTION is a system verb.

Format:

```
OPTION[,LPP=n][,PN=p/ANY][,STAT=ON/OFF].
```

LPP=*n*
: Number of lines per page (0 through 255) for a job listing. If 0 is specified, the current number of lines per page is not changed. The default is an installation parameter.

  This value is used by CRI products that do pagination and is available to user software through the GETLPP subroutine call. It has no effect on I/O processing from user jobs that do not perform their own pagination.

PN=*p*
ANY

Select processor. Select a processor by specifying its number as the argument. Use ANY to indicate that any processor is acceptable. The default is ANY.

If the processor specified by *p* is not available (because it does not exist on the mainframe or is inoperative), an error message appears and the job aborts.

ON
STAT=OFF

Specifies the level of I/O statistics gathered for datasets local to the job. The statistics appear on the user logfile when the dataset is released. The statistics can be on two levels:

- User level statistics (sometimes called accounting information) that identify the type of system requests you made for the dataset.
- System level statistics (sometimes called device information) that indicate how the system handled the requests device by device.

The options are as follows:

ON — User information as defined by the site or, if not defined by the site, as determined by the preset categories of USR. ON is the default if STAT is specified without an option.

OFF — No statistics. OFF is the default if STAT is not specified on the OPTION control statement.

The output is a logfile message of one or more lines with the following format:

```
SY005 - ldn   xWRDS,   xIOS,   xREQ,   xSECTRS,   xx.xxSEC
        ldv   xSECTRS  mode:   xREQ,   xSECTRS,   xx.xxSEC
```

The first line of the message reports the following user-level information (it is issued when STAT equals ON):

ldn — Local dataset name

xWRDS — Size of the dataset in words (decimal)

xIOS — Number of I/O suspensions performed for the dataset by F$RCL

*x*REQ Number of the start I/O requests (FWDC, FRDC, and F$QIO) resulting in queue manager requests

*x*SECTRS Number of sectors moved as a result of the FWDC, FRDC, F$BIO, and F$QIO requests

*xx.xx*SEC Time in seconds that the job spent in I/O suspension waiting for the dataset

Subsequent lines in the message report system level information. Each line corresponds to an I/O transmission to a device on which the dataset resides. A line appears for every device on which the dataset has space allocated. The lines contain the following information:

ldv Logical device name (optional)

*x*SECTRS Number of sectors allocated on the device for the dataset (optional)

mode Direction of I/O data transfer requests: READ or WRITE

*x*REQ Number of data transfer requests issued to the device driver

*x*SECTRS Number of sectors moved as a result of the data transfer requests

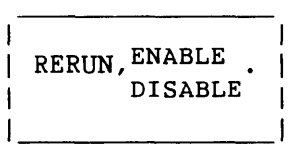
*xx.xx*SEC Time in seconds that the system (queue manager) waited for the device driver to respond to the data transfer requests

STAT gathers I/O statistics on every dataset created or accessed after STAT is specified on the OPTION control statement. The level of statistics gathering that is in effect at the time a dataset is created with ASSIGN or accessed with ACCESS remains in effect until the dataset is released, regardless of subsequent changes to STAT.

7.14 RERUN - UNCONDITIONALLY SET JOB RERUNNABILITY

The RERUN control statement unconditionally declares a job to be either rerunnable or nonrerunnable. If RERUN is used to declare a job rerunnable, the subsequent execution of a nonrerunnable function may cause the system to declare the job nonrerunnable, depending on whether a NORERUN control statement or macro is also present. The RERUN control statement does not affect the monitoring of the user job for nonrerunnable functions. RERUN is a system verb.

Format:



ENABLE
DISABLE — The keywords ENABLE and DISABLE are mutually exclusive. If no parameter is specified on the control statement, the installation option determines if the job is to be rerunnable; the default for the system as released is RERUN,ENABLE.

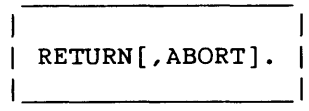
If ENABLE is selected, the system is instructed to consider the job to be rerunnable, regardless of previously executed functions.

If DISABLE is selected, the system marks the job not rerunnable, regardless of previously executed functions.

7.15 RETURN - RETURN CONTROL TO CALLER

The RETURN control statement returns control to the caller. The caller can be a procedure or the job's control statement file. Processing resumes with the caller's next control statement. A RETURN control statement can be embedded anywhere within the called procedure. A RETURN control statement does not have to be placed at the end of the procedure, however, because an EOF record is interpreted as the control statement sequence of an EXIT, RETURN, and RETURN,ABORT. A RETURN encountered in the primary control statement file is ignored. RETURN is a system verb.

Format:



ABORT — After returning to the previous control statement level, ABORT causes COS to issue a job step abort.

7.16 ROLLJOB - ROLL A USER JOB TO DISK

The ROLLJOB control statement protects a job by writing it to disk at any point in its execution so that it can be recovered at that point in the event of a system interruption. The use of ROLLJOB does not guarantee that a job will remain recoverable. It merely ensures that at the current stage there is a recoverable image. Subsequent job activity may invalidate this image. Performing ROLLJOB does not make a job recoverable that has on-line tape datasets accessed.

ROLLJOB is a system verb. There are no parameters.

Format:

```
 ___________
|           |
| ROLLJOB.  |
|___________|
```

7.17 SET - CHANGE SYMBOL VALUE

The SET control statement changes the value of a valid job control language symbol. Valid symbols are those you classify as alterable (U) in table 16-1. A job-step abort occurs if a symbol included in a SET control statement is unknown to the system, can be set only by COS, or is a constant. SET is a system verb.

Format:

```
 ___________________________
|                           |
| SET(symbol=expression)    |
|___________________________|
```

symbol A valid symbol that you can alter

expression
A valid arithmetic, logical, or literal assignment expression. It may be delimited with parentheses to simplify interpretation during control statement evaluation.

Examples:

This example increases the procedure-local register J1 by 1.

```
SET(J1=J1+1)
```

The global register G1 is given an ASCII value that is the low-order 2 characters from the current system revision level (COS *X.XX*).

```
SET(G1=(SYSID.AND.177777B))
```

The global register G3 is assigned a value, depending on the current values of ABTCODE and G2.

```
SET(G3=((ABTCODE.EQ.74).AND.(G2.EQ.0)))
```

7.18 SWITCH - SET OR CLEAR SENSE SWITCH

The SWITCH control statement turns pseudo-sense switches on or off. SWITCH is a system verb.

Format:

```
 _____________
|             |
| SWITCH,n=x. |
|_____________|
```

n Number of switch (1 to 6) to be set or cleared

x Switch position:
- ON Switch *n* is turned on; set to 1.
- OFF Switch *n* is turned off; set to 0.

7.19 TARGET - SPECIFY CPU CHARACTERISTICS

The TARGET control statement:

- Reports the current default settings for CPU characteristics in the job's machine specification table
- Changes the current default settings for the CPU for the job's target machine specification table

The CPU can be any of the following:

- *HOST, the machine on which the job is running
- *TARGET, a site-specified target machine
- A named CPU

At job initiation the *HOST and *TARGET settings are preset to those of the machine on which the job resides. The *TARGET settings can be altered by the user. The actual *HOST and named-CPU characteristics cannot be changed, but a copy of those settings becomes the *TARGET specifications and can be altered.

The characteristics set by TARGET remain in effect for a job until they are changed by another TARGET command or a library request. TARGET is a system verb.

Format:

```
TARGET,CPU=cpu[:EMA|:NOEMA][:CIGS|:NOCIGS][:VPOP|:NOVPOP][:PC|:NOPC]
   [:READVL|:NOREADVL][:VRECUR|:NOVRECUR][:AVL|:NOAVL][:HPM|:NOHPM][:STATRG|:NOSTATRG]
   [:BDM|:NOBDM][:BANKS=banks][:NUMCPUS=numcpus][:IBUFSIZE=ibufsize]
   [:MEMSIZE=memsize][:MEMSPEED=memspeed][:CLOCKTIM=clocktim]
   [:NUMCLSTR=numclstr][:BANKBUSY=bankbusy][,VERIFY=*HOST|*TARGET].
```

CPU=*cpu* Identification of the CPU whose characteristics are to be reported or changed. *cpu* can be *HOST, *TARGET, or a named CPU. The named CPU can be any one of the following:

CRAY-1	CRAY-XMP	CRAY-1A
CRAY-X1	CRAY-1B	CRAY-X2
CRAY-1M	CRAY-X4	CRAY-1S

The CPU parameter is required except when VERIFY is the only parameter specified.

The following parameters that have a NO prefix indicate that the characteristic is not available.

EMA
NOEMA Extended memory addressing

CIGS
NOCIGS Compressed index, gather/scatter

VPOP
NOVPOP Vector population count

PC
NOPC Programmable clock

READVL
NOREADVL Read vector length

VRECUR
NOVRECUR Vector recursion

AVL
NOAVL Additional vector logical functional unit

HPM
NOHPM Hardware performance monitor

STATRG
NOSTATRG Status register

BDM
NOBDM Bidirectional memory

BANKS=*banks*
Number of memory banks

NUMCPUS=*numcpus*
Number of CPUs

IBUFSIZE=*ibufsize*
Instruction buffer size

MEMSIZE=*memsize*
Memory size in words. The words can be expressed as follows (the # represents a number):

#	Words
#K	Words multiplied by 1024
#M	Words multiplied by 1,048,576

Thus, the following values are equal: 1,048,576, 1024K, and 1M.

MEMSPEED=*memspeed*
Memory speed in clock periods

CLOCKTIM=*clocktim*
Clock period in integer picoseconds (10**-12)

NUMCLSTR=*numclstr*
Number of clusters

BANKBUSY=*bankbusy*
Number of clock periods that the memory bank has reserved

```
       *HOST
VERIFY=*TARGET
```
Logfile report of the current settings of *HOST or *TARGET. VERIFY can be the first parameter or the last. If VERIFY is specified without a value, the default is *TARGET.

Example 1:

In this use of TARGET, the only parameter on the control statement requests a report of the current settings for the target machine, a CRAY X-MP computer system. The report follows:

```
TARGET,VERIFY=*TARGET.

TA005 - Primary machine type is:  CRAY-XMP
TA006 - BANKS     =       64
TA006 - NUMCPUS   =        4
TA006 - IBUFSIZE  =       32
TA006 - MEMSIZE   =  8388608
TA006 - MEMSPEED  =       14
TA006 - CLOCKTIM  =     9500
TA006 - NUMCLSTR  =        5
TA006 - BANKBUSY  =        4
TA006 - EMA
TA006 - CIGS
TA006 - VPOP
TA006 - PC
TA006 - READVL
TA006 - NOVRECUR
TA006 - AVL
TA006 - HPM
TA006 - STATRG
TA006 - BDM
```

Example 2:

This use of TARGET changes the specifications for the clock period, number of clusters, availability of vector population count, and availability of additional vector logical functional unit. It also requests a report of the settings, as follows:

```
TARGET,CPU=*TARGET:CLOCKTIM=12500:NUMCLSTR=0:NOVPOP:NOAVL,^
       VERIFY=*TARGET.

TA005 - Primary machine type is:  CRAY-XMP
TA006 - BANKS     =       64
TA006 - NUMCPUS   =        4
TA006 - IBUFSIZE  =       32
```

```
TA006 - MEMSIZE   =  8388608
TA006 - MEMSPEED  =       14
TA006 - CLOCKTIM  =    12500
TA006 - NUMCLSTR  =        0
TA006 - BANKBUSY  =        4
TA006 - EMA
TA006 - CIGS
TA006 - NOVPOP
TA006 - PC
TA006 - READVL
TA006 - NOVRECUR
TA006 - NOAVL
TA006 - HPM
TA006 - STATRG
TA006 - BDM
```

DATASET DEFINITION AND CONTROL 8

The dataset control statements, ASSIGN, HOLD, NOHOLD, and RELEASE, let you define and manage datasets. ACCESS is not used for Integrated Support Processor (ISP) datasets. The ISP control statement gives your jobs access to an ISP, and the CONNECT control statement accesses a specific dataset. Refer to the SUPERLINK/ISP General Information Manual, CRI publication SI-0154, or the SUPERLINK/MVS Users Guide, CRI publication SI-0178, for details.

Control Statement	Function
ASSIGN	Defines characteristics for datasets. ASSIGN can also be used to create a mass storage dataset.
HOLD	Declares that dataset release occurs with implicit HOLD
NOHOLD	Rescinds the effect of the HOLD control statement
RELEASE	Relinquishes access by the job to the named dataset

8.1 ASSIGN - ASSIGN DATASET CHARACTERISTICS

The ASSIGN control statement assigns dataset characteristics for tape and mass storage and can create a mass storage dataset.† If an ASSIGN is used for dataset creation, it must appear before the first reference to the dataset; otherwise, the characteristics of the dataset are defined at the first reference to it. If an ASSIGN is used for a tape dataset, it must follow the tape ACCESS request (see section 9 for a description of ACCESS). ASSIGN is a system verb.

† ASSIGN does not create a dataset that the Fortran OPEN statement recognizes as existing. Refer to the Fortran (CFT) Reference Manual, CRI publication SR-0009, or the CFT77 Reference Manual, CRI publication SR-0018.

Format:

```
ASSIGN,DN=dn,S=size,SZ=size,NOF,BS=bsz,XSZ=xmx:xmn,DV=ldv,DT=dt,
  DF=df,RDM,U,MR,LM=lm,INC=nds,C,DC=dc,BFI=bfi,A=alias,FD=fd,CV=cv,
  CS=cs,F=f,RF=rf,RS=rs,MBS=mbs,DEF=dtl[:dt2:dt3],ST=st,SPD=spd.
```

DN=*dn* Local dataset name beginning with an alphabetic character or $, %, or @, and consisting of 1 to 7 alphanumeric characters. DN is a required parameter.

S=*size* Dataset size. Octal number of sectors (512-word blocks) to be reserved for the dataset. If the dataset size is not given, the space for the dataset is dynamically allocated as needed. The S and SZ options are mutually exclusive. Furthermore, S applies to mass storage datasets only, and is ignored when used for magnetic tape datasets.

SZ=*size* Dataset size. Decimal number of sectors (512-word blocks) to be reserved for the dataset. If the DV option specifies a generic resource or if *ldv* is a controlled device, SZ is the largest number of sectors associated with this dataset that can reside on the device. The mass storage space reservation occurs when the ASSIGN command is processed. If the SZ option is not specified, the space for the dataset is dynamically allocated as needed. The S and SZ options are mutually exclusive. SZ applies to mass storage datasets only and is ignored when used for magnetic tape datasets.

Although the SZ option specifies decimal sectors, disk space is allocated by COS in tracks that are larger than sectors. When an ASSIGN statement declares dataset size, COS rounds the sector count up to an integral multiple of track size and allocates that number of tracks. For example, when ASSIGN(...,SZ=1,...) is specified, COS allocates one track to the dataset, even though the request is for one sector. Track sizes for the various mass storage device types are as follows:

DD-19 disk drive	18 sectors
DD-29 disk drive	18 sectors
DD-39 disk drive	24 sectors
DD-49 disk drive	42 sectors
Extended Buffer Memory	18 sectors
SSD solid-state storage device	32 sectors

SZ=*size* (continued) When the disk device specified on the ASSIGN statement is a controlled device with a generic resource name, the total concurrent use of the device must be declared on the JOB statement as decimal sectors. If the space on the device is divided among several datasets with the SZ option on the ASSIGN statement, a rounding error may occur with each use of the SZ or S options. The result can be an unexpected GENERIC RESOURCE LIMIT EXCEEDED error or an unexpected device overflow. The SZ option can produce other results when it is used with the NOF parameter of ASSIGN. Those results are described under NOF in this section.

If both INC and SZ are specified, SZ is used initially and INC is used thereafter.

To divide space among several datasets on a generic resource such as Buffer Memory or SSD, sector counts should be specified as multiples of track size. Track size is device dependent.

NOF No overflow. When NOF is indicated, the dataset does not span any more than the device specified by the DV parameter. (If a device is not specified, the system selects one.) The SZ and NOF options on the ASSIGN statement produce the following:

SZ and NOF specified	Abort at MIN (Remaining Job Limit, SZ)
SZ specified without NOF	Overflow at MIN (Remaining Job Limit, SZ)
NOF specified without SZ	Abort at Remaining Job Limit
Neither SZ nor NOF specified	Overflow at Remaining Job Limit

BS=*bsz* Buffer size. *bsz* is an octal number that specifies the size of a dataset's circular I/O buffer in 512-word blocks. The default is the value defined by the installation parameter. The U and BS parameters are mutually exclusive.

XSZ=*xmx:xmn*

Transfer sizes. This parameter permits the circular buffer to be partitioned into specific zones, tailoring the I/O to a dataset and the program that uses the dataset.

xmx is the maximum transfer size in octal sectors to a device. If it is omitted, a system default is used; generally half the buffer size.

xmn is the minimum transfer size in octal sectors to a device. If it is omitted, a system default is used: generally one sector.

DV=*ldv* Logical device on which the dataset begins. If a logical device name is not given, one is chosen by the system. *ldv* can also be a generic resource name. Ask site operations for possible logical device names and generic resource names. When *ldv* is a generic resource or a controlled device, the number of sectors consumed by the dataset before overflow is counted against the resource allocation limit specified on the JOB control statement. The DV parameter applies to mass storage datasets only and is ignored when used for magnetic tape datasets.

DT=*dt* Device type. The allowable device types are CRT (interactive) and mass storage (MS). MS is the default. This parameter is ignored when used for magnetic tape datasets.

DF=*df* Dataset format. This parameter is used only on output and is valid only when DT=CRT. This parameter is ignored when used for magnetic tape datasets. The following two formats are supported:

- CB Character blocked; end-of-record (EOR) record control words are converted by the station to the format that the station supports. CB is the default.
- TR Transparent; EOR record control words are not converted. You must insert cursor controls.

RDM Random dataset. If the RDM parameter is present, the dataset is read and written randomly (that is, records may be read or written out of sequence). If the RDM parameter is not specified, only sequential or Fortran direct access I/O is allowed on the datasets. This parameter applies to mass storage datasets only and is invalid for magnetic tape datasets.

U Unblocked dataset structure. If the U parameter is present, the dataset is not in COS-defined blocked format. If the U parameter is absent, the dataset is a COS blocked dataset. (Refer to section 2 for information on unblocked dataset format.) This parameter is invalid for interchange format tape datasets. The U and BS parameters are mutually exclusive.

MR Memory-resident dataset. If this parameter is present, the system I/O routines write the buffers to mass storage only if they become full. If the MR parameter is absent, the dataset is not a memory-resident dataset. MR generates an

error if the U parameter is specified. This parameter applies to mass storage datasets only and is invalid for magnetic tape datasets.

LM=*lm* Maximum size limit for this dataset. *lm* specifies a number of 512-word blocks. The job step is aborted if this size is exceeded. The default and maximum dataset size limits are set by an installation parameter. The default is 100,000 sectors. This parameter applies to mass storage datasets only and is ignored for magnetic tape datasets.

INC=*nds* Number of sectors to allocate each time allocation occurs. The maximum value is 255 sectors. If both INC and SZ are specified, SZ is used initially and INC is used thereafter.

C Contiguous space allocation. Use C to allocate contiguous space requested by the SZ or INC parameter or the default size. If C is not specified, the system tries to find contiguous space on the selected device only. If C is specified, the system searches on every eligible device.

If contiguous space cannot be found when C has been specified, the return status CONTIGUOUS SPACE NOT AVAILABLE appears.

DC=*dc* Disposition code. Disposition of the dataset when it is released. This parameter applies to mass storage datasets only and is ignored for tape datasets. The default is SC (scratch).

dc is a 2-character alphabetic code describing the destination of the dataset as follows:

IN Input queue. The dataset is placed in the input queue of the destination station.

MT Magnetic tape. The dataset is written on magnetic tape at the mainframe of job origin.

PR Print dataset. The dataset is printed on the printer at the mainframe of job origin.

PT Plot dataset. The dataset is plotted on an available plotter at the mainframe of job origin.

PU Punch dataset. The dataset is punched on a card punch available at the mainframe of job origin.

SC Scratch dataset. The dataset is deleted.

ST Stage to mainframe. The dataset is made permanent at the mainframe of job origin.

BFI=*bfi* Blank field initiation. An octal representation of ASCII code that indicates the beginning of a sequence of blanks. BFI=OFF means that blank compression is inhibited. The default code is 33_8 (ASCII ESC code) but can be changed by an installation parameter. BFI is ignored for ISP datasets.

A=*alias* Alternate unit name. Unit names let you refer to a dataset by an alternate name in a program. Each unit name must be a valid COS dataset name.

If the unit name is to be used with Fortran unit numbers, *alias* has the form FT*xx*, where *xx* is the unit number specified. By default, this unit is formatted. If you wish to open it as unformatted, first close it and then reopen it to change the default. The unit number is an integer value in the range 0 through 102. Because unit numbers 100, 101, and 102 are reserved for system use, however, you may designate unit numbers 0 through 99.

Use of this parameter associates the designated unit with the dataset specified by the DN parameter. At job initiation, unit FT05 is associated with dataset $IN and unit FT06 is associated with dataset $OUT. Unit names should not be used as dataset names.

NOTE

If a dataset name is used in place of a unit name or vice versa, Fortran auxiliary statements (that is, OPEN, CLOSE, and INQUIRE) produce unpredictable results.

FD=*fd*† Foreign dataset translation identifier. *fd* is a 3-character code that indicates that foreign dataset translation is to be performed by the libraries on the dataset. This parameter is required for run-time translation. If FD is coded, RF must also be coded. Valid values for *fd* are:

CDC	CDC-compatible tape dataset
IBM	IBM-compatible tape dataset
VMS	VAX/VMS-compatible tape dataset

The default is no translation.

† See Foreign Dataset Conversion on CRAY X-MP and CRAY-1 Computer Systems, publication SN-0236, for more information.

CV=*cv*† Foreign dataset conversion mode. CV indicates if implicit data conversion is to be done by the run-time library. CV values are as follows:

ON Data conversion on. ON causes the library to convert the foreign internal representation to or from Cray internal representation according to the Fortran I/O list.

OFF Data conversion off. The data type is not considered when OFF is specified. Full Cray words are moved to or from the foreign dataset.

The default is OFF.

CS=*cs*† Foreign data character set. This parameter specifies the character set to represent the internal data on the foreign dataset. Run-time library routines convert character data from the *cs* character set to ASCII when implicit data conversion is turned on. The valid *cs* values are as follows:

AS ASCII; AS is the default for VAX/VMS tape file translation.

DC CDC display code; this option is illegal when IBM tape file translation is requested. DC is the default for CDC tape file translation.

EB EBCDIC; EB is the default for IBM tape file translation.

F=*f*† Tape format. *f* is a 1- or 2-character code which describes a CDC tape format type. It is required for CDC tape file translation; no default value is provided for F. Valid F values are as follows:

I Internal tape format
SI System or SCOPE internal tape format

RF=*rf*† Record format, or block and record type. When defined for IBM files, RF refers to record format. *rf* is a 1- to 3-character code that describes an IBM record format. Valid values for RF when defining IBM files are the following:

† See Foreign Dataset Conversion on CRAY X-MP and CRAY-1 Computer Systems, publication SN-0236, for more information.

RF=*rf* (continued)

F	Fixed-length records
FB	Fixed-length blocked records
U	Undefined-length records
V	Variable-length records
VB	Variable-length blocked records
VBS	Variable-length blocked spanned records

No default value is provided, but RF can be omitted when accessing an IBM standard-labeled tape file. In that case, the record format designated by the label is used. If NEW is specified, RF=U.

When defined for CDC tape files, RF refers to block and record type. In this case, *rf* is a 2-character code; the first character of the 2-character code describes the block type:

C	Character-count block type
I	Internal block type

The second character of the 2-character code describes the record type:

S	System-logical record type
W	Control-word record type
Z	Zero-byte record type

No default value is provided. RF is required for CDC tape file translation. The following *rf* values are supported for CDC tape files:

CS	Character-count block type, system-logical record type
CW	Character-count block type, control-word record type
CZ	Character-count block type, zero-byte record type
IW	Internal block type, control-word record type

When defined for VMS files, RF refers to record format. Here, *rf* is a 1- or 2-character code that describes a VMS record format. Values for *rf* are as follows:

F	Fixed-length records
UF	Unblocked fixed-length records
D	ANSI D variable-length records
V	Variable-length records
S	Variable-length segmented records
US	Unblocked variable-length segmented records

Certain formats are valid only on specific applications. See Foreign Dataset Conversion on CRAY X-MP and CRAY-1 Computer Systems, publication SN-0236, for details.

RS=*rs*† Tape dataset record size. *rs* is the length of the record, and its expression varies for IBM and CDC tape files.

When defined for IBM files, *rs* is the length of the record in 8-bit bytes. The default is set according to the requested record format. Table 8-1 shows the defaults for RS for IBM files. No default value is used, however, when accessing an IBM standard labeled tape file. Instead the record size designated by the label is used.

In addition, restrictions may be imposed on IBM files at ASSIGN processing time. Table 8-2 summarizes those restrictions.

When defined for CDC tape files, *rs* is the length of the record in 6-bit characters. *rs* refers to the maximum record length when W is specified as a value for RF. The default, RS=0, implies no maximum record length.

When Z is specified as a value for RF, *rs* becomes the CDC equivalent of the FL parameter: *rs* specifies the length to which zero-byte records are to be extended on input, and the length of a zero-byte record on output. This parameter is required for zero-byte record translation. No default value is provided for *rs* when Z is specified as an RF value.

For CDC system-logical records, *rs* is the maximum record length. The default, RS=0, implies no maximum record length.

For VAX/VMS tape files, *rs* is the length of the record in 8-bit bytes. For fixed-length (F-format) or unblocked fixed-length (UF-format) records, *rs* can be between 1 and 32767. There is no default.

For ANSI D variable-length (D-format) records, *rs* is the maximum record length in 8-bit bytes; *rs* can be between 1 and 9995. The default is a maximum record length of MBS-4 or 9995, whichever is smaller. For variable-length (V format) records, *rs* can be between 1 and 32767. *rs* may not exceed MBS for variable-length (V-format) records.

† See Foreign Dataset Conversion on CRAY-1 and CRAY X-MP Computer Systems, publication SN-0236, for more information.

Table 8-1. RS Defaults for IBM Tape Files

Record Format	Default
Undefined-length Fixed-length Fixed-length, blocked	RS=MBS
Variable-length Variable-length, blocked Variable-length, blocked, spanned	RS≤MBS-4

For variable-length segmented and unblocked variable-length segmented (S and US formats) records, *rs* is the maximum record length in 8-bit bytes. The value of *rs* is unrestricted.

Table 8-2. RS Restrictions for IBM Tape Files

Record Format	Restriction
Undefined-length Fixed-length	RS=MBS
Fixed-length, blocked	RS is multiple of MBS
Variable-length Variable-length, blocked	RS≤MBS-4
Variable-length, blocked, spanned	None

MBS=*mbs* Maximum tape block size. If you request foreign dataset translation by specifying FD (see the description of the FD parameter), values for *mbs* are different. *mbs* values are different for IBM, CDC, and VMS tape files.

MBS=*mbs* (continued) When defined for IBM files, *mbs* is the maximum block length in 8-bit bytes. The only *mbs* restriction for IBM files is that the value be less than or equal to 32760 bytes.

When defined for CDC tape files, *mbs* is the maximum block length in 6-bit characters. The default is 5120 characters. It is recommended that you not override this default value.

When defined for VMS files, *mbs* is the maximum block length in 8-bit bytes. The value must be no greater than 32767.

DEF=*dt1*[*:dt2:dt3*]

User-defined default space. The default space is allocated starting with the first device type specified. If that space is not available, the system tries the next device type. Up to three device types may be specified. The device types are as follows:

DD19	Disk drive
DD29	Disk drive
DD39	Disk drive
DD49	Disk drive
EBM	Extended Buffer Memory
SSD	Solid-state storage device
*	Any available device

If DEF is not specified, the device type defaults to *.

Example 1:

The system attempts to allocate space first on the SSD, next on EBM, and finally on all other default devices. If space is available on the SSD, overflow would be allocated on EBM and subsequent overflow would go to other default space.

```
ASSIGN,DN=A,DEF=SSD:EBM:*.
```

Example 2:

The system attempts to allocate space on the SSD. If space is not available on the SSD, the status NO MORE DISK SPACE AVAILABLE returns.

```
ASSIGN,DN=A,DEF=SSD.
```

ST=*st* User-specified storage for the dataset. The storage types are the following:

SCR	Scratch device
PERM	Permanent space device

The installation parameter I@STYPE defines the default.

Example:

The dataset named A is placed on a scratch device:

```
ASSIGN,DN=A,ST=SCR.
```

SPD=*spd* Sectors per device. *spd* is the number of sectors to allocate to a device before overflowing to a different device that is part of the user-defined default space. Simultaneous transfers can occur from different devices when the request spans more than one device (i.e., pseudo striping). *spd* ranges from a minimum of the number sectors allocated to one track on a device to a maximum of 2047.

If no *spd* is specified or if SPD=0, all data is transferred to the default device. If DV is specified, the SPD function will not occur until the specified DV overflows.

8.2 HOLD - HOLD GENERIC RESOURCE

The HOLD control statement declares that any dataset associated with the indicated generic resource will be released as if HOLD were specified on the RELEASE control statement. The HOLD parameter on the RELEASE control statement prevents the return of the resource allocation to the system pool. The HOLD control statement is useful when the dataset resides on a generic resource, and dataset assignment and release are controlled by applications over which you do not have direct control.

Format:

```
HOLD,GRN=grn.
```

GRN=*grn* Generic resource name

8.3 NOHOLD - RESCIND THE EFFECT OF HOLD

The NOHOLD control statement rescinds the effect of the HOLD control statement for the specified generic resource.

Format:

```
 ____________________
|                    |
| NOHOLD,GRN=grn.    |
|____________________|
```

GRN=*grn* Generic resource name

8.4 RELEASE - RELEASE DATASET

The RELEASE control statement relinquishes access to the named datasets for the job. If a dataset is not permanent and its disposition code is SC (scratch), the system releases the mass storage assigned to the dataset. If the dataset is to be staged, it enters the output queue for staging to the destination station. If the dataset is permanent, the system updates the allocation information in the system catalogs if the size of the dataset has changed since the last SAVE, ACCESS, or ADJUST request. Finally, if the disposition code is *not* scratch (whether or not the dataset is permanent), the system writes an end-of-data (EOD) record to the dataset if it is blocked sequential and the last operation on it was a write.

A dataset associated with a generic resource has a resource allocation as well as a physical allocation. The resource allocation for a tape dataset is one tape unit. The resource allocation for a disk dataset is the number of allocation units used by the dataset. Resources needed for a dataset are counted against the resource allocation limit specified on the JOB control statement during ACCESS (for tape) or ASSIGN (for disk). When a dataset is released, the physical allocation and the resource allocation are released to the system. When HOLD is specified on the RELEASE control statement, the physical allocation is released, but the resource allocation is retained for those datasets specified that are associated with a generic resource. HOLD is ignored for datasets not associated with a generic resource.

Format:

```
RELEASE,DN=dn1:dn2:...:dn8,HOLD.
```

DN=dn_i Name of dataset to be released. A maximum of eight datasets may be specified.

HOLD Hold generic resource. Do not return the resource allocation to the system pool.

8.5 INTEGRATED SUPPORT PROCESSOR (ISP) DATASETS

ISP datasets are controlled by two types of COS control statements:

Control Statement	Description
CONNECT	Provides access to a dataset in the MVS system by a COS job
ISP	Initiates communication with the ISP system on behalf of a COS job

Refer to the SUPERLINK/ISP General Information Manual, CRI publication SI-0154 or the SUPERLINK/MVS User Guide, CRI publication SI-0178, for a complete description of these control statements and their uses.

PERMANENT DATASET MANAGEMENT 9

The permanent dataset management control statements provide methods for creating, protecting, and accessing datasets assigned permanently to mass storage or magnetic tape. Such datasets cannot be destroyed by normal system activity or engineering maintenance.

Section 6 introduces permanent dataset management. This section describes the following permanent dataset management control statements:

Control Statement	Function
ACCESS	Makes an existing permanent dataset local to a job and is used to create a tape dataset
ADJUST	Records the change in any of the size or allocation information for a dataset that might have contracted or expanded
DELETE	Clears all or part of a dataset edition's entry in the system catalogs
MODIFY	Changes the characteristic information for an existing user permanent dataset
PERMIT	Explicitly grants or denies specified users or groups of users access to a permanent dataset
SAVE	Makes a local dataset permanent and defines its associated characteristics for the system

9.1 ACCESS - ACCESS PERMANENT DATASET

The ACCESS control statement makes an existing permanent dataset local to a job and can be used to create a tape dataset. Following the ACCESS statement, all references to the permanent dataset must be by the local dataset name specified by the DN parameter. ACCESS permission parameters ensure that the user is authorized to use the permanent dataset. The ACCESS control statement must precede the ASSIGN control statement or the request call for the dataset. All tape datasets, whether or not they are new, must be made local by an ACCESS control statement or a system request. ACCESS is a system verb.

More than one tape ACCESS control statement with the same dataset name, but a different permanent dataset name, will activate concatenation. Refer to the Concatenated Datasets subsection in section 2 for more information on concatenated datasets.

You do not have to access a permanent dataset entered in the System Directory (SDR). A basic set of datasets is entered into the SDR when the operating system is installed. These datasets include the loaders, Fortran compilers, the CAL assemblers, UPDATE, BUILD, and system utility programs such as copies and dumps (all utilities described in sections 6 through 15 are entered in the SDR). Other datasets can be entered into the SDR according to site requirements. A tape dataset cannot reside in the SDR.

The processing of the ACCESS system request ensures the following:

- The dataset already exists or, for new magnetic tape datasets, the dataset does not already exist.
- The requested permissions are allowed.
- The type of medium on which the dataset resides has been previously allocated by the job, provided the medium is a dedicated resource (such as magnetic tape).

If the Permanent Dataset Archiving feature is enabled, the following factors can cause a delay between the issue of the ACCESS request and its completion while the system recalls the dataset edition to on-line mass storage.

- The dataset edition being accessed has migrated off-line.
- The dataset edition being accessed has been retired off-line and the recall process initiated by a preceding RESTORE statement has not completed.

The Permanent Dataset Manager (PDM) issues a message to your job's logfile indicating the reason for the delay.

Format:

```
ACCESS,DN=dn,NA,ERR,MSG,IR,PDN=pdn,ID=uid,ED=ed,R=rd,W=wt,M=mn,UQ,
   OWN=ov,DT=dt,NEW,MOD,RING=IN/OUT,DEN=den,MF=fes,
   VOL=vol1:vol2:...voln,FSEC=fsec,LB=lb,DF=df,PROT,MBS=mbs,
   XDT=yyddd,RT=rt,FD=fd,CV=cv,CS=cs,F=f,RF=rf,RS=rs,FSEQ=fseq.
```

The following parameters can be used with mass storage datasets:

DN=*dn* Local dataset name. The name the job uses to refer to the dataset named in PDN while it remains local to the job. This parameter must be present and equated to a valid local dataset name not already in use.

NA No abort indicator. This parameter indicates that the job step is not to abort if an error results from the access attempt. If omitted, an error causes the job step to abort. NA is ignored if it is used for magnetic tape datasets.

ERR Error message. If this parameter is specified, error termination messages are suppressed.

MSG Termination message. If MSG is specified, normal termination messages are suppressed.

IR Immediate reply. An ACCESS request cannot always be honored immediately. When this is the case, the operating system automatically delays the request until it can be honored. If IR is specified and the ACCESS control statement cannot be honored immediately, the job will abort and the caller has to reissue the ACCESS request.

PDN=*pdn* Name or file identifier of the permanent dataset to access. For a mass storage dataset, the name can be 1 to 15 characters; for a magnetic tape dataset, 1 to 44 characters. For labeled tape datasets (AL and SL), the rightmost 17 characters of *pdn* are used to match the file identifier from the label group. With front-end servicing, the whole value is generally used as the identifier. If PDN is omitted, the DN value is used.

ID=*uid* Additional user identification; 1 to 8 alphanumeric characters. If *uid* was specified at SAVE time, the ID parameter must be specified on the ACCESS control statement. The default is no user ID. This parameter applies to mass storage datasets only; it is ignored for magnetic tape datasets.

ED=*ed* The edition number of the permanent dataset being accessed; a value from 1 through 4095 was assigned by the dataset creator. If the ED parameter is not specified, the default is the highest edition number known to the system (for this permanent dataset). This parameter applies to mass storage datasets only; it is ignored for magnetic tape datasets.

The following parameters identify the permissions for accessing mass storage permanent datasets.

R=*rd* Read control word as specified at SAVE time; 1 to 8 alphanumeric characters assigned by the dataset creator. The default is no read control word. To obtain read permission, this parameter must be specified on the ACCESS control statement if a read parameter was specified when the dataset was saved. This parameter applies to mass storage datasets only; it is ignored for magnetic tape datasets.

W=*wt* Write control word as specified at SAVE time. To obtain write permission, this parameter must be specified in conjunction with a UQ parameter on the ACCESS control statement if a W parameter was specified when the dataset was saved. Write permission is required for an ADJUST and applies to mass storage datasets only; it is ignored for magnetic tape datasets.

M=*mn* Maintenance control word as specified at SAVE time. This parameter is specified in conjunction with a UQ parameter on an ACCESS control statement if the dataset is to be subsequently deleted. That is, maintenance permission is required to delete a dataset. This parameter applies to mass storage datasets only; it is ignored when used for magnetic tape datasets.

UQ Unique access. This parameter indicates exclusive access to the dataset is desired. If the UQ parameter is specified and the appropriate write or maintenance control words are specified, write, maintenance, and/or read permission is granted. If UQ is not specified, multiple-user read access is granted by default (if at a minimum, the read control word is specified). UQ is required to delete a permanent dataset using the DELETE control statement. This parameter applies to mass storage datasets only; it is ignored for magnetic tape datasets.

Access to the requested dataset edition is delayed if either of the following conditions exist:

- You have requested unique access and another user already has access to the dataset edition.
- You have requested multiple-user read access and another user has unique access to the dataset edition.

When the condition blocking access is resolved, the delay state is cancelled. When multiple-user jobs or tasks are waiting for access to the same dataset edition, the delay

UQ (continued) state is cancelled for all the jobs or tasks at the same time. Thus, you cannot assume that the first of several jobs or tasks to be delayed for the same dataset edition will be the first to access it after a delay state is cancelled.

OWN=*ov* Ownership value. If the OWN parameter is specified and the user has been granted access by the owner, the dataset is made local to the job. OWN is ignored if *ov* matches the active ownership value of the job (users need not be permitted to their own datasets).

The following list describes the parameters available for accessing and/or defining magnetic tape datasets. The DN=*dn* parameter names the dataset.

DT=*dt* Tape dataset generic resource name. This parameter is required for tape datasets. Up to 16 generic resource names can be defined by the installation. Only one generic resource name is configured with the released system:

*TAPE Device capable of 1600 or 6250 b/i

The number of generic resources needed by the job is declared on the JOB control statement, and it is the resource allocation limit. (Refer to the JOB control statement description for details.) When a tape dataset is accessed, the number of tape drives associated with the dataset (usually one) is counted against the resource allocation.

NEW Creation disposition. Selection of this parameter indicates the dataset does not yet exist and is to be created by this job. NEW treats a tape as if it were blank and overwrites an existing tape label. If omitted, it is assumed the dataset already exists. NEW datasets must be written to before any read can occur. NEW and MOD are mutually exclusive. NEW automatically selects RING=IN if ring processing is in effect.

MOD Existing tape dataset modification identifier. This parameter lets you position single volume and multivolume datasets on tape. It specifies that data is to be added at the end of an existing dataset on either labeled or unlabeled tapes. Access requests using MOD for tape volume positioning are successful only if the end of a dataset is indicated by the EOF trailer label for a labeled tape volume, and by a tape mark for an unlabeled tape. MOD and NEW are mutually exclusive. MOD selects RING=IN if ring processing is in effect. MOD cannot be used with the transparent recording format.

RING=$^{IN}_{OUT}$ Tape write ring option. The choices are IN if the tape is to be written and OUT if the tape is only to be read. This parameter is in effect only if the installation parameter I@RNGABT is selected at your site.

DEN=*den* Density of the tape dataset. This parameter applies only to tape datasets; it is ignored when used for mass storage datasets. Density values are:

6250	Dataset density of 6250 b/i, default
1600	Dataset density of 1600 b/i

MF=*fes* Front-end servicing mainframe identifier. This parameter specifies an alternate front-end computer system to which servicing requests are directed. If MF is omitted, the front end from which the job originated is used. Front-end servicing is a mechanism whereby auxiliary servicing (such as updating front-end resident catalogs and tape management systems) of the dataset and/or tape volumes is performed.

The following parameters identify the magnetic tape dataset to be accessed. The PDN=*pdn* parameter names the dataset.

VOL=vol_i Volume identifier list. An optional list of 1- to 6-character volume identifiers (VIs) identify tape volumes where the dataset resides. The list contains up to 255 VIs. If the VI list is omitted for a new tape dataset, the tape volumes on which the dataset is written are selected by the system operator and the front-end servicing routine. This is called a nonspecific volume allocation. If the VI list is omitted for an old tape dataset, the volumes on which the dataset resides are determined by front-end servicing. If front-end servicing has no knowledge of the dataset or is inactive, the omission of the VI list results in a job step abort.

FSEC=*fsec* File section number or volume sequence number. This parameter describes on which volume, relative to the first physical volume of the dataset, to begin processing.

The volume sequence number for the first volume of the dataset is 1. If *fsec* is omitted, a value of 1 is assumed. This parameter has a direct relationship to the VIs specified in the VOL parameter. The volume sequence number corresponds to the first VI identified in the VOL parameter. For example, to access a tape dataset starting with the eighth section, specify FSEC=8 on the ACCESS call.

If both the MOD and FSEC=*fsec* are coded, the FSEC parameter is not used for validating the header label.

FSEC=*fsec* (continued) Instead, it represents the position of the volume serial number in the volume list where MOD processing begins.

For example, the following statement causes processing to start with tape T2.

```
ACCESS,...MOD,VOL=T1:T2:T3,FSEC=2,...
```

LB=*lb* Tape dataset label type that indicates the tape format. If this parameter is omitted, label type NL is assumed. Label types are as follows:

AL	ANSI standard labeled tapes
BP	Bypass label processing†
FAL	Field format with ANSI standard labels
FNL	Field format with no labels
FSL	Field format with IBM standard labels
NL	Unlabeled tapes (default)
SL	IBM standard labeled tapes

Field format tape datasets treat embedded EOFs or tapemarks as data. Tapemarks that are not followed by a label are returned in the data as EOF control words. On output, EOF control words that are not followed by an EOD control word are converted to physical tapemarks.

The following parameters identify the characteristics of a magnetic tape dataset.

DF=*df* Recording format. Identifies the format in which the tape dataset is to be read or written or both. Values for this parameter are the following:

IC	Interchange format
TR	Transparent format (invalid for field format tape datasets)

If DF is omitted, the format is transparent. Refer to section 2 for a description of the formats and the associated properties.

PROT†† Front-end protect indicator. Indicates to the front-end computer system performing the service functions that the tape dataset or its volumes or both are to be protected. PROT is recognized for new tape datasets only. If PROT is omitted, the dataset and its volumes are not protected.

---

† User privilege is required if the system security option (I@SLVL) is in warning mode or full mode. Refer to the CRI site operations staff to acquire this privilege.

†† Station-dependent parameter

MBS=*mbs* Maximum tape block size. If foreign dataset translation is requested by specifying FD, values for *mbs* are different. Refer to the description of the FD parameter. *mbs* values are different for IBM, CDC, and VMS tape files.

When defined for IBM files, *mbs* is the maximum block length in 8-bit bytes. The only *mbs* restriction for IBM tape files is that the value be less than or equal to 32760 bytes.

When defined for CDC tape files, *mbs* is the maximum block length in 6-bit characters. The default is D'5120 characters. It is recommended that you not override this default value.

When defined for VMS files, *mbs* is the maximum block length in 8-bit bytes. The value must be no greater than 32767.

If MBS is omitted and the dataset is new, a default size that has been determined by the site is used. The limiting value of the parameter is also left to site definition. If omitted for an existing labeled tape dataset (AL or SL), the maximum block size is set to the value from the label group. Exceeding this size when writing results in a job abort condition of WRITE FORMAT ERROR. When reading a tape block that is larger than the specified value, a job abort condition of LARGE BLOCK ENCOUNTERED is produced. MBS is rounded up to the next multiple of 4096 bytes for transparent format tape datasets.

XDT=*yyddd* Expiration date. Indicates the date this tape dataset is considered dormant and may be overwritten. *yy* specifies the year and is a number from 0 through 99. *ddd* specifies the day in the year, 001 through 366. If omitted and the dataset is going to be written, the current date is used. This parameter is also used as a means of communicating with a servicing front-end computer system. The XDT and RT parameters are mutually exclusive.

RT=*rt* Retention period. User-defined value from 1 through 4095 specifying the number of days a permanent dataset should be retained by the system. The RT parameter is similar to the XDT parameter but lets you specify relative expiration date. If RT is omitted, the default value is 0. The RT and XDT parameters are mutually exclusive.

The following tape dataset parameters specify that record and data format conversion are to be performed at run time.

FD=*fd*† Foreign dataset translation identifier. *fd* is a 3-character code that indicates foreign dataset translation should be performed on the dataset. This parameter is required for run-time translation. Valid values for *fd* are the following:

CDC	CDC-compatible tape dataset
IBM	IBM-compatible tape dataset
VMS	VAX/VMS-compatible tape dataset

The default is no translation.

CV=*cv*† Foreign dataset conversion mode. CV indicates whether or not implicit data conversion should be done by the run-time library (RTL). CV values are the following:

ON Data conversion turned on. ON causes the library to convert the foreign internal representation to or from Cray internal representation, according to the I/O list.

OFF Data conversion turned off means the data type is not considered. Full Cray words are moved to or from the foreign dataset.

The default is OFF.

CS=*cs*† Foreign data character set specifies the character set to represent the internal data on the foreign dataset. RTL routines convert character data from the *cs* character set to ASCII when implicit data conversion is turned on. The valid *cs* values are the following:

AS ASCII. AS is the default for VAX/VMS tape file translation.

DC CDC display code. DC is the default for CDC tape file translation. This option is illegal when IBM tape file translation is requested.

EB EBCDIC. EB is the default for IBM tape file translation.

F=*f*† Tape format. *f* is a 1- or 2-character code that describes a CDC tape format type. It is required for CDC tape file translation. No default value is provided for F. Valid F values are the following:

I	Internal tape format
SI	System or SCOPE internal tape format

† See Foreign Dataset Conversion on CRAY X-MP and CRAY-1 Computer Systems, publication SN-0236 for more information.

RF=*rf*† Record format, or block and record type. When defined for IBM files, RF refers to record format. *rf* is a 1- to 3-character code that describes an IBM record format. Valid values for RF when defining IBM files are the following:

F	Fixed-length records
FB	Fixed-length blocked records
U	Undefined-length records
V	Variable-length records
VB	Variable-length, blocked records
VBS	Variable-length, blocked, spanned records

No default value is provided, but RF can be omitted when accessing an IBM standard-labeled tape file. In that case, the record format designated by the label is used. If NEW is specified, RF=U.

When defined for CDC tape files, RF refers to block and record type and is a 2-character code. The first character of the 2-character code describes the block type:

C	Character-count block type
I	Internal block type

The second character of the 2-character code describes the record type:

S	System-logical record type
W	Control-word record type
Z	Zero-byte record type

No default value is provided. RF is required for CDC tape file translation. The following *rf* values are supported for CDC tape files:

CS	Character-count block type, system-logical record type
CW	Character-count block type, control-word record type
CZ	Character-count block type, zero-byte record type
IW	Internal block type, control-word record type

When defined for VMS files, RF refers to record format. Here *rf* is a 1- or 2-character code that describes a VMS format. Values for *rf* are as follows:

F	Fixed-length records
UF	Unblocked fixed-length records
D	ANSI D variable-length records
V	Variable-length records
S	Variable-length segmented records
US	Unblocked variable-length segmented records

† See Foreign Dataset Conversion on CRAY X-MP and CRAY-1 Computer Systems, publication SN-0236 for more information.

RS=*rs*† Tape dataset record size. *rs* is the decimal length of the record, and its expression varies for IBM and CDC tape files.

When defined for IBM files, *rs* is the decimal length of the record in 8-bit bytes. The default is set according to the requested record format. No default value is used, however, when accessing an IBM standard labeled tape file. Instead, the record size designated by the label is used. Table 9-1 shows the defaults for which RS is set for IBM files.

Table 9-1. RS Defaults for IBM Tape Files

Record Format	Default
Undefined-length Fixed-length Fixed-length, blocked	RS=MBS
Variable-length Variable-length, blocked Variable-length, blocked, spanned	RS=MBS-4

In addition, restrictions may be imposed on IBM tape files at ACCESS processing time. Table 9-2 summarizes those restrictions. Nonetheless, restrictions are not enforced if the file accessed is an IBM standard labeled tape file, and if neither RS nor MBS is specified.

† See Foreign Dataset Conversion on CRAY X-MP and CRAY 1 Computer Systems, publication SN-0236 for more information.

Table 9-2. RS Restrictions for IBM Tape Files

Record Format	Restriction
Undefined-length Fixed-length	RS=MBS
Fixed-length, blocked	MBS is multiple of RS
Variable-length Variable-length, blocked	RS≤MBS-4
Variable-length, blocked, spanned	None

RS=*rs* (continued)

For CDC tape files, *rs* is the decimal length of the record in 6-bit characters. *rs* refers to the maximum record length when W is specified as a value for RF. The default, RS=0, implies there is no maximum record length.

When Z is specified as a value for RF, *rs* becomes the equivalent of the CDC FL parameter: *rs* specifies the length to which zero-byte records are to be extended with blank characters on input and the length of a zero-byte record on output. This parameter is required for zero-byte record translation. No default value is provided for *rs* when Z is specified as an RF value.

For CDC system-logical records, *rs* is the maximum record length. The default, RS=0, implies that there is no maximum record length.

For VAX/VMS files, *rs* is the length of the record in 8-bit bytes. For fixed-length (F-format) or unblocked fixed-length (UF-format) records, *rs* can be between 1 and 32767. There is no default.

For ANSI D variable-length (D format) records, *rs* is the maximum record length in 8-bit bytes. *rs* can be between 1 and 9995. The default, RS=0, implies a maximum record

RS=*rs* (continued) length of MBS-4 or 9995, whichever is smaller. For variable-length (V format) records, *rs* can be between 1 and 32767; *rs* may not exceed MBS.

For variable-length segmented and unblocked variable-length segmented (S and US formats) records, *rs* is the maximum record length in 8-bit bytes. The value of *rs* is unrestricted. The default, RS=0, implies no maximum record size.

FSEQ=*fseq* File sequence number. This is a 1- to 4-digit number that describes the relative position of the dataset on the tape volume. The default is 1.

9.2 ADJUST - ADJUST PERMANENT DATASET

The ADJUST control statement redefines the size of a mass storage permanent dataset by modifying the information in the Dataset Catalog (DSC) to reflect changes in the dataset size and disk allocation. When a permanent dataset is overwritten, and the dataset size changes, issuing an ADJUST statement informs the system of the dataset's new size. An ADJUST of a permanent dataset can be issued if the dataset has been previously accessed within the job with write permission. ADJUST is a system verb.

Under the appropriate conditions, ADJUST forces any unwritten data to mass storage to ensure that all of the dataset is made permanent. Because this situation occurs when the dataset has recently been written to but not yet closed, ADJUST attempts to close the dataset. CLOSE disposes of current positioning information for that dataset. Therefore, subsequent operations on that dataset must reopen it and begin at the beginning-of-data (BOD). The specific conditions that the dataset must meet are described under the ADJUST macro (refer to the Macros and Opdefs Reference Manual, CRI publication SR-0012).

The ADJUST statement is ignored when used with magnetic tape datasets.

If a dataset's size is reduced sufficiently to require fewer disk allocation units, the unused disk space returns to COS. The size of a disk allocation unit is dependent on the device type.

Format:

```
ADJUST,DN=dn,NA,ERR,MSG.
```

DN=*dn* Local dataset name of a permanent dataset that has been accessed with write permission. This dataset can be closed before the ADJUST statement is processed.

NA No abort. If this parameter is omitted, an error causes the job step to abort.

ERR Error message. If this parameter is specified, error termination messages are suppressed.

MSG Termination message. Normal termination messages are suppressed when MSG is specified.

9.3 DELETE - DELETE PERMANENT DATASET

The DELETE control statement clears all or part of a dataset edition's entry in the system catalogs: the Master Catalog Dataset (MCD), the Dataset Catalog (DSC), and the Backup Catalog (BCD). DELETE's effect depends both on the residence of the dataset and on the parameters specified. The control statement has two formats: one for local datasets and another for nonlocal datasets.

9.3.1 LOCAL DATASET FORMAT

The local dataset format of the DELETE control statement requires that the dataset be accessed as a local dataset with both unique access (the UQ parameter on the ACCESS control statement) and maintenance permission.

For a mass storage resident dataset, the action of DELETE depends on the PARTIAL parameter. If the parameter is specified, the entries in the system catalogs for the dataset are retained, but the allocation information is erased; the dataset itself remains accessible to the job as an empty permanent dataset. If the PARTIAL parameter is omitted, the entries in the system catalogs for the dataset are erased, and the dataset remains accessible to the job as a temporary dataset.

For a magnetic tape resident dataset, DELETE causes COS to send a request to the front-end computer to remove the dataset's definition from its catalogs.

Format:

```
DELETE,DN=dn,NA,ERR,MSG,PARTIAL.
```

DN=*dn* Local dataset name of a permanent dataset accessed with maintenance permission and unique access. This is a required parameter.

NA No abort. If this parameter is omitted, a fatal error causes the job step to abort.

ERR Error message. If this parameter is specified, error termination messages are suppressed.

MSG Termination message. If MSG is specified, normal termination messages are suppressed.

PARTIAL Partial delete. Presence of this parameter causes COS to delete only the mass storage resident data. The DSC entry and the dataset's attributes information are retained. PARTIAL can be specified only for a mass storage dataset; it is ignored for tapes.

9.3.2 NONLOCAL DATASET FORMAT

The nonlocal dataset format of the DELETE control statement is used to permit the deletion of permanent datasets without accessing them in advance. It can be used only for mass storage resident datasets. If you get an error message, it could mean that the system does not have the Master Catalog option enabled. In this case, use the local dataset format of the DELETE control statement.

This form of DELETE erases all record of the specified datasets from the system catalogs; there is no PARTIAL parameter. Deletion from the system catalogs is immediate if the dataset or datasets are not currently accessed. If the datasets are currently accessed, the Permanent Dataset Manager (PDM) processes the request for deletion when the last accessor releases the dataset. In either case, there is no delay to the job issuing the DELETE.

The arguments for PDN, ID, and OWN can use the notations * to indicate any one character and - to indicate an arbitrary string of characters.

Format:

```
DELETE,PDN=pdn,ERR,MSG,ID=id,OWN=owner,ED=ed,M=m.
```

PDN=*pdn* Permanent dataset name; required parameter.

ERR Error message. If this parameter is specified, error termination messages are suppressed.

MSG Termination message. Normal termination messages are suppressed if MSG is specified.

ID=*id* Permanent dataset ID; optional. Omission implies a null ID.

OWN=*owner*
Owner of the permanent dataset. The default is the job owner. If the requester is not the dataset owner, the requester must have maintenance permission.

ED=*ed* Edition number of the dataset. Options for *ed* are as follows:

Specification	Meaning
Unsigned integer (*ed*) Example: ED=2	The specific edition of the dataset
Negative integer (-*ed*) Example: ED=-2	All but the *ed* highest editions
Positive integer (+*ed*) Example: ED=+2	The *ed* highest editions
ED=ALL	All editions of the dataset

The default is the highest edition.

M=*mn* Maintenance control word. Must be specified if the dataset has a maintenance control word.

9.4 MODIFY - MODIFY PERMANENT DATASET

The MODIFY control statement changes permanent dataset information established by the SAVE function or a previously executed MODIFY function. A permanent dataset must be accessed with unique access (UQ) and all permissions before MODIFY can be issued. MODIFY is a system verb.

Once a permanent dataset exists, the read, write, and maintenance control words, public access mode, and access tracking apply to subsequent editions of that permanent dataset.

Parameters are in keyword form; the only required parameter is DN. If any combination of PDN, ID, and ED (including omission of one or more of them) is specified, and the resulting PDN/ID/ED combination already exists, the MODIFY aborts, and no changes are made.

MODIFY applies to mass storage datasets only; it is ignored for tape datasets.

Format:

```
MODIFY,DN=dn,PDN=pdn,ID=uid,ED=ed,RT=rt,R=rd,W=wt,M=mn,NA,ERR,
   MSG,EXO=ON/OFF,PAM=mode,TA=opt,TEXT=text,NOTES=notes,
   RESIDE=ONLINE/OFFLINE,BACKUP=YES/NO .
```

DN=*dn* Local dataset name of a permanent dataset that has been accessed with all permissions; DN is a required parameter.

PDN=*pdn* New permanent dataset name to be applied to the existing dataset. If this parameter is omitted, the existing permanent dataset name is retained.

ID=*uid* New user identification to be applied to the existing permanent dataset; 1 to 8 alphanumeric characters. If this parameter is omitted, the existing user ID is retained. If this parameter is present without a value, user identification is cleared.

ED=*ed* New edition number to be applied to the existing permanent dataset. If this parameter is omitted, the existing edition number is retained.

RT=*rt* New retention period to be applied to the existing permanent dataset. If this parameter is omitted, the current retention period is retained. If this parameter is present without a value, the retention period is set to the installation-defined value.

R=*rd* New read permission control word to be applied to the existing permanent dataset. If this parameter is omitted, the existing read permission is retained. If R is present without a value, the read permission control word is cleared.

W=*wt* New write permission control word to be applied to the existing permanent dataset. If this parameter is omitted, the existing write permission is retained. If W is present without a value, the write permission control word is cleared.

M=*mn* New maintenance permission control word to be applied to the existing permanent dataset. If this parameter is omitted, the existing maintenance permission is retained. If M is present without a value, the maintenance permission control word is cleared.

NA No abort. If this parameter is omitted, an error causes the job to abort.

ERR Error message. If this parameter is specified, error termination messages are suppressed.

MSG Termination message. Normal termination messages are suppressed when MSG is specified.

EXO=ON / OFF Execute-only dataset. This parameter sets or clears the execute-only status of a dataset. EXO only (EXO=ON) causes the dataset to be modified to execute-only. EXO=OFF causes the dataset to be modified to a nonexecute-only dataset. If this parameter is omitted, the execute-only status of a dataset is unchanged.

PAM=*mode* Public access mode. The following options are allowed:

Option	Mode
E	Execute only
M	Maintenance only
N	No public access allowed
R	Read only
W	Write only

Each site controls the default PAM value. Combinations of R, W, and M permissions are allowed; for example, PAM=R:W gives both read and write permissions. PAM=E has the same effect as the EXO or EXO=ON parameter and nullifies any other permissions specified.

TA=*opt* Track accesses. *opt* can be either YES or NO and indicates whether the owner requires that public accesses to the dataset be tracked. Refer to section 6 for a description of public access and access tracking. The default TA value is NO.

TEXT=*text*
Text to be passed to a front-end computer system requesting transfer of the dataset. Specify a maximum of 240 characters. This text information is considered an attribute of the dataset and is retained along with any other attributes. Refer to section 6 for an explanation of all permanent dataset attributes.

To clear the text, specify TEXT without a value.

NOTES=*notes*

Notes to be associated with the dataset. Specify a maximum of 480 characters. There is no other restriction on the contents of *notes*. A caret symbol in *notes* signifies end-of-line and causes AUDIT to advance to a new line when listing the *notes*. The caret symbol is included in the 480-character maximum limit. *notes* is a permanent dataset attribute. Refer to section 6 for an explanation of all permanent dataset attributes.

To clear the notes, specify NOTES without a value.

RESIDE=ONLINE
OFFLINE

The preferred residency of a dataset. ONLINE specifies the dataset should remain on-line. This option requires the SCRESON privilege.

OFFLINE specifies the dataset should receive priority when datasets are selected for migration. The speed with which the dataset migrates depends on factors such as how often the site runs space management. This option does not require a privilege.

To clear the preferred-residency setting, specify RESIDE without a value. This causes the dataset edition to become a candidate for space management based on site-defined criteria.

BACKUP†=YES
NO

Dataset backup. YES specifies the dataset should be backed up after it is created and whenever it is modified. NO specifies that the dataset should not be backed up under any circumstance. A dataset with no backup may be subject to rules defined by the site, especially regarding retention time. The default is YES.

† Deferred implementation

9.5 PERMIT - EXPLICITLY CONTROL ACCESS TO DATASET

The PERMIT control statement explicitly designates who can access a particular permanent dataset. PERMIT applies to all editions of the permanent dataset. The dataset does not need to be local for PERMIT to be executed. PERMIT applies to user permanent mass storage datasets only. Access permission given with a PERMIT control statement takes precedence over the PAM parameter described under SAVE and MODIFY. PERMIT is a system verb.

Format:

```
PERMIT,PDN=pdn,ID=uid,AM=m,RP,USER=ov,ADN=adn,NA,ERR,MSG.
```

PDN=*pdn* Name of an existing user permanent dataset; 1 to 15 characters. PDN is a required parameter.

ID=*uid* Additional user identification; 1 to 8 alphanumeric characters. If ID was specified on the SAVE request, the ID parameter must be specified on the PERMIT control statement. The default is no user ID.

AM=*m* Access mode permitted for alternate user. The options are as follows:

Option	Mode
E	Execute only
M	Maintenance only
N	No public access allowed
R	Read only
W	Write only

Each site controls the default AM value. Combinations of R, W, and M permissions are allowed; for example, AM=R:W gives both read and write permissions. AM=E gives the permitted user execute-only access to the dataset, effectively nullifying any other permissions specified.

RP Remove permit parameter. Removes the permit associated with the specified ownership value.

USER=*ov* User ownership value associated with the user whose access permissions are being specified.

ADN=*adn* Local dataset name of the attributes dataset from which the permit list is copied. The permits are created for the dataset specified by PDN, overwriting existing permits.

NA No abort. If this parameter is omitted, an error causes the job step to abort.

ERR Error message. If this parameter is specified, error termination messages are suppressed.

MSG Termination message. Normal termination messages are suppressed when MSG is specified.

9.6 SAVE - SAVE PERMANENT DATASET

The SAVE control statement makes a local dataset permanent and defines its associated characteristics for the system. For mass storage datasets, saving involves making entries in the system catalogs, which uniquely identify the dataset. For magnetic tape datasets, saving involves front-end servicing on the defined front-end computer system.

Under the appropriate conditions, SAVE forces any unwritten data (left in the output buffer) to be written, ensuring that all the data is made permanent. Because this situation occurs when the dataset has been recently written but not yet rewound or closed, SAVE attempts to close the dataset. CLOSE disposes of current positioning information for that dataset. Therefore, subsequent operations on that dataset must reopen it and begin at the beginning of the dataset (BOD). The specific conditions that the dataset must meet are described under the SAVE macro (refer to the Macros and Opdefs Reference Manual, CRI publication SR-0012). A permanent dataset is uniquely identified by permanent dataset name (PDN), additional user identification (ID), edition number (ED), and ownership value. SAVE is a system verb.

NOTE

Because COS does not identify unblocked and random datasets, these datasets must be assigned as unblocked or random (use the ASSIGN control statement) after they have been accessed.

SAVE creates an initial edition or an additional edition of a permanent dataset.

Format:

```
SAVE,DN=dn,PDN=pdn,ID=uid,ED=ed,RT=rt,R=rd,W=wt,M=mn,UQ,NA,ERR,
        ON
  MSG,EXO=OFF,PAM=mode,ADN=adn(m),TA=opt,TEXT=text,NOTES=notes,
       ONLINE          YES
  RESIDE=OFFLINE,BACKUP=NO .
```

DN=*dn* Local dataset name. The name the job uses to refer to the dataset while it remains local to the job. This dataset can be closed before the dataset is made permanent. This is a required parameter.

PDN=*pdn* Permanent dataset name. The default value is *dn*. The name can be 1 to 15 alphanumeric characters.

ID=*uid* Additional user identification. *uid* can be 1 to 8 alphanumeric characters assigned by the dataset creator. The default is no user ID.

ED=*ed* Edition number. A value from 1 through 4095 assigned by the dataset creator. The default value is:

- 1, if a permanent dataset with the same PDN and ID does not exist
- The current highest edition number plus one, if a permanent dataset with the same PDN and ID does exist

RT=*rt* Retention period. User-defined value from 1 through 4095 specifying the number of days a permanent dataset is to be retained by the system. The default value is an installation-defined parameter.

R=*rd* Read control word; 1 to 8 alphanumeric characters assigned by the dataset creator. The read control word of the highest-numbered existing edition of a permanent dataset applies to all subsequent editions of that dataset. The default is no read control word.

W=*wt* Write control word; 1 to 8 alphanumeric characters assigned by the dataset creator. The write control word of the highest-numbered existing edition of a permanent dataset applies to all subsequent editions of that dataset. To obtain write permission, you must also have unique access (UQ) to that dataset. The default is no write control word.

M=*mn* Maintenance control word; 1 to 8 alphanumeric characters. The maintenance control word must be specified if a subsequent edition of the same permanent dataset is saved. The default is no maintenance control word.

UQ Unique access. If the UQ parameter is specified, only this job can access the permanent dataset at the completion of the SAVE function. Otherwise, multiple-user read access to the permanent dataset is granted.

NA No abort. If this parameter is omitted, an error causes the job to abort.

ERR Error message. If this parameter is specified, error termination messages are suppressed.

MSG Termination message. If MSG is specified, normal termination messages are suppressed.

EXO=$^{ON}_{OFF}$ Execute-only dataset. This parameter sets or clears the execute-only status of the dataset. EXO only or EXO=ON causes the dataset to be saved as execute-only. EXO=OFF or omission of this parameter causes the dataset to be saved as nonexecute-only dataset. When EXO=ON has been specified, it overrides permitted and public access modes.

PAM=*mode* Public access mode. The following options are allowed:

Option	Mode
E	Execute only
M	Maintenance only
N	No public access allowed
R	Read only
W	Write only

Your site controls the default PAM value.

Combinations of R, W, and M permissions are allowed; for example, PAM=R:W gives both read and write permissions. PAM=E has the same effect as the EXO or EXO=ON parameter and nullifies any other permissions specified.

If the dataset is to be used for a segmented load with SEGLDR, use PAM=R (rather than PAM=E) to enable SEGLDR to read the dataset.

ADN=*adn*(*m*)
Name of the attributes dataset from which attributes, indicated by the modifier *m*, are selected. If no modifiers are present, all attributes are selected. Attribute parameters such as NOTES=, TEXT=, PAM=, R=, and so

ADN=*adn(m)*
(continued) on, take precedence over the modifiers. *adn* must be the local dataset name of a permanent dataset. The modifiers must be enclosed with parentheses and separated by colons. The following modifiers are supported:

Modifier	Selection from Attributes Dataset
ALL	All attributes
CW	Control words
NOTES	Notes attribute
PAM	Public access mode attribute
PERMITS	Permit list
TEXT	Text attribute
TRACK	Public access tracking attribute

TA=*opt* Track accesses. *opt* can be either YES or NO and indicates whether the owner requires that public accesses to the dataset be tracked. Refer to section 6 for a description of public access and access tracking. The default TA value is NO.

TEXT=*text* Text to be passed to a front-end computer system requesting transfer of the dataset. A maximum of 240 characters can be specified. This text information is considered an attribute of the dataset and is retained along with any other attributes. Refer to section 6 for an explanation of all permanent dataset attributes.

NOTES=*notes*
Notes to be associated with the dataset. A maximum of 480 characters can be specified. There is no restriction on the content of *notes*. A caret symbol in *notes* signifies end-of-line and causes AUDIT to advance to a new line when listing the *notes*. The caret symbol is included in the 480 character maximum limit. *notes* is a permanent dataset attribute. Refer to section 6 for an explanation of all permanent dataset attributes.

RESIDE=ONLINE
OFFLINE
The preferred residency of a dataset. ONLINE specifies the dataset should remain on-line. This option requires the SCRESON privilege.

OFFLINE specifies the dataset should receive priority when datasets are selected for migration. The speed with which the dataset migrates depends on factors such as how often the site runs space management. This option does not require a privilege.

If RESIDE is not specified, the dataset's selection for migration is based on site-defined criteria established for space management.

BACKUP†=YES / NO

Dataset backup. YES specifies the dataset should be backed up after it is created and whenever it is modified. NO specifies the dataset should not be backed up under any circumstance. The default is YES.

9.7 EXAMPLES OF PERMANENT DATASET CONTROL STATEMENTS

To clarify the permanent dataset management control statements, some examples follow:

Example 1:

A user identified as USERXYZ creates a permanent dataset that no other user can access. All subsequent editions of this dataset share this attribute.

```
SAVE,DN=ABC,PDN=EXAMPLE1,ED=1,PAM=N,TA=NO.
```

Example 2:

A user identified as USERXYZ creates a permanent dataset that can be accessed by all other users in read mode.

```
SAVE,DN=XYZ,PDN=EXAMPLE2,ED=1,PAM=R,TA=NO.
```

Example 3:

An alternate user is accessing the permanent dataset created in example 2.

```
ACCESS,DN=LOCAL,PDN=EXAMPLE2,ED=1,OWN=USERXYZ.
```

The system does not track the alternate user access because the dataset was created with TA=NO.

† Deferred implementation

Example 4:

Allow another user (known in this example as USER1) to access the permanent dataset created in example 1 in read and execute mode only.

```
PERMIT,PDN=EXAMPLE1,USER=USER1,AM=R:E.
```

Example 5:

Enable public access tracking for the permanent dataset created in example 2.

```
ACCESS,DN=LOCAL,PDN=EXAMPLE2,ED=1,UQ.
MODIFY,DN=LOCAL,TA=YES.
```

Example 6:

Permit write mode access for PDN=EXAMPLE2 to users known as USER2 and USER3.

```
PERMIT,PDN=EXAMPLE2,USER=USER2,AM=W.
PERMIT,PDN=EXAMPLE2,USER=USER3,AM=W.
```

Example 7:

Change the permission granted to USER1 in example 4 to AM=W.

```
PERMIT,PDN=EXAMPLE1,USER=USER1,AM=W.
```

Example 8:

Remove the access permission granted to USER1 in example 7.

```
PERMIT,PDN=EXAMPLE1,USER=USER1,RP.
```

Example 9:

User USERXYZ acquires a dataset, then permits another user to use it and subsequently partially deletes the dataset to retain just the PERMITs and TEXT information. Section 10 discusses the ACQUIRE control statement.

```
ACQUIRE,DN=EX9,TEXT='..........',UQ.
PERMIT ,PDN=EX9,USER=SOMEONE,AM=R.
DELETE ,DN=EX9,PARTIAL.
```

Example 10:

User USERXYZ creates a permits template.

```
SAVE,DN=EX10,PDN=PERMS,^
  NOTES='PERMITS TEMPLATE FOR AERO USERS.  '^
  'THESE PERMITS SHOULD BE REMOVED AFTER OCT 31, 1983.',UQ.
PERMIT,PDN=PERMS,USER=USERA,AM=E.
PERMIT,PDN=PERMS,USER=USERB,AM=R.
PERMIT,PDN=PERMS,USER=USERC,AM=W.
DELETE,DN=EX10,PARTIAL.
```

Example 11:

User SOMEONE acquires the dataset that was partially deleted in example 9. Section 10 discusses the ACQUIRE control statement.

```
ACQUIRE,DN=LOCAL,PDN=EX9,OWN=USERXYZ.
```

The TEXT need not be specified and after the dataset has been acquired from the front-end computer system, it is made permanent and belongs to user USERXYZ.

DATASET STAGING CONTROL 10

Staging is the process of transferring COS datasets (jobs and data) from front-end computer systems to Cray mass storage or vice versa. Dataset staging control is introduced in section 6.

Three control statements support staging datasets between Cray mass storage and a front-end system: ACQUIRE, DISPOSE, and FETCH. Another control statement, SUBMIT, directs datasets to the COS input queue.

Control Statement	Function
ACQUIRE	Makes a front-end resident dataset permanent and accessible to the job making the request
DISPOSE	Directs a dataset to the COS output queue for staging to a specified front-end computer system
FETCH	Makes a dataset that resides on a front-end computer system local to the COS job
SUBMIT	Directs a dataset to the COS input queue

10.1 ACQUIRE - ACQUIRE PERMANENT DATASET

The ACQUIRE control statement converts a front-end resident dataset into a permanent dataset so that it is accessible to the job making the request. ACQUIRE is a system verb.

When an ACQUIRE control statement is issued, COS determines if the requested dataset is resident on the front end or permanently resident on Cray mass storage by checking the system catalogs for a dataset with matching PDN, ID, ED, and ownership value fields.

If COS determines that the requested dataset is already permanently resident on Cray mass storage, dataset access is granted to the job making the request if the user has the appropriate access permissions.

If the requested dataset is not a COS mass storage permanent dataset, the request for the dataset is sent to the front-end system.

The front-end system stages the dataset to Cray mass storage if the front end grants the user access. Such access is determined by the front-end operating system and may be dependent on the contents of the TEXT information from a FETCH or ACQUIRE control statement, or of a SAVE or MODIFY control statement preceding a partial DELETE. COS then makes the dataset permanent on Cray mass storage and grants dataset access to the job making the request. Until the dataset is made permanent, processing of the job making the request is delayed.

Format:

```
ACQUIRE,DN=dn,PDN=pdn,AC=ac,ID=uid,ED=ed,RT=rt,R=rd,W=wt,M=mn,UQ,
  TEXT=text,MF=mf,TID=tid,DF=df,OWN=ov,PAM=mode,ADN=adn(m),
                                  ONLINE          YES
  TA=opt,NOTES=notes,ERR,MSG,RESIDE=OFFLINE,BACKUP=NO .
```

DN=*dn* Local dataset name; begins with A-Z, $, @, or %, followed by 1 to 6 alphanumeric characters. The name the job will use to refer to the dataset while it remains local to the job. DN is a required parameter.

PDN=*pdn* Name of the COS permanent dataset to be accessed or staged from a front-end system, saved, and accessed. The permanent dataset name is passed to the front-end system; it is the name saved by the system if the dataset is staged. *pdn* is 1 to 15 alphanumeric characters assigned by the dataset creator. The default for *pdn* is *dn*.

AC=*ac* Acquisition code. The source from which the dataset is to be acquired. If the AC parameter is omitted, the default is ST.

ac is a 2-character alphanumeric code describing the source of the dataset as follows:

- IN Input (job) dataset. Use the SUBMIT control statement to run the job.
- IT Intertask communication
- MT Magnetic tape at the front end designated by the MF parameter
- ST Staged dataset from the front end designated by the MF parameter

NOTE

The dataset acquisitions previously noted are by convention only. Actual dataset acquisition is determined by the front end.

ID=*uid* Additional user identification, 1 to 8 alphanumeric characters assigned by the dataset creator. The default is no user ID.

ED=*ed* Edition number. A value from 1 to 4095 assigned by the dataset creator. The default value is one of the following:

- 1, if a permanent dataset with the same PDN and ID does not currently exist
- The current highest edition number of that dataset if the permanent dataset with the specified PDN and ID does exist

RT=*rt* Retention period. User-defined value from 1 through 4095 specifying the number of days a permanent dataset is to be retained by the system. The default value is an site-defined parameter.

R=*rd* Read control word. 1 to 8 alphanumeric characters assigned by the dataset creator. The default is no read control word.

W=*wt* Write control word. 1 to 8 alphanumeric characters assigned by the dataset creator. The default is no write control word.

M=*mn* Maintenance control word. 1 to 8 alphanumeric characters assigned by the dataset creator. The control word must be specified if a subsequent edition of the permanent dataset is saved and the previous editions have an associated maintenance control word.

UQ Unique access. If the UQ parameter is specified, the job is granted unique access to the permanent dataset; otherwise, multiple-user read access to the permanent dataset is granted. If no staging is performed because the dataset already exists, write, maintenance, and/or read permission can be granted if the appropriate read, write, and/or maintenance control words are specified.

TEXT=*text* Text to be passed to a front-end computer system requesting transfer of the dataset. A maximum of 240 characters can be specified. This text information is considered an attribute of the dataset and is retained along with any other attributes. See section 6 for an explanation of all permanent dataset attributes.

MF=*mf* Identifier for the front-end computer. Two alphanumeric characters. The default is the front end on which the job originated.

TID=*tid* Terminal identifier. 1 to 8 alphanumeric characters identifying the destination terminal. The default terminal is the terminal where the job originated.

DF=*df* Dataset format. This parameter defines whether a dataset is to be presented to the Cray computer system (see the FETCH control statement) in COS blocked format and whether the front-end system is to perform character conversion. The default is CB.

df is a 2-character alphanumeric code defined for use on the front-end system. CRI suggests support of the following codes:

BB Binary blocked. The front-end system blocks the dataset before staging but does not do character conversion.

BD Binary deblocked. The front-end system does not perform character conversion. For ACQUIRE, BD is the same as TR.

CB Character blocked. The front-end system blocks the dataset before staging and performs character conversion to ASCII, if necessary.

CD Character deblocked. The front-end system performs character conversion to ASCII, if necessary.

TR Transparent. No blocking/deblocking or character conversion is performed.

OWN=*ov* Ownership value. If the OWN parameter is specified and the user has been granted access by the owner, the dataset is made local to the job. OWN is ignored if *ov* matches the active ownership value of the job (users need not be permitted to their own datasets).

PAM=*mode* Public access mode. The following options are allowed:

Option	Mode
E	Execute only
M	Maintenance only
N	No public access allowed
R	Read only
W	Write only

Combinations of R, W, and M permissions are allowed; for example, PAM=R:W gives both read and write permissions. Note that PAM=E has the same effect as the EXO or EXO=ON parameter and nullifies any other permissions specified. Each installation controls the default PAM value.

ADN=*adn*(*m*)

Name of attributes dataset from which attributes, indicated by the modifiers *m*, are selected. If no modifiers are present, then all attributes are selected. Attribute parameters such as NOTES=, TEXT= and PAM=, and R= take precedence over the modifiers. *adn* must be the local dataset name of an accessed permanent dataset. The modifiers must be enclosed with parentheses and separated by colons. The following modifiers are supported:

Modifier	Selection from Attributes Dataset
ALL	All attributes
CW	Control words
NOTES	Notes attribute
PAM	Public access mode attribute
PERMITS	Permit list
TEXT	Text attribute
TRACK	Public access tracking attribute

TA=*opt* Track accesses. *opt* can be either YES or NO and indicates whether the owner requires that public accesses to the dataset be tracked. See section 6 for a description of public access and access tracking. The default TA value is NO.

NOTES=*notes*

Notes to be associated with the dataset. A maximum of 480 characters can be specified. There is no other restriction on the content of *notes*. A caret symbol in *notes* signifies end-of-line and causes AUDIT to advance to a new line when listing the *notes*. The caret symbol is included in the 480 character maximum limit. *notes* is a permanent dataset attribute. Refer to section 6 for an explanation of all permanent dataset attributes.

ERR Error message. If this parameter is specified, error termination messages are suppressed.

MSG Termination message. Normal termination messages are suppressed when MSG is specified.

RESIDE=ONLINE / OFFLINE

The preferred residency of a dataset. ONLINE specifies the dataset should remain on-line. This option requires the SCRESON privilege.

OFFLINE specifies the dataset should receive priority when datasets are selected for migration. The speed with which the dataset migrates depends on factors such as how often the site runs space management. This option does not require a privilege.

If RESIDE is not specified, the dataset's selection for migration is based on site-defined criteria established for space management.

BACKUP†=YES / NO

Dataset backup. YES specifies the dataset should be backed up after it is created and whenever it is modified. NO specifies the dataset should not be backed up under any circumstance. A dataset with no backup may be subject to rules defined by the site, especially regarding retention time. The default is YES.

10.2 DISPOSE - DISPOSE DATASET

The DISPOSE control statement directs a dataset to the COS output queue for staging to a specified front-end computer system. You can also use DISPOSE to alter the effects of a previous DISPOSE,DEFER of the same dataset.

Defining the DISPOSE characteristics can be done before the actual staging by using the DEFER parameter. The DEFER parameter saves all selected dispose parameters for use when the dataset is released, which is when the actual staging is initiated. DISPOSE is a system verb.

† Deferred implementation

Format:

```
DISPOSE,DN=dn,SDN=sdn,DC=dc,DF=df,MF=mf,SF=sf,ID=uid,TID=tid,
   ED=ed,RT=rt,R=rd,W=wt,M=mn,TEXT=text,WAIT,NOWAIT,DEFER,NRLS.
```

DN=*dn* Local dataset name. Name by which the dataset is known to the user job. DN is a required parameter.

SDN=*sdn* Staged dataset name. 1- to 15-character name by which the dataset is to be known at the destination front end. The default for *sdn* is *dn*.

DC=*dc* Disposition code. Disposition to be made of the dataset. If the DC parameter is omitted, the default is PR (print).

dc is a 2-character alphanumeric code describing the destination of the dataset as follows:

- IN Input (job) dataset. Dataset is queued as a job on the mainframe specified with the MF parameter.
- IT Intertask communication
- MT Write dataset on magnetic tape at the front end designated by the MF parameter.
- PR Print dataset. Dataset is printed on a printer available at the front end designated by the MF parameter.
- PT Plot dataset. Dataset is plotted on any available plotter at the front end designated by the MF parameter.
- PU Punch dataset. Dataset is punched on any card punch available at the front end designated by the MF parameter.
- SC Scratch dataset. Dataset is released, unless another DISPOSE request is still pending on the dataset. This parameter has the same effect as RELEASE,DN=*dn*.
- ST Stage to front end. Dataset is made permanent at the front end designated by the MF parameter.
- VC Station-specific code. Refer to station documentation for more information.

NOTE

The dataset dispositions previously noted are by convention only. With the exception of SC, actual dataset disposition is determined by the destination front end.

DF=*df* Dataset format. This parameter defines whether a dataset is sent from the Cray computer system in COS-blocked format and whether the front-end system is to perform character conversion. The default is CB (character blocked).

For example, a user wishes to save a dataset on magnetic tape in blocked binary as it appears on COS mass storage. In this case, BB is specified. A user who wants a dataset printed can specify CB if the front-end computer handles deblocking.

df is a 2-character alphanumeric code defined for use on the front-end system. CRI suggests support of the following codes listed below. Other codes can be added by the local site. Undefined pairs of characters can be passed but are treated as transparent mode by COS.

BB Binary blocked. The front-end system does not perform character conversion. The Cray mainframe does not perform deblocking before staging. The front-end system is expected to perform deblocking.

BD Binary deblocked. The front-end system does not perform character conversion. For DISPOSE, BD is the same as TR.

CB Character blocked. No deblocking is performed at the Cray mainframe before staging. The front-end system performs deblocking and character conversion from 8-bit ASCII, if necessary.

CD Character deblocked. The front-end system performs character conversion from 8-bit ASCII, if necessary.

TR Transparent. No blocking, deblocking, or character conversion is performed.

MF=*mf* Front-end computer identifier; 2 alphanumeric characters. Identifies the front end to which the dataset is to be staged. If omitted, the front end where the issuing job originated is used. If MF is given a value of the ID of the Cray mainframe on which the job is running and DC=IN, an error message is issued and the job step is aborted (see the SUBMIT control statement in subsection 10.4).

SF=*sf* Special form information to be passed to the front-end system. 1 to 8 alphanumeric characters. SF is defined by the needs of the front-end system.

ID=*uid* Additional user identification. 1 to 8 alphanumeric characters assigned by the dataset creator. The default is no user ID.

TID=*tid* Terminal identifier. 1 to 8 alphanumeric characters identifying the destination terminal. The default terminal is the terminal where the job originated, where applicable.

ED=*ed* Edition number, meaningful only if DC=ST. A user-defined value from 1 through 4095. The default value depends on the destination front end.

RT=*rt* Retention period, meaningful only if DC=ST. A user-defined value from 1 through 4095 specifying the number of days a dataset is to be retained by the destination front end. The default value depends on the destination front end.

R=*rd* Read control word, meaningful only if DC=ST. 1 to 8 alphanumeric characters. The default is no read control word.

W=*wt* Write control word, meaningful only if DC=ST. 1 to 8 alphanumeric characters. The default is no write control word.

M=*mn* Maintenance control word, meaningful only if DC=ST. 1 to 8 alphanumeric characters. The default is no maintenance control word.

TEXT=*text* Text to be passed to the front-end system requesting transfer of a dataset. The format for TEXT is defined by the front-end system for managing its own datasets or files. Typically, *text* is in the form of one or more control statements for the front-end system; these statements must contain their own terminator for the front end. *text* cannot exceed 240 characters.

NOTE

text specified on the DISPOSE control statement is not the same as the permanent dataset *text* attribute. Any *text* existing as a permanent dataset attribute is ignored by DISPOSE (refer to section 6 for more information).

WAIT Job wait. When this parameter is specified, the job does not resume processing until the disposed dataset has been staged to the front-end system. If the front-end system cancels the transfer, the waiting job is aborted and job step abort processing occurs as described in section 3. If WAIT is not specified, processing can resume immediately upon issue of the DISPOSE, depending upon an installation option. The WAIT parameter is useful in detecting unsuccessful transfers.

NOWAIT When this parameter is specified, the job does not wait until the dataset has been staged to the front-end system but resumes processing immediately. If the front-end system cancels the transfer, no special action is taken; that is, the job is not aborted. If neither WAIT or NOWAIT are specified, processing can resume immediately upon issue of the DISPOSE, depending upon an installation option.

DEFER When this parameter is specified, the disposition occurs when the dataset is released either by a RELEASE request or job termination. The disposition characteristics are saved and used when the dataset is released.

NRLS No release. When this parameter is specified, the dataset remains local to the job after the DISPOSE request has been processed. When NRLS is specified, the dataset cannot be written to until the transfer to the specified front end is completed. Therefore, it is advisable to use WAIT with NRLS.

10.3 FETCH - FETCH LOCAL DATASET

The FETCH control statement makes a dataset that resides on a front-end computer system local to the COS job. The dataset is transferred from the front-end computer system if the front-end system grants access to the

dataset. The dataset is not made permanent on the Cray computer system. The originating job is delayed until the dataset arrives on Cray mass storage.

Format:

```
FETCH,DN=dn,SDN=sdn,AC=ac,TEXT=text,MF=mf,TID=tid,
 DF=df,SF=sf.
```

DN=*dn* Local dataset name. The name the job will use to refer to the dataset while it remains local to the job; 1 to 7 alphanumeric characters, the first of which is A through Z, $, @, or %. DN is a required parameter.

SDN=*sdn* Staged dataset name. Name by which the dataset is known on the front end; 1 to 15 alphanumeric characters. The default for *sdn* is *dn*.

AC=*ac* Acquisition code. The source from which the dataset is to be acquired. If the AC parameter is omitted, the default is ST (staged dataset).

ac is a 2-character alphanumeric code describing the source of the dataset as follows:

- IN Input (job) dataset. Use the SUBMIT control statement to run the job.
- IT Intertask communication
- MT Magnetic tape at the front end designated by the MF parameter
- ST Staged dataset from the front end designated by the MF parameter

NOTE

The dataset acquisitions previously noted are by convention only. Actual dataset acquisition is determined by the front end.

TEXT=*text* Text to be passed to the front-end system requesting transfer of a dataset. The format for TEXT is defined by the front-end system for managing its own datasets or files. Typically, *text* is in the form of one or more control statements for the front-end system; these statements must contain their own terminator for the front end. *text* cannot exceed 240 characters.

MF=*mf* Mainframe computer identifier. 2 alphanumeric characters. The default is the front end of job origin.

TID=*tid* Terminal identifier. 1 to 8 characters identifying the destination terminal. The default is the terminal where the job originated.

DF=*df* Dataset format. This parameter defines whether a dataset is sent from the Cray computer system (see the FETCH control statement) in COS blocked format and whether the front-end system is to perform character conversion. The default is CB (character blocked).

For example, a user who wishes to save a dataset on magnetic tape in blocked binary as it appears on COS mass storage can specify BB. A user who wants a dataset printed can specify CB if the front-end computer handles deblocking.

Other codes can be added by the local site. Undefined pairs of characters can be passed but are treated as transparent mode by COS.

df is a 2-character alphanumeric code defined for use on the station. CRI suggests support of the following codes:

BB Binary blocked. The front-end system blocks the dataset before staging but does not do character conversion.

BD Binary deblocked. The front-end system does not perform character conversion. For FETCH, BD is the same as TR.

CB Character blocked. The front-end system blocks the dataset before staging and performs character conversion to 8-bit ASCII, if necessary.

CD Character deblocked. The front-end system performs character conversion to 8-bit ASCII, if necessary.

TR Transparent. No blocking, deblocking or character conversion is performed.

DF=df (continued) Other codes can be added by the local site. Undefined Pairs of characters can be passed but are treated as transparent mode by COS.

SF=*sf* Special form information to be passed to the front-end system. 1 to 8 alphanumeric characters. SF is defined by the needs of the front-end system.

10.4 SUBMIT - SUBMIT JOB DATASET

The SUBMIT control statement is used by one job to direct another dataset (which must have the structure of a job dataset as defined in section 3) to the COS input queue. The job that is submitted executes independently of the submitting job. SUBMIT is a system verb.

Format:

```
SUBMIT,DN=dn,SID=sf,DID=df,TID=tid,DEFER,NRLS.
```

DN=*dn* Local dataset name. Must be a valid local dataset name. DN is a required parameter.

SID=*sf* Default source identifier; 2 alphanumeric characters. If an MF parameter is not specified in an ACQUIRE or FETCH control statement within the submitted job, the SID parameter defines the default front-end system for the dataset to be acquired. If the MF and SID parameters are omitted, the default source identifier of the submitting job is used.

DID=*df* Default destination identifier; 2 alphanumeric characters. If an MF parameter is not specified in a DISPOSE control statement within the submitted job, the DID parameter defines the default destination front-end system for the dataset to be disposed. If the MF and DID parameters are omitted, the default destination identifier of the submitting job is used.

TID=*tid* Default terminal identifier; 1 to 8 alphanumeric characters that define the default terminal ID for the submitted job. If TID is omitted, the terminal ID of the submitting job is used.

DEFER Deferred submit. This parameter causes the SUBMIT characteristics to be defined, with a release of the dataset actually initiating the submit of the dataset. If DEFER is omitted, the SUBMIT occurs immediately.

NRLS No release. This parameter indicates if the dataset is to remain local to the job after SUBMIT has been processed. If NRLS is omitted, the dataset is released after the SUBMIT. If NRLS is selected, the dataset remains local to the job after the SUBMIT and is available for reading only.

PERMANENT DATASET UTILITIES 11

The following utility routines support permanent datasets:

Utility	Function
AUDIT	Produces a report containing status information for each permanent dataset. AUDIT does not include input or output datasets.
PDSDUMP	Dumps all specified permanent datasets to a user-specified dataset. Input and output datasets can be included in the dump.
PDSLOAD	Loads permanent datasets that have been dumped by PDSDUMP and updates or regenerates the Dataset Catalog (DSC). Input and output datasets are also loaded through PDSLOAD.
RESTORE	Recalls retired or migrated datasets to on-line disk
RETIRE	Declares a dataset retired

All of the permanent dataset utilities permit a shorthand notation for the arguments to the PDN (or PDS), ID, US, and OWN parameters. Using this notation, a dash represents any number of characters or no characters and an asterisk represents any one character.

Examples:

Notation	Description
PDN=ABC-	Lists all permanent dataset names beginning with ABC
PDN=A***	Lists all 4-character permanent dataset names beginning with A
PDN=-A*-	Lists all permanent dataset names containing the letter A followed by one or more other characters
PDN=-	Lists all permanent dataset names
PDN=***-	Lists all permanent dataset names having three or more characters

When permanent dataset privacy is enabled, callers of these utilities are limited to actions on their own datasets unless the CW parameter is present on the control statement. The OWN and NOWN parameters cannot be specified unless CW is also specified. When privacy is enabled, the US value from the JOB or ACCOUNT control statement is an implied dataset selection criterion, unless the CW parameter is present. When privacy is not enabled, the US value from the JOB or ACCOUNT control statement is not used as a selection criterion. CW must be specified if US or OWN is specified on the permanent dataset utility control statement.

11.1 AUDIT - AUDIT PERMANENT DATASETS

The AUDIT utility reports the status of all the permanent datasets known to the system. AUDIT does not include input and output datasets.

If more than one parameter is selected, only those datasets that meet all criteria are listed.

AUDIT can supply the following information on the output listing:

- Permanent dataset name
- Dataset identifier
- Edition number
- User identifications
- Dataset size in words
- Retention time
- Number of accesses
- Public access mode
- Total block count
- Track access flag setting
- Creation date/time
- Last dump date/time
- Last access date/time
- Last modification date/time
- Device name
- *note* information
- *text* information
- Permitted users
- Access counts by user
- Number of datasets selected
- Current residency
- Preferred residency

Format:

```
AUDIT,L=ldn,B=bdn,PDN=pdn,ID=uid,US=usn,ACN=acn,DV=dvn,SZ=dsz,
  ACC=opt:opt,X=mm/dd/yy:'hh:mm:ss',TCR=mm/dd/yy:'hh:mm:ss',
  TLA=mm/dd/yy:'hh:mm:ss',TLM=mm/dd/yy:'hh:mm:ss',CW=cw,
  OWN=ov,LO=opt:...opt,BO=opt:...opt.
```

L=*ldn* Lists dataset name; default is $OUT.

B=*bdn* Name of dataset to receive the binary output. If B is specified alone, the dataset is $BINAUD. If the B parameter is omitted, no binary output is written.

PDN=*pdn* Name of permanent dataset or datasets to be listed

ID=*uid* Lists all permanent datasets with the specified additional user identification. The default is to list all IDs. If ID is present without an equated value, datasets having a null ID are selected.

US=*usn* Lists all permanent datasets with the specified user number. The default is to list all user numbers.

ACN=*acn* Lists all permanent datasets with the specified account number. The default is to list datasets without respect to account number.

DV=*dvn* Lists all permanent datasets on the specified logical device. The default is to list permanent datasets on all devices.

SZ=*dsz* Lists all permanent datasets greater than or equal to the specified size. Size is specified in words. The default is to list all sizes.

ACC=*opt:opt*
Access option parameters. The options are as follows:

AM Lists only those datasets belonging to OWN that have an explicit permit for the job's ownership value

PAM Lists only those datasets belonging to OWN that have any form of public access (R:W:M:E)

If the OWN parameter is omitted, all datasets are searched for the permit or public access. If the CW parameter is specified, the AM includes any permit for any owner value. If the OWN parameter is specified and the CW and ACC parameters are omitted, AUDIT assumes the ACC=AM:PAM parameter on the control statement.

X=*mm/dd/yy*:'*hh:mm:ss*'
Lists all permanent datasets expired as of the specified *mm/dd/yy*:'*hh:mm:ss*'. *mm/dd/yy* can be specified alone. The default expiration date and time are "now" if only X is specified.

TCR=*mm/dd/yy*:'*hh:mm:ss*'
Lists all permanent datasets that have been created since the specified *mm/dd/yy*:'*hh:mm:ss*'. The keyword cannot be specified alone; however, TCR=*mm/dd/yy* is sufficient.

TLA=*mm/dd/yy*:'*hh:mm:ss*'
Lists all permanent datasets that have not been accessed since the specified *mm/dd/yy*:'*hh:mm:ss*'. The keyword cannot be specified alone; however, TLA=*mm/dd/yy* is sufficient.

TLM=*mm/dd/yy*:'*hh:mm:ss*'
Lists all permanent datasets that have been modified since the specified *mm/dd/yy*:'*hh:mm:ss*'. The keyword cannot be specified alone; however, TLM=*mm/dd/yy* is sufficient.

CW=*cw* Site-defined control word regulating the use of AUDIT. If the CW parameter is omitted, only the datasets belonging to the job owner can be listed. If the CW parameter is present and the correct control word is used, any dataset can be listed. If an invalid control word is given, the job step is aborted. When the CW and ACC parameters are omitted, but the OWN parameter is specified, AUDIT assumes the ACC=AM:PAM parameter on the control statement.

OWN=*ov* Lists all permanent datasets with the specified ownership value. If OWN is not specified, the job's ownership value is used.

Output formatting parameters are the following:

LO=*opt*:...*opt*
Listing option selection. S is the default for interactive jobs; L, for batch. The S option cannot be mixed with any others.

The following options can be specified alone or in combination separated by colons:

LO=*opt*:...*opt*
(continued) A Access tracking. Includes accessing owner name, access count, time of last access, and time of first access.

B Backup. Reports the tape volume names on which the current back-up copy resides, the number of space management deletions and reloads, and the status of internal flags indicating whether the dataset is a candidate for backup or recall. Also specify the CW parameter if this option is used.

L Long list. Consists of PDN, ID, ED, size in words, retention time, access count, track access flag, public access mode (PAM), creation, last access, last modification, last dump time, device name, preferred residency (PR), and current residency (CR). L is used for batch jobs when LO is not specified. It lists information for on-line or migrated datasets only.

N Notes list. Displays the dataset catalog *notes* field.

P Permit list. Includes permitted owner name, access mode, access count, time of last access, and time of permit creation.

R Retired datasets listing. Consists of the same categories of information as LO=L but for retired datasets only.

S Short list. Includes PDN, ID, and ED listed two per line. This is used for interactive jobs when LO is not specified.

T Text list. Displays the dataset catalog *text* field.

X Extended long list. Includes everything in the long list (L) plus an indication of the dataset's allocated (ALLOC) size (shown immediately below the dataset's size (SZ)). The extended long list also includes a line immediately below the dataset size summary that gives the number of blocks and words allocated.

BO=*opt*:...*opt*
Binary audit options. These options specify what additional information, if any, is to be added to the standard binary audit file. They are ignored without comment unless a binary audit is requested by the B parameter. If more than one option is desired, separate them with colons. The options are as follows:

BO=*opt*:...*opt*

(continued) A Access tracking. Generates one record for each accessing user for each selected dataset.

B Backup. Reports the tape volume name(s) on which the current back-up copy resides, the number of space management deletions and reloads, and the status of internal flags indicating whether the dataset is a candidate for backup or recall. The CW parameter must be specified if this option is used.

N Notes. Generates one record for each selected dataset that has *notes*.

P Permits. Generates one permit record for each permitted user for each selected dataset.

R Retired datasets listing. Consists of PDN, ID, ED, size in words, retention time, access count, track access flag, PAM, creation, last access, last modification, last dump time, device name, PR, and CR.

T Text. Generates one record for each selected dataset that has *text*.

X Adds a field to the regular binary audit record indicating the allocated word size of the dataset. This is the same value as the ALLOC field on the LO=X output.

Figures 11-1 through 11-6 show some of the LO options as they appear when the listing is directed to a mass storage dataset. Interactive reports omit the page header line. Systems in which the Permanent Dataset Privacy feature is not enabled suppress the owner line unless OWN is used as a control statement parameter.

```
AUDIT          COS 1.16          03/23/87  14:30:09          PAGE      1

 OWN = TNG

PDN              ID          ED        PDN              ID          ED

"DIANE"          U1520        1        "GOTCHA"         U1520        7
DATA             U1520        7        DATA             U1520        8
DATA1234         U1520        2        DI               U1520        1
ENG.SCORES       U1520        1        LETEM            U1520        1
NDAT             U1520        8        NEWDATA          U1520       10
NEWLIB           U1520        1        NLIB             U1520        1
OBJECT           U1520        6        OOPS.34          U1520        2
SDCVALUES_V1     U1520        1        TEST             U1520        1
VOTE             U1520        1        WHATISIT         U1520        1

      18 DATASETS,       34 BLOCKS,              13217 WORDS
       4 DATASETS,        6 BLOCKS,               3072 WORDS ARE ONLINE
      14 DATASETS,       28 BLOCKS,              10145 WORDS ARE OFFLINE

                                                                    1540
```

Figure 11-1. AUDIT, LO=S Listing

```
     PERMITTED USERS FOR PDN =  DI                ID = U1520     ED =     1

     USER             AM     ACC     LAST ACCESS          CREATED

     TNG1520          RWM      0                     03/18/87 15:19:08
     TNG12            RWM      0                     03/18/87 15:24:58

NO REQUESTED INFO FOUND FOR PDN = ENG.SCORES       ID = U1520     ED =     1

     PERMITTED USERS FOR PDN =  LETEM             ID = U1520     ED =     1

     USER             AM     ACC     LAST ACCESS          CREATED

     U1520            RM       0                     11/04/86 11:32:31
     TNGOO            N        0                     03/23/87 12:29:53
     TNG99            E        0                     11/04/86 11:32:33
     RJJ              RWM      0                     03/23/87 11:35:51

                                                                    1541
```

Figure 11-2. AUDIT, LO=P Listing

```
PDN                  ID          ED          LAST      LAST      LAST      DEVICE
        SZ     RT     ACC TA PAM   CREATED ACCESSED MODIFIED  DUMPED    CR    PR
-------------------------------------------------------------------------------

ENG.SCORES       U1520          1  11/04/86 01/23/87          01/24/87
       527     45       3 N  N     14:03:12 11:55:22          02:36:45  MIG   NO

LETEM            U1520          1  10/30/86 03/23/87          03/14/87  39-1-36A
         0     45      34 N  M     14:20:27 13:36:28          02:14:22   ON   NO

      PERMITTED USERS:

      USER              AM      ACC      LAST ACCESS          CREATED

      U1520             RM        0                      11/04/86 11:32:31
      TNG00             N         0                      03/23/87 12:29:53
      TNG99             E         0                      11/04/86 11:32:33
      RJJ               RWM       0                      03/23/87 11:35:51

   NOTES:

   These are permits to beused in exercise 4                          1542
```

Figure 11-3. AUDIT, LO=L:P:N Listing

```
        566     45       6 N  N     16:59:04 10:37:20          02:35:17  MIG   NO

NEWDATA          U1520         10  01/06/87 03/18/87 01/06/87 02/21/87
       532     45      23 N  N     16:40:12 16:19:43 16:40:15 01:45:21  MIG   NO

NEWLIB           U1520          1  01/08/87 01/23/87          01/24/87
      1024     45       4 N  N     11:04:09 11:26:51          02:36:43  MIG   NO

NLIB             U1520          1  11/06/86 03/19/87          03/21/87  49-1-31A
       512     45      20 N  N     17:13:09 16:51:19          01:39:56   ON   NO

OBJECT           U1520          6  03/12/87 03/19/87          03/21/87
       633     45       4 N  N     15:29:51 13:12:49          01:38:06  MIG   NO

OOPS.34          U1520          2  01/06/87 03/23/87 01/19/87 01/24/87  39-1-36A
       512     45       9 N  N     16:04:34 12:47:46 15:27:43 02:36:19   ON   NO

SDCVALUES V1     U1520          1  06/18/86 01/23/87 01/23/87 01/24/87

                                                                    1543
```

Figure 11-4. AUDIT, LO=L Listing

```
NO REQUESTED INFO FOUND FOR PDN = DI                ID = U1520      ED =     1

NO REQUESTED INFO FOUND FOR PDN = ENG.SCORES        ID = U1520      ED =     1

  NOTES FOR PDN = LETEM              ID = U1520      ED =     1
  These are permits to beused in exercise 4

                                                                          1544
```

Figure 11-5. AUDIT, LO=N Listing

```
      1024     45       4 N  N     11:04:09 11:26:51            02:36:43  MIG    NO

NEWS.PL4          U1520          2  11/05/86 01/19/87 01/19/87
      1024     45      10 N  N     09:44:33 15:19:22 15:21:19              RET    NO

NLIB              U1520          1  11/06/86 03/19/87            03/21/87  49-1-31A
       512     45      20 N  N     17:13:09 16:51:19            01:39:56   ON    NO

OBJECT            U1520          6  03/12/87 03/19/87            03/21/87
       633     45       4 N  N     15:29:51 13:12:49            01:38:06  MIG    NO

OOPS.34           U1520          2  01/06/87 03/23/87 01/19/87 01/24/87  39-1-36A
       512     45       9 N  N     16:04:34 12:47:46 15:27:43 02:36:19   ON    NO

POPEYE            U1520          1  10/30/86 03/18/87            01/24/87
       633     45       9 N  N     13:50:11 09:38:57            02:33:54  RET    NO

                                                                          1815
```

Figure 11-6. AUDIT, LO=L:R Listing

11.2 PDSDUMP - DUMP PERMANENT DATASETS

PDSDUMP dumps specified permanent datasets to another dataset that can then be saved or staged to a station. Datasets that have the following characteristics or conditions cannot be dumped:

- Execute-only dataset
- Dataset allocation conflict
- Catastrophic dataset error
- Inconsistent dataset allocation
- Device on which the dataset resides is down
- Inactive dataset entry in the COS Queued Dataset Table (QDT)
- Retired or migrated dataset

When dumping to a tape dataset, the recording format for the tape dataset must be transparent (for example, DF=TR on ACCESS statement). If the dataset is recorded in interchange format, loading of the dumped datasets cannot be performed.

PDSDUMP produces a listing (refer to figure 11-7) on $OUT identifying the datasets dumped or bypassed and summarizing the dump run. The date and time in the heading line refer to the time when the dump run started. The permanent dataset name, edition number, ID, and user number are extracted from the DSC entry for each dataset selected. Each message is followed by the notation DUMPED, DUMPED AND DELETED, or NOT DUMPED. The notation NOT DUMPED indicates the dataset was selected but could not be accessed for dumping. A user logfile message further explains the problem encountered.

Format:

```
PDSDUMP,DN=dn,DV=ldv,PDN/PDS=pdn,ED=ed,CW=cw,ID=uid,US=usn,OWN=ov,

 INC=mm/dd/yy:'hh:mm:ss',ARC=mm/dd/yy:'hh:mm:ss',

 TS=opt,X,C,D,B,SO,I,O,S.
```

DN=*dn* Name of dataset to which dump is written. The default is $PDS. Multiple dumps to a dataset are possible; if the dataset specified already exists, the dump is appended to it.

DV=*ldv* Dumps all datasets residing on logical device *ldv*. Currently only one *ldv* can be specified. (By default, all permanent datasets that could be specified by the parameters are dumped.) Datasets can be limited by the B parameter.

PDN
PDS=*pdn* Dumps all editions of the specified permanent dataset. Editions can be limited by ED parameter.†

ED=*ed* Edition number of permanent dataset dumped; meaningful only if PDS parameter is specified.†

CW=*cw* Site-defined control word regulating use of PDSDUMP. If the CW parameter is omitted, only the datasets belonging to the job owner can be dumped. If the CW parameter is present and the correct control word is used, any dataset can be dumped. If an invalid control word is given, the job step is aborted.

ID=*uid* Dumps all datasets with additional user identification as specified.† If ID is specified without a value, all datasets that meet the rest of the criteria and have a null ID are dumped.

US=*usn* Dumps all datasets with specified user number†

OWN=*ov* Dumps all datasets with specified ownership value†

INC=*mm/dd/yy*:'*hh*:*mm*:*ss*'
Incremental dump. Dumps only datasets modified since the specified date and time.

ARC=*mm/dd/yy*:'*hh*:*mm*:*ss*'
Archive datasets. Dumps and deletes datasets, regardless of the D option, that have not been accessed since the specified date and time.

TS=*opt* Time-stamp conversion option. *opt* may be one of the following:

CURR	Writes time-stamp in whatever format is the current system default for writing time-stamps
NS	Writes time-stamp in nanosecond (new) format
RT	Writes time-stamp in real-time clock (old) format
SAME	Does not convert time-stamp

If TS is not specified, TS=CURR is assumed.

X Dumps expired datasets

C Dumps selected datasets never dumped or datasets modified or adjusted since the last dump of the dataset

D Deletes datasets that are dumped

B Dumps only datasets that begin on the logical device specified by the DV parameter

SO Performs selection only (suppress actual dumping or deletion)

I Dumps system input datasets

O Dumps system output datasets

S Dumps user permanent datasets

† By default, all permanent datasets that match the criteria specified by the parameters are dumped.

NOTE

If none of the I, O, or S parameters is specified, the input, output, and user permanent datasets are all dumped. If any of these parameters is specified, only those datasets of the type specified are dumped.

Multiple calls to PDSDUMP can be made if the dump dataset is to include several permanent datasets requiring specification of different parameters.

Example:

```
PDSDUMP,DN=DUMPA,PDS=LIB1.
PDSDUMP,DN=DUMPA,PDS=LIB2.
```

This example results in a dataset DUMPA that contains all editions of LIB1 and all editions of LIB2.

PDSDUMP produces a listing (refer to figure 11-7) on $OUT identifying the datasets dumped or bypassed and summarizing the dump run. The date and time in the heading line refer to the time when the dump run started. The permanent dataset name, edition number, ID, and user number are extracted from the DSC entry for each dataset selected. Each message is followed by the notation DUMPED, DUMPED AND DELETED, or NOT DUMPED. The notation NOT DUMPED indicates the dataset was selected but could not be accessed for dumping. A user logfile message further explains the problem encountered.

When dumping to a tape dataset, the recording format for the tape dataset must be transparent (for example, DF=TR on ACCESS statement). If the dataset is recorded in interchange format, loading of the dumped datasets cannot be performed.

```
PDSDUMP - PERMANENT DATASET DUMP UTILITY   DUMP ON 08/15/85 AT   14:50:44
AUDPL            ED=0001 ID=QITTYQAT USR=SYSTEM          DUMPED
AUDPL            ED=0002 ID=QITTYQAT USR=SYSTEM          DUMPED
DSCED            ED=0001 ID=QITTYQAT USR=SYSTEM          DUMPED
DSCED            ED=0002 ID=QITTYQAT USR=SYSTEM          DUMPED
TXBUILD          ED=0001 ID=QITTYQAT USR=SYSTEM          DUMPED
TXBUILD          ED=0002 ID=QITTYQAT USR=SYSTEM          DUMPED
TXBUILD          ED=0003 ID=QITTYQAT USR=SYSTEM          DUMPED
LONGDATASETNAME ED=0001 ID=QITTYQAT USR=SYSTEM          DUMPED
LONGDATASETNAME ED=0002 ID=QITTYQAT USR=SYSTEM          DUMPED
LONGDATASETNAME ED=0003 ID=QITTYQAT USR=SYSTEM          DUMPED
LONGDATASETNAME ED=0004 ID=QITTYQAT USR=SYSTEM          DUMPED
DSBUILD          ED=0001 ID=QITTYQAT USR=SYSTEM          DUMPED
DSBUILD          ED=0002 ID=QITTYQAT USR=SYSTEM          DUMPED
DSBUILD          ED=0003 ID=QITTYQAT USR=SYSTEM          DUMPED
DSBUILD          ED=0004 ID=QITTYQAT USR=SYSTEM          NOT DUMPED
AUDPL            ED=0003 ID=QITTYQAT USR=SYSTEM          DUMPED
DSCED            ED=0003 ID=QITTYQAT USR=SYSTEM          DUMPED
TXBUILD          ED=0004 ID=QITTYQAT USR=SYSTEM          DUMPED
AUDPL            ED=0004 ID=QITTYQAT USR=SYSTEM          DUMPED
DSCED            ED=0004 ID=QITTYQAT USR=SYSTEM          DUMPED
             20 DATASETS SELECTED FOR DUMPING
```

Figure 11-7. PDSDUMP Listing

11.3 PDSLOAD - LOAD PERMANENT DATASETS

PDSLOAD loads permanent datasets from a dataset created by PDSDUMP. If any of the permanent datasets already exist on Cray mass storage, they are reloaded only if the RP parameter is present.

Format:

```
                        PDN
PDSLOAD,L=ldn,DN=dn,PDS=pdn,ED=ed,CW=cw,ID=uid,NID=nuid,

  US=usn,OWN=ov,NOWN=nov,DV=dvn,RP,CR,A,I,O,S,NA,SO,TLA.
```

L=*ldn* Lists dataset name. The default is $OUT.

DN=*dn* Name of the dataset from which permanent datasets are to be loaded. The default is $PDS.

PDN
PDS=*pdn* Loads all editions of the specified permanent dataset. Editions can be limited by the ED parameter.†

ED=*ed* Edition number of the dataset to be loaded; meaningful only if the PDS parameter is specified.†

CW=*cw* Installation-defined control word regulating the use of PDSLOAD. If CW is omitted, only datasets belonging to the job owner are loaded.

ID=*uid* Loads all datasets with additional user identification as specified

NID=*nuid* Loads selected datasets with new user identification. This parameter changes the user identification of selected datasets.

US=*usn* Loads all datasets with the specified user number†

OWN=*ov* Loads all datasets with the specified ownership value†

NOWN=*nov* Loads selected datasets to owner *nov*. This parameter changes the ownership value of the selected datasets.

DV=*dvn* Name of logical device the output dataset is assigned before it is opened. If omitted, COS assigns a device at open time. If this parameter is specified, the device name is requested for the output dataset (the one being loaded). COS can choose not to honor this assignment (for example, the device might not be available). This parameter is not involved in selecting a dataset for loading.

RP Replaces a specified existing dataset with the one being loaded

CR Loads the most current version of a dataset, based on creation time. This option allows incremental loads to be performed in any order.

A Loads only active datasets; that is, does not load expired datasets.

I Loads input datasets

O Loads output datasets

† By default, all permanent datasets that could be specified by the parameters are loaded.

S Loads saved datasets

NOTE

If I, O, or S is not specified, the input, output, and saved datasets are loaded. If any one of these parameters is specified, only the datasets of the type specified are loaded.

NA Does not abort if there is not a dataset matching the specifications to load on the $PDS dataset. This parameter applies only to this situation. It does not prevent any other abort condition from occurring or offer reprieve processing of any kind.

SO Performs selection only; suppresses the actual loading of datasets.

TLA Updates the time of the last access as the time that the load was performed

PDSLOAD produces a listing on the list dataset that identifies the datasets loaded or bypassed and summarizing the load run (refer to figure 11-8). The date and time in the heading line refer to the time when the load run started. The permanent dataset name, edition number, ID, and user number are extracted from the Permanent Dataset Definition Table (PDD) for each dataset selected and successfully loaded. Each message is followed by the notation LOADED or NOT LOADED. The notation NOT LOADED indicates the dataset was selected but not loaded. An error message further explains why the dataset was not loaded.

```
 PDSLOAD - PERMANENT DATASET RESTORE UTILITY  LOAD ON 01/07/82 AT 17:13:47
ENTIT           ED=0001 ID=TAQI     USR=SYSTEM          LOADED
DSBUILD         ED=0001 ID=TAQI     USR=SYSTEM          LOADED
TXBUILD         ED=0001 ID=TAQI     USR=SYSTEM          LOADED
AUDPL           ED=0001 ID=TAQI     USR=SYSTEM          LOADED
DSCED           ED=0001 ID=TAQI     USR=SYSTEM          LOADED
              5 DATASETS SELECTED FOR LOADING
```

Figure 11-8. PDSLOAD Listing

11.4 RESTORE - RECALL A DATASET TO ON-LINE DISK

RESTORE recalls retired or migrated datasets to on-line disk. The specified dataset must be present in the Master Catalog and marked as either "retired" or "migrated" and the user must have maintenance permission. RESTORE does not make the dataset local to the job.

The arguments for PDN, ID, and OWN can use the notations * to indicate any one character and - to indicate an arbitrary string of characters.

Format:

```
RESTORE,PDN=pdn,ID=id,ED=ed,OWN=ov,M=m,TYPE=type.
```

The only required parameter is PDN.

PDN=*pdn* Permanent dataset name; required parameter. The keyword cannot appear alone.

ID=*id* Permanent dataset ID. If this parameter is omitted or present without a value, the ID is null.

ED=*ed* Edition number of the permanent dataset. Options for *ed* are as follows:

Specification	Meaning
Unsigned integer (*ed*) Example: ED=2	The specific edition of the dataset
Negative integer (-*ed*) Example: ED=-2	All but the *ed* highest editions
Positive integer (+*ed*) Example: ED=+2	The *ed* highest editions
ED=ALL	All editions of the dataset

The default is the highest edition.

OWN=*ov* Owner of the permanent dataset. The default is the job owner. If the requester is not the dataset owner, the requester must have maintenance permission.

M=*m* Maintenance control word. The default is null. M is required if the dataset has a maintenance control word.

TYPE=*type* Dataset type. The *type* can be RET for a retired dataset or MIG for a migrated dataset. Only on-line datasets can be selected for a PDSDUMP. The default is RET. You can specify both by using RET:MIG.

11.5 RETIRE - RETIRE A DATASET

RETIRE retires a dataset; that is, it moves an on-line or migrated dataset to backup medium. The dataset to be retired does not have to be local to the job. A retired dataset is not recalled to on-line disk by user access or by system or device reload. To recall a retired dataset, use the RESTORE utility.

The arguments for PDN, ID, and OWN can use the notations * to indicate any one character and - to indicate an arbitrary string of characters.

Format:

```
RETIRE,PDN=pdn,ID=id,ED=ed,OWN=ov,M=m,X.
```

The only required parameter is PDN.

PDN=*pdn* Permanent dataset name; required parameter. The keyword cannot appear alone.

ID=*id* Permanent dataset ID. If this parameter is omitted or present without a value, the ID is null.

ED=*ed* Edition number of the permanent dataset. Options for *ed* are as follows:

Specification	Meaning
Unsigned integer (*ed*) Example: ED=2	The specific edition of the dataset
Negative integer (-*ed*) Example: ED=-2	All but the *ed* highest editions
Positive integer (+*ed*) Example: ED=+2	The *ed* highest editions
ED=ALL	All editions of the dataset

The default is the highest edition.

OWN=*ov* Owner of the permanent dataset. The default is the job owner. If the requester is not the dataset owner, the requester must have maintenance permission.

M=*m* Maintenance control word. The default is null. M is required if the dataset has a maintenance control word.

X Specification that the dataset is to be retired only if it is expired; that is, if the retention time has been exhausted.

LOCAL DATASET UTILITIES 12

Local dataset utilities copy, position, or initialize local datasets. The following utilities are available:

Utility	Function
BLOCK	Converts an unblocked dataset to a blocked dataset
COPYD	Copies blocked datasets
COPYF	Copies files of blocked datasets
COPYR	Copies records of blocked datasets
COPYU	Copies unblocked datasets or sectors of unblocked datasets
NOTE	Writes text to a dataset
QUERY	Returns local mass storage dataset status and position information
REWIND	Positions a blocked or unblocked dataset at beginning-of-data, that is, before the first word of the dataset
SKIPD	Skips blocked datasets
SKIPF	Skips files of blocked datasets
SKIPR	Skips records of blocked datasets
SKIPU	Skips sectors of unblocked datasets
UNBLOCK	Converts a blocked dataset to an unblocked dataset
WRITEDS	Initializes a blocked random or sequential dataset

You invoke these utilities by issuing control statements in your JCL. This section describes these control statements.

12.1 BLOCK - CONVERT UNBLOCKED DATASET TO BLOCKED DATASET

BLOCK copies a specified unblocked dataset to a blocked dataset, adding blocked dataset control words as the copy proceeds. For datasets that you did not assign as foreign datasets (with the ASSIGN control statement), a fixed-record length must be provided on a control statement parameter. For datasets previously assigned as foreign, the values for record length and type are taken from the ASSIGN control statement.

Never use BLOCK with tape datasets. To use BLOCK with foreign datasets, see Foreign Dataset Conversion on CRAY-1 and CRAY X-MP Computer Systems, publication SN-0236.

The BLOCK control statement has two mutually exclusive forms, as follows:

Format 1:

```
BLOCK,DN=ldn,BLKSIZE=size.
```

Format 1 is valid for nonforeign datasets only.

DN=*ldn* Name of dataset to be blocked. When the utility terminates, the *ldn* local dataset has been replaced by the blocked copy. (During the copy process, a temporary blocked copy is made in dataset $BLOCK. BLOCK then releases the original *ldn* dataset and $BLOCK is copied back to a new dataset named *ldn*. The *ldn* dataset is rewound before and after processing.

This is a required parameter.

BLKSIZE=*size*
The BLOCK operation on nonforeign datasets merely adds Cray blocking control words to create the blocks of length equal to that specified in the BLKSIZE parameter. The BLKSIZE parameter is only used on nonforeign datasets and describes the record length in 64-bit words of the output dataset.

Format 2:

```
BLOCK,I=idn,O=odn,BLKSIZE=size.
```

I=*idn* Name of the unblocked input dataset. The copy proceeds from the current dataset position throughout the dataset to end-of-data (EOD). This is a required parameter; there is no default.

O=*odn* Name of the local dataset to which the blocked copy is written. If you previously opened this dataset (using, for instance, the job control language (JCL) ASSIGN control statement), BLOCK writes from the current position; otherwise, BLOCK creates the dataset. This is a required parameter.

BLKSIZE=*size*
For foreign datasets, appropriate Cray blocking control information corresponding to the foreign control words in the input dataset are added and the result is written to the output dataset. For datasets previously assigned as foreign, the values for record length and type are taken from the ASSIGN control statement for the input dataset. For these datasets, the BLKSIZE parameter is not permitted.

BLOCK is intended primarily as a post processor for datasets created by or for certain stations.

12.2 COPYD - COPY BLOCKED DATASET

COPYD copies one blocked dataset to another dataset starting at their current positions. Following the copy, both datasets are positioned after the end-of-file (EOF) of the last file copied. The end-of-dataset (EOD) is not written to the output dataset. COPYD expands compressed blanks when writing to the output dataset if an ASSIGN control statement contains BFI=OFF for the output dataset.

Format:

```
COPYD,I=idn,O=odn,S=m.
```

I=*idn* Name of dataset to be copied. The default is $IN.

O=*odn* Name of dataset to receive the copy. The default is $OUT.

S=*m* Shift count. The value *m* is the number of ASCII blanks to insert at the beginning of each line of a character file. The maximum is 132. If S is omitted, the shift count is 0. If S is specified without a value, S=1.

12.3 COPYF - COPY BLOCKED FILES

COPYF copies a specified number of files from one blocked dataset to another dataset starting at the current dataset position. Following the copy, the datasets are positioned after the EOF for the last file copied. COPYF expands compressed blanks when writing to the output dataset if an ASSIGN control statement contains BFI=OFF for the output dataset.

Format:

```
COPYF,I=idn,O=odn,NF=n,S=m.
```

I=*idn* Name of dataset to be copied. The default is $IN.

O=*odn* Name of dataset to receive the copy. The default is $OUT.

NF=*n* Decimal number of files to copy. The default is 1. If the dataset contains fewer than *n* files, the copy terminates on EOD. EOD is not written. If the keyword NF is specified without a value, the copy terminates at the EOD. If the input dataset is positioned midfile, the partial file counts as one file.

S=*m* Shift count. The value *m* is the number of ASCII blanks to insert at the beginning of each line of a character file. The maximum is 132. If S is omitted, the shift count is 0. If S is specified without a value, S=1.

12.4 COPYR - COPY BLOCKED RECORDS

COPYR copies a specified number of records from one blocked dataset to another dataset starting at the current dataset position. Following the copy, the datasets are positioned after the end-of-record (EOR) for the last record copied. COPYR expands compressed blanks when writing to the output dataset if an ASSIGN control statement contains BFI=OFF for the output dataset.

Format:

```
COPYR,I=idn,O=odn,NR=n,S=m.
```

I=*idn* Name of dataset to be copied. The default is $IN.

O=*odn* Name of dataset to receive the copy. The default is $OUT.

NR=*n* Decimal number of records to copy. The default is 1. If the dataset contains fewer than *n* records, the copy terminates on the next EOF. EOF and EOD are not written. If the keyword NR is specified without a value, the copy terminates at the next EOF. If the input dataset is positioned midrecord, the partial record is counted as one record.

S=*m* Shift count. The value *m* is the number of ASCII blanks to insert at the beginning of each line of a character file. The maximum is 132. If S is omitted, the shift count is 0. If S is specified without a value, S=1.

12.5 COPYU - COPY UNBLOCKED DATASETS

COPYU copies a specified number of sectors or all data until EOD. The copy is made to or from the current position on both datasets. At the end of the copy, the datasets remain positioned after the last sector copied.

Format:

```
COPYU,I=idn,O=odn,NS=ns.
```

Parameters I and O are required; they have no defaults.

I=*idn* Name of unblocked dataset to be copied

O=*odn* Name of unblocked dataset to receive the copy

NS=*ns* Decimal number of sectors to copy. The default is 1. If the unblocked dataset contains fewer than *ns* sectors, the copy terminates on EOD. If the keyword NS is specified without a value, the copy terminates at EOD.

12.6 NOTE - WRITE TEXT TO A DATASET

NOTE writes text included in the NOTE control statement to a dataset named in the control statement.

Format:

```
NOTE,DN=dn,TEXT=text.
```

DN=*dn* Name of the dataset to be written. The dataset is written at its current position. If the dataset does not exist, it is created. The dataset is not rewound. If DN is omitted or appears without a value, the dataset defaults to $OUT.

TEXT=*text*
Information to be written to the dataset. The text can have a maximum of 153 characters. It is subject to the same conventions as other strings, as discussed in subsection 16.2.4, Strings.

12.7 QUERY - RETURN STATUS AND POSITION INFORMATION

QUERY determines the current status and position of a local mass storage dataset. QUERY issues this information in the form of a user logfile message. It can also set this information in user-specified symbolic variables for later use in JCL statements.

Format:

```
QUERY,DN=ldn,STATUS=sym,POS=sym.
```

DN=*ldn* Local dataset name, 1 to 7 characters. This is a required parameter.

STATUS=*sym*
JCL symbol name in which the dataset status is to be returned. Symbols are described in subsection 16.2.1.3, Symbolic variables. Return values are as follows:

-1	Dataset is not local
0	Dataset is closed
1	Dataset is open for output
2	Dataset is open for input
3	Dataset is open for I/O

POS=*sym* JCL symbol name in which the dataset position is to be returned. Return values are as follows:

-1	Position indeterminate (dataset is either not local, unblocked format, or closed)
0	Beginning-of-data
1	End-of-data
2	End-of-file
3	End-of-record
4	Mid-record

The logfile message issued has the format:

QU001 - DN: ldn STATUS: status POS: pos

ldn Local dataset name

status UNKNOWN if *ldn* is not local
CLOSED if *ldn* is local and closed
OPEN-O if *ldn* is local and open for output
OPEN-I if *ldn* is local and open for input
OPEN-I/O if *ldn* is local and open for both input and output

pos N/A position is not available (dataset is not local, is closed, or is in unblocked format)
BOD if dataset is at beginning-of-data
EOD if dataset is at end-of-data
EOF if dataset is at end-of-file
EOR if dataset is at end-of-record
MID if dataset is in the middle of a record

12.8 REWIND - REWIND BLOCKED OR UNBLOCKED DATASET

REWIND positions the named datasets at the beginning-of-data (BOD). REWIND opens any of the named datasets that are not open. REWIND is a system verb. The $IN dataset, however, is an exception. After REWIND, $IN is positioned after the control statement file.

REWIND causes an EOD to be written to the dataset if the previous operation was a write or if the dataset is null. If the dataset is not memory resident, the buffers are flushed to mass storage when REWIND follows a write operation. If the dataset is memory resident, the EOD is still placed in the buffer, but the buffer is not flushed. For an on-line magnetic tape dataset, REWIND positions the tape dataset to the beginning of the first volume accessed by the user.

Format:

```
REWIND,DN=dn1:dn2:...:dn8.
```

DN=dn_i Names of datasets to be rewound. A maximum of eight datasets can be specified, separated by colons.

12.9 SKIPD - SKIP BLOCKED DATASET

SKIPD positions a blocked dataset at EOD (after the last EOF of the dataset). It has the same effect as the following statement:

```
SKIPF,DN=dn,NF.
```

If the specified dataset is empty or already at EOD, the statement has no effect.

Format:

```
SKIPD,DN=dn.
```

DN=*dn* Name of dataset to be skipped. The default is $IN.

12.10 SKIPF - SKIP BLOCKED FILES

SKIPF bypasses a specified number of files from the current position of the named blocked dataset.

Format:

```
SKIPF,DN=dn,NF=n.
```

DN=*dn* Name of dataset. The default is $IN.

NF=*n* Number of files to bypass. The default is 1. If the keyword NF is specified without a value, the system positions *dn* after the last EOF of the dataset. If *n* is negative, SKIPF skips backward on *dn*. If *dn* is positioned midfile, the partial file skipped counts as one file.

SKIPF does not bypass an EOD or BOD. If BOD is encountered before *n* files have been bypassed when skipping backward, the dataset is positioned after the BOD. When skipping forward, the dataset is positioned before the EOD of the current file.

This utility is available for use with on-line tapes, except that a negative value cannot be used for NF; for interchange format tapes (DF=IC), NF can only be 1.

For example, if *dn* is positioned just after an EOF, the following control statement positions *dn* after the previous EOF. If *dn* is positioned midfile, *dn* is positioned at the beginning of that file.

```
SKIPF,DN=dn,NF=-1.
```

12.11 SKIPR - SKIP BLOCKED RECORDS

SKIPR bypasses a specified number of records from the current position of the named blocked dataset.

Format:

```
SKIPR,DN=dn,NR=n.
```

DN=*dn* Name of dataset. The default is $IN.

NR=*n* Number of records to skip. The default is 1. If the keyword NR is specified without a value, the system positions *dn* after the last EOR of the current file. If *n* is negative, SKIPR skips backward on *dn*. If *dn* is positioned in the middle of the record, the partial record skipped counts as one record.

SKIPR does not bypass an EOF or BOD. If an EOF or BOD is encountered before *n* records have been bypassed when

NR=*n* (continued) skipping backward, the dataset is positioned after the EOF or BOD. When skipping forward, the dataset is positioned after the last EOR of the current file.

This utility is available for use with on-line tapes except that a negative value cannot be used for NR.

12.12 SKIPU - SKIP UNBLOCKED DATASET

SKIPU bypasses a specified number of sectors or all data from the current position of the named unblocked dataset.

Format:

```
 ____________________
|                    |
| SKIPU,DN=dn,NS=ns. |
|____________________|
```

DN=*dn* Name of unblocked dataset. There is no default value.

NS=*ns* Number of sectors to bypass. The default is 1. If the keyword NS is specified without a value, the system positions *dn* after the last sector of the dataset. If *ns* is negative, SKIPU skips backwards on *dn*.

12.13 UNBLOCK - CONVERT BLOCKED DATASET TO UNBLOCKED DATASET

UNBLOCK copies a specified blocked dataset to an unblocked dataset, removing all blocked dataset control words as the copy proceeds. When you assign the input dataset as foreign, the ASSIGN control statement also causes addition of control words, as appropriate, for the foreign host according to the blocking and record format information from ASSIGN.

Never use UNBLOCK with tape datasets.

The UNBLOCK control statement has two mutually exclusive forms, as follows:

Format 1:

```
UNBLOCK,DN=ldn.
```

This format is illegal for foreign datasets.

DN=*ldn* Name of dataset to be unblocked. During the copy process, a temporary unblocked copy is made in the dataset $UNBLK. The original *ldn* dataset is then released and $UNBLK is copied back to a new dataset named *ldn*. When the utility terminates, the *ldn* local dataset has been replaced by the unblocked copy. The *ldn* dataset is rewound before and after processing.

This is a required parameter.

Format 2:

```
UNBLOCK,I=idn,O=odn.
```

I=*idn* Name of the blocked input dataset. The unblocking copy proceeds from the current dataset position through the dataset to EOD for nonforeign datasets, and to EOF for foreign datasets. The default is $IN.

O=*odn* Name of the local dataset to which the unblocked copy is written. If you previously marked the dataset to be unblocked (using, for instance, the JCL ASSIGN statement), UNBLOCK writes from the current position. Otherwise, UNBLOCK closes the dataset and assigns the unblocked attribute. This has the effect of rewriting the dataset, losing its previous content. This is a required parameter.

The UNBLOCK operation on nonforeign datasets merely discards the blocked dataset control words. (Refer to section 2 for a detailed description of the blocked format and its control words.) For foreign datasets, it also adds appropriate host control information so that you can dispose the dataset (use the DISPOSE control statement) transparently to a supported front end. In a nonforeign dataset containing text, it discards record boundaries. The UNBLOCK utility is intended primarily as a postprocessor for datasets created by or for certain stations.

12.14 WRITEDS - INITIALIZE A BLOCKED RANDOM OR SEQUENTIAL DATASET

WRITEDS initializes a blocked dataset. It writes a dataset containing a single file consisting of a specified number of records of a specified length. This utility is useful only for random datasets, because a record written on a random dataset must end on a preexisting record boundary. Direct-access datasets, implemented in Cray Fortran CFT77 and CFT as defined by the ANSI X3.9-1978 Fortran standard, can be initialized, and even extended, using WRITEDS.

You can also use WRITEDS to write a sequential dataset.

Format:

```
WRITEDS,DN=dn,NR=nr,RL=rl.
```

DN=*dn* Name of dataset to be written. DN is a required parameter.

NR=*nr* Decimal number of records to be written. NR is a required parameter set to the largest value that may be needed, because a dataset is generally not extended when it is in random mode.

RL=*rl* Decimal record length (the number of words in each record). The default is zero words, which generates a null record.

If the record length is nonzero, the first word of each record is the record number, represented as a binary integer starting with 1.

ANALYTICAL AIDS 13

The following utilities provide analytical aids to the programmer:

Utility	Function
COMPARE	Compares two blocked datasets and lists all differences
DDA†	Dynamic Dump Analyzer. Allows interactive symbolic analysis of a dump. The Symbolic Debugging Package Reference Manual, CRI publication SR-0112, describes DDA in detail.
DEBUG	Produces a symbolic dump. The Symbolic Debugging Package Reference Manual, CRI publication SR-0112, describes DEBUG in detail.
DSDUMP	Dumps all or part of a dataset to another dataset. The input dataset may be either blocked or unblocked.
DUMP and DUMPJOB	Generally used together to examine the contents of registers and memory as they were at a specific time during job processing. DUMPJOB captures the information so that DUMP can later format selected parts of it.
FLODUMP	Dumps flowtrace tables when a program aborts with flowtracing active. Refer to the COS Performance Utilities Reference Manual, publication SR-0146, for a description of FLODUMP.
FTREF	Analyzes Fortran source code to show the calling tree, common block usage, and information for multitasking. Refer to the COS Performance Utilities Reference Manual, publication SR-0146, for a description of FTREF.
ITEMIZE	Inspects library datasets and generates statistics about them. Section 5 describes libraries; section 15 describes library dataset management.

† Deferred implementation

MTDUMP	Produces formatted listings of dumps of the multitasking history buffer. Refer to the CRAY X-MP Multitasking Programmer's Manual, publication SN-0222, for more information.
PERFMON	Monitors machine activity in detail, by means of the performance monitor that is part of most CRAY X-MP computer systems. Refer to the COS Performance Utilities Reference Manual, publication SR-0146.
PRINT	Writes the value of an expression to the logfile
SPY	Indicates approximate amounts of time used by different loops and code segments, including a histogram to show "spikes." Refer to the COS Performance Utilities Reference Manual, publication SR-0146.
SYSREF	Generates a global cross-reference listing for a group of Cray Assembly Language (CAL) or APML programs

You can invoke these aids by including a control statement in your JCL. This section describes these control statements.

13.1 COMPARE - COMPARE DATASETS

COMPARE compares two blocked datasets and lists all differences found. The output consists of a listing of the location of each discrepancy, the contents of the differing portions of the datasets, and a message indicating the number of discrepancies. Refer to the COS Message Manual, CRI publication SR-0039.

Keyword parameters let you specify the maximum number of errors and the amount of context to be listed.

If portions of two datasets are being compared, the portions must be copied to separate datasets before comparison; COMPARE compares complete datasets only.

COMPARE rewinds both input datasets before and after the comparison.

Format:

```
COMPARE,A=adn,B=bdn,L=ldn,DF=df,ME=maxe,CP=cpn,
  CS=csn,CW=cw1:cw2,ABORT=ac.
```

A=*adn* and B=*bdn*
: Input dataset names. If *adn*=*bdn*, COMPARE issues an error message and aborts the job step. Both A and B are required parameters.

L=*ldn*
: Dataset name for the list of discrepancies. *ldn* must be different from *adn* and *bdn*. The default is $OUT.

DF=*df*
: Input dataset format. The default is T. *df* is a 1-character alphabetic code as follows:

  B Binary. The input datasets are compared logically to verify that they are identical. If they are not identical, the differing words are printed in octal and as ASCII characters. Nonprinting characters appear as blanks in the ASCII representation. The location printed is a word count. The first word of each dataset is called word 1.

  T Text. The input datasets are compared to see if they are equivalent as text. For example, a blank-compressed record and its expansion are considered equivalent. If the two datasets are not equivalent, the differing records are printed as text. The location is printed as a record count. The first record of each dataset is called record 1.

ME=*maxe*
: Maximum number of differences printed. The default is 100.

CP=*cpn*
: Amount of context printed. *cpn* records to either side of a difference are printed. The CP parameter applies only if DF=T; if DF=B and CP are specified, an error message is generated. The default is 0.

CS=*csn*
: Amount of context scanned. *csn* records to either side of a discrepancy are scanned for a match. The CS parameter applies only if DF=T; if DF=B and CS are specified, an error message is generated. The default is 0.

  If a match is found within the defined range, subsequent comparisons are made at the same interval. That is, if record 275 of dataset A is equivalent to record 277 of dataset B, the next comparison is between record 276 of dataset A and record 278 of dataset B.

NOTE

If identical records occur within *csn* records of each other, the pairing is ambiguous and COMPARE can match the wrong pair.

CW=*cw* or CW=cw_1:cw_2
: Compare width. If CW=*cw* is specified, columns 1 through *cw* are compared. If CW=cw_1:cw_2 is specified, columns cw_1 through cw_2 are compared. Specifying CW without a value is not permitted. The default is to compare columns 1 through 133, but this can be changed by installation option. The CW parameter applies only if DF=T; if DF=B and CW are specified, an error message is generated.

ABORT=*ac*
: If *ac* or more differences are found, the job step aborts. Specifying ABORT alone is equivalent to ABORT=1 and causes an abort if any differences are found. Specifying ABORT does not prevent the listing of up to *maxe* differences.

13.2 DSDUMP - DUMP DATASET

DSDUMP dumps specified portions of a dataset to another dataset. A disk dataset can be dumped in either blocked or unblocked format. A tape dataset can be dumped only in blocked format.

Unblocked format is used to dump a disk dataset without regard to whether it is blocked. Dumping a blocked dataset in unblocked format (by sectors) is possible. A group of sectors within the dataset or a group of words within each sector can be selected. The initial word and initial sector numbers are relative to the beginning of the dataset. Specifying an initial sector greater than 1 causes sectors to be skipped from the beginning of the dataset; specifying an initial word greater than 1 (or 0, if the control statement includes the Z parameter) causes words to be skipped from the beginning of each sector. Following a dump in unblocked format, the dataset is closed.

For a blocked format, a group of words within a record, a group of records within a file, or a group of files within a dataset can be selected. The initial word number, initial record number, and initial file number are relative to the current dataset position. Specifying an initial number greater than 1 (or 0, if the control statement includes the Z parameter) causes words, records, or files to be skipped starting from the current position.

Because the initial word, record, or file number is relative to the current position of the dataset, the dataset must be positioned properly before calling DSDUMP. If you rewind the dataset before calling DSDUMP, the initial word, record, and file numbers are relative to the beginning of the dataset. When DSDUMP is completed, the input dataset is positioned after the last record dumped.

Two groups of DSDUMP parameters require the specification of numbers: the values of the initial word, record, file, and sector (I values) and their counts (N values). These values may be specified in three ways:

- Explicit decimal number (for example, D'1234' or D1234)
- Explicit octal number (for example, O'1234' or O1234)
- Simple number (for example, 1234). This is interpreted as a decimal number.

The following lines reference the same first word:

```
DSDUMP,...,IW=4096.
DSDUMP,...,IW=D'4096'.
DSDUMP,...,IW=O'10000'.
```

Format:

```
DSDUMP,I=idn,O=odn,DF=df,IW=n,NW=n,IR=n,NR=n,IF=n,
  NF=n,IS=n,NS=n,Z,DB=db,DSZ=sz.
```

The only required parameter is I.

I=*idn* (or DN=*idn*)
: Name of dataset to be dumped. This is a required parameter.

O=*odn* (or L=*odn*)
: Name of dataset to receive the dump. The default is $OUT.

DF=*df*
: Dump format. The default is B.

  B Blocked
  U Unblocked

IW=*n*
: Decimal or octal number (*n*) of the initial word for each record or sector on *idn*. The default is 0 if Z is specified; 1 if Z is not specified.

NW=*n*
: Decimal or octal number (*n*) of the words per record or sector to dump. Specifying NW without a value dumps all words to the end of a record or sector. The default is 1.

IR=*n*
: Decimal or octal number (*n*) of the initial record for each file on *idn*. Applicable only if DF=B. The default is 0 if Z is specified; 1 if Z is not specified.

NR=*n* Decimal or octal number (*n*) of the records per file to dump. Specifying NR without a value dumps all records to the end of the file. Applicable only if DF=B. The default is 1.

IF=*n* Decimal or octal number (*n*) of the initial file of the dataset on *idn*. Applicable only if DF=B. The default is 0 if Z is specified; 1 if Z is not specified.

NF=*n* Decimal or octal number (*n*) of the files on *idn* to dump. Specifying NF without a value dumps all files to the end of the dataset. Applicable only if DF=B. The default is 1.

IS=*n* Decimal or octal number (*n*) of the initial sector on *idn*. Applicable only if DF=U. The default is 0 if Z is specified; 1 if Z is not specified.

NS=*n* Decimal or octal number (*n*) of the sectors to dump. Specifying NS without a value dumps all sectors to the end of the dataset. Applicable only if DF=U. The default is 1.

Z Zero-based initial-value parameters (IW, IR, IF, and IS). If Z is specified, the value for each I parameter is 0, and output referring to word, record, file, and sector numbers begins at 0. The following lines reference the same first word:

```
DSDUMP,...,IW=4096.
DSDUMP,...,Z,IW=4095.
```

If Z is not specified, the value for each I parameter is 1.

The Z parameter does not affect the Nx parameters.

DB=*db* Numeric base in which to display the data words

OCTAL or O	Octal (base 8)
HEX or X	Hexadecimal (base 16)

The default is OCTAL.

DSZ=*sz* Size of the data items to dump

WORD or W	Cray 64-bit words
PARCEL or P	Cray 16-bit parcels

The default is WORD.

For blocked format, each record from *idn* dumped to *odn* is preceded by a header specifying the file and record number in both octal and decimal. For unblocked format, each sector is preceded by a header specifying the sector number in both octal and decimal.

Table 13-1 summarizes the DSDUMP output records according to the specification of DB and DSZ parameters.

A row of five asterisks indicates that one or more groups of 4 words have not been formatted because they are identical to the previous 4 words. Only the first group is formatted. The number of words not formatted can be determined from the word counts of the formatted lines before and after the asterisks. The final group of 4 or fewer words is always formatted.

Table 13-1. DSDUMP Output Format

DB,DSZ	Word Count	Number Interpretation	ASCII Interpretation
OCTAL,WORD	†	Four 22-digit octal numbers	One 32-character interpretation
HEX,WORD	†	Four 16-digit hexadecimal numbers	One 32-character interpretation
OCTAL,PARCEL	†	Sixteen 6-digit octal numbers	None (insufficient space)
HEX,PARCEL	†	Sixteen 4-digit hexadecimal numbers	One 32-character interpretation

† If the Z parameter is used, the word count is 0-based and octal. If the Z parameter is not used, the word count is 1-based and decimal.

13.3 DUMP - DUMP REGISTERS AND MEMORY

DUMP reads and formats selected parts of the memory image that is contained in $DUMP and writes the information to another dataset. The DUMP control statement can be placed anywhere in the control statement file after $DUMP has been created by the DUMPJOB control statement.

Normally the DUMPJOB and DUMP control statements are placed after an EXIT control statement. This ensures the dump is performed no matter which part of the job causes an error exit. The use of DUMP and DUMPJOB is not, however, restricted to this purpose.

DUMP can be called any number of times within a job. This might be done to dump selected portions of memory from a single $DUMP dataset or it might be done if $DUMP has been created more than once in a single job.

Format:

```
DUMP,I=idn,O=odn,FWA=fwa,LWA=lwa,JTA,NXP,V,DSP,FORMAT=f,CENTER,
  BIAS=address,BUFFER.
```

I=*idn* Name of the dataset containing the memory image. The default dataset $DUMP is created by DUMPJOB but any dataset in the $DUMP (unblocked) format is acceptable.

O=*odn* Name of the dataset to receive the dump; default is $OUT.

FWA=*fwa* First word address of memory to dump. The default is word 0 of the Job Communication Block (JCB).

LWA=*lwa* Last word address of memory to dump. The default is word 200 of the JCB. Specifying the keyword LWA without a value causes the limit address to be used. Specifying LWA=0 causes no memory to be dumped.

JTA Dump Job Table Area. The default is no JTA dump.

NXP No dump of Exchange Package, B registers, T registers, cluster registers, or semaphore registers dumped. The default causes the Exchange Package, B registers, T registers, cluster registers, and semaphore registers to be dumped. Cluster registers and semaphore registers are available only on CRAY X-MP mainframe types. NXP overrides the V parameter if the two are used together.

V Dumps vector registers. The default is no dump of V registers.

DSP Dumps Logical File Tables (LFTs) and Dataset Parameter Tables (DSPs). The default is no LFTs and DSPs are dumped.

FORMAT=*f* Format for the part of memory selected by FWA and LWA. All of the following options except I are appropriate for formatting a data dump. The I format is for dumping program instructions only. O is the default.

D Decimal numbers and ASCII character

FORMAT=*f* (continued)

G Floating-point or exponential numbers, depending on the value of the number, and ASCII character

I Instruction format. CAL instruction mnemonics are printed with ASCII characters.

M Mixed hexadecimal and octal numbers and characters written in ASCII. Each 16-bit parcel is represented as 5 characters; the first character is a hexadecimal digit representing the high-order 4 bits, and the next 4 are octal characters representing the low-order 12 bits.

O Octal numbers and ASCII characters

P Dump is given in 16-bit parcels (4-word boundaries are forced for FWA and LWA)

X Hexadecimal numbers and ASCII characters

CENTER Dump 100_8 words on each side of the address in the P register of the Exchange Package. The format is P.

BIAS=*address*
Print of dump will begin at user *address*

BUFFER Dump I/O buffers

Example 1:

The following example is a portion of a data dump obtained using format O, the default format type.

```
0000100 0524611523106020200000 0360000001403200125000 0000100003100000030200 0000163230400000030144 U1520A  <          2  0   i   0d
0000104 0000010000000000000000 1144507000650140022000 0415172462006113430466 0000000000000000000000          (  A $ COS 1.16
0000110 0000000000000000000000 0000000000000000000000 0000000000000000000000 0374400000000000000000                      ?
0000114 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0000120*0000000000000000000000 THRU 0000157
0000160 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0000164 0000000000000000000000 0000000000000000000000 0300711363107113634066 0310601643146416430064                09/29/8620:34:04
0000170*0000000000000000000000 THRU 0000177
0000200 0425302024652023042040 0104000010045217620000 0002211104247237625601 0250021000000045422102 EXAMPLE     *?    SE: + *     $B
```

1794

Example 2:

The same portion of the dump in format D.

```
0000100     6138746256756899840     4323455745791142400       2252014562062464       4056682510430308 U1520A  <          2  0   i   0d
0000104      281474976710656     -7410426839300037632     4850186721430483254                      0          (  A $ COS 1.16
0000110                    0                        0                        0     4548635623644200960                      ?
0000114                    0                        0                        0                        0
0000120*0000000000000000000000 THRU 0000157
0000160                    0                        0                        0                        0
0000164                    0                        0     3474860480247314486     3616454492372480052                09/29/8620:34:04
0000170*0000000000000000000000 THRU 0000177
0000200     4996815586883028256     1224979653404336128       40853751375801217     3027017083928323138 EXAMPLE     *?    SE: + *     $B
```

1795

Example 3:

The same portion of the dump in format X.

```
0000100       5531353230410000       3C0000181A00AA00       0008003200003080       000E698800003064 U1520A  <           2  0   i   0d
0000104       0001000000000000       9928E00D41802400       434F5320312E3136       0000000000000000           (  A $ COS 1.16
0000110       0000000000000000       0000000000000000       0000000000000000       3F2000000000000                        ?
0000114       0000000000000000       0000000000000000       0000000000000000       0000000000000000
0000120*0000000000000000000000000 THRU 0000157
0000160       0000000000000000       0000000000000000       0000000000000000       0000000000000000
0000164       0000000000000000       0000000000000000       30392F32392F3836       32303A33343A3034                09/29/8620:34:04
0000170*0000000000000000000000000 THRU 0000177
0000200       455841AD504C4520       110000812A3F2000       009124453A7F2B81       2A022000000962442 EXAMPLE     *?    $E: + *     $B
```

1796

Example 4:

The same portion of the dump in format G.

```
0000100    0.254311434435+1633    0.204573742730-311    0.0000000000000E+00    0.0000000000000E+00 U1520A  <           2  0   i   0d
0000104    0.0000000000000E+00    0.0000000000000E+00    0.3047206886818+255    0.0000000000000E+00           (  A $ COS 1.16
0000110    0.0000000000000E+00    0.0000000000000E+00    0.0000000000000E+00    0.0000000000000E+00                        ?
0000114    0.0000000000000E+00    0.0000000000000E+00    0.0000000000000E+00    0.0000000000000E+00
0000120*0000000000000000000000 THRU 0000157
0000160    0.0000000000000E+00    0.0000000000000E+00    0.0000000000000E+00    0.0000000000000E+00
0000164    0.0000000000000E+00    0.0000000000000E+00    0.254398439672-1216    0.821520619736-1065                09/29/8620:34:04
0000170*0000000000000000000000 THRU 0000177
0000200    0.1643314277866+412    0.0000000000000E+00    0.0000000000000E+00    0.198625290891-1695 EXAMPLE     *?    $E: + *     $B
```

1797

Example 5:

The same portion of the dump in format P.

```
0000100 052461 032462 030101 000000  036000 000030 015000 125000 U1520A  <
0000102 000010 000062 000000 030200  000016 064610 000000 030144    2  0   i   0d
0000104 000001 000000 000000 000000  114450 160015 040600 022000          (  A $
0000106 041517 051440 030456 030466  000000 000000 000000 000000 COS 1.16
0000110 000000 000000 000000 000000  000000 000000 000000 000000                ?
0000112 000000 000000 000000 000000  037440 000000 000000 000000
0000114 000000 000000 000000 000000  000000 000000 000000 000000
0000116 000000 000000 000000 000000  000000 000000 000000 000000
0000120*000000000000000000000000 THRU 0000157
0000160 000000 000000 000000 000000  000000 000000 000000 000000
0000162 000000 000000 000000 000000  000000 000000 000000 000000
0000164 000000 000000 000000 000000  000000 000000 000000 000000
0000166 030071 027462 034457 034066  031060 035063 032072 030064 09/29/8620:34:04
0000170*000000000000000000000000 THRU 0000177
0000200 042530 040515 050114 042440  010400 000201 025077 020000 EXAMPLE     *?
0000202 000221 022105 035177 025601  025002 020000 000226 022102   $E: + *     $B
```

1798

Example 6:

The same portion of the dump in format M.

```
0000100       5531353230410000       3C0000181A00AA00       052461152310602020000 0360000001403200125000 U1520A  <
0000102       0008003200003080       000E6988U0003064       000010000310000003020O 000016323040000000030144    2  0   i   0d
0000104       0001000000000000       9928E00D41802400       000001000000000000000 114450700065014002200          (  A $
0000106       434F5320312E3136       0000000000000000       041517246200611343046G 000000000000000000000000 COS 1.16
0000110       0000000000000000       0000000000000000       000000000000000000000 000000000000000000000000
0000112       0000000000000000       3F2000000000000        000000000000000000000 037440000000000000000000          ?
0000114       0000000000000000       0000000000000000       000000000000000000000 000000000000000000000000
0000116       0000000000000000       0000000000000000       000000000000000000000 000000000000000000000000
0000120*0000000000000000000000 THRU 0000157
0000160       0000000000000000       0000000000000000       000000000000000000000 000000000000000000000000
0000162       0000000000000000       0000000000000000       000000000000000000000 000000000000000000000000
0000164       0000000000000000       0000000000000000       000000000000000000000 000000000000000000000000
0000166       30392F32392F3836       32303A33343A3034       030071136310711363406G 031060164314641643006 4 09/29/8620:34:04
0000170*0000000000000000000000 THRU 0000177
0000200       455841AD504C4520       110000812A3F2000       042530202465202304244O 010400001004521762O000 EXAMPLE     *?
0000202       009124453A7F2B81       2A022000000962442      000221110424723762560I 025002100000004542210Z   $E: + *     $B
```

1799

Example 7:

A dump of program instructions in format I.

```
0000070*0000000000000000000000 THRU 0000077
0000100a SO       S4<61          A4        A6*A2          052461152310602020000    U1520A
         A1       0*A1           ERR       000
0000101a TOO,A0   ,A0            ERR       030            036000001403200125000    <
         JSN      25200a
0000102a ERR      010            ERR       062            000010000310000030200      2  0
         ERR      090            A2        0*1
0000103a ERR      016            S6        S1*1SB         000016323040000030144      i  0d
         ERR      000            A1        A4+A4
0000104a ERR      001            ERR       000            000001000000000000000
         ERR      000            ERR       000
0000105a 12160015,A4, A4                                  114450700065014002200      (  A $
         S6       22000
0000106a S5       14126337                                041517246200611343046      COS 1.16
         A4       A5*A6          A4        A6+A6
0000107a ERR      000            ERR       000            000000000000000000000
         ERR      000            ERR       000
```

1800

13.4 DUMPJOB - CREATE $DUMP

DUMPJOB creates the local dataset $DUMP, if it does not already exist. When the DUMPJOB statement is encountered, $DUMP receives an image of the memory assigned to the job (the Job Table Area (JTA) and user field). Placing the DUMPJOB statement after a system verb, excluding the * (comment) and EXIT statements, causes a dump of the Control Statement Processor (CSP). A DUMPJOB to an execute-only dataset is rejected.

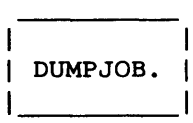
If the $DUMP dataset already exists, it is overwritten each time a DUMPJOB control statement is processed. If $DUMP is permanent and the job does not have write permission, DUMPJOB aborts. If $DUMP is permanent and the job has write permission, the dataset is overwritten.

If the DUMPJOB/DUMP sequence fails because of such situations as destroyed system-managed Dataset Parameter Areas (DSPs), rewind $DUMP before the job step for which the dump is to be written and save it with unique access. DUMPJOB writes to $DUMP, and job termination automatically adjusts $DUMP. $DUMP can then be inspected in a separate job. This procedure applies only to situations in which the user overwrites certain system tables without the detection of the system.

DUMPJOB creates $DUMP as an unblocked dataset so it can be used by DUMP, FLODUMP, DEBUG, and DDA.† DUMPJOB is a system verb and cannot be continued to subsequent statements.

There are no parameters.

Format:



13.5 ITEMIZE - INSPECT LIBRARY DATASETS

ITEMIZE prints a formatted report of the contents of a dataset generated by compilers, loaders, assemblers, UPDATE, or BUILD. For additional information about the contents of an UPDATE PL, use AUDPL. Refer to the UPDATE Reference Manual, CRI publication SR-0013.

† Deferred implementation

A header containing the jobname, ITEMIZE version number, date, time, and page number appears at the top of every page. The line shown below appears following the header on page 1. The line gives the local dataset name of the dataset being processed.

ITEMIZE OF *dn*

ITEMIZE normally produces file-level output. For binary library datasets, however, it produces a more detailed record-level output. The following subsections describe both levels of output.

Restrictions:

- An UPDATE PL is recognized only if it is the only item in a dataset. A PL created by the UPDATE utility consists of many files. The last file of the dataset must be a PL directory. If NF is not specified on the control statement, ITEMIZE prints information only for the first file, although it has examined the last file. Again, the dataset must contain only a PL.

- ITEMIZE does not operate on a tape dataset.

Format:

```
ITEMIZE,DN=dn,L=odn,NREW,NF=n,T,BL,E,B,X.
```

DN=*dn* Local dataset name of the dataset to be listed. The default is $OBL.

L=*odn* Local dataset name where listing is written. If L is omitted or is specified alone, $OUT is used.

NREW No rewind. Specifies the dataset is not rewound. If NREW is omitted, the dataset to be listed is rewound before and after ITEMIZE is executed.

NF=*n* Number of files within a dataset to be listed. If NF is used alone, the contents of all files within the dataset are listed. If NF=*n*, the contents of *n* files within the dataset are listed. The default is NF=1.

T Truncation. Specifying this parameter truncates lines on the listing dataset to 80 characters. Optional parameter; however, specifying this parameter precludes specifying the E, B, and X parameters.

BL Burstable listing. When this parameter is specified, each dataset heading starts at the top of a page. The default is a compact listing in which a page eject occurs only when the current page is nearly full.

E Entry points. Specifying E causes all entry points to be included in the listing. Use for binary library datasets only.

B Blocks. Specifying B causes all entry points, code, and common block information to be included in the listing. Use for binary library datasets only. B overrides E.

X Externals. Specifying X causes all entry points, code, common block, and external information to be included in the listing. X overrides B.

13.5.1 FILE-LEVEL OUTPUT

ITEMIZE prints one line for each file examined (up to the maximum specified by the NF parameter or the default of 1). A second header line appears on each page and contains the column headings shown in figure 13-1.

Figure 13-1 is an example of ITEMIZE operating on a program library (PL). The control statement used to generate the listing was ITEMIZE,BL,NF. The list following figure 13-1 describes the contents of each column.

```
                Itemize 1.16        11/10/86        09:37:55    Page   1

            Itemize of COSPL

    File    Records     Type        Length     Check      Part       Date

      1         60       PL           245      7314       7314     10/15/86

  File count limit (NF parameter) reached.

                       Sum=           245      7314       7314

      ******************************************************
      * Dataset is UPDATE PL -- use AUDPL for more details *
      ******************************************************
```

1743

Figure 13-1. Sample Listing of ITEMIZE for a Program Library

Heading	Description
FILE	Sequence number of the file within the dataset
RECORDS	Number of records within the file
TYPE	Type of information contained within the file. If the file is a member of a PL, the column contains PL. Other values that may appear in this column are ABS, REL, DAT, and ???. ABS and REL indicate absolute and relocatable program modules, respectively. DAT indicates data, and ??? is used for otherwise unrecognized files.
LENGTH	Length of the file in words
CHECK	Checksum of the data within the file
PART	Same as CHECK for file-level output
DATE	Date of the PL from its directory; blank if other types of datasets.

13.5.2 OUTPUT FOR BINARY LIBRARY DATASETS

A binary library is a collection of binary records recognized by the existence of a Program Description Table (PDT). For binary library datasets, ITEMIZE operates record-by-record rather than file-by-file. The second header line for binary library datasets contains the column headings.

Figure 13-2 is an example of ITEMIZE operating on a binary library dataset. The list following figure 13-2 describes the contents of each column. The control statement used to generate the listing was ITEMIZE,BL,NF,X. If the control statement had been ITEMIZE,BL,NF., lines with no entry in the REC column would not have appeared.

```
TITEMA      Itemize 1.16        11/19/86        16:51:56    Page   1

        Itemize of TESTLIB                    File     1

    Rec     Name      Type       Length    Check     Part      Date


      1   DUMMY1      REL           75     6737     0234    11/19/86 16:51:55    CFT 1.
                                 Hardware requirements : CRAY-XMP  EMA
          * ENT *               DUMMY1
          * BLK *  DUMMY1             9
          * BLK *  #TB                4
          * BLK *  #CL                2
          * BLK *  #ST                0
          * BLK *  #RG                0
          * BLK *  #DA                3
          * EXT *               DUMMY2              DUMMY3

      2   DUMMY2      REL           70     1230     0274    11/19/86 16:51:55    CFT 1.
                                 Hardware requirements : CRAY-XMP  EMA
          * ENT *               DUMMY2
          * BLK *  DUMMY2             8
          * BLK *  #TB                4
          * BLK *  #CL                1
          * BLK *  #ST                0
          * BLK *  #RG                0
          * BLK *  #DA                3
          * EXT *               DUMMY3

      3   DUMMY3      REL           63     2431     0241    11/19/86 16:51:55    CFT 1.
                                 Hardware requirements : CRAY-XMP  EMA
          * ENT *               DUMMY3
          * BLK *  DUMMY3             7
          * BLK *  #TB                4
          * BLK *  #CL                0
          * BLK *  #ST                0
          * BLK *  #RG                0
          * BLK *  #DA                3

      1   * EOF *                  208     0531     0026

TITEMA      Itemize 1.16        11/19/86        16:51:56    Page   2

        Itemize of TESTLIB                    File     2

    Rec     Name      Type       Length    Check     Part      Date


      1   * DIR *     D01            31     2564     2564

Dir entry:DUMMY1      REL        No. of blocks    :     5
                                 No. of entries   :     1
                                 No. of externals :     2
          * ENT *               DUMMY1
          * BLK *  #TB
          * BLK *  #CL
          * BLK *  #ST
          * BLK *  #RG
          * BLK *  #DA
          * EXT *               DUMMY2              DUMMY3

Dir entry:DUMMY2      REL        No. of blocks    :     5
                                 No. of entries   :     1
                                 No. of externals :     1
          * ENT *               DUMMY2
          * BLK *  #TB
          * BLK *  #CL
          * BLK *  #ST
          * BLK *  #RG
          * BLK *  #DA
          * EXT *               DUMMY3

Dir entry:DUMMY3      REL        No. of blocks    :     5
                                 No. of entries   :     1
                                 No. of externals :     0
          * ENT *               DUMMY3
          * BLK *  #TB
          * BLK *  #CL
          * BLK *  #ST
          * BLK *  #RG
          * BLK *  #DA

      2   * EOF *                   31     2564     2564
          * EOD *    Sum=           239     0664     0173
/EOF
```

1015

Figure 13-2. Sample Listing of ITEMIZE for a Binary Library Dataset with X and NF Parameters

Heading	Description
REC	Sequence number of the record within the file
NAME	Name of the program from the PDT
TYPE	ABS or REL, which indicate absolute and relocatable program modules, respectively
LENGTH	Length of the record in words
CHECK	Checksums
PART	Checksums
DATE	Date of compilation from the PDT

One line containing the data previously listed is generated for each record. If you specify any of the E, B, or X options on the control statement, several additional lines can be printed. The information in these lines is labeled separately:

- When you specify E, B, or X, the comment field of the PDT is printed on a separate line. The hardware required for the module to execute correctly is listed on a separate line. In addition, the entry point names are printed with five names per line.
- When you specify B or X, a separate line is printed for each block containing its name and length.
- When you specify X, the externals referenced by the program are printed with five external names per line.

A binary library dataset contains a second directory file containing one record. If E, B, or X is specified on the control statement, a line is printed specifying the directory ID and length. In addition, entries, blocks, and externals are printed as described previously for program records.

13.6 PRINT - WRITE VALUE OF EXPRESSION TO LOGFILE

PRINT writes the value of an expression on the logfile. The value of the expression is written in three different formats: as a decimal integer, as a 22-digit octal value, and as an ASCII string. PRINT is a system verb.

Format:

```
 ____________________
|                    |
| PRINT(expression)  |
|____________________|
```

expression
Any JCL expression (refer to section 16). The maximum length is 8 characters. This parameter is required.

Logfile format:

UT060 *decimal octal ASCII*

UT060	Message code indicating the origin is a PRINT statement
decimal	A 16-digit decimal representation of the evaluated expression
octal	A 22-digit octal representation of the evaluated expression
ASCII	An 8-character ASCII representation of the evaluated expression

13.7 SYSREF - GENERATE GLOBAL CROSS-REFERENCE LISTING

SYSREF generates a global cross-reference listing for a group of CAL or APML programs. The number of CAL or APML programs that can be included in such a group is limited by the amount of Cray computer system memory allocated to a user.

SYSREF reads special binary symbol tables written by CAL or APML and produces a single cross-reference listing for the program modules represented in the tables. When the X parameter appears on a CAL or APML statement, a record is written for each program unit assembled. The records are written to a dataset specified by the X parameter ($XRF by default or if X appears alone). Each record has a header containing the name of the program unit. The rest of the record consists of cross-reference information for every global symbol used in that program.

Format:

```
SYSREF,X=xdn,L=ldn.
```

X=*xdn* Name of dataset whose first file (normally the only file) contains one or more symbol records written by CAL or APML. The default is $XRF.

L=*ldn* Name of output dataset. The default is $OUT.

13.7.1 USE OF SYSREF

SYSREF is usually used to process symbol records written by CAL and/or APML earlier in the same job. To do so, add X parameters to each CAL or APML control statement and follow them with a SYSREF control statement:

```
CAL,X.
APML,X.
CAL,X.
SYSREF,L=XROUT.
```

$XRF is used as the default in all cases.

To process symbol records written in an earlier job, the following sequence is used.

The first job:

```
CAL,X.
APML,X.
SAVE,DN=$XRF,ID=XX.
```

The second job:

```
ACCESS,DN=$XRF,ID=XX.
SYSREF,L=XROUT.
```

To add more symbol records before invoking SYSREF, use:

```
ACCESS,DN=$XRF,ID=XX,UQ.
SKIPR,DN=$XRF,NR.
CAL,X.
SYSREF.
```

The previous format has the same effect as if the CAL step had been done before the SAVE step.

13.7.2 GLOBAL CROSS-REFERENCE LISTING FORMAT

The global cross-reference listing contains only global symbols. A symbol is global if it is any one of the following:

- Named in an ENTRY or EXTERNAL statement
- Defined before an IDENT statement and after any preceding END statement
- Defined within a system text such as $SYSTXT
- Defined within a section of source code bracketed by TEXT and ENDTEXT pseudo instructions

The order of the symbols in the global cross-reference listing is lexicographic, based first on the symbol name and then (within each symbol name) on the module name. An exception to the order is made for symbol names beginning with N@, S@, or W@. These symbol names are sorted as if @ is the most significant (leftmost) character and the N, S, or W is the least significant character. The listing displays the symbol name correctly. The effect is a grouping of all the N@, S@, and W@ symbols that refer to the same field in a table.

The global cross-reference listing consists of 13 columns:

Column	Heading	Contents
1	Value	The symbol's value
2	Symbol	The symbol's name
3	Origin	The IDENT of the system text in which the symbol is defined; or the label of the TEXT block in which the symbol is defined; or *GLOBAL*, if the symbol is defined outside any program unit; or blank.
4	Module	The IDENT of the module within or before which the symbol is defined or referenced
5-11	References	A list of the lines on which the symbol is defined or referenced

The symbol's name, value, and references appear in a format similar to that of a CAL or APML listing. The page number in each reference is a local page number that starts at 1 for each module. In a CAL or APML listing, this is the page number that appears in parentheses to the right of the second title line on each page.

CREATING AN EXECUTABLE PROGRAM 14

The COS Relocatable Loader is a utility program that executes within the user field. It is used for loading and linking, in memory, relocatable modules from datasets on mass storage.

The relocatable loader is called with the LDR control statement when you need to load a program in relocatable format. Absolute load modules can also be loaded. The design of the COS loader tables and relocatable loader allows program modules to be loaded, relocated, and linked to externals in a single pass over the dataset being loaded. This minimizes the time spent in loading activities on the Cray computer system. The loader allows the immediate execution or the creation of an absolute binary image of the object module on a specified dataset.

The relocatable loader can also generate a partially relocated module. This module, referred to as a relocatable overlay, is described later in this section.

Most applications that require more than 4 Mwords of Central Memory cannot be loaded by LDR. LDR messages LD064, LD065, LD066, or all three are issued if problems are found. These applications may have to be loaded with the Segment Loader (for details refer to the Segment Loader (SEGLDR) Reference Manual, CRI publication SR-0066). You can use the LD2 control statement to change from the LDR control statement to SEGLDR.

14.1 LDR CONTROL STATEMENT

The LDR control statement begins execution of the loader. Parameters of the control statement determine the functions to be performed by the loader.

Format:

```
LDR,DN=dn,LIB=ldn,NOLIB=ldn,LLD,AB=adn,MAP=op,SID[='string'],T=tra,

   NX,DEB=l,C=com,OVL=dir,CNS,NA,USA,L=ldn,SET=val,E=n,I=sdir,

   NOECHO,SECURE,GRANT=sc1:sc2:...:scn,BC=bc,PAD=pad,NORED,

   STK[=initial size[:increment]],MM[=initial size[:increment]],
                          AFTER
   MMEPS=epsilon,MMLOC=BEFORE.
```

DN=*dn* Dataset containing modules to be loaded. The default is $BLD. Modules are loaded in $BLD unless you specify a block name with a Fortran PROGRAM, SUBROUTINE, BLOCK DATA, or FUNCTION statement. Loading continues until an end-of-file (EOF) is reached. Duplicate blocks are skipped and an informative message is issued.

Multiple files from the same dataset can be loaded by specifying the dataset name multiple times separated by colons. You can indicate a maximum of eight files.

Datasets specified by the DN parameter are closed at the end of the load process. Closing a dataset has the effect of rewinding the dataset and releasing I/O tables and buffers.

Modules to be loaded can be relocatable or absolute; but do not mix the two types of modules. Neither LD2 nor SEGLDR supports absolute modules.

For example, the following statement causes the loading of all modules in the first file of datasets LOAD1, then LOAD2, and then $BLD:

```
DN=LOAD1:LOAD2:$BLD
```

Normally the dataset is rewound before loading; however, consecutive occurrences of a dataset name inhibit subsequent rewind operations. Therefore, the following statement causes the loading of all modules in the first two files of dataset LOAD3:

```
DN=LOAD3:LOAD3
```

The DN parameter takes on a special quality when OVL is specified. Then only one *dn* can be specified. The dataset named is the initial LOAD file used by the overlay loader. (Refer to the description of overlay loading later in this section for more information.)

LIB=*ldn* The LIB parameter names the dataset from which unsatisfied externals are loaded. A maximum of eight datasets can be named, with the dataset names separated by colons.

Any default libraries are automatically included in the library list unless the NOLIB parameter is specified. LDR accesses the default libraries from the COS System Directory (SDR) if they are not local to the job; no ACCESS statement is required.

Datasets specified by the LIB parameter are closed at the end of the load process. Closing a dataset has the effect of rewinding the dataset and releasing I/O tables and buffers.

NOTE

Use the BUILD utility to generate object datasets specified by the LIB parameter to prevent unnecessary overhead in the loader.

The libraries cannot be tape datasets.

NOLIB=*ldn* The NOLIB parameter value names the specific default library to be excluded from the load. Selecting NOLIB with no value specifies the exclusion of all default system libraries. If NOLIB is not specified, any default libraries that a site has are automatically included in the library list, along with any libraries specified on the LIB parameter.

LLD Specifying the LLD parameter will retain any libraries included in the load as local datasets when the load is completed. These local datasets remain open. Datasets automatically accessed are not released when the load is completed. If the LLD parameter is not specified, the loader closes all libraries and releases automatically accessed datasets at load completion.

LD2 uses the LLD parameter to inhibit the release of datasets generated to assist in using SEGLDR. Use the LLD parameter to keep these datasets for subsequent conversion to SEGLDR.

AB=*adn* Absolute binary object module generation. This parameter causes an absolute binary object module to be written to the named dataset after the load process is complete. Selecting AB does not imply no execution (NX). Unless NX is also selected, the loaded program begins execution after the binary is generated. Specifying AB without *adn* causes the module to be written on a dataset named $ABD, the default dataset. The dataset is not rewound before or after the file is written.

If the AB parameter is omitted, no binary generation occurs.

If OVL is specified on the LDR statement, the OVLDN directive replaces AB; any value specified for AB is ignored in overlay mode. Overlay loading is described later in this section.

MAP=*op* Map control. The MAP parameter causes the loader to produce a map of the loaded program on the specified dataset. MAP can take any of the following values:

ON Produces a block list and an entry list including all cross-references to each entry

OFF No map is produced. Default is MAP=OFF.

FULL Same as MAP=ON

PART Produces a block list only. Equivalent to MAP with no value specified.

SID[='*string*']
Debug routine loading. The SID parameter indicates the system debugging routines are to be loaded with the code. These routines comprise an additional binary dataset loaded after all DN specified datasets and before any libraries.

The '*string*', if given, is passed to SID for evaluation as a control statement. The verb and initial separator are not required. For example, SID='I=IN,ECH=ELIST.' is a proper string specification; the period is a required terminator. Refer to the Symbolic Interactive Debugger (SID) User's Guide for a complete description of SID parameters. If only SID is specified, all keyed default SID control statement parameter values are used.

T=*tra* Transfer name. Lets you specify an entry name for the loader to transfer control at completion of the load.

The T parameter also specifies the entry included in absolute binary object modules. The entry name is 8 characters maximum. If no T parameter is specified, the loader begins object program execution at either the entry specified by the first encountered START pseudo from a CAL routine or at the entry of the first main program in Fortran compiled routines. If no START entries are encountered, a warning message is issued and the first entry of the first relocatable or absolute module is used.

NOTE

When the SID parameter is used, the load transfer is to the system debugger, and the T parameter is ignored. If T is coded, however, a warning message is issued to the user logfile.

NX No execution. This parameter inhibits execution of the loaded program.

DEB=*l* Job Communication Block (JCB) length. The default length is 200_8 words. Specifying DEB without a value changes the JCB length to 3000_8.

C=*com* Compressed load. Allows control of the starting locations of modules and common blocks. An align bit is set for each relocatable module and common block that contains an ALIGN pseudo-op. Refer to the CAL Assembler Version 1 Reference Manual, CRI publication SR-0000, or the Fortran (CFT) Reference Manual, CRI publication SR-0009.

C can take on any of the following values:

ON Forces the loading of each module and common block to begin at the next available location after the previous module or common block, ignoring the align bit. Equivalent to C with no value specified.

PART Forces the loading of each module and common block with the align bit set to an instruction buffer boundary.† If the align bit is not set, that module or common block is loaded at the next available location after the previous module or common block. C=PART is the default.

OFF Forces the loading of every module to an instruction buffer boundary.† Common blocks are forced to instruction buffer boundaries only if the align bit is set.

OVL=*dir* Overlay load. Indicates an overlay load sequence is specified on *dir*. Overlay loading is explained in detail later in this section. If the OVL keyword is specified without a value, the loader examines the next file of $IN for an overlay load sequence. The default is no overlay load. Selecting OVL implies NX (no execution).

CNS Crack next control statement record image. Allows the loader to pass parameters to the loaded program for analysis and to use the parameters during execution of the loaded program. The control statement that is cracked follows the LDR control statement and is not available for processing by the Control Statement Processor (CSP) after processing by the loaded program.

NOTE

When the SID parameter is specified, the CNS parameter is ignored and a warning message is written to the user logfile if CNS is present. SID prompts for the control statement for the code being debugged.

NA No abort. If this parameter is omitted, a caution or higher level loader error causes the job to abort.

USA Unsatisfied external abort. When USA is specified, the loader aborts at the end if it finds one or more unsatisfied externals. If called for, a load map listing all unsatisfied externals is produced.

† Instruction buffer sizes are 40_8 words for the CRAY X-MP computer system and 20_8 words for all CRAY-1 S models.

L=*ldn* Listing output. Lets you specify the name of the dataset, *ldn*, to receive the map output. If L=0, all output is suppressed. The default dataset is $OUT.

SET=*val* Memory initialization. Variables, named and blank common blocks, and storage areas defined by DIMENSION statements are set to 0, -1, or an out-of-range floating-point value during loading. The default is SET=ZERO.

SET=ZERO Memory is set to binary zeros.

SET=ONES Memory is set to -1 (all bits set).

SET=INDEF Memory is set to a value that causes an out-of-range error if the word is referenced as a floating-point operand. The ones complement of each memory address is placed in the low-order 24 bits of the respective word to aid in reading register and memory dumps. An example, in octal, of the value loaded into memory word 13216 is: 0605050037740177764561.

E=*n* List error messages. Indicates the highest level of loader-produced error messages to be suppressed. One of five levels of severity can be suppressed, where *n* is the highest level to be suppressed. The default for this parameter is E=1.

Level	Type	Description
1	COMMENT	Error does not hinder program execution
2	NOTE	Error probably hinders program execution
3	CAUTION	Job aborts when load process completes unless NA is selected; program might not execute properly.
4	WARNING	Job aborts when load process completes unless NX is selected; program execution is not possible.
5	FATAL	Job aborts immediately. FATAL messages are never suppressed.

Example:

E=2 suppresses COMMENT and NOTE messages and allows CAUTION and WARNING messages to appear.

I=*sdir* Selective load. Modules from other datasets can be loaded according to a set of directives. *sdir* indicates the dataset containing the directives. If the I keyword is specified without a value, the directives are taken from the next file of $IN. The selective load directives INCLUDE and EXCLUDE are described later in this section.

NOECHO Suppresses writing the current control statement to the user logfile (that is, the control statement that invoked the actual loading into memory is not written to the logfile).

SECURE Defines each dataset created during this job step to be *secure* (that is, to be released during job advancement unless specifically overridden with a F$DSD operating system request).

GRANT=$sc_1:sc_2:...:sc_n$
Grants the privileges defined as parameters if this module is loaded from the System Directory (SDR). (These privileges are merged with the user's only for the duration of the job step.) The following parameters are defined if security is enabled. They are operative only if the dataset is executed from the SDR.

Parameter	Privilege
SCACES	Accesses user-saved dataset without passwords
SCDIAG	Allows F$DIAG request for on-line diagnostics
SCDTIM	Allows use of PDM "set time of PDSDUMP" function
SCDUMP	Allows F$DJA requests anytime
SCENTR	Allows ENTER option on ACCESS
SCERCH	Allows F$DRIVER requests
SCERQM	Allows SDT queue manipulation
SCISPT	Allows F$TRB requests for Integrated Support Processor (ISP) testing
SCLUSR	Loads user dataset
SCMLOG	Lets you send messages to another user's logfile
SCNVOK	Invokes job class structure
SCPDAD	Allows access of system catalog dataset
SCPRIV	Allows special system requests
SCQDXT	Allows LINK DXT requests
SCQSDT	Allows dequeuing and queuing of SDT requests
SCRDSC	Allows reading of Dataset Catalog (DSC) page

Parameter	Privilege
SCRESIDE	Allows declaring a dataset to be on-line, preventing it from being migrated or retired
SCRESON	Allows you to request that a dataset reside on-line
SCSPOL	Allows SAVE/ACCESS/DELETE/LOAD/DUMP spooled dataset
SCSYSJ	Allows a job to be a system job
SCSYSPRG	Allows system programmer functions such as F$PROF and F$CMEM
SCTPBLP	Allows bypass label processing for magnetic tape
SCUPDD	Allows access user dataset for PDSDUMP
SCURID	Allows use of reserved ID in interjob communication
SCWNSC	Allows you to randomly seek a direct access dataset beyond an area to which you have written

BC=*bc* Blank common. *bc* specifies the number of words to be added to the size of blank common when the program is loaded for execution. The default is 0.

PAD=*pad* Pad. *pad* specifies the number of words of unused space to be made available in the job when the program is loaded for execution. After the program is loaded with its requested extra space, the job is placed in user-managed field-length reduction mode for the duration of the job step. The default is 0.

NORED No field-length reduction. Before the program is loaded, the job is placed in user-managed field-length reduction mode for the duration of the job step.

STK[=*initial size*[:*increment*]]
Initializes for stack processing. STK is a run-time memory management parameter.

initial size indicates the initial size of a stack in number of words. An installation parameter defines the default value. If the *initial size* value is less than 128, LDR substitutes the default value.

increment specifies the size of additional segments to a stack (in number of words) if a stack overflows. An installation parameter defines the default value. A value of 0 indicates that overflow is prohibited.

MM[=*initial size*[:*increment*]]

Initializes for managed memory processing. The values assigned to MM specify the number of words available to the heap manager.

initial size indicates the number of words initially available to the heap manager. An installation parameter defines the default value. The loader changes the specified value if the heap is not allowed to grow and if there is no room for heap and stack overhead.

increment specifies the minimum size, in words, of a request to the operating system for additional memory if the heap overflows. Zero means that the size of the heap is fixed. An increment other than zero cannot be specified if the heap is before blank common. An installation parameter defines the default value. If the BEFORE value is specified for the MMLOC parameter, the default value is 0.

MMEPS=*epsilon*

epsilon is the smallest block that can be left on the list of available space in the heap. If a request for additional memory from the heap is made by the run-time routines, and the request leaves a memory fragment of less than *epsilon* words, the additional words are given to the request. The value must be at least 2. An installation parameter defines the default value.

MMLOC=AFTER
MMLOC=BEFORE

Specifies the location of the heap. AFTER specifies that the heap is located after blank common; default. If the heap is located before blank common, BEFORE is specified.

14.2 LD2 CONTROL STATEMENT

LD2 is a utility program that converts programs using LDR to programs using SEGLDR. You will find LD2 useful with applications that require more than 4 Mwords of Central Memory and with applications (especially those that use overlays) being migrated to the Cray operating system UNICOS, where LDR is not available. Normally, LD2 builds auxiliary CAL source files, SEGLDR directive files, and COS job control language (JCL) files, and automatically invokes the COS JCL file. This has the effect of retaining LDR control statements, directive files, and overlay methods while actually using SEGLDR.

Use the LLD parameter on the LD2 control statement to capture the CAL source and SEGLDR directive intermediate files. This simplifies creation

of a hybrid job using SEGLDR more directly. This can save time for jobs with large, complex overlay programs by removing the need to execute LD2 each time.

You might detect some differences between LDR and LD2. In general, the LD2 control statement produces the same result as the LDR control statement; however, LD2 does not convert programs that rely on loading one or more common blocks at a specific address. Unlike LDR, LD2 does not allocate the first common block encountered in the first module loaded at 200(8).

Before using LD2, remove from your program any names that conflict with the following LD2 output names:

- Program names of the form Z0000001 through Z9999999
- OVERLAY. If you have a private copy of the library module OVERLAY, you must remove it before using LD2.
- Dataset names $ILDR, $DLDR, $XLDR, and $A00001 through $A99999

LD2 does not fully support multiple file object datasets, although it does handle many straightforward cases.

The LD2 control statement has the same parameters as the LDR control statement with some exceptions. These are: LD2 does not support the SID parameter, but it does support the VIEW and CMD parameters. Refer to the LDR parameter descriptions for other minor differences. The LD2 VIEW and CMD parameters function as follows:

VIEW=*level* Echoes the LDR directives being converted to the SEGLDR listing dataset. This produces a sometimes large listing detailing the joint actions of LD2 and SEGLDR. The *level* specifies the degree of detail desired in the report. By default, *level* is 1. The range is from 1 to 255; currently, however, useful values are 1 and those greater than 8. Larger values produce more detailed information. A *level* greater than 8 writes to the listing dataset a possibly voluminous report of each dataset examined.

CMD=*string* CMD lets you specify one or more SEGLDR directives. The *string* is passed to SEGLDR as its first directive. This permits you to obtain SEGLDR specific load maps, for example. Occasionally, you must use the CMD parameter to supply the proper SLT count for SEGLDR (for example, you must code "CMD='SLT=number'", where number can be obtained from a SEGLDR error message in an earlier, failed run). For more information on the SLT directive and on the CMD parameter, refer to the Segment Loader (SEGLDR) Reference Manual, CRI publication SR-0066.

14.3 LOAD ORDER FOR LDR AND LD2

Loaders (LDR and LD2) load in the following order:

- Routines you supply are loaded first. These routines usually come from $BLD. You may specify other datasets with the DN parameter in both LDR and LD2 or use the BIN directive with SEGLDR.

- If any externals remain unresolved, the libraries are scanned, in this order:

  - Libraries you supply with the LIB parameter are scanned first, in the order in which you gave them.

  - The default libraries are scanned next, in this order:

    $IOLIB
    $UTLIB
    $SYSLIB
    $ARLIB
    $FTLIB
    $PSCLIB
    $SCILIB
    $SLLIB

Loaders load only one module with a given external name. LDR and LD2 use different methods to select the module that is loaded if you have duplicate external names, either within your own libraries or in the complete set of libraries. LDR loads the first module encountered after the external call becomes known; this is not always the first module in the library scan order. SEGLDR (and hence LD2) loads the first module in the library scan order and generates a warning message for duplicates that are ignored.

Example:

Suppose your main program references FOO, which you expect to satisfy from library USER2. FOO, in turn references BAR. You specified LIB=USER1:USER2 on the LDR or LD2 control statement and there are instances of BAR in both USER1 and USER2. For LDR, USER2 is used, because at the time USER1 was scanned, FOO had not yet been encountered and the need for BAR was not known. For LD2 (SEGLDR), USER1 is used.

NOTE

Because of differences between loaders, and because Cray Research reserves the right to modify, reorganize, and reorder standard libraries, you are cautioned against developing applications that depend on how loaders process duplicate entry points.

14.4 LOAD MAP

Each time the loader is called, you have the option of requesting a listing, called a load map. This load map describes where is module is loaded and what entry points and external symbols are used for loading.

Specify the contents of the map or the dataset to receive the map by setting the LDR control statement parameters. The MAP parameter of the LDR control statement lets you specify the contents of the map requested. The Segment Loader (SEGLDR) Reference Manual, CRI publication SR-0066, decribes the load maps produced by LD2 and SEGLDR. MAP=ON or MAP=FULL produces a block list and an entry list. The block list gives the names, beginning addresses, and lengths of the program and subroutines loaded on this loader call; the entry list includes all cross-references to each entry. MAP=PART supplies the block map only.

When a load map is requested, it is printed even if fatal errors abort the load. In this case, the map contains only those modules loaded up to the point where the fatal load error occurred.

Figure 14-1 shows the load map generated by the following LDR statement:

```
LDR,DN=$BLD:LOAD2,LIB=MYLIB,MAP=FULL,MM=16000:4000,STK=1280:128
```

The block list consists of items 1 through 16; the entry list includes items 17 through 23.

```
TOTAL: 1321                              LDRMAP          LDR X.14  84251                09/24/84            11:54:11          PAGE 1
                     RELOCATABLE LOAD

LOAD TRANSFER IS TO (6)   AT  ( (7) )

DATASET    BLOCK      ADDRESS     LENGTH     DATE         OS REV      PROCSSR      VER.          Comment
           *SYSTEM         0      200
$BLD       LDRMAP        200      1321       09/24/84     COS X.14    CFT 1.13     09/21/84
LOAD2      ABCDEFGH     1521      36         09/24/84     COS X.14    CFT 1.13     09/21/84
MYLIB      X1           1557      41         09/24/84     COS X.14    CFT 1.13     09/21/84
           X2           1620      41         09/24/84     COS X.14    CFT 1.13     09/21/84
           .
           .
           .
MODULE NAME     ENTRIES      ENTRY VALUE REF. MODULE        ABSOLUTE     REFERENCES

     LDRMAP          LDRMAP         717a
     ABCDEFGH        ABCDEFGH       1525a     LDRMAP    1425a
     X1              X1             1570a     ABCDEFGH  1531a
     .
     .
     .
                     NLERP%         3234a
     $FDP            $FDP           4640       $WUT     10603b
     $WFD            $WFI           5451a     LDRMAP    1410a        1416d

*** MANAGED MEMORY STATISTICS ***
INITIAL STACK SIZE:                 1280(10)  2400(8) WORDS
STACK INCREMENT SIZE:               128(10),  200(8) WORDS
INITIAL MANAGED MEMORY SIZE:        16000(10),        37200(8) WORDS
MANAGED MEMORY INCREMENT SIZE:  4000(10),          7640(8) WORDS
MANAGED MEMORY EPSILON:                               2(10),
2(8) WORDS
BASE ADDRESS OF MANAGED MEMORY/STACK:   15566(10),             36316(8)
WORDS
MANAGED MEMORY/STACK LOCATION: AFTER BLANK COMMON

*** LOAD IMAGE STATISTICS ***
ABSOLUTE BINARY LENGTH: 31438(10),        75316(8) WORDS
PROGRAM IMAGE: FWA =            200(8), LWA =      75516(8)
```

1009

Figure 14-1. Load Map Example

1. Job name from the JOB control statement
2. Loader level and the assembly date of the loader
3. Date and time of loader execution
4. Page number
5. Load type; either relocatable, absolute, or overlay.
6. Entry name to which initial transfer is given
7. Entry address where initial transfer is made
8. Name of load or library dataset containing modules to be loaded

(9) Names of blocks loaded from the named dataset. These are common blocks (identified by the slashes around their names, for example, /LABEL/) are names of program blocks.

*SYSTEM is always the first block listed in a relocatable load. It consists of the first 200_8 words of the user field, which is reserved for the Job Communication Block (JCB). For an absolute load, *SYSTEM is not allocated. The CAL user must set the origin to 200_8 with an ORG pseudo instruction to allow space for the JCB. If this is not done, the job aborts.

Blank common, indicated as //, is allocated last and appears at the end of the list (if it has been defined).

(10) Starting address of the block, in octal

(11) Word length of the block, in octal

(12) Date the object module was generated

(13) Operating system revision date at the time the object module was generated

(14) Name and revision level of the processor that generated the object module

(15) Revision date of the processor that generated the object module

(16) Comment (if any) from CAL COMMENT pseudo included in the load module

(17) Name of program block referenced

(18) Entry points in the program block

(19) Word address, parcel address, or value of each entry point

(20) Module name of reference to each entry point

(21) Absolute parcel addresses of references to each entry point. Eight references are listed per line; some entry points have no references.

(22) Managed memory statistics. The numbers in parentheses indicate the base: decimal (10) and octal (8).

(23) Actual length of the binary; the minimum amount of memory required to load the program. FWA is the first word address of the load image. LWA is the last word address of the load image. The numbers in parentheses indicate the base: decimal (10) and octal (8).

14.5 SELECTIVE LOAD

If the I keyword is present on the LDR control statement, one or more INCLUDE and/or EXCLUDE directives are examined in the specified dataset.

Formats:

```
INCLUDE,SDN=sdn,FN=fn,MOD=md1:md2:...:md50.
```

```
EXCLUDE,SDN=sdn,FN=fn,MOD=md1:md2:...:md50.
```

SDN=*sdn* Name of the dataset containing modules to be selectively loaded. If SDN is specified without a value, the first dataset specified on the DN parameter of the LDR statement is the default. If the SDN parameter is omitted, an error message results and the directive is skipped; the load does not abort. The SDN and FN parameters must refer to the same dataset.

FN=*fn* File number of the specified dataset; a number from 0 to 7. *fn* refers to the file by its numerical position in SDN or in the DN parameter of the LDR statement.

For example, if DN=D1:D1:D2, the first file of D1 has an *fn* of 0, and the second file of D1 has an *fn* value of 1. If FN is specified without a value, the default is 0. If FN is omitted, all of *sdn* is searched for the correct module; a message is issued for a complete *sdn* search. The SDN and FN parameters must refer to the same dataset.

To load a module from the first file of D1, the directive can include the parameter FN=0; however, if FN is specified without a value, the default is to load a module from the first file.

MOD=*md* Module name or entry point to a module to be included or excluded from the load. Up to 50 modules can be specified; the modules must be separated by colons. If the MOD parameter is omitted, an error message results, and the directive is skipped.

Example: Given the LDR statement

```
LDR,DN=D1:D1:D2,...,I.
```

A directive to load a module from the second file of dataset D1 includes the following directive in the next file of $IN:

```
INCLUDE,SDN=D1,FN=1,MOD=... .
```

Selective load messages are never suppressed.

14.6 OVERLAYS

Very large programs may not fit in the available user memory space or might not use large portions of memory while other parts of the program are in execution. For such programs, the COS relocatable loader includes the ability to define and generate *overlays*, separating modules that the user creates and then calling and executing as necessary.

Two types of overlays are available:

- *Type 1 overlays* are generated by using the directives ROOT, POVL, and SOVL. Two levels of overlays in addition to the root overlay are allowed with calls to a maximum of 999 primary overlays and up to 999 secondary overlays per primary overlay.
- *Type 2 overlays* are generated by using the directive OVLL. Ten levels of overlays in addition to the root overlay are allowed with calls to a maximum of 63 overlays per branch.

The overlay structure, rules for overlay generation, and overlay calls for both types are described in this subsection. The control statements used to generate the overlay and the directives common to both types of overlays are described first. Specific rules for generating Type 1 and Type 2 overlays are described separately in the following subsections.

Overlay generation consists of a load operation in which the loader performs relocatable loading and writes the resulting binary image to disk. One named absolute binary record is written per root and each overlay.

If the LDR control statement has the parameter OVL=*dir*, the loader finds the overlay generation directives on the named dataset, *dir*. If no dataset is given, the loader reads overlay generation directives from $IN.

Format:

```
LDR,...,OVL=dir,... .
```

dir Name of the dataset containing the overlay generation directives

14.6.1 OVERLAY DIRECTIVES

An overlay directive consists of a keyword and a parameter. A blank, comma, or open parenthesis must separate the keyword from the parameter. A period, closed parenthesis, or two consecutive blanks serve as the terminator. A caret at the end of the directive line indicates that the next line is a continuation of the current directive. The caret cannot be preceded by a blank; it must immediately follow the last character of the line.

14.6.1.1 FILE directive

The FILE directive indicates the dataset, *dn*, containing the routines to be loaded. This directive's function is similar to that of the DN parameter on the LDR control statement. It is generally the first directive on the directives dataset but appears at any time and as often as necessary thereafter. If no FILE directive appears, the loading proceeds from the dataset specified on the DN parameter of the LDR control statement. If that too has been omitted, loading initially occurs from $BLD. This directive is common to both overlay types.

Format:

```
 _____________
|             |
| FILE,dn.    |
|_____________|
```

dn Name of the dataset containing the routines to be loaded

14.6.1.2 OVLDN directive

The function of the OVLDN directive is similar to that of the AB parameter on the LDR control statement. This directive names the dataset, *dn*, on which overlays are written. The *dn* parameter must be present. If no OVLDN directive is present, the default overlay binary dataset ($OBD) is assigned. All overlays generated following an OVLDN directive reside as separate binary records on dataset *dn*. OVLDN directives appear as often as desired on the LDR control statement. The LD2 control statement accepts only the first OVLDN directive that you specify; it silently ignores any others. This directive is common to both overlay types.

Format:

```
 ____________
|            |
| OVLDN,dn.  |
|____________|
```

dn Name of the dataset on which overlays are written

14.6.1.3 SBCA directive

The SBCA directive sets the blank common starting address to the specified address. This directive lets you place blank common after all load modules in the current overlay structure. The address specified must be larger than any address used in the overlay structure. This directive must appear before any overlay generation directive, such as ROOT or OVLL. The SBCA directive is mutually exclusive.

Format:

```
 ________________
|                |
| SBCA,address.  |
|________________|
```

address Address assigned to blank common, in octal. For LD2, even though the octal address is ignored, it must be present. SEGLDR can automatically determine blank common location.

14.6.1.4 SMMA directive

The SMMA directive sets the managed memory (heap) address to the specified address. This directive lets you place managed memory after all load modules in the current overlay structure. The address specified must be larger than any address used in the overlay structure. This directive must appear before any overlay generation directive, such as ROOT or OVLL. The SMMA directive is mutually exclusive.

Format:

```
 ________________
|                |
| SMMA,address.  |
|________________|
```

address Octal address assigned to the heap

14.6.2 TYPE 1 OVERLAY STRUCTURE

Each Type 1 overlay is identified by a pair of decimal numbers, each from 0 to 999. There must be one and only one root overlay; its level numbers are (0,0). This root remains in memory throughout program execution. Primary overlays all have level numbers (*n*,0), where *n* is in the range 1 through 999.

Primary overlays are called at various times by the root and are loaded at the same address immediately following the root. A secondary overlay is associated with a specific primary overlay, and it can be called only by the corresponding primary overlay. The secondary level numbers are (*n*,*m*), where *n* is the primary level, and *m* is in the range 1 through 999. All secondary overlays associated with a given primary (that is, the same *n*) are loaded at the same address immediately following that primary.

Only the root, one primary overlay, and one secondary overlay can be in memory at one time.

Figure 14-2 is a diagram of a sample Type 1 overlay loading. The primary and secondary overlays are shown in time sequence. The sequence of generation does not imply that the routines are loaded into memory in the same sequence or that they remain in memory for a set period of time when they are executed.

All external references must be directed toward an overlay nearer to the root. For example, overlay (1,0) can contain references to the root (0,0) but not to overlay (1,1). Overlay (1,1) can contain references to both (1,0) and (0,0).

LDR places named common before the routine that first references it. All named common references must be directed toward a lower-level routine. The lowest level routine with a named common block must contain data statements for that block.

For example, in figure 14-2,

MAIN	Can reference named common A only
SUB1 and SUB2	Can reference named common A and B only
TEST	Can reference named common A, B, and C

LDR allocates blank common immediately after the first overlay where it is declared. If blank common is declared in the root overlay (0,0), it is allocated at the highest address of the root overlay and is accessible to all overlays. If blank common is first declared in primary overlay (1,0) and not declared in the root (0,0), it is accessible only to the (1,*x*) overlays. Allocation and placement of blank common is also manipulated by the user through the SBCA directive.

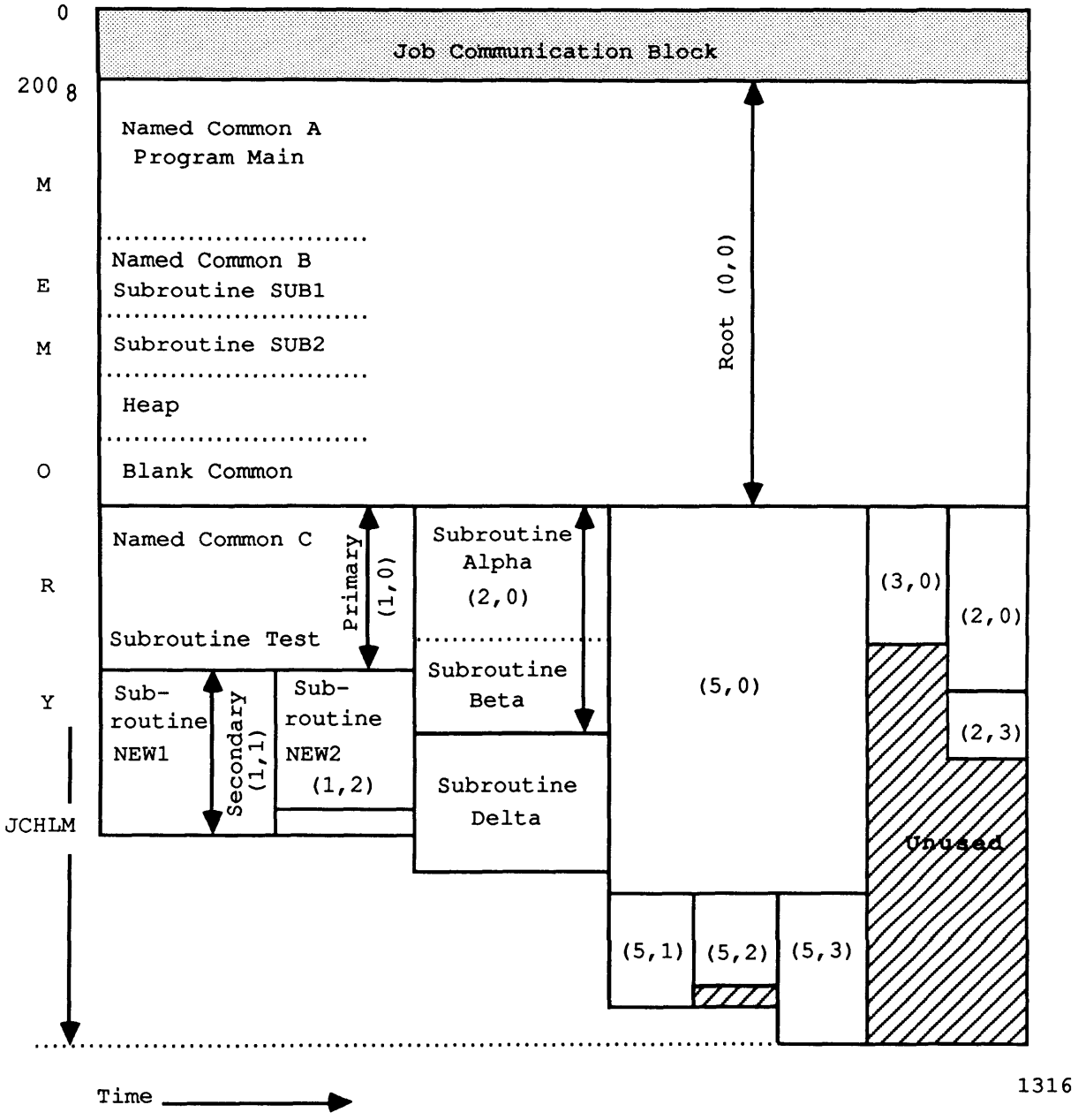


Figure 14-2. Type 1 Overlay Loading Example

JCHLM is set to the highest address of the root overlay before loading. If a subsequent overlay module requires additional memory, JCHLM is reset to the highest address of that module.

14.6.3 TYPE 1 OVERLAY GENERATION DIRECTIVES

The overlay generation directives define the structure of the overlay. Included in this class are the ROOT, POVL, and SOVL directives.

14.6.3.1 ROOT directive

This directive defines programs, subroutines, and entry points, that comprise the load from *dn*. For programs written in CAL, list each entry referenced. Fortran programs need the program name only. All members for this directive reside on the same dataset, *dn*, as defined by the FILE directive.

Format:

```
| ROOT,member1,member2,...membern. |
```

member$_i$ Module names for inclusion in the root

14.6.3.2 POVL directive

This directive causes relocatable loading of the named blocks to the primary overlay with the name *plevel*:000. The size of the root determines the base location. All members for this directive reside on either the dataset specified in the last FILE directive or, if none was named there, the dataset specified on the LDR statement. The first member in the list is the one that receives control when the overlay is loaded. For routines written in CAL, the first entry point of the first routine receives control.

Format:

```
| POVL,plevel,member1,member2,...membern. |
```

plevel Primary overlay name; between 1 and 999.

member$_i$ Module names for inclusion in the primary overlay number *plevel*

14.6.3.3 SOVL directive

This directive causes relocatable loading of the named blocks to the secondary overlay with the name *plevel*:*slevel*. The length of POVL (*plevel*:000) determines the base location. All members for this directive reside on the same dataset, *dn*. The first member in the list is the one that receives control when the overlay is loaded. For routines written in CAL, the first entry point of the first routine receives control.

Format:

```
SOVL,slevel,member1,member2,...,membern.
```

slevel Secondary overlay name; between 1 and 999.

member$_i$ Module names for inclusion in the secondary overlay number *slevel*

14.6.3.4 Generation directive example

In the following example:

- DSET1 contains routines THETA, TEST, GAMMA, SUB1, MAIN, and SUB2.
- DSET2 contains routines NEW2, ALPHA, OVER, NEW1, DELTA, EPSILON, SIGMA, and BETA.

Format of the control statement that initializes overlay generation follows:

```
LDR,...,OVL=OVLIN,....
```

Dataset OVLIN contains the following directives:

Directive	Description
FILE,DSET1.	The loader selectively loads from dataset DSET1.
OVLDN,LEV00.	The following overlay modules are written to the dataset LEV00.

Directive	Description
ROOT,MAIN,SUB1 ,SUB2.	The absolute binary of MAIN,SUB1,SUB2 is written as the first record on dataset LEV00.
POVL,1,TEST.	The binary of TEST is named 001:000 and is binary record 2 on dataset LEV00.
FILE,DSET2.	The loader selectively loads from dataset DSET2.
SOVL,1,NEW1.	The binary of NEW1 is named 001:001 and is binary record 3 on dataset LEV00.
OVLDN,LEV12.	The subsequent overlay modules are written to the dataset LEV12.
SOVL,2,NEW2.	The binary of NEW2 is named 001:002 and is binary record 1 on dataset LEV12.
POVL,2,ALPHA,BETA.	The binary of ALPHA,BETA is named 002:000 and is record 2 on dataset LEV12.
.	
.	
.	
EOF	End of overlay load sequence

14.6.4 TYPE 1 OVERLAY GENERATION RULES

The Type 1 overlay generation rules are as follows:

1. Overlay members are loaded from datasets named in FILE directives. Members are searched for in the most recently mentioned dataset only. In the absence of a FILE directive, members are loaded from the dataset specified on the LDR control statement. If that is also omitted, loading initially occurs from $BLD. Currently, the relocatable modules of all members for any overlay level must reside on the same file.

2. The overlays are generated in the order of the directives.

3. There must be only one root.

4. Level hierarchy must be maintained. The root overlay must be generated first; hence, the ROOT directives appear first. Following the root generation, a primary overlay (POVL) is generated. No limitation is placed on which primary overlay number (*plevel*) is generated; however, all secondary overlays (SOVL) associated with the *plevel* must follow. The secondary overlay *slevels* can be generated in any order following their respective primary level.

5. An EOF in the directives file ends the input of overlay directives; hence, overlay generation.

6. Any directive other than FILE, OVLDN, SBCA, ROOT, POVL, or SOVL causes a fatal error.

7. The list of members can be continued to another line by using a caret (^) immediately following the last nonblank character at the end of the directive line. The ^ does not replace a separator and must not appear within a member name.

8. Any number of lines can be used to name the members of an overlay.

9. A secondary overlay can only be called by the corresponding primary overlay.

14.6.5 TYPE 1 OVERLAY EXECUTION

A control statement call of the dataset containing the ROOT overlay initiates its loading and execution. If no OVLDN directives are used before generating the ROOT, the dataset $OBD contains the ROOT overlay.

The following sequence executes the root overlay after generation:

```
LDR,...,OVL=dir,... .
$OBD.
```

During overlay generation, the members are loaded from the FILE dataset in the order they appear on the dataset, regardless of their order of appearance in the members list. The entry for POVL and SOVL overlays is defined by the first member listed on the generation directive. Control is transferred to this address after loading by the $OVERLAY routine during program execution. The ROOT entry is named using the T parameter on the LDR control statement.

You call for the loading of overlays from within the program, and the method by which they are called depends on the program language in use (Fortran or CAL). OVERLAY is a subroutine of the root overlay and is loaded into memory with the root.

14.6.5.1 Fortran language call

A Fortran program calls for the loading of overlays as follows:

```
CALL OVERLAY(dn,level1,level2,r)
```

dn Dataset name or unit number that contains the file. Must be a character constant, integer variable, or an array element containing Hollerith data of not more than 7 characters.

$level_1$ Primary level number of the overlay

$level_2$ Secondary level number of the overlay

r An optional recall parameter. If you wish to reexecute an overlay without reloading it, enter 6LRECALL. If the overlay is not currently loaded, it is loaded.

14.6.5.2 CAL language call

A sample call sequence from a CAL program follows:

```
|Location|Result    |Operand
|        |          |
|        |EXT       |OVERLAY
|        |.         |.
|        |.         |.
|        |.         |.
|        |CALL      |OVERLAY,(OVLDN,PLEV,SLEV)
|        |.         |.
|        |.         |.
|        |.         |.
|OVLDN   |CON       |A'LEV12'L
|PLEV    |CON       |2
|SLEV    |CON       |0
```

OVLDN is the address of the dataset name, PLEV is the address of the primary level, and SLEV is the address of the secondary level. If recall is desired, the address of the literal 'RECALL' is transmitted as the fourth argument.

Example:

```
|Location |Result    |Operand                  |Comment
|  1      |10        |20                       |35
|         |          |                         |
|         |CALL      |OVERLAY,(OVLDN,PLEV,SLEV,RECL)
|         | .        | .                       |
|         | .        | .                       |
|         | .        | .                       |
|RECL     |CON       |'RECALL'L                |
```

For both Fortran and CAL language calls, during execution of the ROOT(0,0) program MAIN, the statement

```
CALL OVERLAY(5LLEV12,2,0)
```

or the preceding CAL sample call causes OVERLAY to search dataset LEV12 for the absolute binary named 002:000. OVERLAY positions the dataset LEV12 to the location of the absolute binary named 002:000 using information supplied by the loader, loads the overlay, and transfers control to the first member specified on the POVL or SOVL directive. After execution of the overlay, control returns to the statement in MAIN immediately following the CALL statement. Following the load, dataset LEV12 is positioned immediately after the EOR for the overlay (2,0). If overlay (2,0) is not on dataset LEV12, a fatal error results.

Placing a call for a secondary overlay for which the corresponding primary overlay is not already loaded causes a fatal error. A fatal error also results if the primary and secondary overlays are not both on the named *ovldn*.

14.6.6 TYPE 2 OVERLAY STRUCTURE

Figure 14-3 shows the tree structure of the Type 2 overlay. There is only one root overlay, and its level number is 0. The root overlay remains in memory during program execution and calls only level 1 overlays. Only one branch is in memory at any time. Overlay (2,1) under overlay (1,1) is different from the (2,1) under (1,5). Moreover, overlay (2,1) under overlay (1,1) can be called only by overlay (1,1).

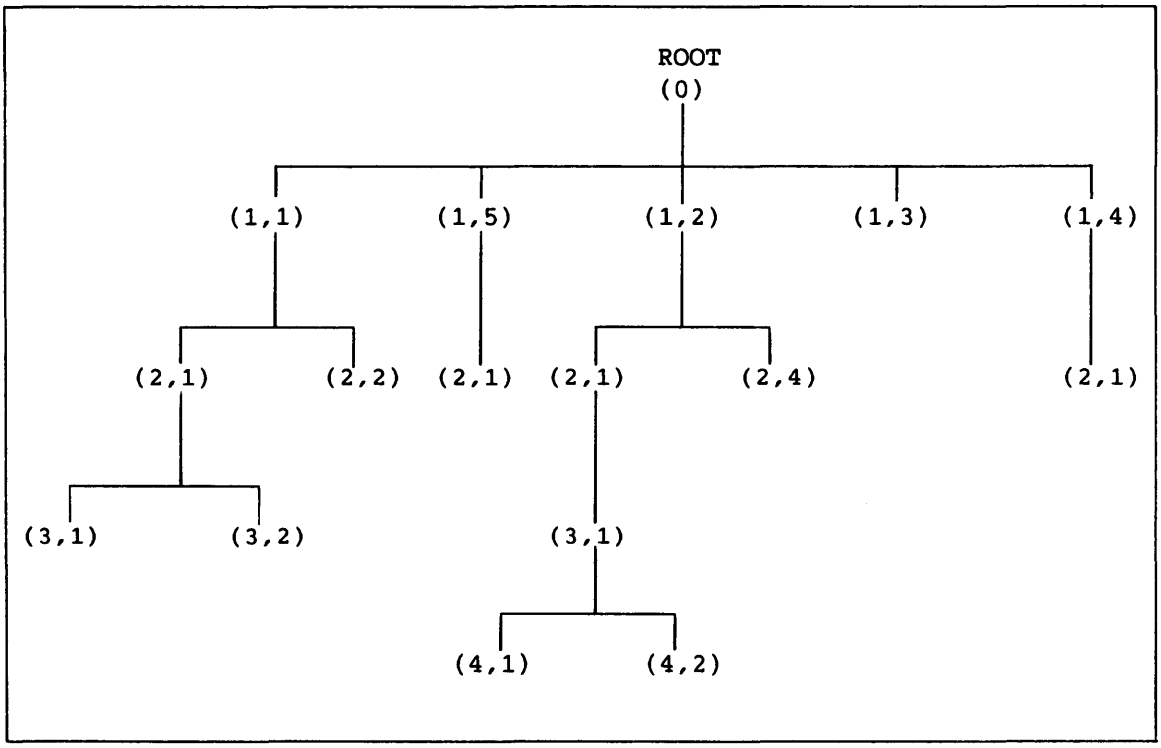


Figure 14-3. Type 2 Overlay Tree Example

Figure 14-4 shows a sample Type 2 overlay loading diagram. The overlays are shown in time sequence. The sequence of generation does not imply that the programs are loaded into memory in the same sequence or that they remain in memory for a set period of time when they are executed.

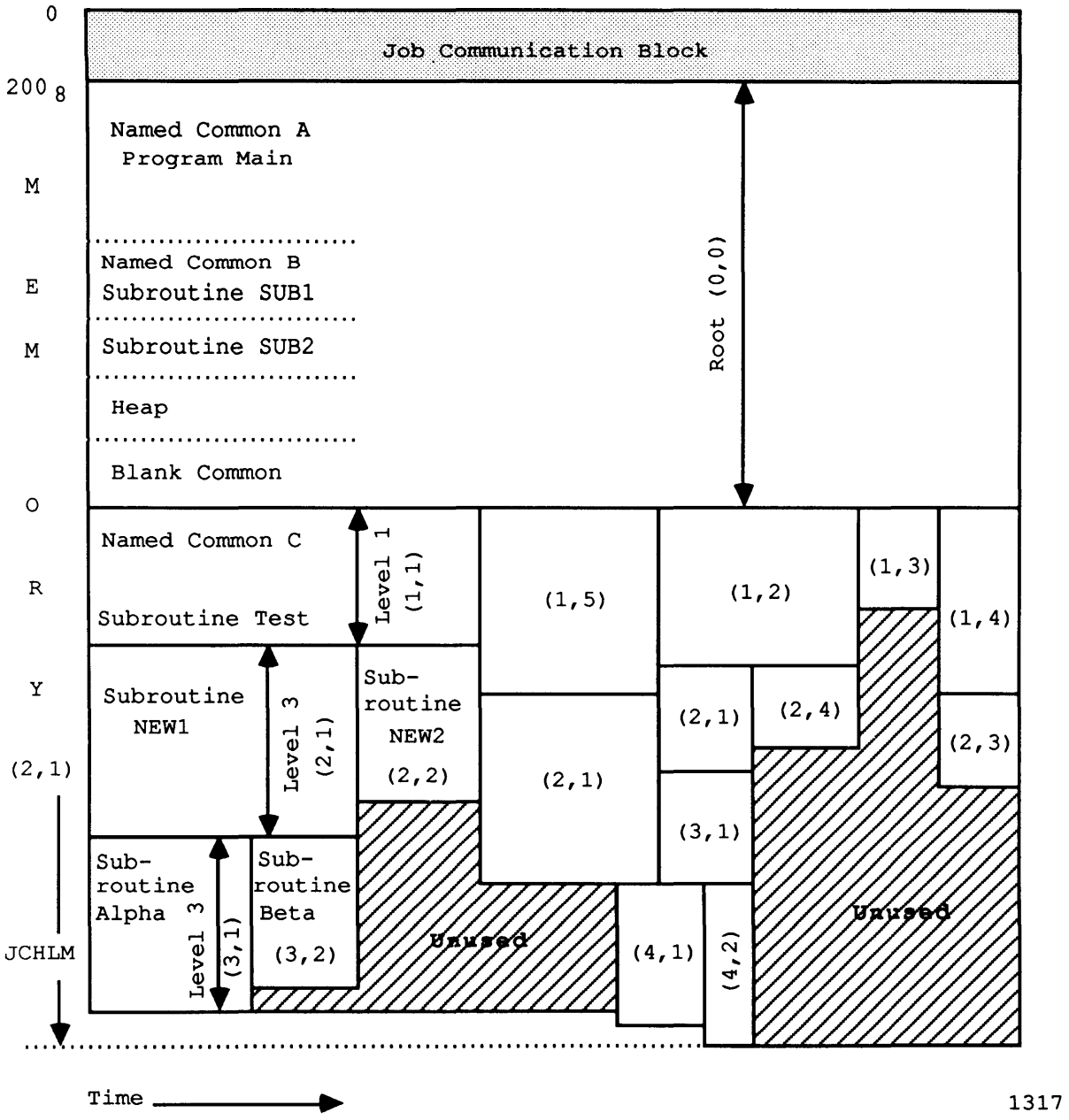


Figure 14-4. Type 2 Overlay Loading Example

Level 1 overlays are called at various times by the root overlay. Each call loads the named overlay at the same address, immediately following the location of the root. The first level overlay must be called by the root. Each upper-level overlay must be called by the associated overlay at the adjacent lower level. A hierarchy exists among overlay levels; an upper-level overlay is subordinate to the proximate lower-level overlay. An upper-level overlay associated with overlay (2,1) might be (3,2), (3,3), or (3,4). Upper-level overlays appear on the page *after* the lower-level overlays.

An overlay can call into memory any overlay in the next higher level; it cannot call an overlay more than one level above it in the hierarchy. For example, overlay (2,1) can call (3,1) through (3,63), but it cannot call (4,1). Each call for an overlay loads the named overlay at the same address location immediately following the location of the calling overlay. Only the root and one overlay at each level can be in memory concurrently.

All external references must be directed toward an overlay nearer the root overlay. Overlay (1,1) can contain references to the root overlay but not to overlay (1,2) or overlay (2,1). The (2,1) overlay can reference externals in both the (1,1) overlay and the root overlay.

The loader places named common blocks before the routine that first references it. All named common references must be directed toward a lower-level routine (toward the root overlay). If blank common is declared in the root overlay, it is allocated at the highest address of the root and is accessible to all overlays. If blank common is declared first in a level 1 overlay, for example, and is not declared in the root overlay, it is accessible only to level 1 and upper-level overlays.

JCHLM is set to the highest address of the root overlay before loading. If a subsequent overlay module requires additional memory, JCHLM is reset to the highest address of that module.

14.6.7 TYPE 2 OVERLAY GENERATION DIRECTIVE

The Type 2 overlay directive defines the structure of the overlay within the directive format.

14.6.7.1 OVLL directive

This directive causes relocatable loading of the named blocks of an overlay. The size of the lower-level overlays in the group determines the base location. All members for this directive reside on the same dataset, *dn*, specified by the FILE directive. The first member in the list is the one that receives control when the overlay is loaded. For programs written in CAL, the first entry point of the first routine receives control.

Format:

```
OVLL,level,number,member1,member2,...membern.
```

level Either a level number of the overlay (1 to 10), or the root phase (0). If the root phase is being generated, *number* must be omitted.

number Number of the overlay (1 to 63) within the level

member$_i$ Module names for inclusion in the individual overlays

14.6.7.2 Generation directive example

In the following example:

- DSET1 contains routines THETA, TEST, GAMMA, SUB1, MAIN, and SUB2.
- DSET2 contains routines NEW2, ALPHA, OVER, NEW1, DELTA, EPSILON, SIGMA, and BETA.

Format of the control statement that initializes overlay generation:

```
LDR,...,OVL=OVLIN,...
```

Dataset OVLIN contains the following directives:

Directive	Description
FILE,DSET1.	The loader selectively loads from dataset DSET1.
OVLDN,LEV00.	The following overlay modules are written to the dataset LEV00.
OVLL,0,MAIN,SUB1, SUB2.	The absolute binary of MAIN,SUB1,SUB2 is the first record on dataset LEV00.
OVLL,1,1,TEST.	The binary of TEST is binary record 2 on dataset LEV00.
FILE,DSET2.	The loader selectively loads from dataset DSET2.
OVLL,2,1,NEW1.	The binary of NEW1 is binary record 3 on dataset LEV00.

Directive	Description
OVLDN,LEV12.	The subsequent overlay modules are written to the dataset LEV12.
OVLL,3,1,ALPHA.	The binary of ALPHA is binary record 2 on dataset LEV12.
OVLL,3,2,BETA.	The binary of BETA is binary record 3 on dataset LEV12.
OVLL,2,2,NEW2.	The binary of NEW2 is binary record 1 on dataset LEV12.
.	
.	
.	
EOF	End of overlay load sequence

14.6.8 TYPE 2 OVERLAY GENERATION RULES

The Type 2 overlay generation rules are as follows:

1. Overlay members are loaded from datasets named in FILE directives. Members are searched for in the most recently mentioned dataset only. In the absence of a FILE directive, members are loaded from the dataset specified on the LDR control statement. If that is also omitted, loading initially occurs from $BLD.

2. The overlays are generated in the order of the directives.

3. There must be one and only one root per dataset.

4. Level hierarchy must be maintained. The root overlay must be generated first. Following the root generation, a first level overlay is generated. No limitation is placed on which overlay number is generated; however, all overlays associated with that first level overlay must follow. The overlays can be generated in any order; the same restrictions apply for all levels of overlays (1 to 10).

5. The first level overlay must be called by the root. An overlay can call into memory any overlay in the next higher level; however, an overlay cannot call an overlay that is more than one level above it in the hierarchy.

6. An EOF ends the input of overlay directives.

7. Any directive other than FILE, OVLDN, SBCA, or OVLL causes a fatal error.

8. The list of members can be continued to another line by using a caret immediately following the last character at the end of the directive line (that is, no blanks). The caret does not replace a separator and must not appear within a member name.

9. Any number of lines can name the members of an overlay.

14.6.9 TYPE 2 OVERLAY EXECUTION

A control statement call of the dataset containing the root overlay initiates the root overlay's loading and execution. If no OVLDN directives are used before generating the root, the dataset $OBD contains the root overlay. All overlays reside on the datasets specified on the overlay directives. The entry for higher-level overlays is defined by the first member listed on the generation directive. Control is transferred to this address after loading by the $OVERLAY routine during program execution. The root entry is named using the T parameter on the LDR control statement.

The following sequence executes the root overlay after generation:

```
LDR,...,OVL=dir,... .
$OBD.
```

When the program is to be executed, the root overlay is brought into memory as a result of a control statement call in the job deck. Thereafter, additional overlays are called into memory by the executing program. Overlay loading allows any overlay to call for the loading of an adjacent upper-level overlay.

You call for the loading of Type 2 overlays from within the program, and the method by which they are called depends on the program language in use (Fortran or CAL). OVERLAY is a subroutine of the root overlay and is loaded into memory with the root.

14.6.9.1 Fortran language call

A Fortran program calls for the loading of Type 2 overlays as follows:

```
CALL OVERLAY(dn,level,number,r)
```

dn Name of the dataset in which this overlay resides. The name must be left-adjusted and zero-filled.

level Level number of the overlay

number Number of the overlay within the level

r Optional recall parameter. If you wish to reexecute an overlay without reloading it, enter 6LRECALL. If not currently loaded, it is loaded.

14.6.9.2 CAL language call

A sample call sequence from a CAL program is as follows:

```
|Location |Result    |Operand
|         |          |
|         |EXT       |OVERLAY
|         |.         |.
|         |.         |.
|         |.         |.
|         |CALL      |OVERLAY,(OVLDN,PLEV,SLEV)
|         |.         |.
|         |.         |.
|         |.         |.
|OVLDN    |CON       |A'LEV12'L
|PLEV     |CON       |2
|SLEV     |CON       |0
```

OVLDN is the address of the dataset name, PLEV is the address of the primary level, and SLEV is the address of the secondary level. If recall is desired, the address of the literal 'RECALL' is transmitted as the fourth argument.

Example:

```
|Location |Result    |Operand          |Comment
|1        |10        |20               |35
|         |          |                 |
|         |CALL      |OVERLAY,(OVLDN,PLEV,SLEV,RECL)
|         | .        | .               |
|         | .        | .               |
|         | .        | .               |
|RECL     |CON       |'RECALL'L        |
```

For both Fortran and CAL language calls, during execution of the ROOT program MAIN, the statement

```
CALL OVERLAY(5LLEV12,1,2)
```

or the preceding CAL sample call causes OVERLAY to search dataset LEV12 for the absolute binary named 2. OVERLAY positions the dataset LEV12 to the location of the absolute binary named 2 using information supplied by the loader, loads the overlay, and transfers control to the first member specified on the OVLL directive. After execution of the overlay, control returns to the statement in MAIN immediately following the CALL statement. Following the load, dataset LEV12 is positioned immediately after the EOR for the overlay 2. If overlay 2 is not on dataset LEV12, a fatal error results.

14.6.10 OVERLAY GENERATION LOG

When MAP is specified on the LDR control statement, a listing is obtained describing where each module is loaded and what entry points and external symbols are used for loading. This listing is an overlay load map and is similar to the map of a nonoverlay load. A log of the directives used follows the map of the last overlay generated. If overlay loading aborts, the directives are not listed.

BUILD UTILITY 15

BUILD is a utility program used for generating and maintaining library datasets. A *library dataset* contains a program file followed by a directory file. The program file is composed of loader tables for one or more absolute or relocatable program modules. The directory file contains an entry for each program. The entry contains the name of the program module, the relative location of the program module in the dataset, and block, entry, and external names. Library datasets primarily provide the loader with a means of rapidly locating and accessing program modules.

The BUILD program constructs a library from one or more input datasets named in the BUILD control statement. A library dataset created by one BUILD run can be used as input to a subsequent BUILD run. Through BUILD directives, you designate the program modules to be copied from the input datasets to the new library and their order in the library.

No directives or control statement parameters are needed for the most frequent application of BUILD, which is to add new binaries from $BLD to an existing library of binary programs, replacing the old binaries where necessary.

BUILD does not use tape datasets.

15.1 BUILD CONTROL STATEMENT

Keywords can be in any order.

Format:

```
BUILD,I=idn,L=ldn,OBL=odn,B=bdn,NBL=ndn,SORT,NODIR,REPLACE.
```

I=*idn* *idn* is the name of the data containing BUILD directives, if any. Directives can be included in the $IN dataset, or they can be submitted in a separate dataset. BUILD directives are discussed later in this section.

If the I parameter appears alone or is omitted, all directives are taken from the $IN dataset, starting at its

I=*idn* (continued) current position and stopping when an end-of-file (EOF) is read.

If I=*ddn*, all directives are taken from the specified dataset, *ddn*, stopping when an EOF is read.

If I=0, no directives are read. The most common condition is to merge the modules from *odn* (the OBL parameter dataset) with those from *bdn* (the B parameter dataset), replacing OBL modules with B modules whenever the names conflict, and to write the output to *ndn* (the NBL parameter dataset). The input dataset specified by the B parameter corresponds to the binary output from CAL and Fortran, also designated by B.

L=*ldn* Name of list output dataset. If the L keyword appears alone or is omitted, list output is written to $OUT. If L=*ldn*, list output is written to *ldn*. If L=0, no list output is written.

OBL=*odn* Name of the first input dataset, usually a previously created library dataset. If the OBL parameter is omitted or appears alone, the first dataset read is $OBL. If OBL=*odn*, the first dataset read is *odn*. If OBL=0, no old binary library exists; this is a creation run.

B=*bdn* Name of the second input dataset, whose modules are added to or replace the modules in the first dataset. If the B parameter appears alone or is omitted, the second dataset read is $BLD. If B=*bdn* is specified, the second dataset read is *bdn*, which is read to the first EOF. If B=0, no modules are being added; this run edits an old library.

NBL=*ndn* Name of the output dataset, usually a new library dataset. If the NODIR parameter is also present, *ndn* is not in library format. If the NBL parameter appears alone or is omitted, output is written to $NBL. If NBL=*ndn*, output is written to *ndn*. If NBL=0, no output is written.

SORT Specifies that all modules will be listed alphabetically according to their new names. The default is to list the modules in the order they are first read. SORT applies only to the list dataset and not to the output library.

NODIR Specifies that no directory is to be appended to the output dataset, resulting in an ordinary sequential dataset like $BLD. The default is to append the directory.

The dataset *ndn* specified by NBL is not rewound if NODIR is specified.

REPLACE Specifies that the output library is to contain modules in the same order as the old library. If REPLACE is omitted, the new library contains modules from the old library that are not replaced by modules from the input binary dataset. These are followed by modules from the input dataset, whether the modules from the input dataset are new or replace modules from the old library. The modules appear in the order encountered on the input dataset.

Build aborts if any of the following errors occur:

- A module specified explicitly in a COPY or OMIT directive is not in the current input dataset.
- A module specified explicitly in a COPY directive has already been selected for output.
- Improper syntax is used in the BUILD control statement or in the directive dataset.
- An unrecognized directive or control statement keyword is used.
- A dataset name or module name is too long or contains illegal characters.

15.2 PROGRAM MODULE NAMES

BUILD directives refer to program modules by their names (as given in the directory) or, if the directory is missing or is unrecognizable, by the names given in the program modules.

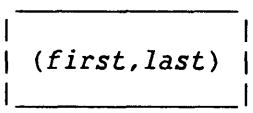
15.3 PROGRAM MODULE GROUPS

In the COPY and OMIT directives, program modules with names containing one or more identical groups of characters can be specified together. To accomplish this, variable parts of each name are replaced by one or more hyphens. For example, XYZ- represents all names beginning with XYZ, including XYZ itself. The extreme case is a name consisting of only a hyphen which represents all possible names.

In addition, up to eight asterisks can be used anywhere in a name as wild characters matching any character other than a blank. For example, GE* specifies a group of modules having 3-character names including GET and GEM but not GE or GEMS. GE*S would represent GEMS.

15.4 PROGRAM MODULE RANGES

To make it easy to copy large numbers of contiguous program modules, the COPY directive allows use of a range specifier instead of a single name or group specifier. The range specifier has the following general format:



This means skip to the first module specified and copy all modules from the first up to and including the last module specified.

15.5 FILE OUTPUT SEQUENCE

If the SORT parameter appears in the BUILD control statement, all modules are copied alphabetically according to their new names. In the absence of a SORT parameter, modules are written in the order they are originally read from the input datasets.

The order of the entries in the directory is always the same as the order of the modules themselves.

15.6 FILE SEARCHING CONSIDERATIONS

You do not need to know the order of modules in the input dataset unless two or more modules have the same name or a range is specified in a COPY directive.

If two or more modules with the same name are in the input datasets, the last of the modules read is the one that survives, unless you specifically omit that last module while its original dataset is the currently active input dataset.

The concept of *current position* in the input file is used to interpret range specifiers where the first name is omitted as in (,*last*) or (,). In such cases, the current position is defined to be either immediately after the last module copied or at the beginning of the dataset if no modules have yet been copied.

15.7 BUILD DIRECTIVES

BUILD is controlled through directives in a dataset defined by the I parameter on the BUILD control statement. A directive consists of a keyword and, if the keyword requires it, a list of dataset names or module names. When names are required, the keyword must be separated from the first name by a blank; subsequent names (if any) in the list are separated from each other by commas. Extra blanks are optional except within the keyword.

A line can contain more than one directive. Use periods or semicolons to separate directives on the same line from each other. You cannot continue a directive from one directive line to the next.

Examples of directives:

```
OMIT ENCODE,DECODE

COPY **CODE.
```

Examples of multiple directives on one line:

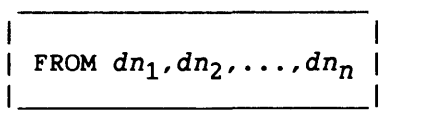
```
FROM OLDLIB; LIST; OMIT ENCODE,DECODE,XLATE

FROM $BLD. LIST.
```

15.7.1 FROM DIRECTIVE

A FROM directive names a single dataset, which is used as the input dataset for succeeding COPY, OMIT, and LIST directives, or it lists several datasets that (except for the last dataset in the list) are to be copied in their entirety to the output dataset ($NBL). The last dataset in the list is established as the current input dataset, just as if it were specified alone in the FROM directive. If no COPY or OMIT directive follows, the last dataset is also copied in its entirety to the output dataset.

An input dataset can be a library (with a directory) or an ordinary sequential dataset (such as $BLD). BUILD always determines whether a directory is present at the end of the dataset and attempts to use it if it is there. A library dataset is treated as sequential if its directory file is unrecognizable.

Format:



The following rule lets you copy several datasets with one FROM directive or omit COPY (which means copy all) when it would be the only directive (except for OMIT directives) in the range of a particular FROM directive:

> If any dataset named on a FROM directive is not acted on by any LIST or COPY directive, BUILD copies all of the modules belonging to that dataset. BUILD takes this action when it encounters the next FROM dataset name or the end of the directive file, whichever comes first.

If there are two input datasets to be read as soon as BUILD begins to execute (that is, if neither OBL=0 nor B=0 is specified), the modules from these two datasets are treated as if they belong to a single dataset as far as the OMIT, COPY, and LIST directives are concerned. If either of them is named in a FROM directive, however, it is treated as a separate dataset and OMIT, COPY, and LIST directives apply only to whichever is the current input dataset.

15.7.2 OMIT DIRECTIVE

The OMIT directive lets you specify that certain modules, that would otherwise be included in a group, be omitted from the group on subsequent copy operations. An OMIT affects modules on the current input dataset only; its effect ends when a FROM directive is encountered.

Format:

```
 _____________________________
|                             |
| OMIT fn1,fn2,...,fnn        |
|_____________________________|
```

Each fn_i can be one of the following:

- A single name, such as $AB@CDEF or CAB22, by which binary records can be explicitly prevented from being copied
- A group name, such as F$- or *AB**, by which binary records are prevented from being copied unless they are specified explicitly (that is, singly) in a COPY directive. (Refer to subsection 15.3, Program Module Groups, for a description of * and - usage.)

If an *fn* parameter specifies a module not in the input dataset or a group of modules having no representatives in the input dataset, a diagnostic message is included in the list output and BUILD aborts.

15.7.3 COPY DIRECTIVE

COPY directives cause BUILD to select the specified modules for copying from the current input dataset to the output dataset. You can specify single modules, groups of modules, or ranges of modules to be copied. If you specify a module not in the current input dataset, a diagnostic message is included in the list output and BUILD aborts.

Format:

```
| COPY fn1,fn2,...,fnn |
```

Each fn_i is either of the two forms valid in OMIT directives:

- A single module name by which modules are explicitly selected for copying even if they belong to a group named in a previous OMIT directive
- A group specifier by which all the modules in the group are selected for copying unless they are specified either explicitly or implicitly in a previous OMIT directive

In addition, two special forms are allowed for each fn_i in COPY directives:

- A form to rename a single module whose old name is specified explicitly; for example, OLDNAME=NEWNAME. (The name is changed both in the output directory and in the module's Program Description Table.)
- A form to copy an inclusive range, as in (FIRST,LAST), by which all the modules in the range are selected for copying unless they are specified either explicitly or implicitly in a previous OMIT directive.

These two forms are mutually exclusive. A module copied by being included in a range cannot simultaneously be renamed. Both forms cannot accept a hyphen or an asterisk specifying a group of modules.

Examples:

BUG=ROACH	Copies BUG, renaming it to ROACH
(LOKI,THOR)	Copies all modules from LOKI through THOR

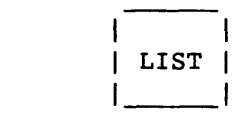
(THOTH,)	Copies all modules from THOTH to the end of the input dataset
(,ISIS)	Copies all modules from the current dataset position through ISIS
(,)	Copies all modules from the current dataset position to the end of the input dataset

The current dataset position is defined as the beginning of the input dataset if no modules have been selected for copying yet. Otherwise the position is the beginning of the record immediately after the last module that has been selected for copying.

15.7.4 LIST DIRECTIVE

The LIST directive tells BUILD to list the characteristics of the modules in the current input dataset. Its effect is immediate. (BUILD's standard list output describes the contents of the output dataset and is produced at the end of the run so as not to interfere with output triggered by LIST directives.)

Format:



15.8 EXAMPLES

The following are examples of various uses of the BUILD program:

- Creating a new library dataset, using as input whatever binary modules have been written out to $BLD (for example, by CAL or Fortran, or both)

  Control statements:

```
BUILD,OBL=0,I=0.
SAVE,DN=$NBL,PDN=MYLIB.
     .
     .
     .
```

- Adding one or more modules to an already existing library dataset, again taking the input from $BLD

  Control statements:

```
ACCESS,DN=$OBL,PDN=MYLIB.
BUILD,I=0.
SAVE,DN=$NBL,PDN=MYLIB.
     .
     .
     .
```

  Any modules whose names were already in the directory of MYLIB are replaced by the new binaries from $BLD in the new edition of MYLIB that is created by BUILD and saved by the SAVE control statement.

- Merging several libraries

  Control statements:

```
ACCESS,DN=LIBONE,PDN=HERLIB.
ACCESS,DN=LIBTWO,PDN=HISLIB.
ACCESS,DN=ANOTHER,PDN=ITSLIB.
ACCESS,DN=LASTONE,PDN=MYLIB.
BUILD,I,OBL=0,B=0.
SAVE,DN=$NBL,PDN=NEWLIB.
     .
     .
     .
```

  Directives:

```
FROM LIBTWO,ANOTHER,LIBONE,LASTONE
```

  The order of the dataset names in the FROM directives, not the order of the ACCESS control statements, determines the order of processing. If two datasets contain modules of the same name, the surviving module is the one in the dataset whose name occurs later in the FROM directive. (Any module could be renamed in order to prevent it from begin discarded before input from a succeeding dataset is begun. See the File Searching Considerations subsection for a description of the interaction with OMIT directives.)

- Deleting a program module from a library

  Control statements:

```
ACCESS,DN=$OBL,PDN=MYLIB.
BUILD,B=0.
SAVE,DN=$NBL,PDN=MYLIB.
     .
     .
     .
```

  Directive:

```
OMIT BADPROG
```

- Extracting a program module from a library for input to the system loader, using the local dataset name $BLD as the intermediate file

  Control statements:

```
ACCESS,DN=XXX,PDN=MYLIB.
BUILD,I,OBL=XXX,B=0,NBL=$BLD,NODIR.
     .
     .
     .
```

  Directive:

```
COPY RUNPROG
```

JOB CONTROL LANGUAGE STRUCTURES 16

This section discusses three aspects of Job Control Language (JCL) structures:

- Control statement logic structures
- JCL expressions
- Procedures

16.1 CONTROL STATEMENT LOGIC STRUCTURES

The COS JCL allows three fundamental logic structures:

- *Simple control statement sequences.* Control statements are processed one after another.
- *Conditional control statement blocks.* A sequence of control statements is processed only if the specified condition is met.
- *Iterative control statement blocks.* A sequence of control statements is processed repetitively until the specified condition is met.

Most computer algorithms can be expressed in terms of these structures or as combinations of them.

16.1.1 SIMPLE CONTROL STATEMENT SEQUENCES

A simple control statement sequence is a series of one or more of the control statements described in sections 6 through 15. The individual control statements are processed sequentially as described in section 3.

16.1.2 CONDITIONAL CONTROL STATEMENT BLOCKS

A conditional control statement block is a group of control statements that is processed only if a specified condition is met. The control statements ELSE, ELSEIF, ENDIF, EXITIF, and IF allow other control statements to be placed in a conditional block structure, as follows:

- IF defines the beginning of a conditional block.
- ENDIF defines the end of a conditional block.
- EXITIF defines a condition which causes an escape from a conditional block.
- ELSE defines an alternate condition.
- ELSEIF defines an alternate condition to test when the previous one tested is false.

ELSE, ELSEIF, and EXITIF sequences are optional.

16.1.2.1 ELSE - Defines alternate condition

The ELSE control statement defines an alternate condition. An IF statement, as well as any ELSEIF statements, must precede the ELSE control statement. If all conditions specified by the IF and ELSEIF statements that precede the ELSE in the conditional block test as false, the statements that follow the ELSE statement are executed.

Within a conditional block, only one ELSE sequence is permitted. The ELSE statement, if present, must follow any ELSEIF statement. ELSE is a system verb. (System verbs are defined in subsection 4.2, Control System Verbs.) There are no parameters.

Format:

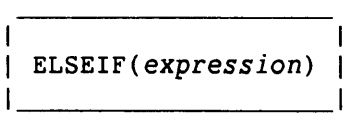
```
 _______
|       |
| ELSE. |
|_______|
```

16.1.2.2 ELSEIF - Defines alternate condition

The ELSEIF control statement defines an alternate condition to test if the previously tested condition was false. The sequence of statements following the ELSEIF statement is executed when the ELSEIF expression is true. All ELSEIF control statements must precede the optional ELSE control statement for a conditional block. An ELSEIF statement without a previously processed IF statement results in a job step abort. An unlimited number of ELSEIF sequences can be used in a conditional block.

ELSEIF is a system verb.

Format:



expression
A valid JCL expression (discussed later in this section). This parameter is required.

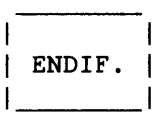
The block of control statements following an ELSEIF statement is processed under the following conditions:

- The expression for the IF statement is false.
- All preceding ELSEIF statement expressions are false.
- The ELSEIF expression is true.

16.1.2.3 ENDIF - End conditional block

The ENDIF control statement defines the end of a conditional block. ENDIF is a system verb. There are no parameters.

Format:

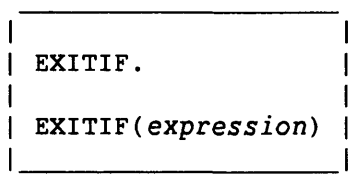


16.1.2.4 EXITIF - Exit from conditional block

The EXITIF control statement defines the conditions that must be met so that the remaining control statements in the conditional block are skipped. EXITIF is a means of skipping to the ENDIF statement without regard to EXIT statements. If the EXITIF expression is true, the remainder of the conditional block is skipped; if the expression is false, the control statements which follow the EXITIF statement are executed.

EXITIF may appear anywhere within a conditional block. An EXITIF statement that is not within a conditional block causes a job step abort. When conditional blocks are nested, the EXITIF control statement applies to the innermost conditional block that contains it. EXITIF is a system verb.

Formats:

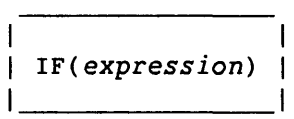


expression
A valid JCL expression (discussed later in this section). If *expression* is omitted, the remainder of the block is skipped unconditionally.

16.1.2.5 IF - Begin conditional block

The IF control statement defines the beginning of a conditional block. Each IF control statement must have a corresponding ENDIF control statement. IF is a system verb.

Format:



expression
A valid JCL expression (discussed later in this section). This parameter is required.

16.1.2.6 Conditional block structures

The conditional block is first scanned to verify the validity of the block's syntax. If any syntax errors exist, the block is skipped without being evaluated and a job step abort error occurs. Any EXIT control statements within the conditional block are ignored when a syntax error exists in that conditional block. This validation occurs when the control statement file, where it is contained, is invoked. (Validation occurs at job initiation if the control statement file is $CS; it can also occur at the time that a procedure is invoked, or when a CALL statement is encountered.)

Null sequences (for example, an ELSE statement immediately following an ELSEIF) are ignored without comment.

Conditional blocks can be constructed in the following ways:

- Basic conditional block
- Conditional block with ELSE
- Conditional block with ELSEIFs
- Conditional block with ELSEIFs and ELSE

Basic conditional block - The format of a basic conditional block begins with an IF statement and ends with an ENDIF statement. When the IF statement expression is true, the control statement sequence that follows is processed. When the expression is false, the control statement sequence is not processed. The following example shows the conditional block structure.

Examples:

```
ACCESS,DN=MYPROG,NA.
IF(PDMST.NE.1)
  FETCH,DN=MYPROG,MF=...
  SAVE,DN=MYPROG.
ENDIF.
```

In the preceding example, the JCL symbolic variable PDMST would be 1 if the ACCESS succeeded. The IF condition would be evaluated as false, and control statement execution would continue with the line following the ENDIF. If the ACCESS did not succeed, the IF condition would be evaluated as true, and the dataset would be accessed from the front-end system with FETCH and written to the Cray system with SAVE.

The following example shows a conditional block using EXITIF.

```
ACCESS,DN=MYPROG,NA.
IF(PDMST.NE.1)
    UPDATE(Q=MYPROG)
    CFT(I=$CPL,ON=A)
    LDR(AB=MYPROG,NX,USA)
    SAVE(DN=MYPROG,NA)
    EXITIF.
    EXIT.
*.
*.        ERROR GENERATING MYPROG
*.
   EXIT.
ENDIF.
MYPROG.
```

In this example, a conditional block is used to generate a dataset if that dataset is not found. EXITIF is used to skip the remaining statements in the conditional block if the dataset is generated successfully; otherwise, the job terminates.

Conditional block with ELSE - The second conditional block structure includes the ELSE control statement. The control statement sequence is processed if the expression on the IF statement is true. If the expression is not true, the sequence following the ELSE statement is processed. An example of a conditional block structure using the ELSE statement follows.

Example:

```
ACCESS,DN=INITJCL.
ACCESS,DN=PREPROG.
ACCESS,DN=PROG.
PREPROG.
IF(JSR.NE.0)
  CALL,DN=INITJCL.
  SWITCH,1=ON.
ELSE.
  SWITCH,1=OFF.
ENDIF.
PROG.
```

After PREPROG is executed, the conditional block determines if PREPROG has successfully executed (by its setting of JSR). The procedure INITJCL is executed and a sense switch is set if JSR is nonzero. The sense switch is cleared if PREPROG set JSR to zero.

Conditional block with ELSEIF - The third conditional block structure includes one or more ELSEIF statements. Each logical expression on the IF and ELSEIF statements is tested in sequence until a true condition is found; then the corresponding control statement sequence is processed.

A conditional block can contain any number of ELSEIF control statements. The block of control statements following an ELSEIF statement is processed under the following conditions:

- The expression for the IF statement is false.
- All preceding ELSEIF statement expressions are false.
- The ELSEIF expression is true.

Example:

```
IF(SYSID.EQ.'COS 1.12')
  ACCESS,DN=$FTLIB,ID=V112.
ELSEIF(SYSID.EQ.'COS 1.13')
  ACCESS,DN=$FTLIB,ID=V113.
ELSEIF(SYSID.EQ.'COS 1.14')
  ACCESS,DN=$FTLIB,ID=V114.
ENDIF.
LDR,NOLIB,LIB=$FTLIB.
```

This conditional block tries to access the correct version of the Fortran library, $FTLIB, for the execution of the loader following the conditional block.

Conditional block with ELSEIF and ELSE - The fourth conditional block structure uses ELSEIF and the ELSE statements. A block can contain any number of ELSEIF statements but can contain only one ELSE, which must be the last conditional statement before the ENDIF.

The ELSE control statement sequence in this case is processed only if both of the following are true:

- The expression on the IF statement is false
- All ELSEIF statement expressions are also false

The following example is an expansion of the example for the third format and allows execution of the compiled program if there is enough time left and if the correct library is accessible. On a successful run, the dataset called RESULTS is disposed as a staged dataset.

Example:

```
IF(TIMELEFT.GT.175)
  IF(SYSID.EQ.'COS 1.12')
    ACCESS,DN=$FTLIB,ID=V112.
  ELSEIF(SYSID.EQ.'COS 1.13')
    ACCESS,DN=$FTLIB,ID=V113.
  ELSE.
    *.
    *.  CURRENT SYSTEM LEVEL NOT RECENT ENOUGH
    *.
    EXIT.
  ENDIF.
  LDR,NOLIB,LIB=$FTLIB.
  SET,J1='YES'L.
ELSE.
  SET,J1='NOTIME'L.
ENDIF.
IF(J1.EQ.'YES'L)
  DISPOSE,DN=RESULTS,DC=ST.
ELSE.
  *.
  *.  JOB DID NOT RUN TO NORMAL COMPLETION
ENDIF.
EXIT.
```

16.1.3 ITERATIVE CONTROL STATEMENT BLOCKS

An iterative control statement block is the third fundamental logic structure allowed by COS JCL. It contains a control statement sequence that will process more than once during the processing of a job.

- LOOP defines the beginning of an iterative block.
- EXITLOOP defines the conditions under which the control statement block iteration is to end.
- ENDLOOP defines the end of an iterative control statement block.

16.1.3.1 ENDLOOP - End iterative block

The ENDLOOP control statement terminates an iterative control statement block. If an ENDLOOP control statement occurs without a preceding LOOP statement at the same nesting level, a job step abort occurs. Execution of the ENDLOOP statement results in control being passed to the preceding LOOP statement, which begins another iteration of the loop.

There are no parameters.

Format:

```
ENDLOOP.
```

16.1.3.2 EXITLOOP - End iteration

The EXITLOOP control statement defines the conditions under which the control statement block iteration is to end. If its expression is true, the loop is exited; if it is false, the control statements that follow are executed.

An EXITLOOP statement that does not appear within an iterative block causes a job step abort. When nesting iterative control statement blocks, the EXITLOOP control statement defines the exit conditions for only the most immediate iterative block. EXITLOOP is a system verb.

Formats:

```
EXITLOOP.

EXITLOOP(expression)
```

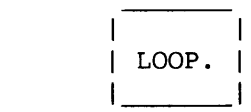
expression
Optional valid JCL expression (discussed later in this section). If omitted, an unconditional exit from the iterative block occurs.

16.1.3.3 LOOP - Begin iterative block

The LOOP control statement defines the beginning of an iterative block. An ENDLOOP control statement is required at the same nesting level to terminate the iterative block. LOOP is a system verb.

There are no parameters.

Format:



Iterative blocks are prescanned for syntax errors before actual processing begins. Any errors in the block structure cause a skipping of that block followed by a job step abort. If an iterative block is included within a conditional block, it must be totally contained within that conditional block.

The following example merges the two datasets DSIN1 and DSIN2 for 60 records.

Example:

```
SET,J1=0.
SET,J2=60.
LOOP.
  EXITLOOP(J2.EQ.0)
  IF(J1.EQ.0)
   COPYR,I=DSIN1,O=OUTDS.
   SET,J1=1.
  ELSE.
   COPYR,I=DSIN2,O=OUTDS.
   SET,J1=0.
  ENDIF.
  SET,J2=J2-1.
ENDLOOP.
REWIND,DN=DSIN1:DSIN2:OUTDS.
```

16.2 JOB CONTROL LANGUAGE EXPRESSIONS

Much of the power of the control statements described in this section is derived from the use of *expressions*. Expressions allow operations such as incrementing counters, checking error codes, and comparing strings.

An *expression* is a *string* (strings are described in the Strings subsection below) consisting of *operands* and *operators*. Expressions are evaluated from left to right, honoring nested parentheses and operator hierarchy. This subsection begins by defining operands and operators, and it ends by discussing expression evaluation and strings.

16.2.1 OPERANDS

Expression *operands* are of four types:

- Integer constants
- Literal constants
- Symbolic variables
- Subexpressions

16.2.1.1 Integer constants

An *integer constant* is a character string with two possible forms:

$\pm$ *ddd...*

*nnn...*B

d is a decimal digit and *n* is an octal digit.

An integer constant has an approximate decimal range $0 \leq |I| \leq 10^{19}$. Range overflow is not detected and overflow results may be unpredictable.

16.2.1.2 Literal constants

A *literal constant* is a string of 1 to 8 characters of the form:

'*ccc...*'L
'*ccc...*'R
'*ccc...*'H

c is a character code with an ordinal number in the the range 040_8 through 176_8. The value of a character constant corresponds to the ASCII character codes positioned within a 64-bit word. Alignment is indicated by the following suffixes:

L Left-adjusted, zero-filled
R Right-adjusted, zero-filled
H Left-adjusted, space-filled

If no suffix is supplied, H is assumed.

16.2.1.3 Symbolic variables

A *symbolic variable* is a string of 1- to 8-alphanumeric characters, beginning with an alphabetic character.

A symbolic variable always has an associated value. COS defines a set of symbols when the job is initiated. Symbols are mnemonics for values maintained by COS or the user or both. You can manipulate the group of symbols listed in table 16-1 through COS control statements or through system requests.

Certain symbols allow communication between COS and the job being processed. Used in the JCL block control statements defined in this section, these symbols provide you with powerful tools for analyzing the progress of a job. For example, a job can request the reason for an abort situation and proceed, based on the reply from COS, through the use of conditional control statements.

Symbols that are preserved over subprocedure calls are called *local* to a procedure; they are saved when a subprocedure is called. Those that are not preserved are *global* over all procedures and can be altered by any procedure. *Constants* are symbols that are never altered.

Information on predefined symbols is summarized in table 16-1; the only local symbols are J0 through J7.

Table 16-1. Symbolic Variable Table

Symbol	Set By	Range	Description
ABTCODE	S†	System error codes 0-*nnn*	COS job abort code; abort code corresponding to most recent job step abort. The abort code corresponds to the abort message number (the *nnn* in AB*nnn*) issued by COS. Refer to the COS Message Manual, CRI publication SR-0039, for the error codes.
DATE	S	Literal value	Date in the form *mm/dd/yy*
FALSE	I††	0	False value
FL	S†	$0\text{-}77777777_8$	Current job field length; can be set with MEMORY statement.
FLM	S	$0\text{-}77777777_8$	Maximum job field length; determined by JOB statement.
G0-G7	U†††	Any 64-bit value	Global job pseudo-registers; represent user-alterable data global over all procedure levels. Data can be passed or into returned from procedures with the G registers.
J0-J7	U††	Any 64-bit value	Job pseudo-registers; represent user-alterable data local to a procedure. Each procedure level can be considered to have its own set of J registers.
JSR	U	Any 64-bit value	Job status register; previous job step completion code (normally 0).
NOTEXT	U	Any 64-bit value	Text field not echoed (text field secured). Default is ON.

† The letter S designates COS.
†† The letter I designates a system constant.
††† The letter U designates user.

Table 6-1. Symbolic Variable Table (continued)

Symbol	Set By	Range	Description
PDMFC	U	64-bit value	Most recent user-issued PDM request.
PDMST	S	64-bit value	Status of most recent Permanent Dataset Manager request.
SID	I	Literal value	Mainframe identifier for front end of job origin; 2 right-justified ASCII characters.
SN	I	64-bit integer	CPU serial number
SSW*n*	U	($1 \leq n \leq 6$)	Job pseudo sense switch settings; can be set with the SWITCH statement.
SYSID	I	Literal value	COS system level of the form 'COS X.XX'
TRUE	I†	-1	True value
TIME	S	Literal value	Time of day in the form *hh:mm:ss*
TIMELEFT	S	64-bit integer	Job time remaining (in milliseconds) as an integer value
XM	I	64-bit value	System-build module level

† The letter I designates a system constant.

16.2.1.4 Subexpressions

A *subexpression* is an expression that is evaluated so that its result becomes an operand.

16.2.2 OPERATORS

Expression *operators* are of three types:

- Arithmetic
- Relational
- Logical

These operators are used in the Fortran sense. Table 16-2 details the expression operators.

Table 16-2. Expression Operator Table

Type	Function	Symbol	Results
A†	Addition	+	64-bit sum of operands
A	Unary plus	+	Following integer operand is positive
A	Subtraction	-	64-bit difference of operands
A	Unary minus	-	Following integer operand is negative
A	Multiplication	*	64-bit product of operands
A	Division	/	64-bit quotient of operands
R††	Equal	.EQ.	True/false
R	Not equal	.NE.	True/false
R	Less than	.LT.	True/false
R	Greater than	.GT.	True/false
R	Less than or equal	.LE.	True/false
R	Greater than or equal	.GE.	True/false
L†††	Inclusive OR	.OR.	A 1 bit in either operand sets corresponding bit in the result.
L†††	Intersection	.AND.	A 1 bit in both operands sets corresponding bit in the result.
L	Exclusive OR	.XOR.	A 1 bit is set in the result if either (but not both) corresponding bit in the operands is 1.
L	Unary complement	.NOT.	A 1 bit (or 0) is set in the result if the corresponding operand bit is 0 (or 1).

† The letter A designates arithmetic.
†† The letter R designates relational.
††† The letter L designates logical.

16.2.2.1 Arithmetic operators

All *arithmetic operations* are performed on 64-bit integer quantities. Care must be used with arithmetic operators because of the following conditions:

- Multiplication/division underflow or overflow of the result is not detected
- Division by 0 produces a zero result
- Intermediate and final results are truncated (for example, 2*(13/2) yields 12, whereas (2*13)/2 yields 13)

16.2.2.2 Relational operators

Relational operations return a -1 value for a TRUE result and a 0 value for a FALSE result. A value produced by an arithmetic or logical operation is considered true if it is a negative value.

16.2.2.3 Logical operators

Logical operations return a 64-bit result. Their functions are performed on a bit-by-bit basis.

16.2.3 EXPRESSION EVALUATION

Expressions are evaluated from left to right, honoring nested parentheses. The operator hierarchy is as follows:

1. Multiplication and division
2. Addition, subtraction, and negation
3. Relational operation
4. Complement (.NOT.)
5. Intersection (.AND.)
6. Inclusive OR (.OR.)
7. Exclusive OR (.XOR.)

Parentheses can be used to change the order of evaluation. For example, 2+3*4 is evaluated as 14, whereas (2+3)*4 is evaluated as 20.

****

CAUTION

Because COS does not check for type, the results of expression evaluation may not be as expected. For example, although both J1.EQ.1 and J2.EQ.2 are TRUE, (J1 .AND. J2) is FALSE.

****

16.2.4 STRINGS

A *string* is a group of characters that is to be taken literally as a parameter value as follows:

- Strings are normally delimited with apostrophes, in which case they are referred to as *literal strings*.
- Strings can also be delimited with open and close parentheses, in which case they are referred to as *parenthetic strings*.

Characters in a string can be any ASCII graphic characters (codes 040_8 through 176_8).

16.2.4.1 Literal strings

Apostrophes are never treated as part of a literal string during evaluation except when doubled or when the literal string is a part of an expression (refer to the examples). To continue literal strings to another line, place an apostrophe followed by a continuation character at the end of the line, and place the remainder of the string on the next line preceded by an apostrophe. Characters otherwise recognized as separators are not evaluated as such when part of a literal string. Doubled apostrophes within a literal string are interpreted as a single apostrophe. A literal string without characters is the null string.

Examples:

String	Interpretation
...'LITERAL STRING'	LITERAL STRING
...'LITERAL STRING'^ 'ACROSS LINE BOUNDARIES'	LITERAL STRINGACROSS LINE BOUNDARIES
...'WON''T SHOW'	WON'T SHOW

String	Interpretation
...'LITERAL^ STRING'	LITERAL^ STRING
...''	Interpreted as a null string
...IF (G0.EQ.'COS1.01'L)	G0.EQ.'COS1.01'L

16.2.4.2 Parenthetic strings

There are two main differences between parenthetic strings and literal strings: in parenthetic strings, blank spaces are removed, and some separators are evaluated. The separators are evaluated as follows:

- If apostrophes appear in a parenthetic string, the enclosed characters are interpreted as a literal string.
- The continuation character is interpreted within a parenthetic string.
- Nested parentheses within a parenthetic string are not treated as separators.

Examples:

String	Interpretation
...(LITERAL STRING)	LITERALSTRING
...(LITERAL STRING^ ACROSS LINE BOUNDARIES)	LITERALSTRINGACROSSLINEBOUNDARIES
...(WON''T SHOW)	WONTSHOW
...((NESTED PARENTHESES))	(NESTEDPARENTHESES)
...(STRING 'LITERAL STRING')	STRINGLITERAL STRING
...(CLOSED PARENTHESIS ')')	CLOSEDPARENTHESIS)
...(KEYWORD=ABC.DEF)	KEYWORD=ABC.DEF
...( )	Interpreted as a null string
...()	Interpreted as a null string

16.3 PROCEDURES

Just as Fortran programs can be divided into separate modules called subprograms, control statement sequences can be divided into modules called *procedures*. A procedure is a sequence of control statements, data, or both that have been saved for processing at a later time. Procedures simplify control statement use in three ways:

- Generalized procedures can be written to perform many similar tasks. Work is saved because a new control statement sequence need not be written to perform each separate task.

- Complex control statement structures can be decomposed into separate subtasks, with a separate procedure written for each subtask. Such modularization reduces the job's design complexity and allows each subtask to be individually tested.

- Procedure libraries can be built. Procedures need be defined only once and placed in a library; different jobs and users can use the procedures and make them part of their own control statement structures.

Procedures have two formats:

- A *simple procedure* consists of only the control statement body.

- A *complex procedure* consists of a prototype definition statement, control statement body, and optional data.

16.3.1 SIMPLE PROCEDURES

A simple procedure is a series of control statements that does not reside in the primary control statement dataset ($CS). No parameter substitution occurs in a simple procedure.

Because a simple procedure has no name associated with it, a simple procedure can only reside in a nonlibrary dataset. It therefore must be invoked with the CALL control statement without the CNS parameter.

Example:

The first file of dataset MOVER contains five control statements. The five control statements can be executed with the following procedure calling statement:

```
CALL,DN=MOVER.
```

In the preceding example, interpretation of control statements from dataset MOVER terminates when a RETURN statement is encountered (refer to section 7), when the end of the first file (in dataset MOVER) is reached, or when an EXIT statement is encountered while COS is not skipping control statements due to an error condition.

16.3.2 COMPLEX PROCEDURES

A complex procedure allows replacing values within the procedure body with values supplied from the procedure call. These values are called *substitution parameters* and are governed by the prototype statement of the procedure.

A complex procedure can reside in a library or nonlibrary dataset; they are invoked (executed) in one of two fashions:

- Procedure name call. The procedure must first reside in a known control statement library (either $PROC or a local dataset named with a LIBRARY control statement); the procedure is called (invoked) by using the procedure name as the control statement verb.

- CALL statement call. The procedure must reside in the first file of a separate dataset; the dataset is named in the CALL control statement. The CNS (crack next statement) parameter must be used for the operating system to properly recognize and process the procedure prototype statement. PROC and ENDPROC are not used with CALL.

Complex procedures can be defined within the control statement stream (*in-line definition*) or as input to the BUILD utility. BUILD currently does not support procedure entries in libraries. When an in-line procedure definition is encountered in the JCL control statement file, it is processed and written to the system default library $PROC. Refer to example 8 later in this section for an example of how to create a user permanent procedure library.

A complex procedure can contain *formal parameters* that define what substitution is to occur in the procedure body. A character string that is eligible for substitution is listed in the prototype statement as a *formal parameter specification*. This name, when preceded by an ampersand in the definition body, indicates that a value is to be substituted during procedure invocation. COS replaces the ampersand and parameter name with the corresponding value supplied by the procedure invocation. If the parameter listed in the prototype statement is not preceded by an ampersand in the body, substitution does not occur. If two ampersands precede the string, one is removed and substitution is inhibited.

Any string consisting of 1 to 8 characters (ampersand included) can be selected for substitution.

When a statement in the current control statement file calls a procedure, COS searches the definition body for the character strings preceded by ampersands. For each occurrence, COS substitutes the values supplied by either the calling statement or the prototype statement.

Whereas simple procedures consist only of a control statement body, complex procedures contain five elements as shown in the following example:

Example:

```
PROC.
Prototype statement
Control statements
  .
  .
  .
&DATA,dn.
Data
*                    Optional; may be repeated for more than one dn
  .
  .
  .
ENDPROC.
```

- PROC defines the beginning of an in-line procedure definition block.

- The prototype statement specifies the name of the procedure and identifies character strings within the procedure that are to be substituted when the procedure is called. COS uses values supplied with the procedure call and default parameter values from the prototype statement to replace these strings.

- The procedure definition body is a sequence of COS control statements processed as part of the current control statement file when the procedure is called. It can optionally include lines of text data preceded in the definition body by an &DATA control statement.

- &DATA introduces text information to be included in the procedure definition body, and it names the dataset to be created and written to when the procedure is invoked. If the dataset already exists, it will be overwritten. When the procedure is invoked, the named dataset is created and the text information is available in that local dataset, including any substitutions resulting from the call. This temporary dataset remains local and allows programs such as CAL or CFT to use the temporary dataset as source data.

- ENDPROC indicates the end of an in-line procedure definition block.

The first control statement in an in-line procedure is PROC; the last is ENDPROC. A prototype statement follows PROC providing the name of the procedure and optionally a list of parameters that identify the substitution values within the definition body.

In addition to defining the values to be substituted, the prototype statement parameters control the selection or omission of the parameters and define the default value assignments. The control statements and data to be processed are contained in the definition body. The control statements are grouped in a sequence.

If data is included in a procedure, the data is preceded by an &DATA statement and follows the control statement sequence. The &DATA statement also includes the name of the dataset to which the data is to be written after processing so that programs can use the data as source data.

A definition can be placed within a definition; such nesting can occur to any level. Nested definitions do not become defined, however, until the outermost procedure is invoked.

16.3.2.1 PROC - Begin procedure definition

The PROC control statement defines the beginning of an in-line procedure definition block. PROC is a system verb. There are no parameters.

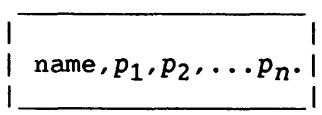
Format:

```
 _______
|       |
| PROC. |
|_______|
```

16.3.2.2 Prototype statement - Introduce a procedure

The prototype control statement specifies the name of the procedure and provides the *formal parameter specifications* that define where substitution is to occur within the definition body. Value substitution is described later in this section.

Format:



name Procedure name; 1 to 8 alphanumeric characters. The name should not be the same as a system verb; if it is, the results are unpredictable.

p_i Formal parameter specifications, using one of the formats which follow. A formal parameter identifies a character string within the definition body. All formal positional parameters, if any, must precede all formal keyword parameters; if they do not, the procedure definition is in error and the job aborts.

pos_i Positional formal parameter specification

key_i=*dvalue*:*kvalue*
Keyword formal parameter specification as follows:

key_i Formal keyword parameter

dvalue Optional default value; this value is substituted if entire keyword parameter is omitted from the calling statement.

kvalue Optional keyed default value; this value is substituted if the keyword is present but no value is specified.

Special cases:

key_i= Provides no default values and requires the caller to provide a nonnull value

key_i=: Provides no default values, but lets you specify key_i= or just key_i

16.3.2.3 Procedure definition body

The procedure definition body consists of a sequence of COS control statements processed as part of the current control statement file when the procedure is called. (It can optionally include lines of text data preceded in the definition body by an &DATA control statement. Refer to &DATA, which follows.)

The prototype statement identifies character strings within the procedure that are to be substituted when the procedure is called. COS uses values supplied with the procedure call and default parameter values from the prototype statement to replace these strings.

An ampersand (&) must precede each parameter to be substituted (*substitution parameter*) within the definition body. If a parameter appears in the prototype, and a matching string in the body is found but is not preceded by an ampersand, substitution does not occur.

16.3.2.4 &DATA - Procedure data

Data can be included within the procedure definition body after the &DATA statement.

The *dn* parameter creates a temporary dataset composed of the data identified in the procedure, including any substitutions resulting from the call. This temporary dataset allows programs such as CAL or CFT to use it as source data.

Format:

```
 ___________
|           |
| &DATA,dn. |
|___________|
```

dn Name of dataset to contain the data that follows; *dn* is required.

The initial separator for an &DATA statement can be a blank, a comma, or an opening parenthesis; the statement terminator can be a blank, a period, or a closing parenthesis.

An &DATA specification cannot be continued to subsequent lines. All lines following an &DATA statement up to the next &DATA statement or an ENDPROC statement are written to the specified dataset after string substitution is performed. Refer to example 7 later in this section.

16.3.2.5 ENDPROC - End procedure definition

The ENDPROC control statement indicates the end of an in-line procedure definition block. ENDPROC is a system verb.

Format:

```
 ___________
|           |
| ENDPROC.  |
|___________|
```

16.3.3 PARAMETER SUBSTITUTION

Formal parameter specifications can be selected for substitution. Character strings to be substituted are delimited by any character other than numerals, alphabetics, commercial at (@), dollar sign ($), and the percent sign (%). An ASCII underline is used as a string delimiter when the next character is one of these characters. (Refer to example 3 later in this section.) COS deletes the underline after evaluating the string it delimits. Thus, the underline concatenates the strings it delimits.

Formal parameter specifications can be in positional or keyword format.

16.3.3.1 Positional parameters

Positional formal parameters let you list the strings within the body that can be substituted. The calling statement lists values to be substituted for these strings in the same order in which they are listed in the prototype statement. The value supplied with the calling statement is substituted for every occurrence of the corresponding formal positional parameter within the definition body. If the caller passes too few positional parameters, null strings are substituted for the remaining formal positional parameters. If too many positional parameters are passed, the procedure call is in error and the job aborts.

16.3.3.2 Keyword parameters

Keyword formal parameters are listed in any order after all positional parameters are given on the prototype statement and the calling statement. A keyword formal parameter let you specify substitution values on the prototype statement that are to be used when one is not given on the calling statement.

If the keyword formal parameter is included in the calling statement with a value, that value is substituted. If the entire keyword formal parameter is omitted from the calling statement, the *default value* on the prototype statement is substituted. If a default value is not provided on the prototype statement, the character string within the body corresponding to that formal parameter is not included in the procedure expansion.

If only the keyword portion of the keyword formal parameter (the character string itself) is included in the calling statement, without a value assigned to it, a *keyed default value* from the prototype statement is substituted. If a keyed default value is not provided on the prototype statement, again the character string within the body corresponding to that formal parameter is not included in the procedure expansion.

A keyword parameter enclosed in apostrophes ('*KEY=value*') is considered a positional parameter. Table 16-3 summarizes the forms of keyword substitution.

Table 16-3. Keyword Substitution After Expansion

Calling Statement \ Keyword Format for Prototype Statement	*key* (positional)	*key=dv:kv* *key=:kv* *key=:(kv)*	*key=(dv):* *key=dv:(kv)*	*kv key=dv* *key=(dv)*
1. *name, value.*	*value*	CS119†	CS119†	CS119†
2. *name, key.*	CS121†	*kv*	*kv*	CS121†
3. *name.* *name, ().*	Null	Null	*dv*	*dv*
4. *name, key=value.*	*value*	*value*	*value*	*value*
5. *name, key=.*	Null	Null	Null	Null

† Error message number. Refer to the COS Message Manual, publication SR-0039, for an explanation of each message.

kv=keyword value
dv=default value

Error messages:

CS119 - EXTRA POSITIONAL PARAMETER: *value*
CS121 - KEYWORD USED WITHOUT ASSIGNING IT A VALUE: *key*
CS122 - NO VALUE WAS ASSIGNED TO *key*

16.3.3.3 Positional and keyword parameters

When supplying both positional and keyword parameters, all positional parameters must precede all keyword parameters; COS evaluates the call's positional parameters first. The end of the caller's list of positional parameters is signaled by the appearance of a keyword parameter, statement terminator, or by specifying all positionals.

16.3.3.4 Apostrophes and parentheses

Sometimes parameter values in a procedure definition or a procedure calling statement require a special format. If a literal string (a string delimited with apostrophes) appears in either of these statements, it is processed as if it were a literal constant. That is, all apostrophes in the value remain when the value is substituted. Refer to example 5 later in this section.

To avoid erroneous processing, use parentheses as string delimiters in these statements. Outermost parentheses preceded by the initial, parameter, equivalence, or concatenation separators are removed during value substitution. This procedure delays processing of any separator characters in the string until the statement itself, with substituted values, is processed.

This delay is also required when specifying multiple values for the default value or keyed default value parameters on a procedure definition statement. (Refer to examples 1, 2, 4, and 6.) Parentheses are advised in the procedure calling statement when the use of the value in the procedure statements is unknown. (Refer to examples 4, 5, and 6.)

Table 16-4 summarizes the forms of parenthetical substitution.

Table 16-4. Expansion of Parenthetic and Literal String Values

Invocation	Expansion
value	*value*
(*value*1=*value*2)	*value*1=*value*2
*value*1'.'*value*2	*value*1'.'*value*2
*value*1(.)*value*2	*value*1.*value*2

The following examples demonstrate the COS control statement procedure substitution process.

Example 1:

Consider a single statement procedure called LOAD defined as follows:

Definition

```
PROC.
LOAD,NOGO=:NX,LIBRARY=($FTLIB:$SYSLIB):MYLIB.   Prototype statement
LDR,&NOGO,LIB=&LIBRARY.                         Definition body
ENDPROC.
```

The prototype statement in this example defines two formal parameters, both of which are in keyword format. The keyword NOGO has a null value when omitted from the calling statement and a value of NX when included on the calling statement in keyword-only format. The keyword LIBRARY has the default value of $FTLIB:$SYSLIB. When LIBRARY is used in the calling statement without a value, the keyed default value, MYLIB, is substituted.

When the LOAD procedure is invoked, it expands to a single statement whose form depends on the choice of parameters:

Invocation	Expansion
LOAD,NOGO.	LDR,NX,LIB=$FTLIB:$SYSLIB.
LOAD.	LDR,,LIB=$FTLIB:$SYSLIB.
LOAD,LIBRARY=THISLIB.	LDR,,LIB=THISLIB.
LOAD,LIBRARY,NOGO.	LDR,NX,LIB=MYLIB.

Example 2:

The following in-line procedure definition creates a procedure called BLDABS.

Definition

```
PROC.
BLDABS,SOURCE,LIST,GO='NO':'YES',LIB=^
   :($SYSLIB:$FTLIB),MAP=FULL:PART.          Prototype statement
REWIND,DN=$BLD:&SOURCE.
CAL,I=&SOURCE,L=&LIST,ABORT.
LDR,NX,LIB=&LIB,MAP=&MAP,L=&LIST,AB=$ABD.
REWIND,DN=$ABD:&LIST.                        Definition body
SAVE,DN=$ABD,PDN=MYPROGRAM.
IF(&GO.EQ.'YES')
$ABD.
ENDIF.
ENDPROC.
```

Invocation	Expansion
BLDABS,WORK,,GO,LIB=VLIB2.	REWIND,DN=$BLD:WORK. CAL,I=WORK,L=,ABORT. LDR,NX,LIB=VLIB2,MAP=FULL,L=. REWIND,DN=$ABD:. SAVE,DN=$ABD,PDN=MYPROGRAM. IF('YES'.EQ.'YES') $ABD. ENDIF.

Example 3:

This procedure shows the proper use of the underscore character for the definition of a formal parameter. It creates a procedure called AUDJCL.

Definition

```
PROC.
AUDJCL,DN,LEVEL,L=$OUT:AUDLST.               Prototype statement
AUDIT,PDN=&DN&LEVEL_JCL,ID=JCL,L=&L.         Definition body
ENDPROC.
```

Invocation	Expansion
AUDJCL,-,05.	AUDIT,PDN=-05JCL,ID=JCL,L=$OUT.

Example 4:

Parentheses are required when specifying multiple values for a single parameter value on a procedure definition prototype statement or on a calling statement. In these cases, the colon separates default and Boolean values in a keyword parameter. For example:

Procedure-definition Prototype Statement

```
MYPROC,POS1,KEY=(DEF1:DEF2):(B001:B002).
```

Invocation

```
MYPROC,(POS1A:POS1B).
```

When substitution occurs during this call, POS1A:POS1B replaces all POS1 occurrences within the definition body. Both values (POS1A and POS1B) are evaluated separately during control statement evaluation. If apostrophes are on the call, 'POS1A:POS1B' is evaluated as one literal string.

Example 5:

The following procedure definition shows the use of literal strings instead of parenthetical strings.

Definition

```
PROC.
PURGER,PDN,ID,ED,M.                                          Prototype
ACCESS,DN=$PURGE,PDN=&PDN,ID=&ID,ED=&ED,M=&M,UQ,NA.
DELETE,DN=$PURGE,NA.                                         Definition body
RELEASE,DN=$PURGE.
ENDPROC.
```

Invocation	Expansion
PURGER,'SOURCE.MAIN',PROJECT	ACCESS,DN=$PURGE,PDN='SOURCE.MAIN', ID=PROJECT,ED=,M=,UQ,NA. DELETE,DN=$PURGE,NA. RELEASE,DN=$PURGE.

The apostrophes remain as part of the string in the expansion. If parentheses had been used in the invocation instead of apostrophes for the permanent dataset name, (SOURCE.MAIN), the value when the ACCESS statement is evaluated would be SOURCE.MAIN because the outermost parentheses are removed when preceded by a valid separator. This action would cause an error because the period in SOURCE.MAIN would be evaluated as a statement terminator during evaluation.

Example 6:

The following example shows the use of parenthetical strings instead of literal strings in a procedure definition.

Definition

```
PROC.
LGO,CALSORC,ABS,NLIB=$SCILIB:($SCILIB:^
  $SYSLIB:$FTLIB).                                  Prototype
CAL,I=&CALSORC.
LDR,NX,AB=&ABS,NOLIB=&NLIB.                         Definition body
ENDPROC.
```

Invocation	Expansion
LGO,,,NLIB.	CAL,I=. LDR,NX,AB=,NOLIB=$SCILIB:$SYSLIB:$FTLIB.

Parentheses were not included for the expansion of the NLIB keyed default value because parentheses are removed during processing when preceded by the concatenation delimiter (:).

If apostrophes had been used instead of parentheses for the NLIB parameter value, the colons would have been ignored as separators during expansion. Also, apostrophes are treated as part of the value when included in a procedure definition prototype statement or a calling statement. Therefore, if apostrophes had been used, the following expansion would have occurred.

```
CAL,I=.
LDR,NX,AB=,NOLIB='$SCILIB:$SYSLIB:$FTLIB'.
```

When the LDR statement is executed, the value assigned to the NOLIB parameter is the literal string '$SCILIB:$SYSLIB:$FTLIB' which violates the syntax for the NOLIB parameter.

Example 7:

Consider the following procedure definition. This procedure retrieves specified source decks from an UPDATE program library by the use of the &DATA option as follows:

```
PROC.
GDECK,PLNAME,MASTERCH,DECKRNGE.          Prototype statement
ACCESS,DN=&PLNAME.
UPDATE,I=QZRRZQ2,Q,C=0,S,P=&PLNAME.
RELEASE,DN=QZRRZQ2:&PLNAME.              Definition body
&DATA QZRRZQ2
&MASTERCH_COMPILE &DECKRNGE
ENDPROC.
```

Two sample invocations and their expansions follow:

Invocation	Expansion
GDECK,COSPL,*,(ST,CT).	ACCESS,DN=COSPL. UPDATE,I=QZRRZQ2,Q,C=0,S,P=COSPL. RELEASE,DN=QZRRZQ2:COSPL. (Dataset QZRRZQ2 contains: *COMPILE ST,CT)
GDECK,FTLIBPL,*,(COS.RFD).	ACCESS,DN=FTLIBPL. UPDATE,I=QZRRZQ2,Q,C=0,S,P=FTLIBPL. RELEASE,DN=QZRRZQ2:FTLIBPL. (Dataset QZRRZQ2 contains: *COMPILE COS.RFD)

Example 8:

This example shows one mechanism for defining and maintaining user procedure libraries. The new procedure library is saved on mass storage for later use.

```
ACCESS,DN=GENLIB.
CALL,DN=GENLIB.
```

The permanent dataset GENLIB contains the following:

```
ECHO,OFF.
RELEASE,DN=$PROC.
*.
*.        Define procedure for ACCESS of commonly used ID.
*.
PROC.
UQ,DN,ED=:1,PDN=:GENLIB,R=:READCW,W=:WRITECW,M=:MAINCW,NA=:NA.
ACCESS,DN=&DN,ID=MYUID,PDN=&PDN,ED=&ED,R=&R,W=&W,M=&M,NA=&NA.
RETURN.
EXIT.
RETURN,ABORT.
ENDPROC.
*.
*.        Edit a local dataset.
*.
PROC.
ED,DN,AC=:'ACCESS'.
IF(&AC.EQ.'ACCESS')
  UQ,&DN.
ENDIF
TEDI,DN=&DN.
RETURN.
EXIT.
RETURN,ABORT.
ENDPROC.
*.
*.        End of definitions
*.
UQ,PROCLIB,NA.
SAVE,DN=$PROC,PDN=PROCLIB,ID=MYUID.
DELETE,DN=PROCLIB,NA.
RELEASE,DN=$PROC.
ACCESS,DN=PROCLIB,ID=MYUID.
LIBRARY,DN=*:PROCLIB.
ECHO,ON.
```

APPENDIX SECTION

JOB USER AREA A

The table diagrams and their field descriptions were generated by the Table Diagram Generator using definitions from COSPL and SYSDFPL version V116BF1M of December 1986. Subsequent releases and local modifications could compromise the accuracy of the tables printed here.

The table diagrams use the following symbols:

$ When two appear on the same line, indicates a range of words not shown. When one appears at the end of one line and another at the beginning of the next, indicates a field crossing a word boundary.

* Indicates that a field is too short to contain its label.

/ Indicates an unused area of a table. Hashed areas can contain information used elsewhere in the system, such as a front-end station.

Numbers in table descriptions are denoted as follows:

O' Indicates an octal number.

D' Indicates a decimal number.

Throughout this appendix, word numbers are shown in octal. Bit numbers are decimal.

The tables appear in alphabetical order according to field prefix (a 2-character string before the table name). Some program library (PL) decks contain multiple tables. When multiple tables are contained in a single program library deck, they are presented in the order in which they occur. This causes some tables to be out of sequence. An example of this is program library deck COMIJ, which contains definitions for the F$IJMSG parameter block, the Node Control Block, the Receptive Control Block, and the Inter-job Communication Message Buffer. Please check the table of contents if a table does not appear where expected.

A table's prefix is included in every field. To save space, however, the prefix is left off the field names in the bit description.

BGN Table. This table is input to the F$BGN call which provides a mechanism for a user program to indicate to the Operating System the location of the executable binary and a P address which the CPU can be released to. In addition, the old BGN format is supported for this release. The following functions are currently supported with the new BGN format:

A) Load a dataset from mass storage as specified by the DSP.
B) Copy memory from a source base address to target base address for lengths specified.
C) Preset memory with supplied pattern from preset base address for lengths specified.
D) Support for target machine characteristics.

Support is included for the separation of instruction and data segments. Instruction segments are currently supported and any attempt to load a data segment will be aborted.

Define the F$BGN Function codes:

```
BGNLOAD   =  O'1              Load from dataset function code
BGNCOPY   =  O'2              Copy from source to destination
BGNFMAX   =  BGNCOPY          Set max Function Code value
```

BG Begin Code Execution Table - BGN

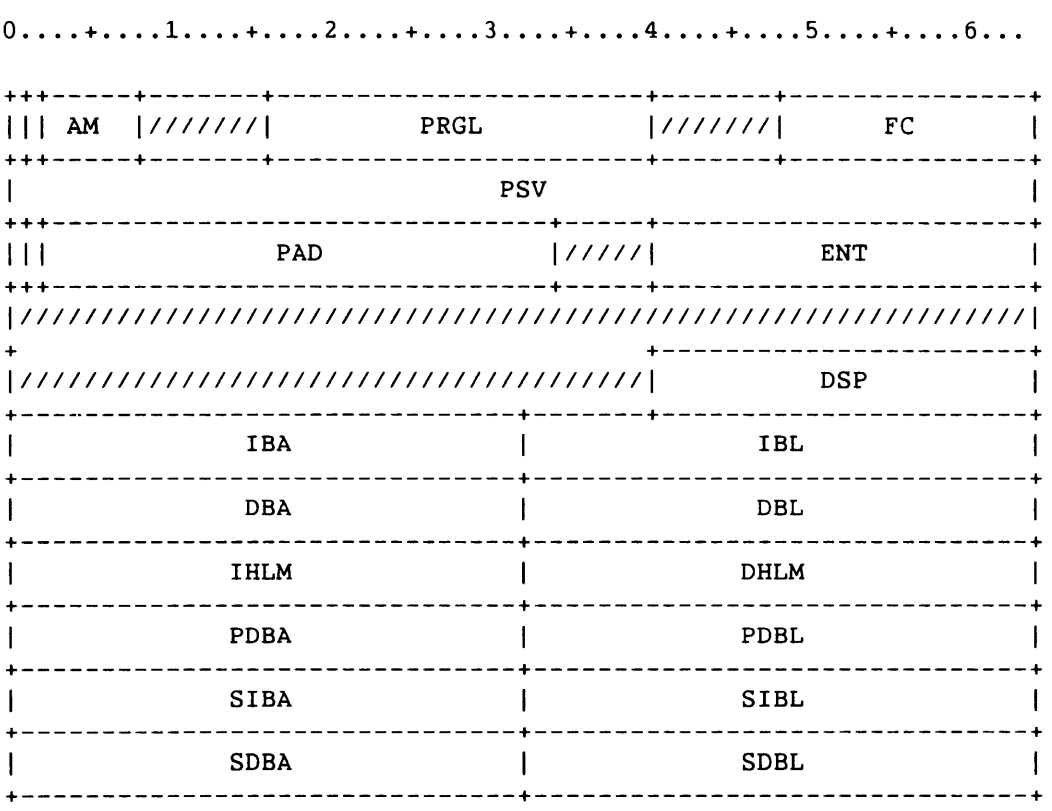


Figure A-1. Begin Code Execution Table

Field	Word(base8)	Bits	Description
BGPSF	0	0	Preset value flag If=1, preset segment
BGEMA	0	1	EMA setting for new calls, 1=ENABLE
BGAM	0	2-7	Additional modes field
BGAMF	0	2	Additional modes flag (1=modes set)
BGBDM	0	3	Bi-directional memory 1=ENABLE
BGAVL	0	4	Additional vector logical 1=ENABLE
BGORI	0	5	Operand range interrupt 1=ENABLE
BGFI	0	6	Floating point interrupt 1=ENABLE
BGPS	0	7	Program state
BGPRGL	0	16-39	Program length(Old BGN Format only)

BG Begin Code Execution Table - BGN

Field	Word(base8)	Bits	Description
BGFC	0	48-63	BGN Function Code(0 for old)
BGPSV	1	0-63	Preset value
BGBP	2	0	Breakpoint flag
BGNRD	2	1	No reduce bit
BGPAD	2	2-33	Pad value
BGENT	2	40-63	Entry point for instruction segment
New BGN table field definitions			
BGDSP	4	40-63	DSP address of load dataset
BGIBA	5	0-31	Instruction base address to load to
BGIBL	5	32-63	Instruction segment length
BGDBA	6	0-31	Data base address to load to
BGDBL	6	32-63	Data segment length
BGIHLM	7	0-31	Instruction segment HLM value
BGDHLM	7	32-63	Data segment HLM value
BGPDBA	10	0-31	Preset data base address for pattern
BGPDBL	10	32-63	Preset data length for pattern
BGSIBA	11	0-31	Source Instruction base address(COPY)
BGSIBL	11	32-63	Source Instruction length(COPY)
BGSDBA	12	0-31	Source Data base address(COPY)
BGSDBL	12	32-63	Source Data length(COPY)

DD Dataset Definition List - DDL

A Dataset Definition List in the user field must accompany any create DNT (F$DNT) request.

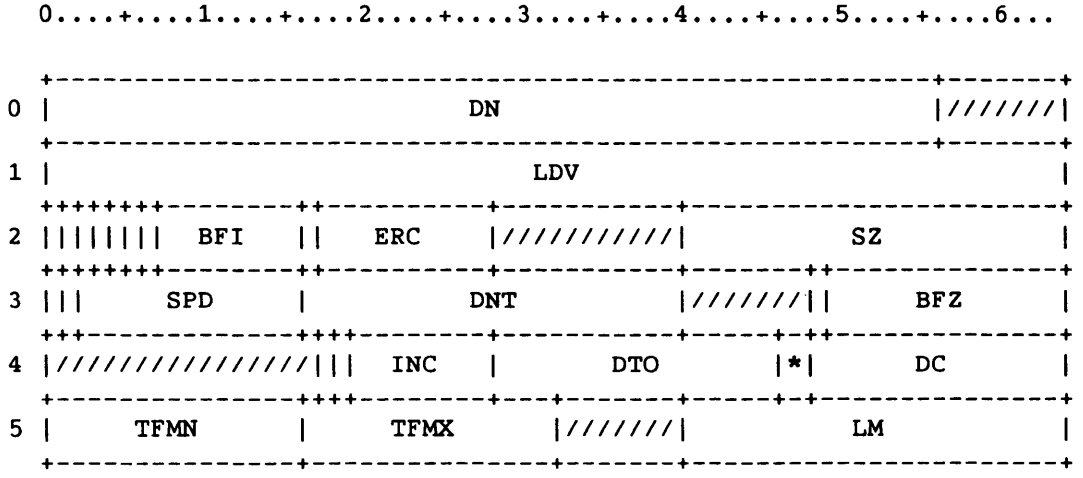


Figure A-2. Dataset Definition List
NUMDT=3 Max number of desired device types

Field	Word(base8)	Bits	Description
DDDN	0	0-55	Dataset name
DDLDV	1	0-63	Logical device name
DDRDM	2	0	Random dataset flag: 0 Sequential 1 Random
DDUDS	2	1	Undefined dataset structure: 0 COS blocked dataset structure 1 Undefined structure
DDNFE	2	2	Return error if dataset does not exist. Register S0 returned nonzero if DNT does not exist; no DNT is created.

DD Dataset Definition List - DDL

Field	Word(base8)	Bits	Description
DDSTAT	2	3	Request dataset statistics; ignored unless DDNFE=1 (see DDDNT)
DDMR	2	4	Dataset is to be memory resident
DDIA	2	5	Interactive type dataset
DDTRAN	2	6	Transparent mode for interactive dataset
DDBFI	2	7-15	Blank field indicator (octal) for character I/O: Value Symbol Meaning 000 BFI=I@BFI 1-377 This ASCII character 400 BFI@ZER BFI=<000> >400 Disabled 777 BFI@OFF Disabled
DDNA	2	16	No-Abort flag
DDERC	2	17-27	Error code if No-Abort set
DDSZ	2	40-63	Dataset size in 512-word blocks
DDSEQ	3	0	Change a dataset from random to sequential. Valid only if dataset is currently random, ignored if sequential.
DDBLK	3	1	Change a dataset form unblocked to blocked. Valid only if dataset is currently unblocked, ignored if blocked.
DDSPD	3	2-15	Sectors to allocate before switching devices"STRIPING"
DDDNT	3	16-39	Address of DNT image returned by F$DNT when DDNFE=1 and DDSTAT=1
DDNOF	3	48	No Overflow flag
DDBFZ	3	49-63	Buffer size in 512-word blocks $SYSTXT name

DD Dataset Definition List - DDL

Field	Word(base8)	Bits	Description
DDC	4	17	allocate contiguous space for request
DDSL	4	18	Superlink dataset flag
DDINC	4	19-27	sectors to allocate per request
DDDTO	4	28-45	Default devices wanted
DDDT1	4	28-33	Desired device type for storage
DDDT2	4	34-39	2nd preferred type for storage
DDDT3	4	40-45	3rd preferred type for storage
DDST	4	46-47	Storage type
DDSCR	4	46	Scratch storage space preferred
DDPERM	4	47	Permanent storage space necessary
DDDC	4	48-63	Disposition code (two characters):
DDTFMN	5	0-15	Transfer minimum
DDTFMX	5	16-31	Transfer maximum
DDLM	5	40-63	Dataset size limit in 512-word blocks

Logical I/O requires the presence of a DSP for the dataset in the user's field. Refer to COS Version 1 Reference Manual, publication SR-0011, for details of DSP use.

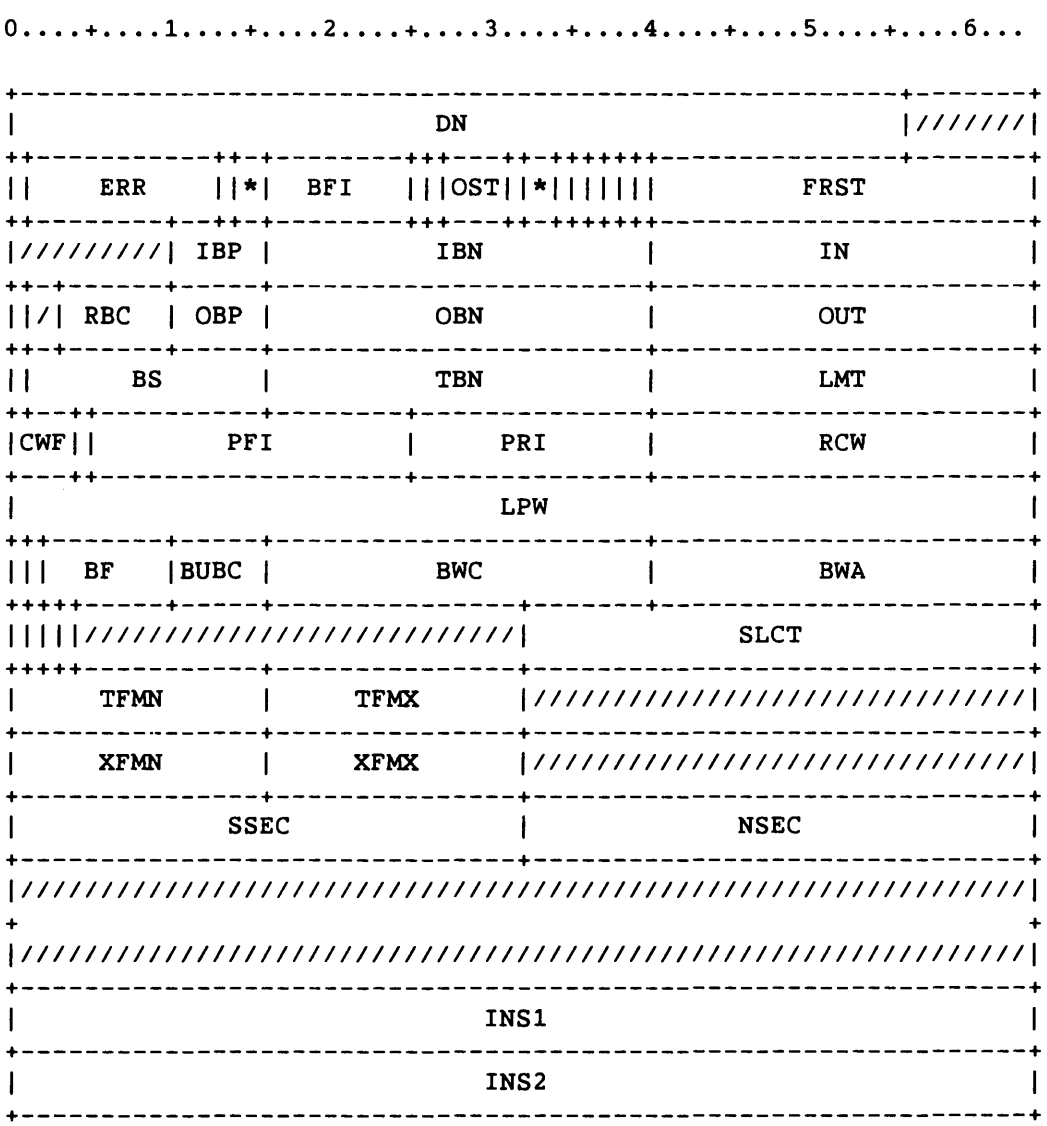


Figure A-3. Dataset Parameter Table

DP Dataset Parameter Table   - DSP

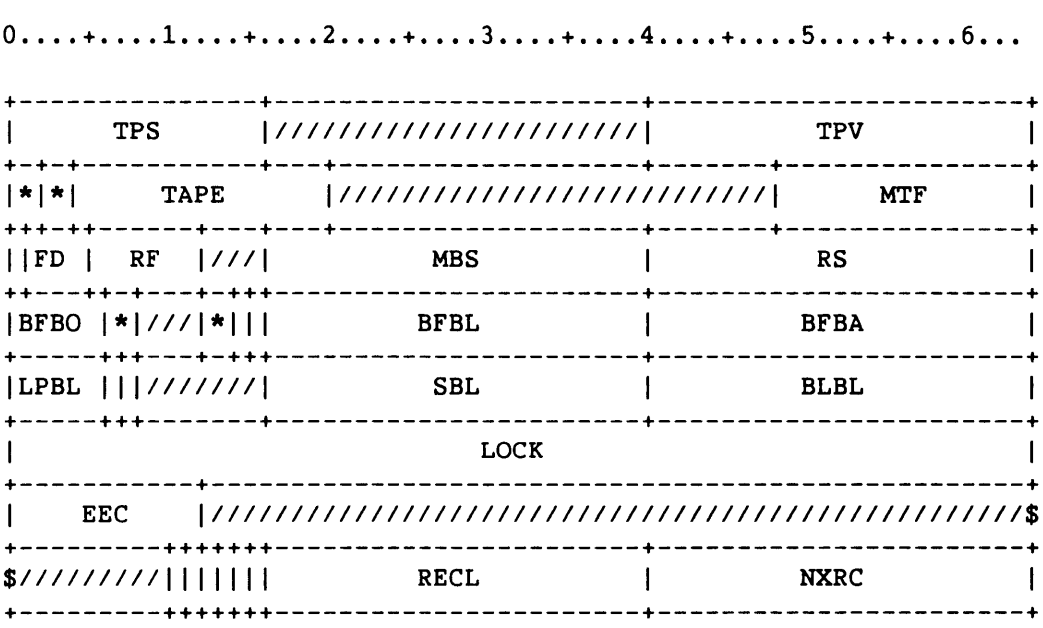


Figure A-3. Dataset Parameter Table

Field	Word(base8)	Bits	Description
DPDN	0	0-55	Dataset name
DPBSY	1	0	Busy flag, circular I/O: 0 Not busy 1 Busy
DPERR	1	1-12	Error flags:
DPEOI	1	1	End of data on read; write past allocated disk space on write.
DPENX	1	2	Dataset does not exist
DPEOP	1	3	Dataset not open
DPEPD	1	4	Invalid processing direction
DPEBN	1	5	Block number error
DPEDE	1	6	Unrecovered data error
DPEHE	1	7	Unrecovered hardware error
DPERW	1	8	Attempted read after write or past EOD
DPEPT	1	9	Dataset prematurely terminated

DPELE	1	10	Unrecovered logical data error Reserved
DPEEP	1	12	Extended error (see DPEEC)
DPSTS	1	14-15	Status: 00 Closed 01 Open for output (O) 10 Open for input (I) 11 Open for I/O
DPBFI	1	16-24	Blank compression character in ASCII (BFI=O'777 implies no compression)
DPISP	1	25	ISP dataset flag
DPQIO	1	26	Queued I/O Request Flag
DPOST	1	27-30	Open status
DPABD	1	31	Accept bad data flag
DPTP	1	32-33	Tape dataset (online/staged)
DPTRAN	1	34	Transparent mode for interactive dataset
DPIA	1	35	Dataset is interactive
DPMEM	1	36	Dataset is memory resident
DPRDM	1	37	Random dataset flag: 0 Sequential dataset 1 Random dataset
DPUDS	1	38	Undefined dataset structure: 0 COS-blocked dataset structure 1 Undefined dataset structure
DPEND	1	39	Write end-of-data flag
DPFRST	1	40-63	Address of first word of buffer
DPIBP	2	10-15	Input bit position

Field	Word(base8)	Bits	Description
DPIBN	2	16-39	Block number, read request. System reads from block number until buffer is filled. DPIBN is then set to the next block number.
DPIN	2	40-63	Address of current input word
DPSPOS	3	0	Asynchronous SETPOS busy flag
DPRBC	3	3-9	Remaining blank count
DPOBP	3	10-15	Bit position in current output word (character I/O only)
DPOBN	3	16-39	Block number, write request. System writes from block number until buffer is empty. The next block number is then in DPOBN.
DPOUT	3	40-63	Address of current output word
DPUEOF	4	0	Uncleared end-of-file (EOF)
DPBS	4	1-15	Buffer size (in D'512 word sectors)
DPTBN	4	16-39	Temporary block number; used by random I/O for last block read
DPLMT	4	40-63	Address of last word+1 of buffer. LMT minus FRST defines buffer size.
DPCWF	5	0-3	Control word types detected
DPEOR	5	0	End Of Record
DPEOF	5	2	End Of File
DPEOD	5	3	End Of Data
DPRW	5	4	Previous operation read/write flag: 0 Read 1 Write
DPPFI	5	5-24	Previous file index; backward index to block containing previous EOF.

Field	Word(base8)	Bits	Description
DPPRI	5	25-39	Previous record index; backward index to block containing previous EOR.
DPRCW	5	40-63	Control word address: Previous RCW address if in write mode Next RCW if in read mode
DPLPW	6	0-63	Last partial word; used for character mode I/O
DPBIO	7	0	Buffered I/O busy: 0 Buffered I/O operation complete 1 Buffered I/O operation incomplete
DPBER	7	1	Buffered I/O error flag
DPBF	7	2-9	Function code: BIOFRRP = 0 Read partial record BIOFRR = O'10 Read record BIOFWRP = O'40 Write partial record BIOFWR = O'50 Write record BIOFEOF = O'52 Write EOF BIOFEOD = O'56 Write EOF
DPBPD	7	4	Processing direction: 0 Read 1 Write
DPBEO	7	6-9	Termination condition: 00 Partial 10 Record 12 File, write only 16 Dataset, write only
DPBUBC	7	10-15	Unused bit count; must be specified on a write record request. Value returned on a read request.

DP Dataset Parameter Table - DSP

Field	Word(base8)	Bits	Description
DPBWC	7	16-39	Word count; number of words at DPBWA to read or write. Field contains actual number of words read when request is completed.
DPBWA	7	40-63	Word address of user data area
DPXV	10	0	Extended DSP validation if set
DPNSN	10	1	New sector number processing mode
DPRMIO	10	2	I/O mode is record flag
DPSL	10	3	Superlink dataset flag
DPSLCT	10	32-63	Pointer to Superlink Connection Tables
DPTFMN	11	0-15	Minimum buffer transfer size (sectors)
DPTFMX	11	16-31	Maximum buffer transfer size (sectors)
DPXFMN	12	0-15	Active transfer minimum size (sectors)
DPXFMX	12	16-31	Active transfer maximum size (sectors)
DPSSEC	13	0-31	Starting sector number (FSS copy)
DPNSEC	13	32-63	Number of sectors to copy (FSS copy)
DPINS1	16	0-63	Reserved for installation
DPINS2	17	0-63	Reserved for installation
DPTPS	20	0-15	Online tape status
DPTPV	20	40-63	Tape pointer to label definition table
DPTPD	21	0-1	Tape density
DPTPF	21	2-3	Tape format

Field	Word(base8)	Bits	Description
DPTAPE	21	4-19	Tape status
DPAEV	21	4	User is at tape end of volume
DPTOR	21	5	Tape off reel
DPTMS	21	6	Tape mark status
DPBLT	21	7	Blank tape
DPEOVR	21	8	EOV READ
DPBTM	21	9	tape is before tape mark

MASKS FOR TESTING TAPE STATUS FIELD

TS$EOV=O'100000	EOV mask
TS$TOR=O'040000	Tape off reel mask
TS$TMS=O'020000	Tape mark status mask
TS$BLT=O'010000	Blank tape detected mask
TS$EOVR=O'004000	Read completed in EOV processing
TS$BTM=O'002000	tape is before tape mark

DPMTF	21	48-63	Maintenance test field
DPCV	22	0	Data conversion flag
			DPCVOFF=0 Data conversion off
			DPCVON=1 Data conversion on
DPFD	22	1-4	Translation identifier
			DPFDNONE=0 NO foreign file translation
			DPFDIBM=1 IBM file translation
			DPFDCDC=2 CDC file translation
			DPFDVMS=3 VMS file translation

Field	Word(base8)	Bits	Description
DPRF	22	5-11	Record format (if DPCT nonzero)
			DPRFUNKN=O'177 Unknown record format
			DPRFIU=0 IBM undefined
			DPRFIF=1 IBM fixed
			DPRFIFB=2 IBM fixed blocked
			DPRFIV=3 IBM variable
			DPRFIVB=4 IBM variable blocked
			DPRFIVBS=5 IBM variable block span
			Values 21 through 37 are reserved for ANSI record types:
			DPRFIIW=O'00 I tape format, I blocks, W records
			DPRFICW=O'10 I tape format, C blocks, W records
			DPRFICZ=O'11 I tape format, C blocks, Z records
			DPRFICS=O'12 I tape format, C blocks, S records
			DPRFSIIW=O'40 SI tape format, I blocks, W records
			DPRFSICW=O'50 SI tape format, C blocks, W records
			DPRFSICZ=O'51 SI tape format, C blocks, Z records
			DPRFSICS=O'52 SI tape format, C blocks, S records
			DPRFVF=1 VMS F format
			DPRFVUF=2 VMS UF format
			DPRFVD=3 VMS D format
			DPRFVV=4 VMS V format
			DPRFVS=5 VMS S format
			DPRFVUS=6 VMS US format
DPMBS	22	16-39	Maximum block size
DPRS	22	40-63	Record length
DPBFBO	23	0-5	User data area current bit offset

Field	Word(base8)	Bits	Description
DPCS	23	6-7	Character set (if DPCT nonzero) DPCSAS=0 ASCII DPCSEB=1 EBCDIC DPCSDC=2 CONTROL DATA display code
DPSCC	23	12-13	Record continuation code
DPBDF	23	14	Bad data flag
DPPCR	23	15	Process-characters-remaining flag
DPBFBL	23	16-39	User data area current bit length
DPBFBA	23	40-63	User data area current address
DPLPBL	24	0-5	Last partial word bit length
DPEOLR	24	6	Foreign dataset end of logical record
DPEOLF	24	7	Foreign dataset end of logical file
DPSBL	24	16-39	Current segment/record bit length
DPBLBL	24	40-63	Current tape block bit length
DPLOCK	25	0-63	Multitasking lock (nonzero TIB address
DPEEC	26	0-11	Error code if DPEEP is set; correspond to EXP abort codes.
DPSEQ	27	10	FORTRAN sequential access flag
DPFMT	27	11	FORTRAN formatted I/O flag
DPDEL	27	12	FORTRAN file status: 0 Keep 1 Delete
DPBLNK	27	13	FORTRAN numeric input blank conversion: 0 Null 1 Zero
DPDIR	27	14	FORTRAN direct access flag

DP Dataset Parameter Table - DSP

Field	Word(base8)	Bits	Description
DPUFMT	27	15	FORTRAN unformatted I/O flag
DPRECL	27	16-39	FORTRAN direct access record length (in number of characters)
DPNXRC	27	40-63	FORTRAN direct access next record number

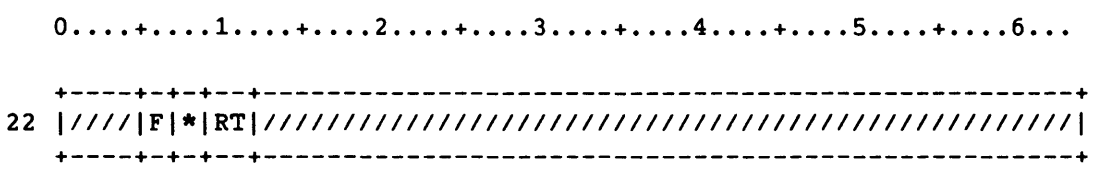


Figure A-4. CDC Record Format

Field	Word(base8)	Bits	Description
DPF	22	5-6	Tape format DPFNONE=0 No tape format DPFI=1 Internal DPFSI=2 System or scope internal
DPBT	22	7-8	Block type DPBTI=0 Internal DPBTC=1 Character count
DPRT	22	9-11	Record type DPRTW=0 Control word DPRTZ=1 Zero byte DPRTS=2 System-logical

DP Dataset Parameter Table   - DSP

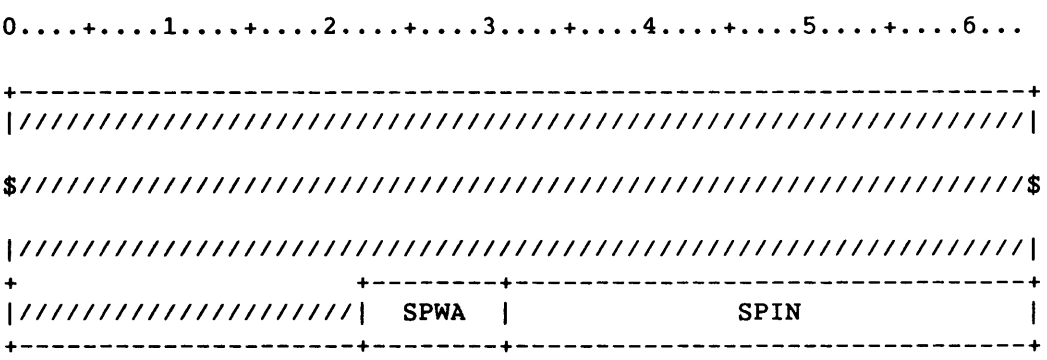


Figure A-5. Save Areas Used by Asynchronous SETPOS

Field	Word(base8)	Bits	Description
DPSPWA	6	22-30	Word address save areas used
DPSPIN	6	31-63	by asynchronous SETPOS

DR Disk Reservation Table - DRT

STP contains a Disk Reservation Table (DRT) for each logical mass storage device known to the system. The table (figure A-6) consists of a header and a bit map. Each bit in the bit map represents one allocation unit (AU), such as one track on a disk. A set bit implies that the the AU is in use.

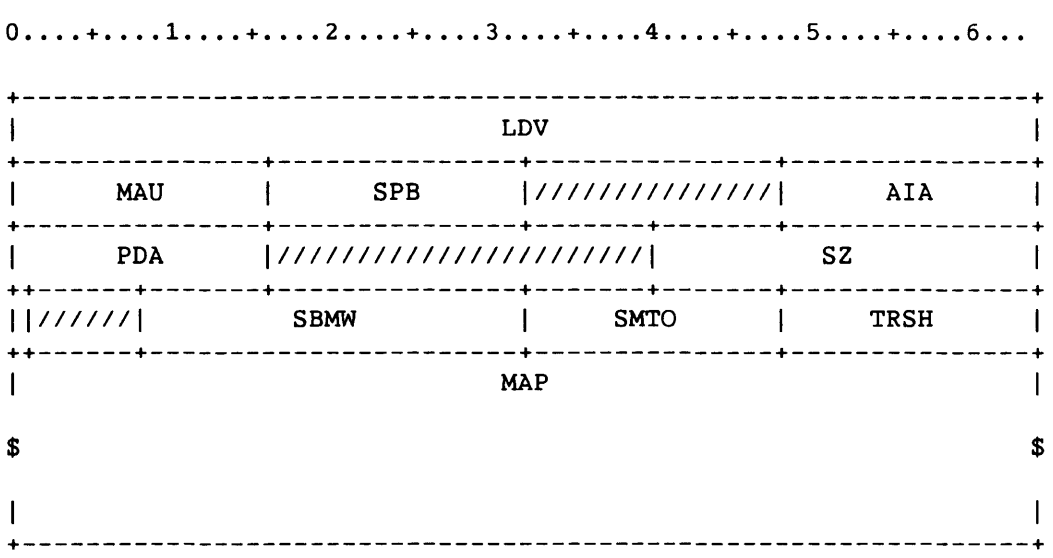


Figure A-6. Disk Reservation Table

Field	Word(base8)	Bits	Description
DRLDV	0	0-63	Logical device name
DRMAU	1	0-15	Maximum allocation units (AU) less flaws
DRSPB	1	16-31	SECTORS PER RESERVATION BIT
DRAIA	1	48-63	Total available AUs (number of unused bits)
DRPDA	2	0-15	Number of AUs used for permanent dataset
DRSZ	2	40-63	DRT map size in words

DR Disk Reservation Table - DRT

Field	Word(base8)	Bits	Description
DRSMAA	3	0	Space Manager already activated
DRSBMW	3	8-31	STP rel address to start map search
DRSMTO	3	32-47	Space Manager TXT ordinal
DRTRSH	3	48-63	Space Manager activation threshold, expressed as minimum available AIs
DRMAP	4-n	0-63	Bit map, one bit per track

ER F$ERCL parameter block - ERPB

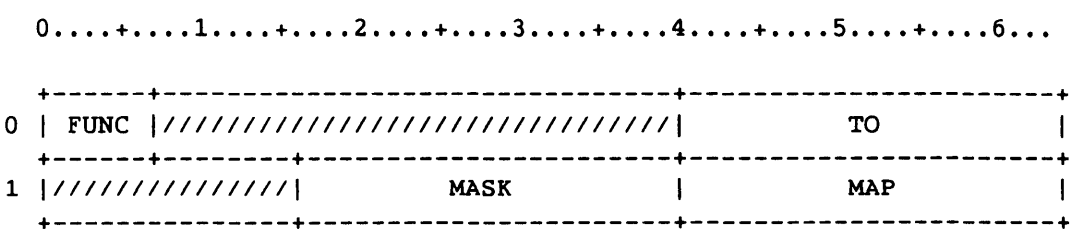


Figure A-7. F$ERCL Parameter Block

Field	Word(base8)	Bits	Description
ERFUNC	0	0-6	Subfunction code

The functions range from ERCL$$MI to ERCL$$MA-1. When subfunctions are added adjust the ERCL$$ symbols as needed.

ERCL$DIS=01	Disable event monitoring
ERCL$ENA=02	Enable event monitoring
ERCL$RCL=03	Recall untill event
ERCL$RET=04	Return occurred-events map
ERCL$$MI=01	minimum subfunction
ERCL$$MA=05	maximum subfunction+1

ERTO	0	40-63	Timeout value (milliseconds)
ERMASK	1	16-39	Event selection mask

ERCL$$ values must be changed when new events are added.
Bits zero thru ERM$$MAX-1 must always be defined.
Bits ERM$$FP thru ERM$$LP-1 must always be defined.

```
        ERMSIJ      1       16     Inter-job message arrived
        ERMSUO      1       17     Unsolicited oper msg arrived
        ERMSOR      1       18     Operator reply arrived
        ERMSIP      1       19     IPC request done
        ERMSSE      1       20     Site defined event (for local code)
                                     ERM$$MAX=D'20+1    Last non-
                                                        privileged bit+1
                                     ERM$$FP=D'26 First privileged bit
        ERMSCH      1       26     Channel function done
        ERMSIQ      1       27     SDT placed in INPUT queue
        ERMSOQ      1       28     SDT placed in OUTPUT queue
        ERMSAE      1       29     Archiving System Event
                                     ERM$$LP=D'29+1    Last
                                                       privileged bit+1

      ERMAP         1    40-63   Occurred-events map
        ERMPIJ      1       40     Inter-job message arrived
        ERMPUO      1       41     Unsolicited oper msg arrived
        ERMPOR      1       42     Operator reply arrived
        ERMPIP      1       43     IPC request done
        ERMPSE      1       44     Site defined event (for local code)
        ERMPCH      1       50     Channel function done
        ERMPIQ      1       51     SDT placed in INPUT queue
        ERMPOQ      1       52     SDT placed in OUTPUT queue
        ERMPAE      1       53     Archiving System Event

                                   On return from F$ERCL,
                                   S0 can have the following
                                   values.

                                     ERER=00   Okay
                                     ERER$MT=01   Prohibited to
                                                  multitasking job
                                     ERER$PV=02   Not a privileged job
                                     ERER$BFN=03  Bad function
                                     ERER$UDB=04  Mask contains
                                                  undefined bits
                                     ERER$MDI=05  Monitoring not enabled
```

IJ F$IJMSG Parameter Block - IJPB

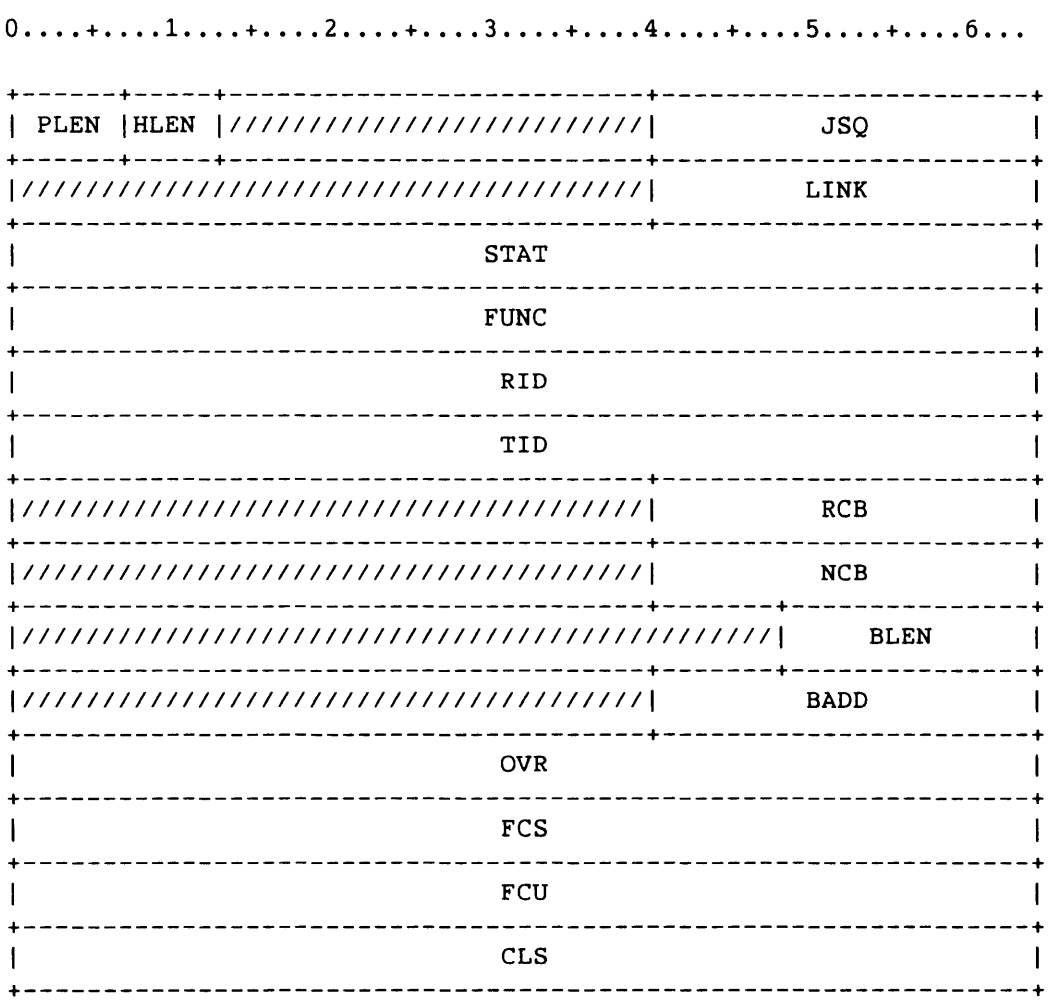


Figure A-8. F$IJMSG Parameter Block

IJ F$IJMSG Parameter Block - IJPB

Field	Word(base8)	Bits	Description
IJPLEN	0	0-6	length of the parameter block
IJHLEN	0	7-12	message buffer header length (LH@MHB)
IJJSQ	0	40-63	JSQ of target (OPEN, ACCEPT, REJECT)
IJLINK	1	40-63	link to next parameter block
IJSTAT	2	0-63	status

IJMS$OK=00 completed with no error

The following responses do not terminate a request chain.
If any values are changed, SYSLIB must be changed also.

IJMS$AR=01 ID is already receptive
IJMS$AU=02 ID is in use
IJMS$BA=03 buffer address or length bad
IJMS$BN=04 NCB is bad
IJMS$BNA=05 NCB address is bad
IJMS$BP=06 path is busy
IJMS$HL=07 HLEN error
IJMS$IF=08 IPT full
IJMS$INR=09 ID not registered
IJMS$INS=10 ID not specified
IJMS$MC=11 bad log message class
IJMS$ML=12 bad message length
IJMS$NA=13 ID is not attached
IJMS$NE=14 path is not open
IJMS$NO=15 no outstanding open request
IJMS$NP=16 path does not exist
IJMS$NR=17 ID is not receptive
IJMS$OO=18 outstanding OPEN was found
IJMS$PE=19 path is already established
IJMS$PF=20 memory pool is full
IJMS$PR=21 ID is privileged
IJMS$RB=22 bad RCB address
IJMS$RF=23 RIT full

IJMS$TA=24	target's buffer address is bad
IJMS$TL=25	target's buffer length is bad

The following responses terminate a request chain.

IJMS$BE=32	IJPB length error
IJMS$BF=33	undefined function
IJMS$LA=34	bad link address
IJMS$MT=35	more than one active TXT
IJMS$NC=36	RIT or IPT has zero entries
IJMS$PV=37	privileged function
IJMS$TP=38	more than I@MPBS parameter blocks
IJMS$MAX=39	maximum status value + 1

IJFUNC 3 0-63 subfunction code

If any values are changed, SYSLIB must be changed also.

IJM$NOP=00	no op
IJM$REC=01	request receptivity state
IJM$OPEN=02	open a communication path
IJM$ACCE=03	accept an IJM$OPEN request
IJM$REJE=04	reject an IJM$OPEN request
IJM$SNDM=05	send a message
IJM$CLOS=06	close a communication path
IJM$END=07	ends the receptivity state
IJM$$HNP=07+1	maximum value + 1 of unprivileged subfunctions

IJM$$MIP=32 minimum privileged function value
IJM$SNDL=32 send a logfile message (privileged)
IJM$$MAX=32+1 maximum subfunction value + 1

IJRID	4	0-63	ID of the requesting job
IJTID	5	0-63	ID of the target job
IJRCB	6	40-63	RCB address
IJNCB	7	40-63	NCB address
IJBLEN	10	48-63	message buffer length
IJBADD	11	40-63	message buffer address
IJOVR	12	0-63	log message over-ride flag
IJFCS	13	0-63	log message to system log
IJFCU	14	0-63	log message to user log
IJCLS	15	0-63	log message class

NC Node Control Block - NCB

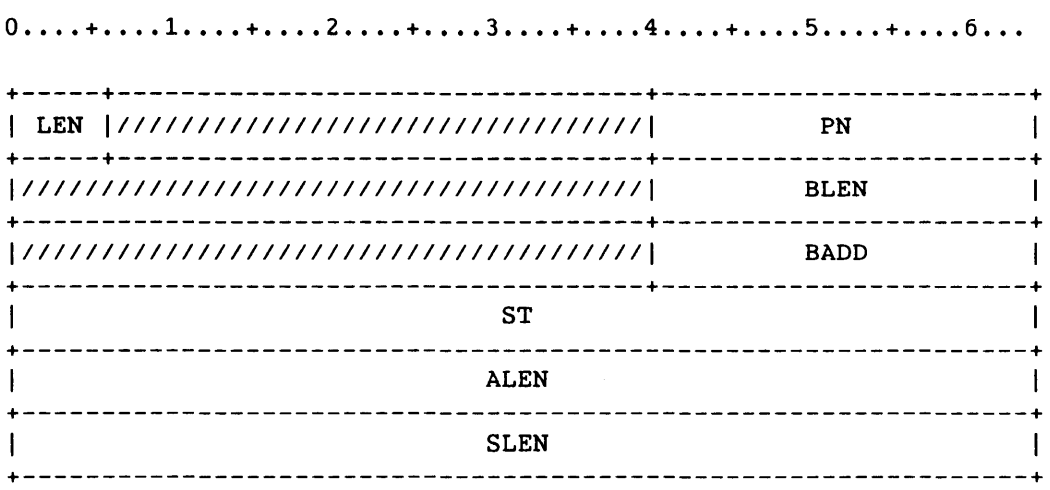


Figure A-9. Node Control Block

Field	Word(base8)	Bits	Description
NCLEN	0	0-5	length of the NCB (L@NCB)
NCPN	0	40-63	IPT offset for this path
NCBLEN	1	40-63	length of the node buffer
NCBADD	2	40-63	address of the node buffer
NCST	3	0-63	status
NCMS	3	0	message status
NCOS	3	48-63	open status

If any values are changed, SYSLIB must be changed also.

NCB$ACC='AC'R	open request accepted
NCB$REJ='RJ'R	open request rejected
NCB$CLO='CL'R	path was closed

NCALEN	4	0-63	length of message put into buffer

Field	Word(base8)	Bits	Description
NCSLEN	5	0-63	length of the message sent L@NCMH=2 length of the message header

RCB Receptive Control Block   - RCB

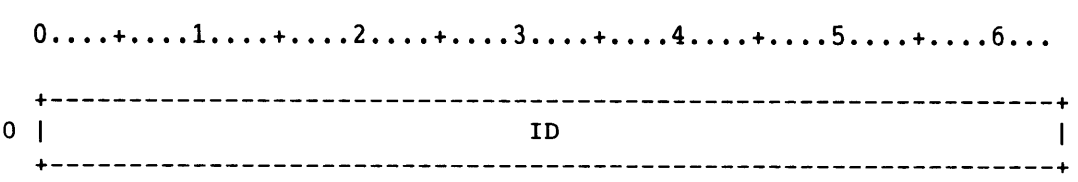


Figure A-10. Receptive Control Block

Field	Word(base8)	Bits	Description
RCBID	0	0-63	ID of the job requesting connection

MH Inter-job Communication Message Buffer - MHB

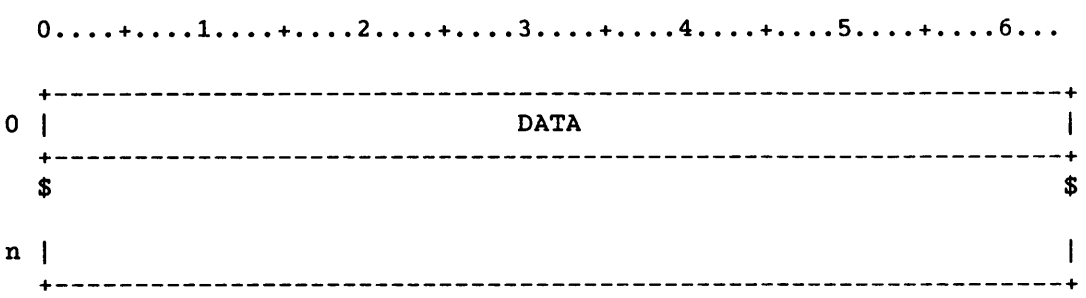


Figure A-11. Inter-job Communication Message Buffer

Field	Word(base8)	Bits	Description
MHDATA	0	0-63	first word of the message The message length is defined by fields in the NCB
MHLAST	0-n	0-63	last word of the message

The 1-word JCL Block Information Table (JBI) is generated in the user field and has two formats: one for conditional information (figure JB-1) and the other for iterative information (figure JB-2).

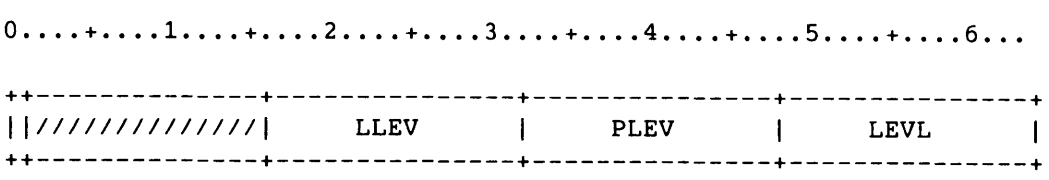


Figure A-12. JBI Conditional Format

Field	Word(base8)	Bits	Description
JBEXC	0	0	Conditional sequence is in execution
JBLLEV	0	16-31	Conditional is contained in this iterative nesting level
JBPLEV	0	32-47	Iterative is contained in this procedure level
JBLEVL	0	48-63	Current iterative nesting level

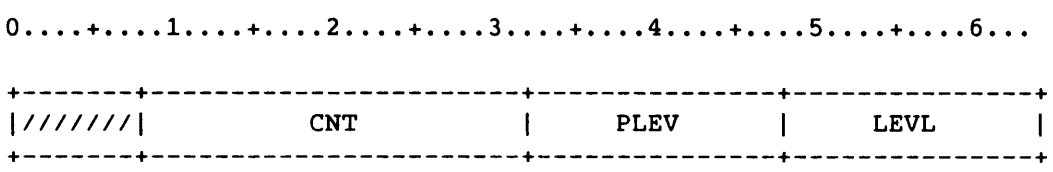


Figure A-13. JBI Iterative Format

Field	Word(base8)	Bits	Description
JBCNT	0	8-31	Iteration count
JBPLEV	0	32-47	Iterative is contained in this procedure level
JBLEVL	0	48-63	Current iterative nesting level

The first 128 words of each user field comprise the Job Communication Block. The JCB is accessible to the user.

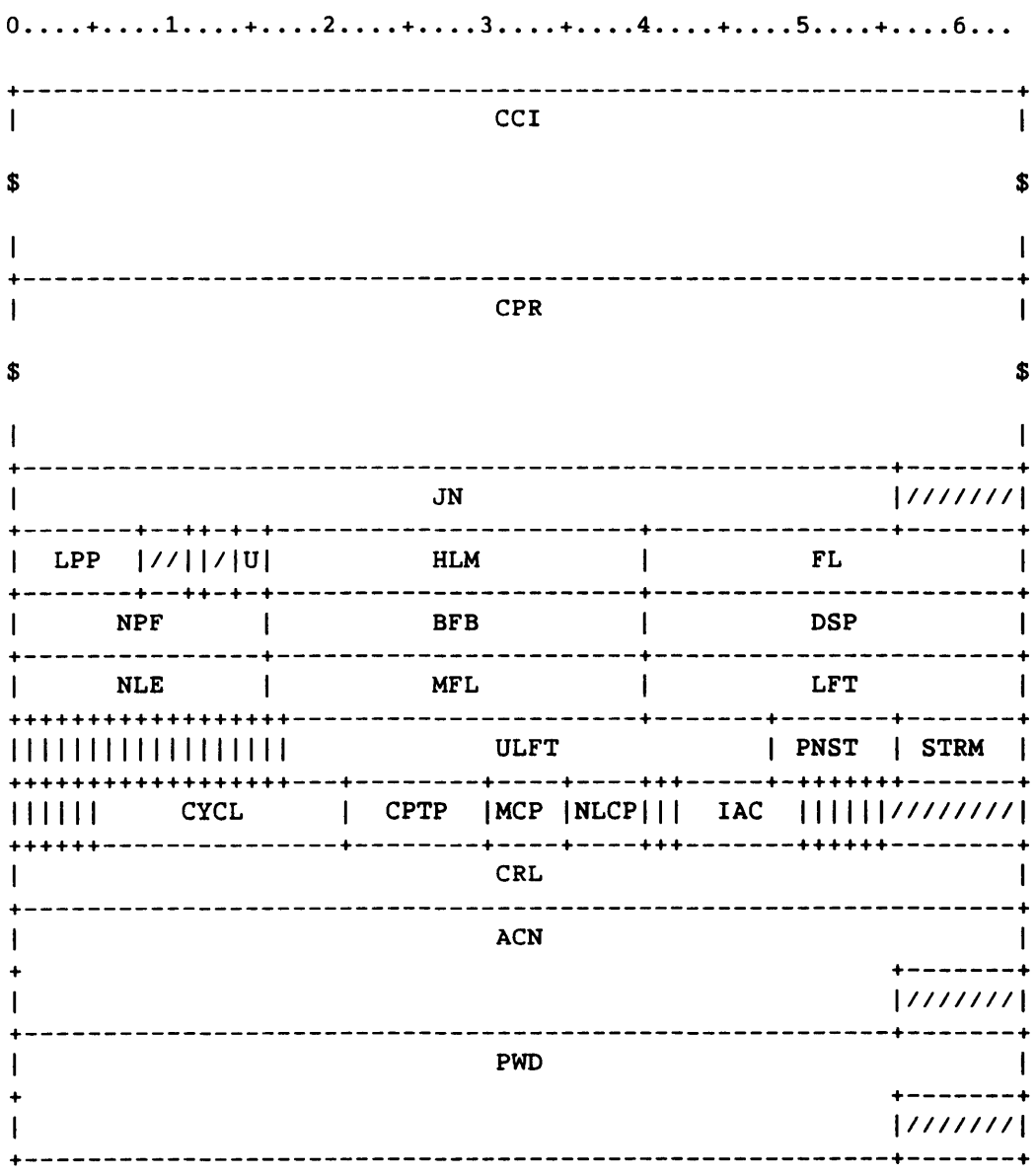


Figure A-14. Job Communication Block

JC Job Communication Block   - JCB

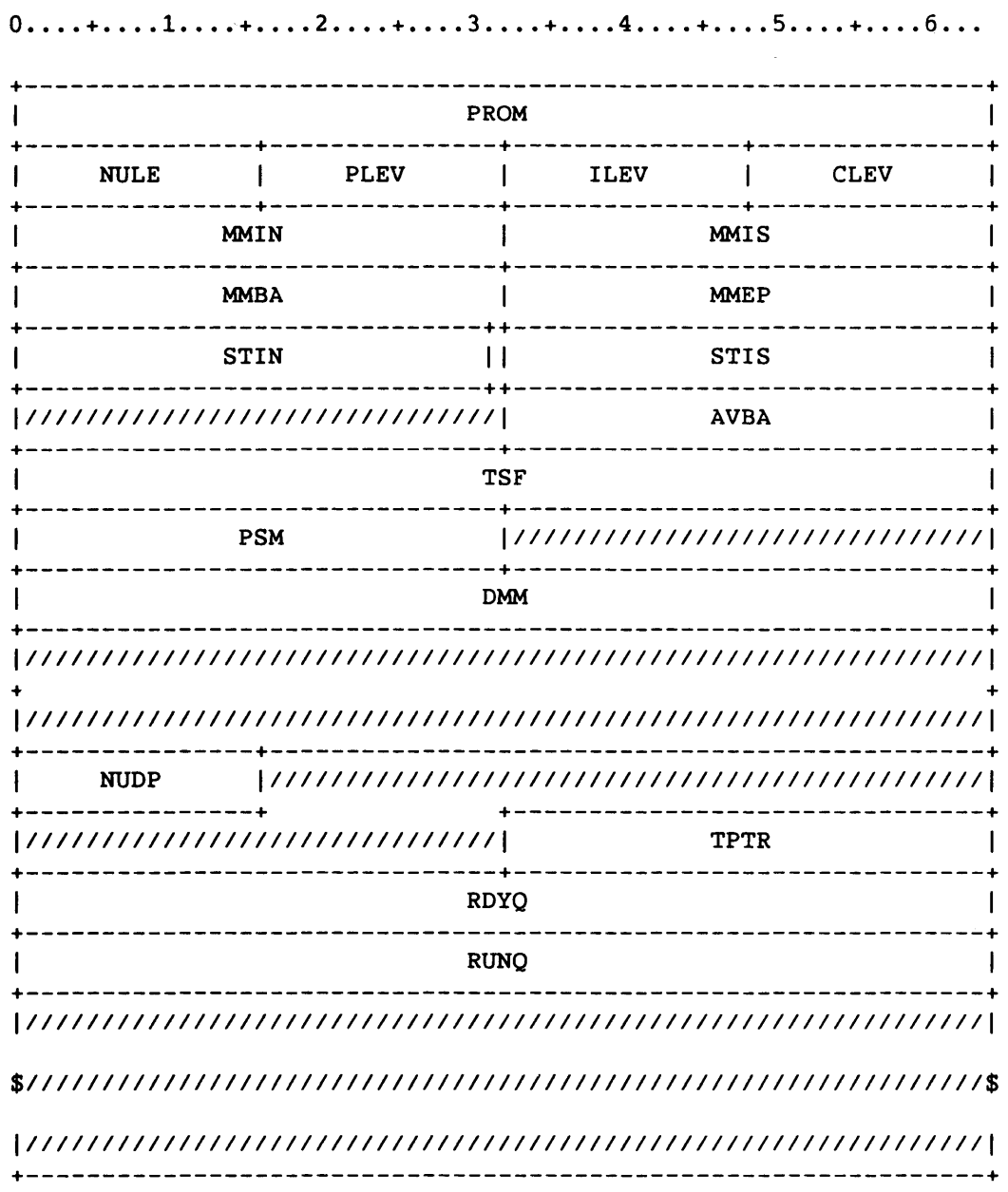


Figure A-14. Job Communication Block

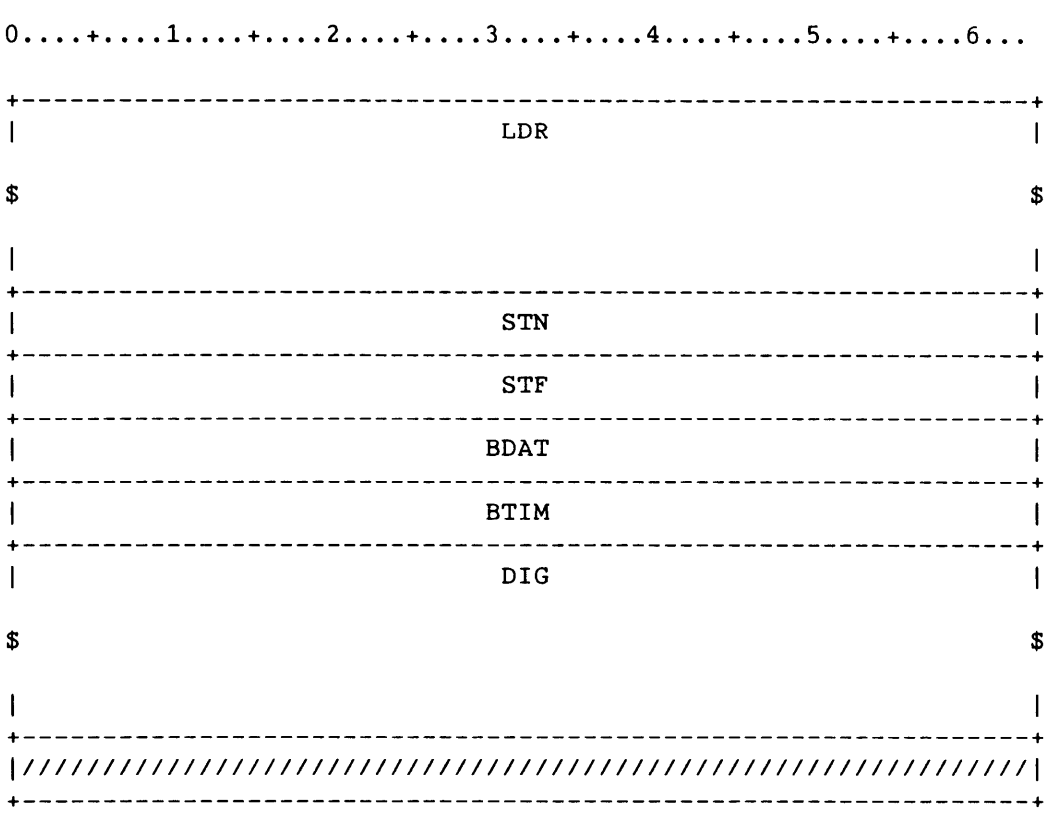


Figure A-14. Job Communication Block

B@JCB=0 Symbol for JCB base, relative to BA

The first five words of the JCB are assigned as a save area for the BGN table that is used by F$BGN.

Field	Word(base8)	Bits	Description
JCCCI	5-17	0-63	Control statement image packed 8 characters per word
JCCPR	20-77	0-63	Control statement parameters, expanded to two words per parameter
JCJN	100	0-55	Job name; bits 56-63 must be 0.
JCLPP	101	0-7	Lines per page

Field	Word(base8)	Bits	Description
JCRMSG	101	11	RFL messave sent
JCU	101	14-15	User mode indicator:
JCUL	101	14	Local
JCUG	101	15	Global
JCHLM	101	16-39	High limit of user code
JCFL	101	40-63	Current field length
JCNPF	102	0-15	Number of physical buffers and datasets
JCBFB	102	16-39	Base address of I/O buffers
JCDSP	102	40-63	Base address of DSP area
JCNLE	103	0-15	Number of entries in LFT
JCMFL	103	16-39	Maximum FL allowed
JCLFT	103	40-63	Base of LFT
JCDCS	104	0	CSP dynamic control statement flag
JCCSDB	104	1	CSP debug flag
JCBP	104	2	JOB statement breakpoint (BP) flag
JCNTB	104	3	CSP traceback suppression flag
JCIOAC	104	4	I/O area current status flag: 0 User's I/O area is unlocked 1 User's I/O area is locked
JCIOAP	104	5	I/O area previous status flag: 0 User's I/O area is unlocked 1 User's I/O area is locked
JCIA	104	6	Interactive flag
JCCHG	104	7	Execute CHARGES utility for trailer message.

Field	Word(base8)	Bits	Description
JCJBS	104	8	JOB statement flag (if set, JOB statement just processed)
JCCSIM	104	9	Flag is set when CRAY-1 simulator is running.
JCDLIT	104	10	Display literal delimiters in control statement crack.
JCRPRN	104	11	Retain level 1 parentheses.
JCVSEP	104	12	Last character was valid separator.
JCSDM	104	13	NOECHO of current control statement
JCPDMS	104	14	Suppress PDM user logfile messages
JCCSQ	104	15	New CFT calling sequence in effect
JCOVT	104	16	Overlay type
JCULFT	104	17-47	Base of user LFTs (JCB-REL)
JCPNST	104	48-55	Parentheses nesting level for current control statement
JCSTRM	104	56-63	Statement termination for current control statement
JCEFI	105	0	Enable floating-point interrupt flag; used by $ARLIB math routines to reset floating-point interrupt flag
JCOVL	105	1	Overlay flag
JCSBC	105	2	SBCA flag
JCBDM	105	3	Enable bidirectional mode flag
JCORI	105	4	Interrupt on operand range flag
JCCYCL	105	5-20	CPU cycle time, in picoseconds

Field	Word(base8)	Bits	Description
JCCPTP	105	21-29	CPU type, @CRAYxxx
JCMCP	105	30-34	Maximum number of logical CPUs that
JCNLCP	105	35-39	Current number of logical CPUs asg'd
JCEMA	105	40	1=Extended memory addressing enabled
JCAVL	105	41	1=Additional vector logical unit enab.
JCIAC	105	42-49	Number of account processing retries allowed for an interactive job
JCACRQ	105	50	Accounting mandatory flag
JCPWRQ	105	51	Password mandatory flag
JCRYPT	105	52	Encryption flag
JCSLVL	105	53	Security level flag
JCSJOB	105	54	S on job card
JCCRL	106	0-63	COS revision level
JCCRLS	106	32-63	COS revision number
JCACN	107-110	0-63	1 through 15 character account number
JCACN1	107	0-63	Characters 1 through 8 of account number
JCACN2	110	0-55	Characters 9 through 15 of account number
JCPWD	111-112	0-63	1 through 15 character password
JCPWD1	111	0-63	Characters 1 through 8 of password
JCPWD2	112	0-55	Characters 9 through 15 of password

Field	Word(base8)	Bits	Description
JCPROM	113	0-63	Current user job interactive prompt, justified, zero-filled. 64 bits of binary zeroes disables user job prompt. Set to system default at beginning of each job step.
JCNULE	114	0-15	Number of user LFT entries (below HLM)
JCPLEV	114	16-31	Current procedure nesting level
JCILEV	114	32-47	Current iterative nesting level
JCCLEV	114	48-63	Current conditional nesting level
The next four words are used by the run-time memory manager:			
JCMMIN	115	0-31	Size of increments to the managed memo
JCMMIS	115	32-63	Initial size of memory to be managed
JCMMBA	116	0-31	Base address of managed space
JCMMEP	116	32-63	Size of smallest block added to availa
JCSTIN	117	0-30	Size of increments to a stack
JCSTRT	117	31	Flag to indicate stack for root task
JCSTIS	117	32-63	Initial size of a stack
JCAVBA	120	32-63	Base of available space
JCTSF	121	0-63	Task scheduling flag
JCPSM	122	0-31	Pseudo semaphore registers 1 A&B, 1/S
JCDMM	123	0-63	Don't move memory when nonzero
JCNUDP	126	0-15	Number of system DSPs in user
JCTPTR	127	32-63	Pointer to list of all tasks
JCRDYQ	130	0-63	Multitasking ready queue header

Field	Word(base8)	Bits	Description
JCRUNQ	131	0-63	Multitasking run queue header
JCLDR	146-163	0-63	Unsatisfied externals
JCSTN	164	0-63	Job step count
JCSTF	165	0-63	Job step failure flag
JCBDAT	166	0-63	Date of absolute load module generation
JCBTIM	167	0-63	Time of absolute load module generation
JCDIG	170-177	0-63	Reserved for diagnostics

The presence of this figure adds no information. It is required by the table diagram generator to improve the appearance of the table while still supplying the S@JCDIG and N@JCDIG tags.

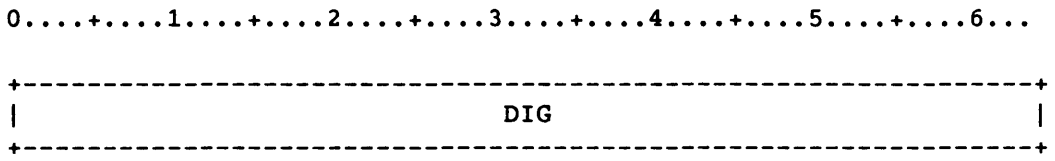


Figure A-15. Additional Tags for Diagnostics

Field	Word(base8)	Bits	Description
JCDIG	170	0-63	

The 4-word JST contains information about system and user symbols.

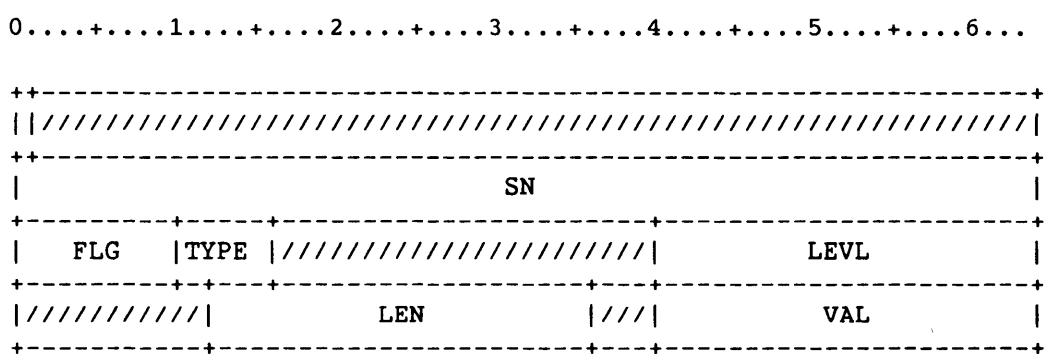


Figure A-16. JCL Symbol Table

Field	Word(base8)	Bits	Description
JSCRE	0	0	Create if not found. Available only for system use.
JSSN	1	0-63	Symbol name
JSFLG	2	0-9	Symbol flag fields
JSLOC	2	0	Local or global. If set, symbol is procedure local.
JSCON	2	1	Constant or variable. If set, symbol is constant.
JSSRS	2	2	System reserved. If set, the symbol name is reserved by system.
JSUSR	2	3	User settable. If set, symbol may be modified by the job.
JSSYS	2	4	System settable. If set, the symbol may be modified by COS.
JSTYPE	2	10-15	One of the following symbol types: SYMTUND=O'00 Undefined - no type SYMTBOO=O'01 Boolean - logical SYMTINT=O'02 Decimal integer SYMTLIT=O'03 ASCII literal; 1-8 characters. SYMTBIN=O'04 Binary
JSLEVL	2	40-63	Procedure definition level

JS JCL Symbol Table - JST

Field	Word(base8)	Bits	Description
JSLEN	3	12-35	Length of value
JSVAL	3	40-63	Base of value buffer

* Job Table Area (JTA)

The Job Table Area records all information about a job which needs to be present whenever the job is rolled into memory.

There is a fixed portion, followed by a memory pool which holds entries allocated as the jobs needs grow.

Figure A-17 shows the JTA. The display of field JTDTM is in error. JTDTM is shown as one word, while it in fact occupies the apparently undefined words below it as well.

Figure A-18 shows the detailed structure of the user breakpoints (JTBKP).

Figure A-19 shows the detailed structure of the pointer fields within the memory pool areas for the JTA DNTs.

Figures A-20 and A-21 provide additional tags for the JTUSR and JTGRN fields. They provide no additional information and exist only for the convenience of the table diagram generator.

Assumed sizes of other tables referenced.

LE@SCTR	=	D'512	Disk sector length in words
C@CLSIZE	=	D'17	XMP cluster register save area size
LH@DNT	=	D'01	Length of DNT linkage word
LE@DNTSK	=	D'40	I/O stack length

JT Job Table Area   - JTA

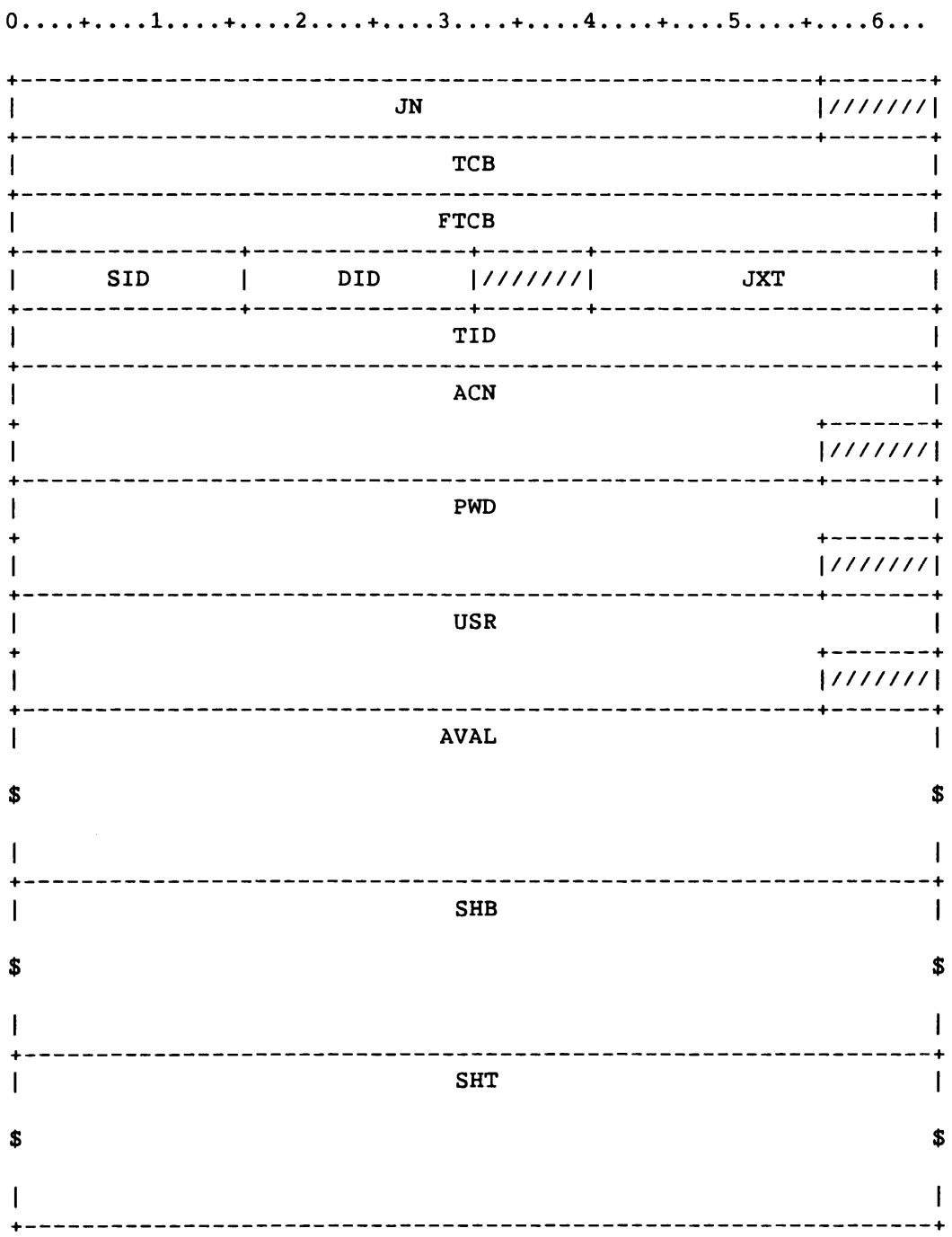


Figure A-17. Job Table Area

JT Job Table Area   - JTA

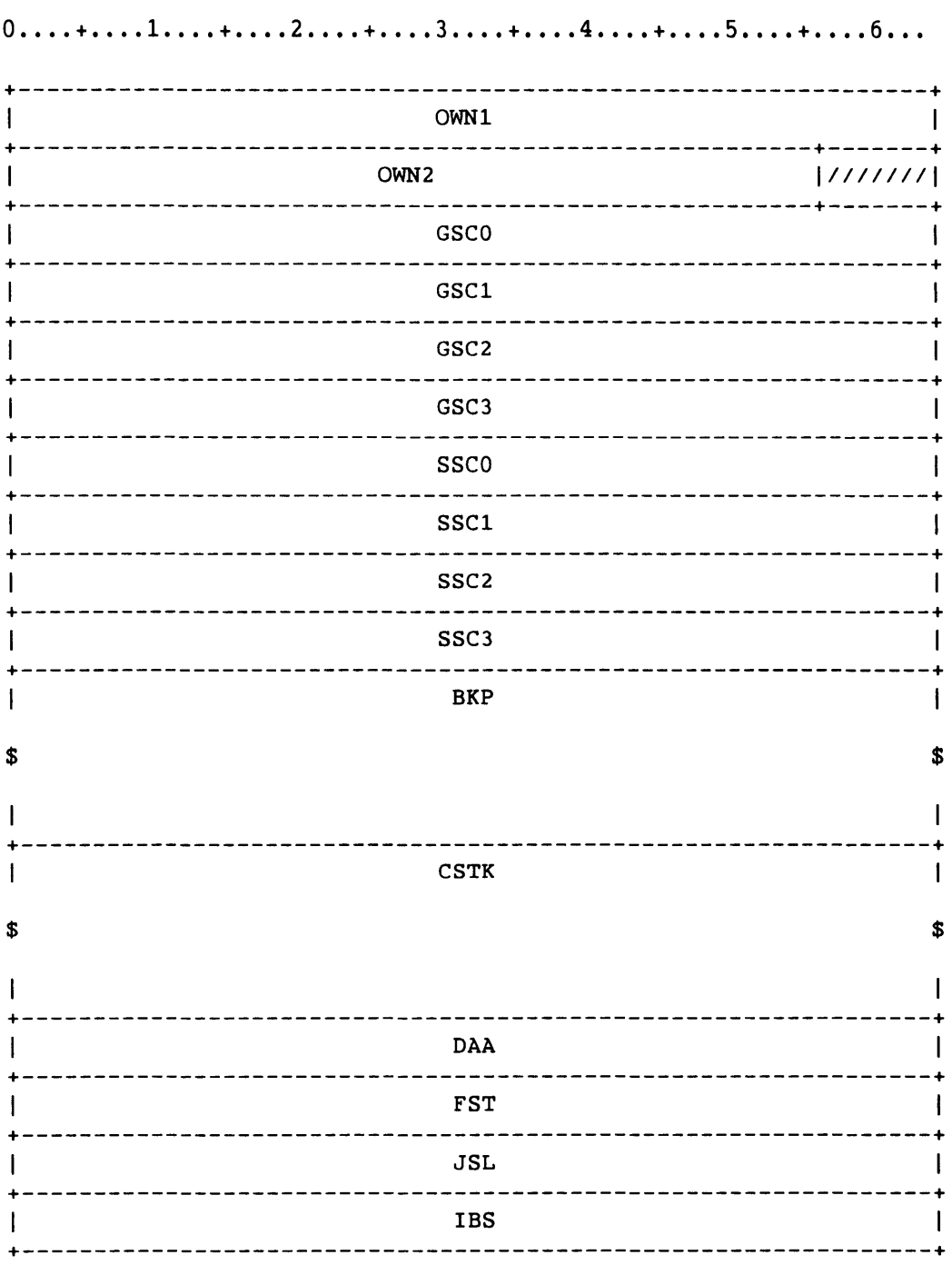


Figure A-17. Job Table Area

JT Job Table Area   - JTA

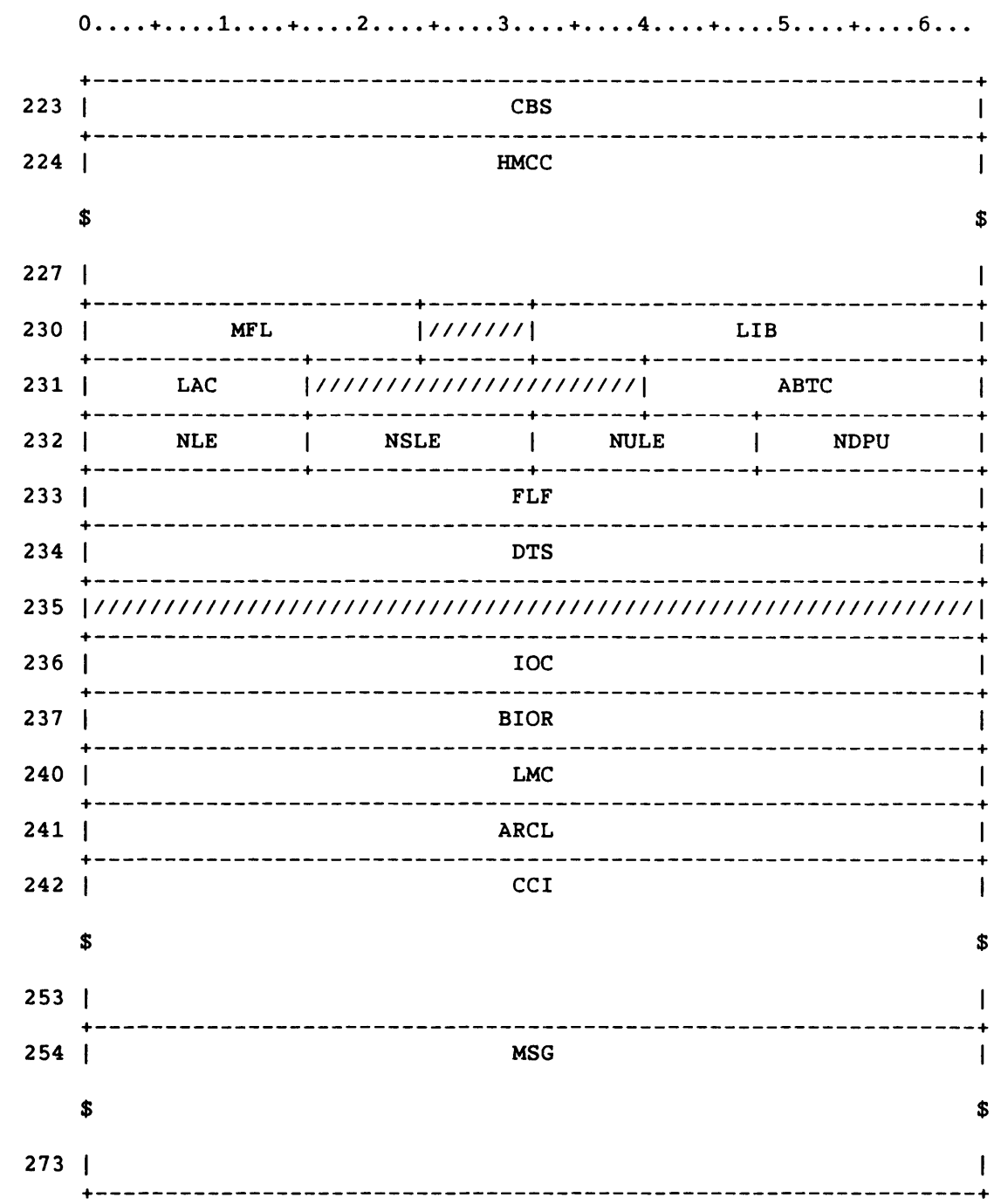


Figure A-17. Job Table Area

JT Job Table Area   - JTA

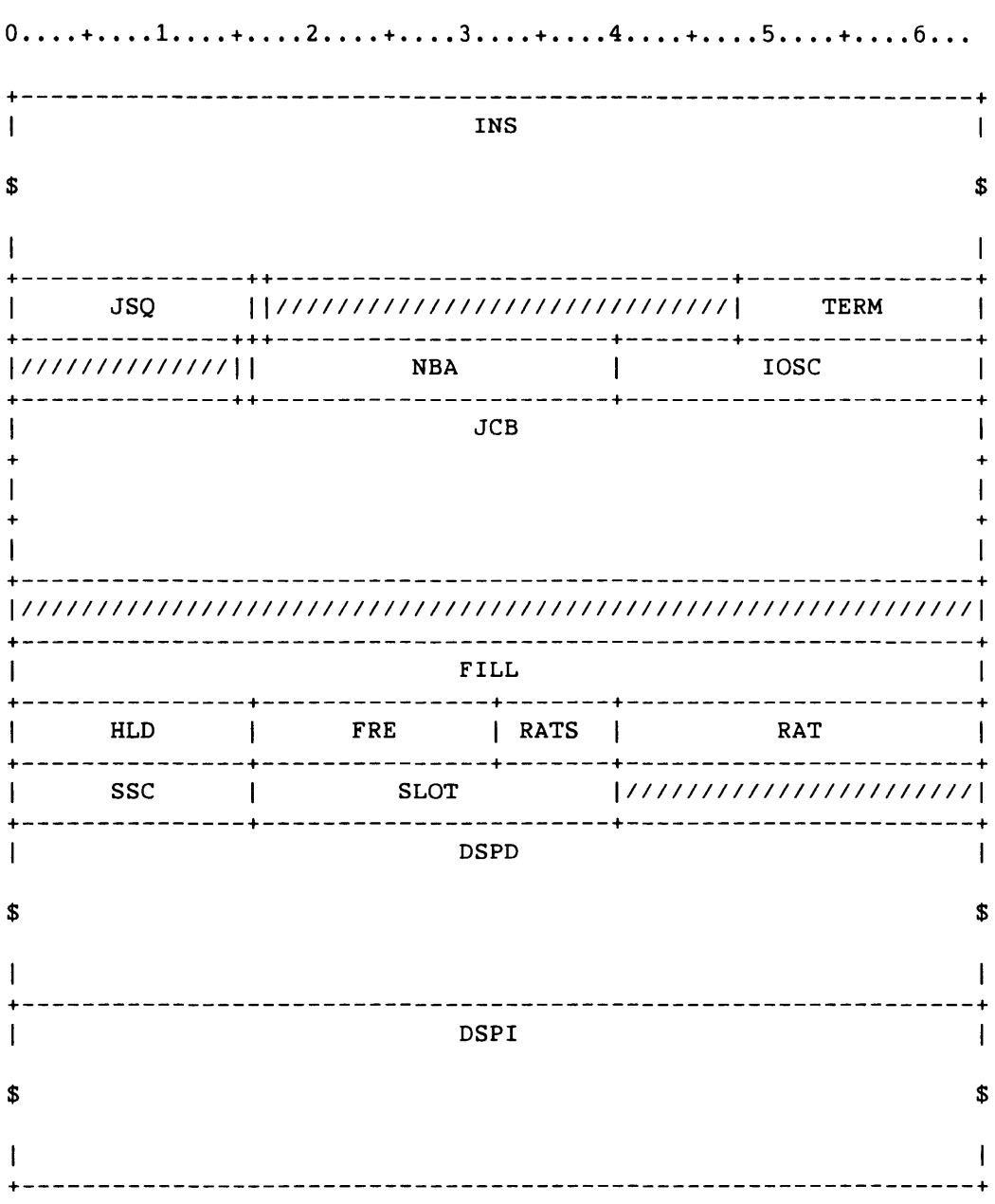


Figure A-17. Job Table Area

JT Job Table Area - JTA

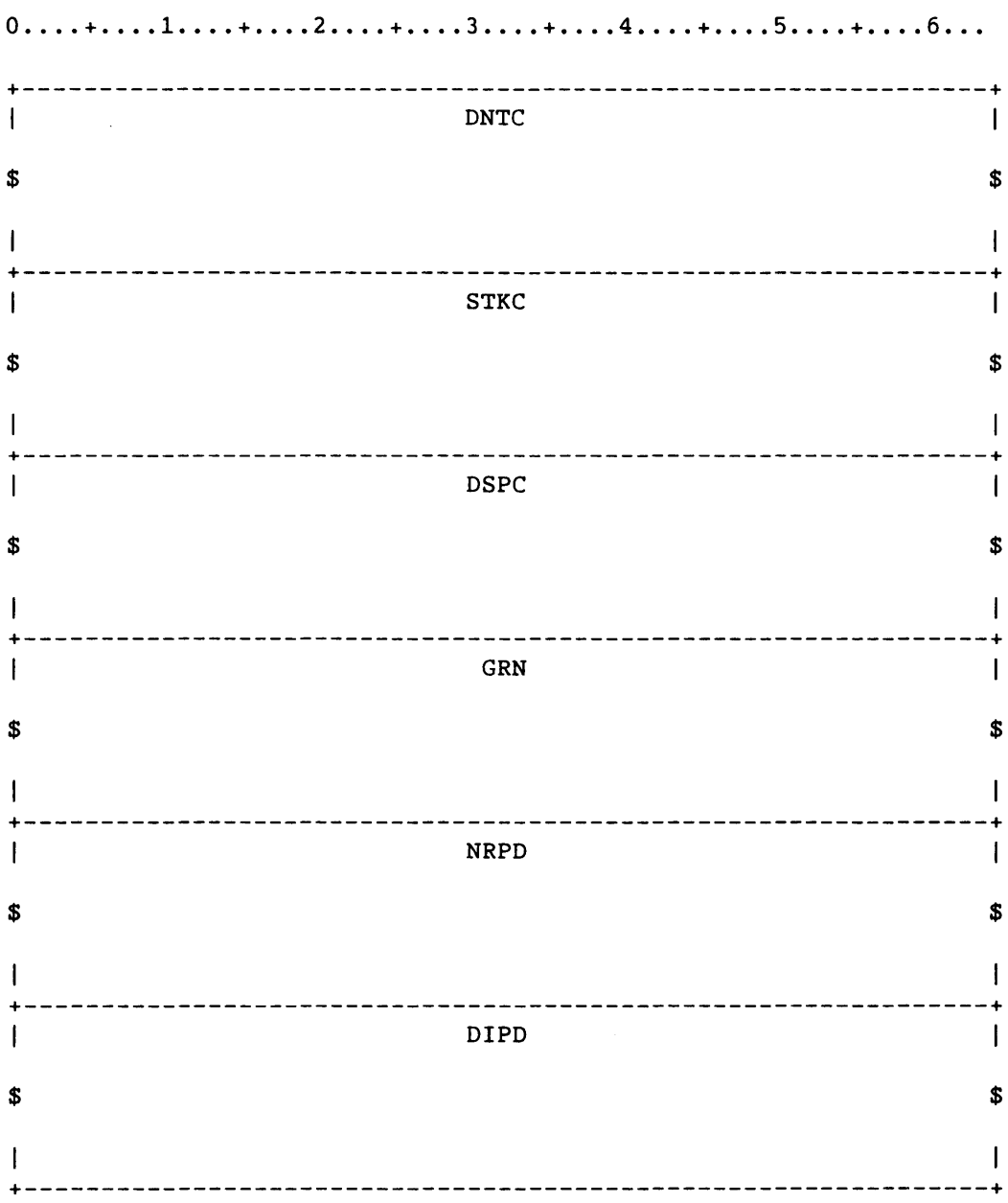


Figure A-17. Job Table Area

JT Job Table Area - JTA

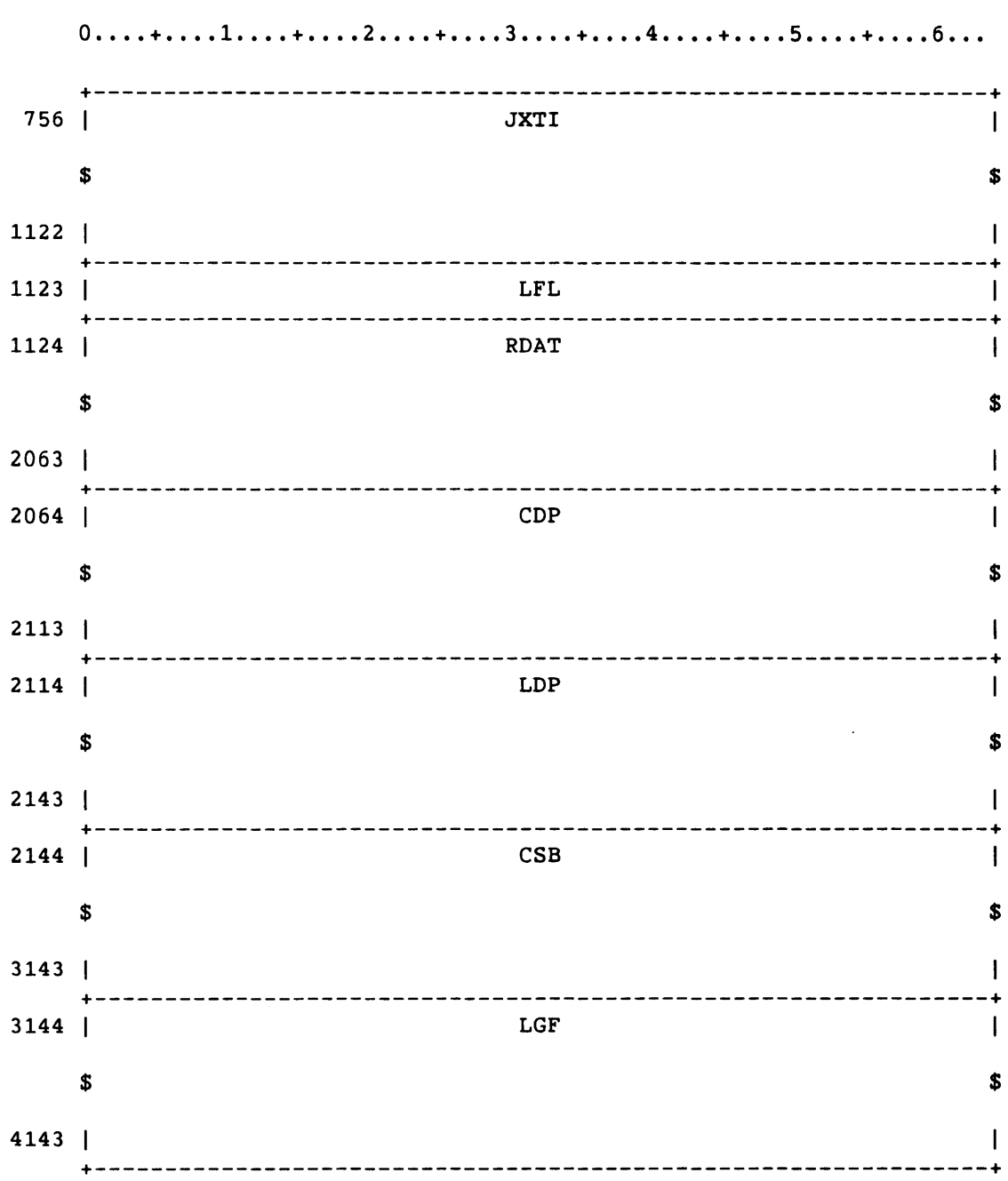


Figure A-17. Job Table Area

JT Job Table Area   - JTA

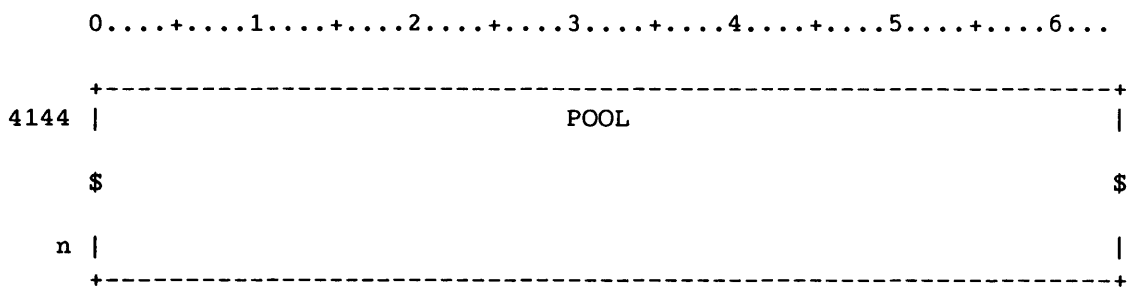


Figure A-17. Job Table Area

Identifying information.

Field	Word(base8)	Bits	Description
JTJN	0	0-55	Job name
JTTCB	1	0-63	JTA offset of first TCB
JTFTCB	2	0-63	JTA offset of free TCB chain
JTSID	3	0-15	Two character source ID
JTDID	3	16-31	Two character destination ID
	3	32-39	Reserved for expansion of JTJXT
JTJXT	3	40-63	Address of JXT entry
JTTID	4	0-63	Terminal ID
JTACN	5-6	0-63	Fifteen character account number
JTACN1	5	0-63	First eight characters
JTACN2	6	0-55	Last seven characters
JTPWD	7-10	0-63	Fifteen character password:
JTPWD1	7	0-63	First eight characters
JTPWD2	10	0-55	Last seven characters
JTUSR	11-12	0-63	Fifteen character user number

JT Job Table Area - JTA

Field	Word(base8)	Bits	Description
JTUSR1	11	0-63	First eight characters
JTUSR2	12	0-55	Last seven characters
			The following fields contain ASCII field names plus the values of the symbols for aid in debugging and for DUMP.
JTAVAL	13-144	0-63	L@JTAVAL=D'64
			Job statistics. These are aggregate task statistics.
JTTSX	113	0-63	Time spent executing (cycles)
JTTSWS	114	0-63	Time spent waiting semaphore (Cycles)
JTDTSX	115	0-63	Sum of all deleted tasks' time spent executing
JTTSW	116	0-63	Time spent waiting to execute(cycles)
JTTSD	117	0-63	Time spent waiting for I/O completion
JTXMI	120	0-63	(CPU time)*(memory size) floating
JTDMI	121	0-63	(I/O wait time)*(memory size) floating
JTSMI	122	0-63	(Wait sem) * (Memory size) floating
JTIOB	123	0-63	Disk sectors transferred
JTIOF	124	0-63	FSS sectors transferred
JTIOR	125	0-63	User I/O requests made
JTBIOC	126	0-63	Number of F$BIO requests made
JTIOS	127	0-63	Number of I/O suspend requests to CIO
JTDLI	130	0-63	Count of deadlock interrups for job
JTMXM	131	0-23	Maximum job size

JT Job Table Area - JTA

Field	Word(base8)	Bits	Description
JTMIM	131	24-47	Minimum job size
JTOPC	131	48-63	Number of open calls by user
JTPFA	132	0-23	Permanent file space accessed
JTPFS	132	24-47	Permanent file space saved
JTCLC	132	48-63	Number of close calls by user
JTBRF	133	0-23	No. of sectors received from front end
JTBSF	133	24-47	No. of sectors sent to front end
JTDDRO	133	48-63	Number of data to disk replies owed
JTTFS	134	0-23	Temporary file space used
JTMRD	134	24-39	Number of memory resident datasets
JTL	134	40-63	Length of job table area
JTMXFL	135	0-23	Maximum field length used
JTMIFL	135	24-47	Minimum field length used
JTMXJT	136	0-23	Maximum JTA used
JTMIJT	136	24-47	Minimum JTA used
Flags of every size and flavor.			
JTSEC	137	0	Security flag. CSP is executing
JTTLE	137	1	Initial time limit expired
JTADV	137	2	Job in advance
JTITRM	137	3	Intend to terminate
JTEOF	137	4	End of file on $CS
JTKIL	137	5	Job killed

JT Job Table Area - JTA

Field	Word(base8)	Bits	Description
JTRRN	137	6	Job rerun
JTEPD	137	7	DAT changed by DIA task
JTWUC	137	10	Waiting for user channel reply
JTIDP	137	11	Inhibit dumpjob processing
JTEXO	137	12	Execute only dataset open
JTRST	137	13	Reprieve processing set single thread
JTDLM	137	14	Disable log messages
JTTRM	137	15	Job in termination
JTABT	137	16	Job abort
JTCMSG	137	17	Enable conditional messages
JTLGFL	137	18	$LOG size exdeeded I@LGUSZ
JTJCBX	137	19	Bad JCB detected
JTSTAT	137	20	Request dataset statistics
JTTRM1	137	21	Second pass through TRM
JTCXDS	137	22	Close Execute-only datasets (EXP/ENDP)
JTINIT	137	23	Job initiated
JTNRR	137	24	Job not rerunnable
JTNRO	137	25	Disable no rerun
JTIPC	137	26	Set if F$IPC request made
JTOSUP	137	28	Interactive output suspended flag
JTISUP	137	29	Interactive input suspended flag
JTIA	137	30	Interactive flag

Field	Word(base8)	Bits	Description
JTSKP	137	31	Control statement skip flag
JTMAC	137	32	Move AC.NO./PW. to JTA flag
JTDNR	137	33	Device-not-ready flag
JTLPP	137	34-41	Lines per page
JTSDR	137	42	Module is from SDR
JTSSM	137	43	Module wants secure datasets
JTVFLG	137	44	Security violation occured flag
JTETRM	137	45	Internal termination flag
JTSCM	137	46	EXU control statement msg flag: 0=issued, 1=not issued
JTFUA	137	47	Force-unique-access (for AQR)
JTSSF	140	16-33	Subsystem feature flags
JTIJF	140	16	Set if any F$IJMSG function is used
JTIJC	140	18-33	Inter-job connection count
JTIRT	140	40-63	POINTER TO IRT CHAIN
JTVIO MUST BE A FULL WORD			
JTVIO	141	0-63	Number of security violations
Job-related reprieve information			
JTST	142	0-63	Reprieve status word
JTFEFW	143	0-63	Reprieve fatal error flags:
JTFE03	143	1	No DAT space
JTFE10	143	2	No disk space
JTFE11	143	3	System directory is full
JTFE23	143	4	Job time limit exceeded
JTFE24	143	5	Operator dropped user job
JTFE41	143	6	Enter allowed on access only
JTFEXX SUBFIELD 7,1			** UNASSIGNED **

JTFE51	143	8	LFT chain pointer invalid
JTFE43	143	9	User log size exceeded
JTFE94	143	10	HARDWARE ERROR WHILE WRITING $LOG
JTF260	143	11	Dataset not recoverable after offload
JTFENR	143	63	Not reprievable

Cluster registers for job.

JTSEM	144	0-31	Semaphore registers
JTSHB	145-154	0-63	Shared B registers
JTSHT	155-164	0-63	Shared T registers

Security information

JTOWN1	165	0-63	Dataset owner ID, characters 1-8
JTOWN2	166	0-55	Dataset owner ID, characters 9-15
JTGSC0	167	0-63	Global security flags
JTGSC1	170	0-63	
JTGSC2	171	0-63	
JTGSC3	172	0-63	
JTSSC0	173	0-63	Job step security flags
JTSSC1	174	0-63	
JTSSC2	175	0-63	
JTSSC3	176	0-63	

Breakpoint control information.

		L@JTBKP=D'8	Length of breakpoint information
JTBKP	177-206	0-63	User breakpoints
		MAXPRLVL=7	Maximum nesting level, with $CS
JTCSTK	207-216	0-63	Control statement file stack base

Field	Word(base8)	Bits	Description
JTDAA	217	0-63	Pointer to device name table
JTFST	220	0-63	Pointer to FSS accounting table
JTJSL	221	0-63	JCL symbol list chain control word
JTIBS	222	0-63	Iterative block stack chain control
JTCBS	223	0-63	Conditional block stack chain control
JTHMCC	224-227	0-63	Hardware perf.mon. chain control
JTATCC	225	0-63	Active TCB chain control
JTFTCC	226	0-63	Free TCB chain control
JTTACC	227	0-63	Task accounting chain control
JTMFL	230	0-23	Maximum FL
JTLIB	230	32-63	Library search JTA offset
JTLAC	231	0-15	Last abort code
JTABTC	231	40-63	Job step abort code (ABxxx)
JTNLE	232	0-15	Number of LFT entries in JTA
JTNSLE	232	16-31	NUMBER OF JTA LFTS WHICH POINT TO SYSTEM-AREA USER LFTS
JTNULE	232	32-47	NUMBER OF JTA LFTS WHICH POINT TO USER-AREA USER LFTS
JTNDPU	232	48-63	NUMBER OF USER-AREA SYSTEM DSPS
JTFLF	233	0-63	JTA offset of first link in LFT chain
JTDTS	234	0-63	RT clock at rollout
JTIOC	236	0-63	Count of active I/O requests/functions
JTBIOR	237	0-63	Number of active F$BIO requests

Field	Word(base8)	Bits	Description
JTLMC	240	0-63	Lock-in-memory counter
JTARCL	241	0-63	Recall-on-any user task bit map DNTMSK L@JTCCI=D'80/D'8 80 character buffer for control stmt L@JTMSG=D'128/D'8 128 character buffer for last $LOG msg L@JTINS=O'20 JTA installation words
JTCCI	242-253	0-63	Control statement being prescanned
JTMSG	254-273	0-63	Last logfile message issued
JTINS	274-313	0-63	Reserved for installation
JTJSQ	314	0-15	Job sequence number
JTTRM2	314	16	MSG flag to terminate job immediately
JTTERM	314	48-63	Termination status
JTMR	315	15	Outstanding memory request flag
JTNBA	315	16-39	New buffer address
JTIOSC	315	40-63	I/O suspend counter
JTJCB	316-320	0-63	JCB save area
JTCHLM	316	16-39	End of user code, JCB relative
JTCFL	316	40-63	Current field length, in words
JTCNDP	317	0-15	Number of DSPs in system area
JTCBFB	317	16-39	Base of system buffers, JCB relative
JTCDSP	317	40-63	Base of sytem DSPs, JCB relative
JTCNLE	320	0-15	Number of LFTs in system area
JTCMFL	320	16-39	Maximum field length, in words
JTCLFT	320	40-63	Base of system LFTs, JCB relative

JT Job Table Area - JTA

Field	Word(base8)	Bits	Description
JTFILL	322	0-63	
JTHLD	323	0-15	Implicit hold bit map
JTFRE	323	16-31	G.R. - first request encountered
JTRATS	323	32-39	Size if RAT save area
JTRAT	323	40-63	RAT save area in JTA pool
JTSSC	324	0-15	Station slot word count
JTSLOT	324	16-39	Station slog address (JTA-relative)

Allocate DNT/DSP space for the datasets that the system performs the I/O on.

JTDSPD	325-354	0-63	$DUMP Dataset parameter table (DSP)
JTDSPI	355-404	0-63	Submit dataset parameter table (DSP)
JTDNTC	405-445	0-63	$CSP Dataset Name Table (DNT)
JTSTKC	446-515	0-63	I/O stack for CSP reads
JTDSPC	516-545	0-63	$CSP Dataset parameter table (DSP)

Allocate space for various tables.

JTGRN	546-565	0-63	Pointers to G. R. accounting tables
JTNRPD	566-661	0-63	PDD for NORERUN
JTDIPD	662-755	0-63	PDD for diagnostic requests L@JTRDAT=D'16*D'30 Length of roll image DAT space
JTJXTI	756-1122	0-63	JXT image at time of rollout
JTLFL	1123	0-63	Last word of roll image
JTRDAT	1124-2063	0-63	DAT for roll dataset

Allocate the space for $CS and $LOG DSPs and circular buffers.

JT Job Table Area - JTA

Field	Word(base8)	Bits	Description
JTCDP	2064-2113	0-63	$CS Dataset parameter table (DSP)
JTLDP	2114-2143	0-63	$LOG Dataset parameter table (DSP)
JTCSB	2144-3143	0-63	$CS Circular buffer base
JTLGF	3144-4143	0-63	$LOG Circular buffer base

Dynamic area of JTA. Initialize with the DNTs for $CS (control statements) and $LOG (logfile messages).

JTPOOL	4144-n	0-63	First word of JTA pool, header word

The POOL initially contains DNTs for $CS and $LOG. These are not shown due to problems with the table diagram generator.

Detailed structure of user breakpoints

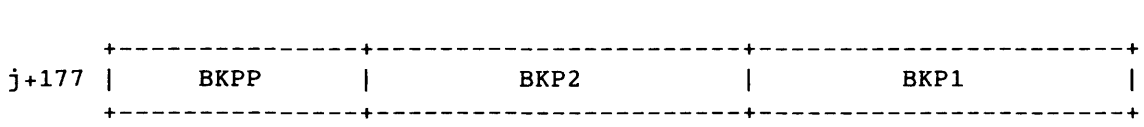


Figure A-18. JTA User Breakpoints

Field	Word(base8)	Bits	Description
JTBKPP	j+177	0-15	Contents of replaced parcel
JTBKP2	j+177	16-39	Breakpoint reset address
JTBKP1	j+177	40-63	Breakpoint address

DEFINE THE POINTER FIELDS WITHIN THE MEMORY POOL
AREAS FOR THE JTA DNT'S

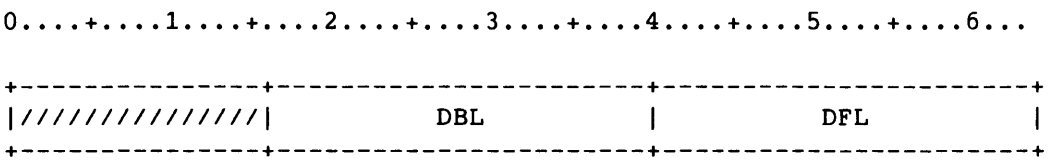


Figure A-19. JTA DNTs

Field	Word(base8)	Bits	Description
JTDBL	0	16-39	DNT BACKWARD LINK
JTDFL	0	40-63	DNT FORWARD LINK

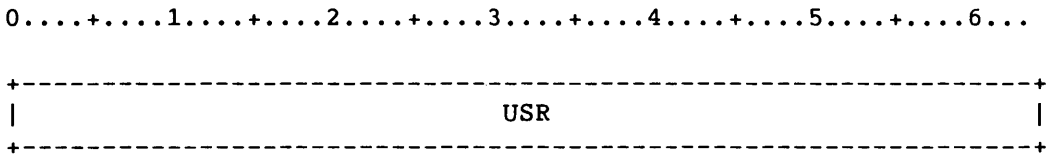


Figure A-20. Provide Tags for JTUSR

Field	Word(base8)	Bits	Description
JTUSR	11	0-63	

JT Job Table Area   - JTA

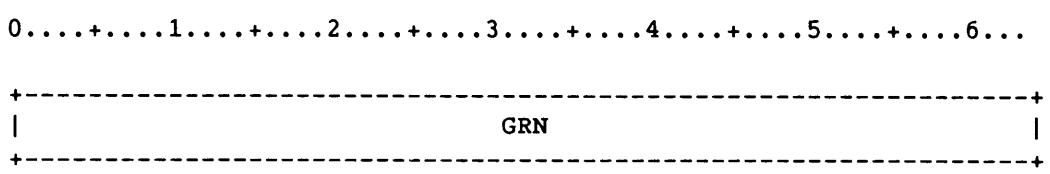


Figure A-21. Provide Tags for JTGRN

Field	Word(base8)	Bits	Description
JTGRN	546	0-63	

The Label Definition Table describes the tape label, and consists of four parts: the LDT header, volume header, header which points to the other entries, these entries are optional and can appear anywhere after the header. The following conditions must be met for constructing a Label Definition Table (LDT):

- The header must be present.
- The header must precede each entry.
- Each entry must be pointed to by the offset value in the LDT header. Zero is used for absent fields.
- The lengths of the whole LDT and of each entry must be set in the proper fields.
- The length value for volume 1 must be at least the length of the entire first VSN. The length value for either header 1 or header 2 must be at least the defined length of the respective entry.

LD Label Definition Table - LDT

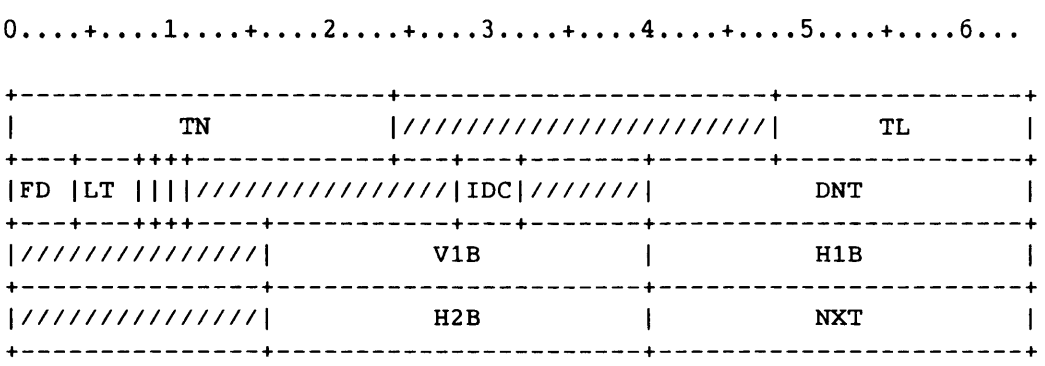


Figure A-22. Label Definition Table Header

Field	Word(base8)	Bits	Description
LDTN	0	0-23	Table name ('LDT' in ASCII)
LDTL	0	48-63	Table length (variable)
LDFD	1	0-3	Foreign dataset translation identifier This field is used to indicate whether run time foreign dataset translation should be performed on this dataset.
LDLT	1	4-7	Requested label type: 0 TPLNL Non-labeled 1 TPLAL ANSI-standard label 2 TPLSL IBM standard labels 3 TPLBP BY-PASS LABEL 4 TPLFR UNSUPPORTED LABEL 5 TPLFAL FIELD ANSI LABELED 6 TPLFNL FIELD NON LABELED 7 TPLFSL FIELD IBM LABELED
LDPROT	1	8	Protected access indicator. If non-zero for a new tape dataset then the dataset is to be protected on the servicing front-end.
LDCAT	1	9	Cataloged dataset indicator

Field	Word(base8)	Bits	Description
LDCV	1	10	Dataset data conversion flag. This field is used to indicate whether implicit data conversion shall be done by the run time library.
LDIDC	1	28-31	Initial dataset desposition 0 TPOLD Old dataset 1 TPNEW New dataset
LDDNT	1	40-63	Dataset name table (DNT) pointer. The field value is JTA-relative.
LDV1B	2	16-39	Offset of volume 1 entry, relative to LDT base. If the LDT does not contain a VOL1 entry, this field must be zero.
LDH1B	2	40-63	Offset of header 1 entry, relative to LDT base; must be zero if there is no HDR1 entry
LDH2B	3	16-39	Offset of header 2 entry, relative to LDT base; must be zero if there is no HDR2 entry
LDNXT	3	40-63	PTR TO NEXT LDT FOR CONCATENATION

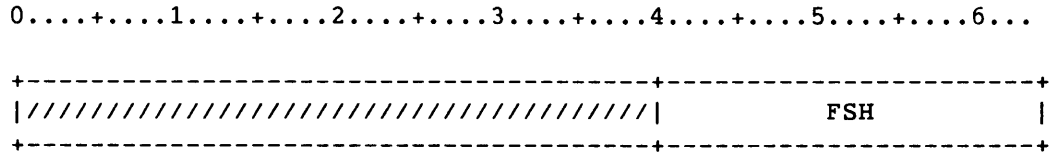


Figure A-23. Header Redefiniton of LDDNT

Field	Word(base8)	Bits	Description
LDFSH	1	40-63	Front-end service header offset

VOLUME 1 ENTRY

The volume 1 entry corresponds to volume 1 labels for all volumes in the dataset. The volume 1 entry can be placed anywhere after the header, as long as the LDV1B header field points to it properly. The volume 1 entry is optional.

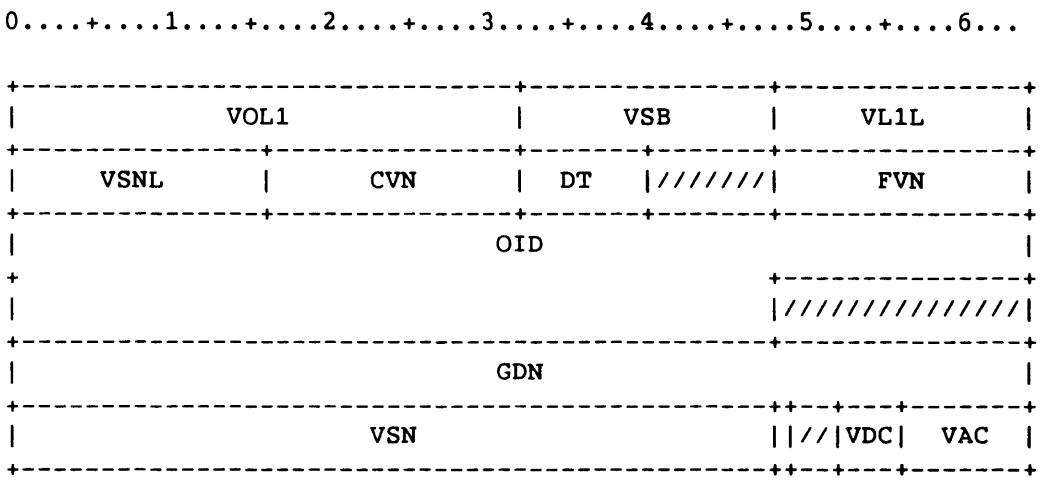


Figure A-24. VOL1 Entry Description

Field	Word(base8)	Bits	Description
LDVOL1	0	0-31	Entry name ('VOL1' in ASCII)
LDVSB	0	32-47	Volume serial list base offset
LDVL1L	0	48-63	Volume 1 length
LDVSNL	1	0-15	Number of VSNs in entry
LDCVN	1	16-31	Current VSN ordinal
LDDT	1	32-39	Device type LDDT6250=0 0 TPD62 6250 BPI LDDT1600=1 1 TPD16 1600 BPI LDDT3480=2 2 3480 DEVICE

LD Label Definition Table - LDT

Field	Word(base8)	Bits	Description
LDFVN	1	48-63	Final VSN ordinal: ordinal of VSN corresponding with the volume sequence number in access condition
LDOID	2-3	0-63	Owner identifier
LDOID1	2	0-63	Characters 1-8
LDOID2	3	0-47	Characters 9-14
LDGDN	4	0-63	Generic device name
LDVSN	5	0-47	Beginning VSN
LDVRG	5	48	Volume-registered flag, set by a servicing front-end. When set, the VSN is from front-end catalog.
LDVDC	5	52-55	Volume disposition 0 TPOLD Existing dataset 1 TPNEW New volume to dataset
LDVAC	5	56-63	Volume accessibility character, obtained from the label group

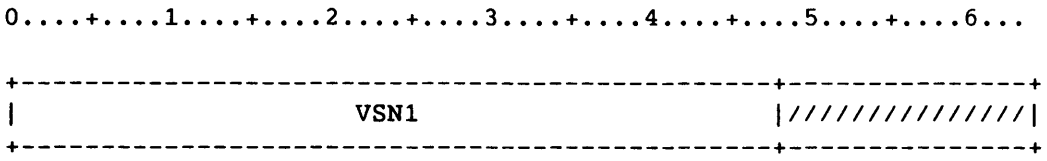


Figure A-25. Redefinition of LDVSN?

Field	Word(base8)	Bits	Description
LDVSN1	5	0-47	LE@VOL1=W@LDVSN+I@TMV

HEADER 1 ENTRY

The header 1 entry describes dataset attributes and corresponds to the HDR1, EOF1, and EOV1 labels for all volumes in the dataset. Header 1 shows numeric fields in both binary and ASCII. COS uses ASCII for generating and validating the label group. If a field is changed, both versions must be changed. ASCII fields are right-justified with leading zeros. The header 1 entry is optional and can be placed anywhere after the header, provided it is pointed to by header field LDH1B.

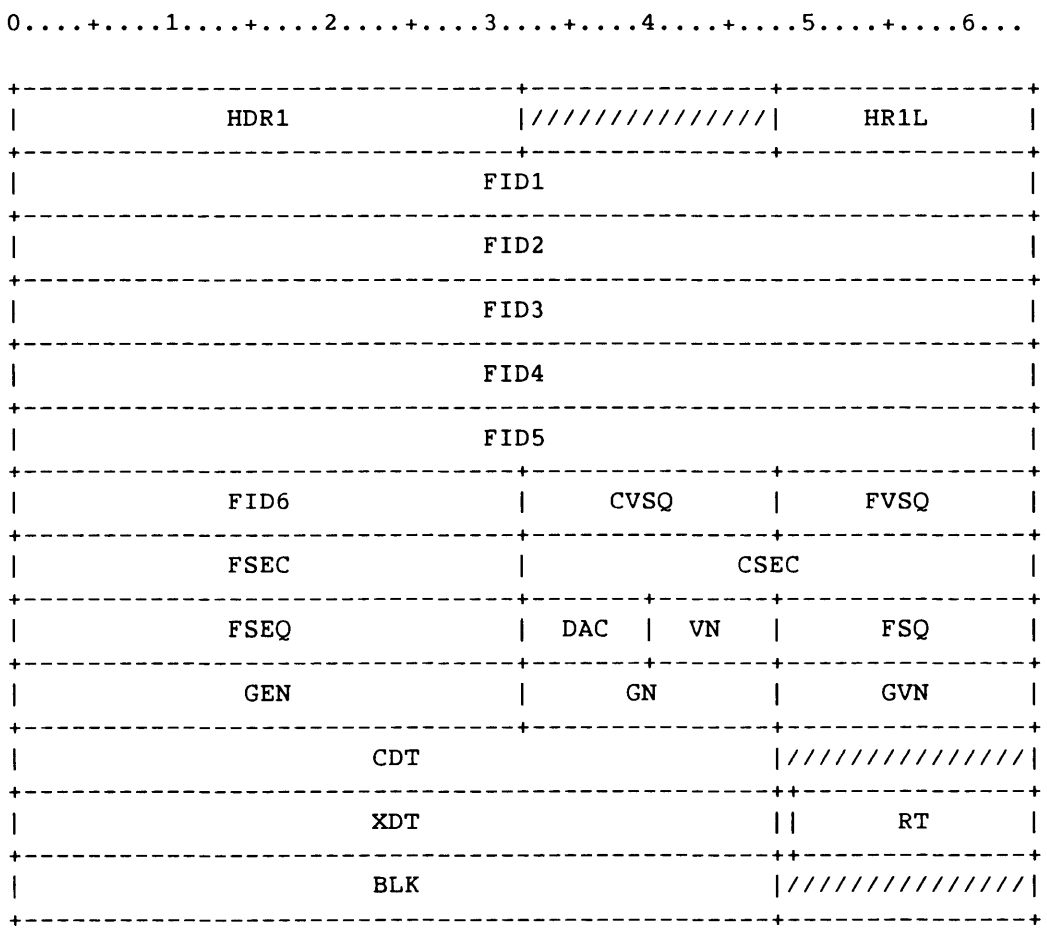


Figure A-26. HDR1 Entry Description

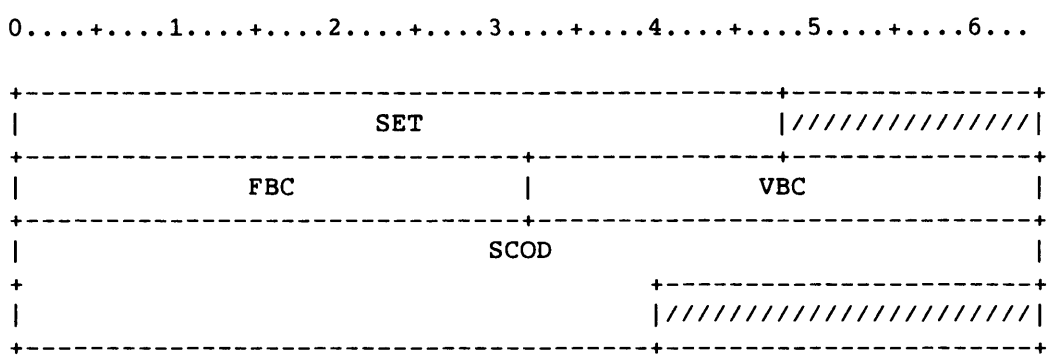


Figure A-26. HDR1 Entry Description

Field	Word(base8)	Bits	Description
LDHDR1	0	0-31	Entry name ('HDR1' in ASCII)
LDHR1L	0	48-63	Header 1 length
LDFID1	1	0-63	Characters 1-8
LDFID2	2	0-63	Characters 9-16
LDFID3	3	0-63	Characters 17-24
LDFID4	4	0-63	Characters 25-32
LDFID5	5	0-63	Characters 33-40
LDFID6	6	0-31	Characters 41-44
LDCVSQ	6	32-47	Current volume sequence number (file section number), binary equivalent of LDCSEC
LDFVSQ	6	48-63	First volume sequence number (file section number), binary equivalent of LDFSEC
LDFSEC	7	0-31	First file section number (volume sequence number) in ASCII, the ordinal number of the volume to be mounted first

Field	Word(base8)	Bits	Description
LDCSEC	7	32-63	Current file section number (volume sequence number) in ASCII, the ordinal number of the currently mounted volume
LDFSEQ	10	0-31	File sequence number (ASCII) ordinal of the dataset being accessed. If FSEQ > 1, volume should have more than one dataset.
LDDAC	10	32-39	Dataset accessibility character.
LDVN	10	40-47	Generation version number, numeric equivalent of LDGVN
LDFSQ	10	48-63	File sequence number, numeric equivalent of LDFSEQ
LDGEN	11	0-31	Generation number. Any value other than one indicates that a dataset is in a generation data group.
LDGN	11	32-47	Generation number, numeric equivalent of LDGEN
LDGVN	11	48-63	Generation version number (ASCII). Any value other than 0 indicates that the dataset is in a generation data group.
LDCDT	12	0-47	Creation date (ASCII). This field indicates the creation date of the dataset in the julian form: 'yyddd'. Note the space (LDCSP) must be present.
LDCSP	12	0-7	Space
LDCYR	12	8-23	Year
LDCDY	12	24-47	Day
LDXDT	13	0-47	Expiration date; same format as creation date above

LDXSP	13	0-7	Space
LDXYR	13	8-23	Year
LDXDY	13	24-47	Day
LDUXD	13	48	User specified XDT (expiration date) flag
LDRT	13	49-63	Retention period, integer days
LDBLK	14	0-47	Volume block count (ASCII): number of user data blocks present, read from or written into the label. Can be inaccurate because overflow causes it to be cleared; see LDVBC for an accurate count.
LDSET	15	0-47	File set identifier, normally set to the serial number of first volume in the dataset
LDFBC	16	0-31	File block count (binary)
LDVBC	16	32-63	Volume block count (binary), number of blocks written on volume so far
LDSCOD	17-20	0-63	System identification code, to identify the operating system or computer system that generated the tape
LDSCD1	17	0-63	Character 1-8
LDSCD2	20	0-39	Character 9-13 identify the operating system or computer system that generated the tape

LE@HDR1=W@LDSCD2+1

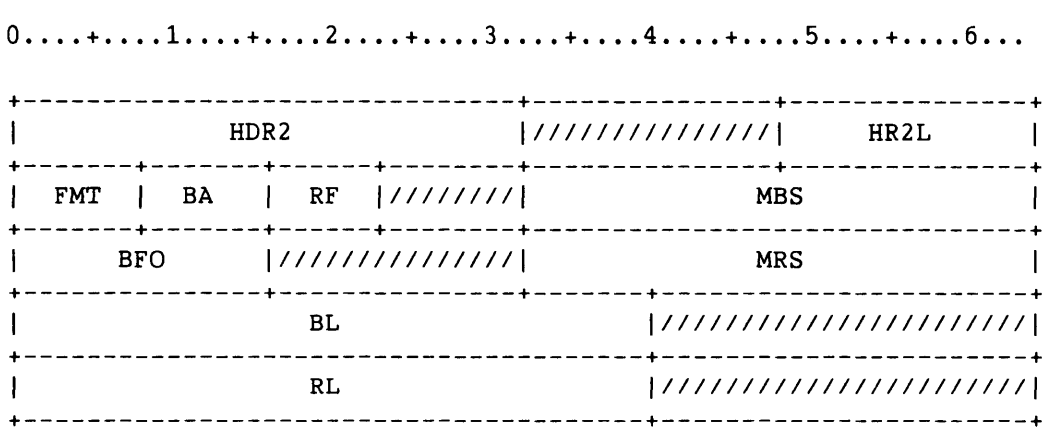


Figure A-27. HDR2 Entry Description

Field	Word(base8)	Bits	Description
LDHDR2	0	0-31	Entry name ('HDR2' in ASCII)
LDHR2L	0	48-63	Header 2 length
LDFMT	1	0-7	Record format, two types IBM label types: F Fixed-length records V Variable-length records U Undefined record format ANSI label types: F Fixed-length records D Variable-length records S Records span tape blocks
LDBA	1	8-15	Blocking attributes, IBM label types only: B Blocks are an integral multiple of the record size S Records span tape blocks R Records span tape blocks, and the blocks are an integral multiple of the record size
LDRF	1	16-22	Record format.

Field	Word(base8)	Bits	Description
LDMBS	1	32-63	Maximum block size (binary), maximum size of any tape block that can be read or written
LDBFO	2	0-15	Buffer offset, ANSI only (not currently supported by COS)
LDMRS	2	32-63	Maximum record size (binary), maximum size of any record that can be read or written
LDBL	3	0-39	Maximum block size (ASCII), maximum number of bytes in a tape block, read from or written into the label. Can be inaccurate because overflow causes it to be cleared; see LDMBS for an accurate count.
LDRL	4	0-39	Maximum record size (ASCII), maximum number of bytes in a tape record, read from or written into the label. Can be inaccurate because overflow causes it to be cleared; see LDMRS for an accurate count.

```
LE@HDR2=W@LDRL+1
```

LF Logical File Table - LFT

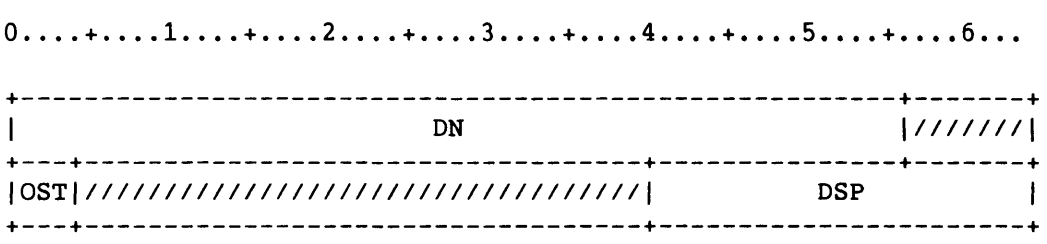


Figure A-28. Logical File Table

Field	Word(base8)	Bits	Description
LFDN	0	0-55	Dataset name
LFOST	1	0-3	DATASET OPEN STATUS
LFDSP	1	40-63	DSP address

OD Open Dataset Table - ODN

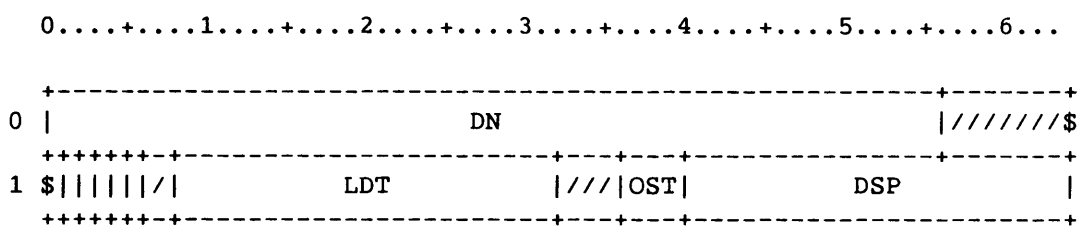


Figure A-29. Open Dataset Table

Field	Word(base8)	Bits	Description
ODDN	0	0-55	Dataset name
ODV	1	1	Close volume
ODM	1	2	Open for 'mod' (append)
ODS	1	3	Close or open with saved position
ODH	1	4	Hold resources
ODUDS	1	5	Open as unblocked flag
ODLDT	1	8-31	LDT address
ODOST	1	36-39	TYPE OF OPEN REQUESTED OSTSA=0 Create DSP/LFT buffer in system area OSTUA=1 Create DSP/LFT/buffer in user area OSTMSY=2 DSP/LFT/buffer moved to system area
ODDSP	1	40-63	DSP pointer: Negative: negative offset Positive: absolute address

OP Parameter Block for F$OPT - OPT

This table is passed for an F$OPT call.

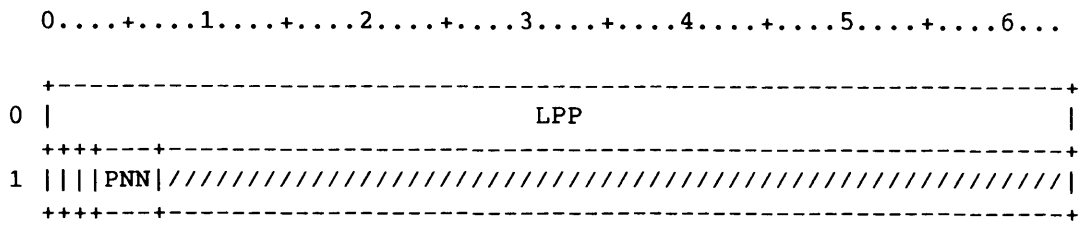


Figure A-30. Parameter Block for F$OPT

Field	Word(base8)	Bits	Description
OPLPP	0	0-63	Page length
OPSTAT	1	0	Dataset statistics enabled
OPPNCH	1	1	NZ if OPTION,PN selected
OPPNAS	1	2	NZ if PN=n, ZR if PN=ANY
OPPNN	1	3-6	Processor number (if @OPPNAS NZ)

A PDD is a parameter list that accompanies a Permanent Dataset Management request.

The PDD illustrated in table A-1 is used for all save, access, dump access, load, modify, permit, rewrite SDT, pseudo-access, and permanent dataset name requests.

The PDD illustrated in figure A-31 is used for both DSC and DXT page requests, and for dump time requests.

The PDD illustrated in figure A-32 is used for all delete, release, and adjust requests.

The PDD illustrated in figure A-33 is used for queue and dequeue SDT requests, and for get and link DXT requests.

The PDDs starting with figure A-34 are function oriented; most are used for archive feature support.

Table A-1. Permanent Dataset Function Codes

Symbol	Octal Code	Function
PMFCSU	10	Save user dataset
PMFCSI	12	Save input dataset
PMFCSO	14	Save output dataset
PMFCAU	20	Access user dataset
PMFCAI	26	Access spooled dataset
PMFCAO	26	Access spooled dataset
PMFCDU	30	Delete user dataset
PMFCDI	36	Delete spooled dataset
PMFCDO	36	Delete spooled dataset
PMFCPG	40	DSC Page request
PMFCPX	41	DXT Page request
PMFCLU	50	Load user dataset
PMFCLI	52	Load input dataset
PMFCLO	54	Load output dataset
PMFCRL	60	PDS/Release request
PMFCPN	70	PDN request
PMFCPNI	72	PDN request - input datasets
PMFCPNO	74	PDN request - output datasets
PMFCDT	100	Dump time request
PMFCDQ	110	Dequeue SDT

PMFCEA	120	Queue SDT to available queue
PMFCEI	122	Queue SDT to input queue
PMFCEO	124	Queue SDT to output queue
PMFCAD	130	Adjust user dataset
PMFCMD	140	Modify user dataset
PMFCRSDT	150	Rewrite input SDT entry
PMFCPSAC	160	Pseudo-access for RRJ
PMFCPU	170	Access user saved dataset for PDSDUMP
PMFCPO	176	Access output dataset for PDSDUMP
PMFCPI	176	Access input dataset for PDSDUMP
PMFCPE	200	Permit Request
PMFCLKDX	210	Link DXT Request
PMFCCTXT	221	Copy Text to buffer
PMFCCSLT	222	Copy Station Slot to buffer
PMFCCTAS	223	Copy Text and Station Slot to buffer
PMFCACDC	231	Access Dataset Catalog
PMFCACDX	232	Access Dataset Catalog Extension
PMFCACMC	233	Access Master Catalog
PMFCACBC	234	Access Backup Catalog
PMFCLDMC	243	Load Master Catalog
PMFCLDBC	244	Load Backup Catalog
PMFCONBU	250	Logon Backup System Job
PMFCONSM	251	Logon Space Manager System Job
PMFCONRC	252	Logon Recall System Job
PMFCONCU	253	Logon Cleanup System Job
PMFCONBH	254	Logon Backup Helper Job
PMFCONSH	255	Logon Space Manager Helper Job
PMFCONRH	256	Logon Recall Helper Job
PMFCONCH	257	Logon Cleanup Helper Job
PMFCOFBU	260	Logoff Backup System Job
PMFCOFSM	261	Logoff Space Manager System Job
PMFCOFRC	262	Logoff Recall System Job
PMFCOFCU	263	Logoff Cleanup System Job
PMFCOFBH	264	Logoff Backup Helper Job
PMFCOFSH	265	Logoff Space Manager Helper Job
PMFCOFRH	266	Logoff Recall Helper Job
PMFCOFCH	267	Logoff Cleanup Helper Job
PMFCSDEI	270	Set Dataset Edition Interlock
PMFCCDEI	300	Clear Dataset Edition Interlock
PMFCRET	311	Retire Dataset Edition
PMFCMIG	312	Migrate Dataset Edition
PMFCDEL	313	Delete Dataset Edition
PMFCSBRS	321	Set Backup Required Status
PMFCCBRS	322	Clear Backup Required Status
PMFCSRLD	330	Set Reload Requested Status

PMFCBUAC	340	Backup Access
PMFCRLD	350	Reload Dataset Edition
PMFCWRBC	360	Write Backup Catalog
PMFCGLDV	370	Get Logical Device Information
PMFCGRRL	400	Get Recall/Restore List
PMFCSRET	411	Set Retirement Requested Status
PMFCSRES	412	Set Restore Requested Status
PMFCSDEL	413	Set Delete Requested Status
PMFCARCL	420	Abort Recall Requests
PMFCGKEY	430	Return hash key and region FWA
PMFCDAU	440	Copy DAT to STP and place address in DNT (User permanent dataset) (System request only)
PMFCDAI	441	Copy DAT to STP and place address in DNT (Input spooled dataset) (System request only)
PMFCDAO	442	Copy DAT to STP and place address in DNT (Output spooled dataset) (System request only)
PMFCCDWU	450	Set/clear DCDWN bit in DSC (User permanent dataset) (System request only)
PMFCCDWI	451	Set/clear DCDWN bit in DSC (Input spooled dataset) (System request only)
PMFCCDWO	452	Set/clear DCDWN bit in DSC (Output spooled dataset) (System request only)

PM Permanent Dataset Definition - PDD

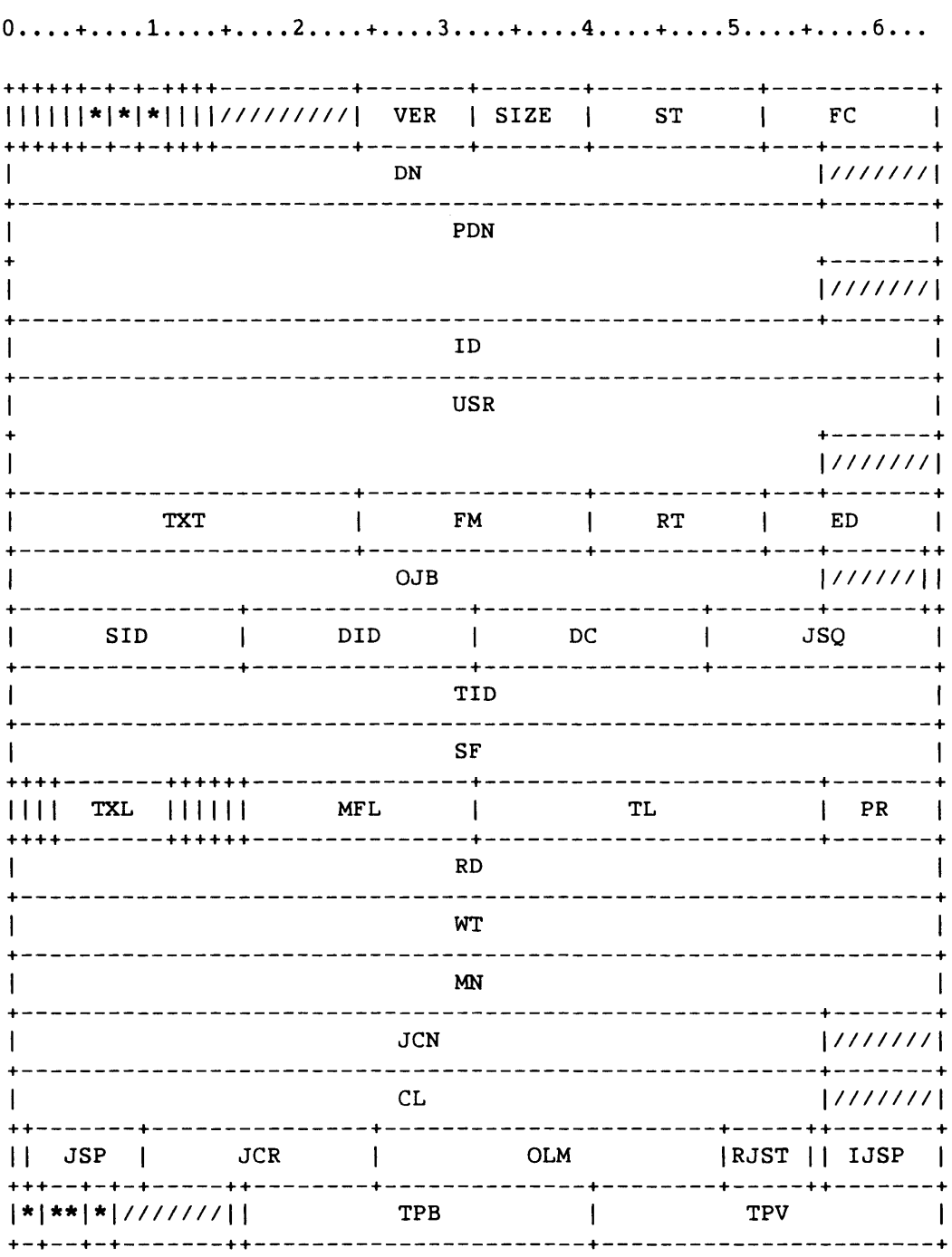


Figure A-31. Permanent Dataset Definition

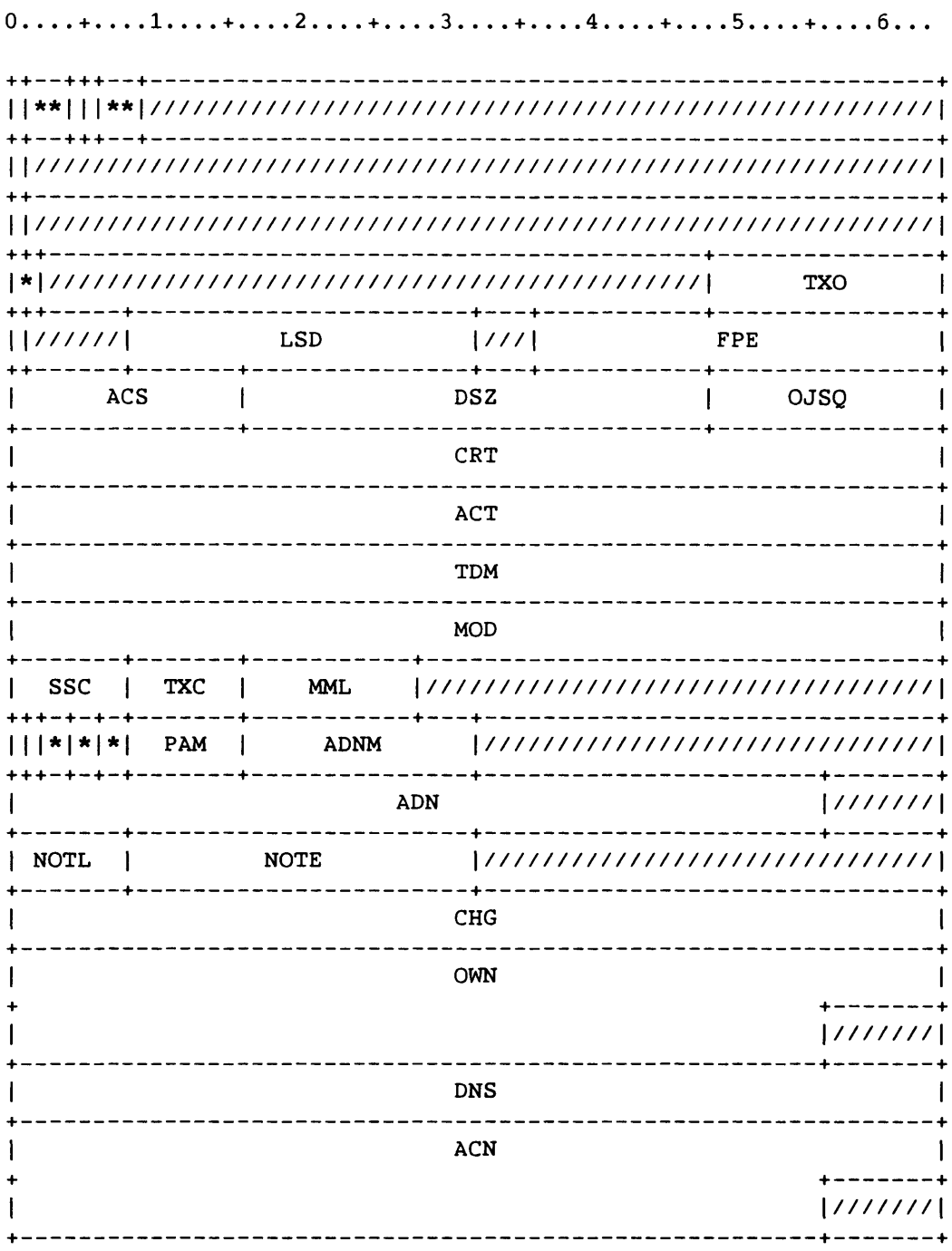


Figure A-31. Permanent Dataset Definition

PM Permanent Dataset Definition - PDD

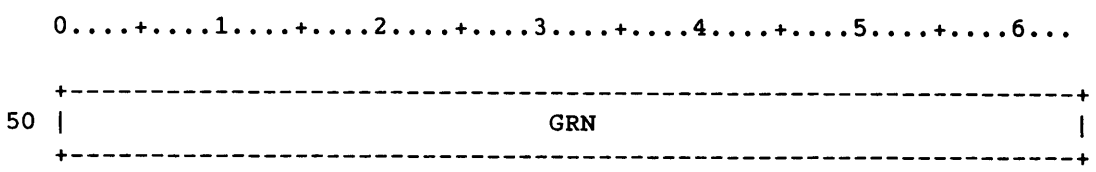


Figure A-31. Permanent Dataset Definition
LE@MPDD=1 Minimum PDD size
LE@PDD11=D'31 COS 1.11 PDD size

Field	Word(base8)	Bits	Description
PMSG	0	0	Normal completion message suppression indicator
PMERR	0	1	Error message suppression indicato
PMWAIT	0	2	WAIT flag for a disposed dataset
PMNRLS	0	3	No release of dataset on DISPOSE
PMAQR	0	4	Acquire flag for accounting
PMTP	0	5-6	Tape dataset (online/staged)
PMTCS	0	7-8	Tape dataset character set
PMEXO	0	9-10	Execute only
PMDTR	0	11	Update dump-time on PDSDUMP access
PMSMT	0	12	Submit flag
PMDFFL	0	13	Job-used-MFL-default flag
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMDN	1	0-55	Local dataset name

PM Permanent Dataset Definition - PDD

Field	Word(base8)	Bits	Description
PMPDN	2-3	0-63	Permanent dataset name
PMPDN1	2	0-63	Characters 1-8
PMPDN2	3	0-55	Characters 9-15
PMID	4	0-63	User identification
PMUSR	5-6	0-63	User number
PMUSR1	5	0-63	Characters 1-8
PMUSR2	6	0-55	Characters 9-15
PMTXT	7	0-23	Address of optional text field
PMFM	7	24-39	Format designator (two characters) FMCD=CD Character/deblocked FMCB=CB Character/blocked FMBD=BD Binary/deblocked FMBB=BB Binary/blocked
PMRT	7	40-51	Retention period; 0-4095 days.
PMED	7	52-63	Edition number (0-4095)
PMOJB	10	0-55	Originating job name
PMDWN	10	63	New state of DCDWN bit (FC=45x)
PMSID	11	0-15	Source ID; 2 characters.
PMDID	11	16-31	Destination ID; 2 characters.
PMDC	11	32-47	Disposition code; 2 characters. DCIN=IN Job dataset DCST=ST Dataset to be staged DCSC=SC Scratch dataset DCPR=PR Print dataset DCPU=PU Punch dataset DCPT=PT Plot dataset DCMT=MT Magnetic tape dataset
PMJSQ	11	48-63	Job sequence number

PM Permanent Dataset Definition - PDD

Field	Word(base8)	Bits	Description
PMTID	12	0-63	Terminal ID; 1-8 characters.
PMSF	13	0-63	Special forms
PMUQ	14	0	Unique access required
PMENT	14	1	Enter in System Directory
PMIR	14	2	Immediate reply requested
PMTXL	14	3-10	Number of words of text
PMNRR	14	11	Job rerun flag; set if job cannot rerun (input entries only).
PMINIT	14	12	Job initiate flag; set if job has been initiated.
PMIA	14	13	Interactive flag
PMDFR	14	14	Deferred disposition indicator
PMNA	14	15	No abort flag. If set, processing continues even if an error is encountered.
PMMFL	14	16-31	MFL parameter from job card (input
PMSGFL	14	16	All available memory requested
PMFL	14	17-31	Field length/512
PMTL	14	32-55	Time limit (input datasets)
PMPR	14	56-63	Priority (input datasets)
PMRD	15	0-63	Read permission control word
PMWT	16	0-63	Write permission control word
PMMN	17	0-63	Maintenance permission control wor
PMJCN	20	0-55	Job class name
PMCL	21	0-55	CL parameter from JOB statement

Field	Word(base8)	Bits	Description
PMSYS	22	0	System job
PMJSP	22	1-8	JOB statement priority
PMJCR	22	9-24	Job class rank
PMOLM	22	25-48	Size of $OUT in 512-word block
PMRJST	22	49-54	Job status flag
PMIJSP	22	56-63	Original job card priority
PMTPD	23	0-1	Tape density
PMTPL	23	2-4	Tape label type
PMTPF	23	5-6	Tape format
PMTPC	23	15	Tape cataloged dataset
PMTPB	23	16-39	Tape maximum block size in bytes
PMTPV	23	40-63	Tape pointer to label definition table
PMTPM	24	0	Tape online maintenance access
PMTPP	24	1-3	Tape parallel device count
PMTP2	24	4	Tape second device assignment
PMTPH	24	5	Tape hold assigned device
PMIDC	24	6-8	Tape initial disposition code
PM2164	25	0	Unused
PM2264	26	0	Unused
PMTSCV	27	0-1	Timestamp conversion specification TSCVTHIS=0 Convert to current COS system

TSCVRT=1 Convert to RT-based timestamp

TSCVNS=2 Convert to NS-based timestamp

TSCVSAME=3 No conversion -- leave timestamp alone

PMTXO	27	48-63	TXT ORDINAL OF USER TASK
PMOCC	30	0	Operator-changed-class flag
PMLSD	30	8-31	Temporary SDT address for load input/output
PMFPE	30	36-63	First DSC page/entry for dataset
PMFPP	30	36-59	First DSC page for dataset
PMFEN	30	60-63	First entry for dataset
PMACS	31	0-15	Number of accesses (load saved datasets only)
PMDSZ	31	16-47	Size of dataset as reflected by DS DAT bodies (used only when a pse access is performed during the recovery of rolled jobs)
PMOJSQ	31	48-63	Originating job sequence number
PMCRT	32	0-63	Creation time in cycles (load request only)
PMACT	33	0-63	Time of last access in cycles (loa request only)
PMTDM	34	0-63	Time of last dump in cycles (load request only)
PMMOD	35	0-63	Time of last modification in cycle (load request only)
PMSSC	36	0-7	Station slot word length
PMTXC	36	8-15	Text field word length

Field	Word(base8)	Bits	Description
PMMML	36	16-27	Interactive maximum message length
PMPDE	37	0	Partial delete flag
PMREM	37	1	Remove permit flag
PMTRA	37	2-3	Track accesses flag: TRAKNO=1 Do not track accesses TRAKYE=2 Do track accesses
PMRESD	37	4-5	Preferred residency RESON=1 Online residency preferred RESOF=2 Offline residency preferred RESNP=3 No residency preference
PMBACK	37	6-7	Backup requirement BACKNO=2 No backup required BACKYE=3 Backup is required
PMPAM	37	8-15	Public/permit access mode: PAMEX=O'011 Execute only PAMRE=O'001 Read permission PAMWR=O'002 Write permission PAMMA=O'004 Maintenance permission PAMNO=O'200 No permissions MAXPAM=5
PMADNM	37	16-31	ADN propagate attributes mask: PACW=O'000001 Control words PAPAM=O'000002 Public access mode PATRK=O'000004 Track accesses PAPER=O'000010 Permits PATXT=O'000020 Text PANTS=O'000040 Notes PAALL=O'000077 All of the above PANO=O'100000 None MAXPA=D'8 Maximum allowable attributes

PM Permanent Dataset Definition - PDD

Field	Word(base8)	Bits	Description
PMADN	40	0-55	Attributes dataset name
PMNOTL	41	0-7	Notes length in words
PMNOTE	41	8-31	Pointer to notes text LE@NOTE=D'60 Allow 480 characters for notes
PMCHG	42	0-63	Last modification time (PDSLOAD)
PMOWN	43-44	0-63	Dataset Owner
PMOWN1	43	0-63	Owner (char 1-8)
PMOWN2	44	0-55	Owner (char 9-15)
PMDNS	45	0-63	Reserved for installation
PMACN	46-47	0-63	Account Number
PMACN1	46	0-63	Characters 1-8 of account number
PMACN2	47	0-55	Characters 9-15 of account number
PMGRN	50	0-63	Generic resource counters

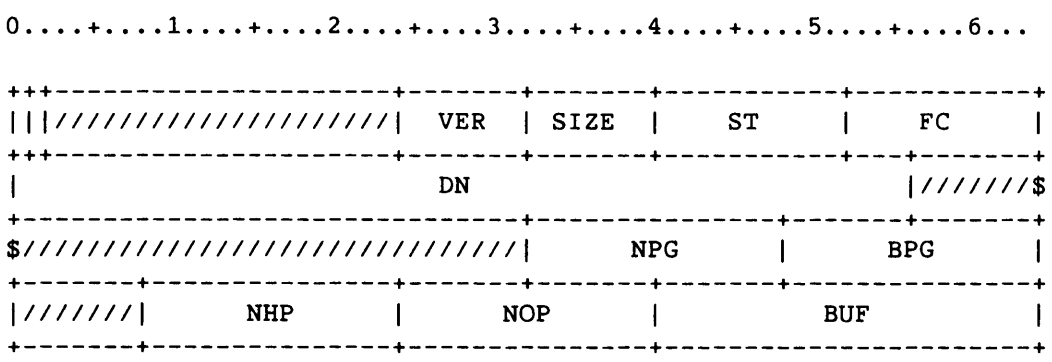


Figure A-32. PDD Format 2

Field	Word(base8)	Bits	Description
PMSG	0	0	Normal completion message suppression indicator
PMERR	0	1	Error message suppression indicator
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMDN	1	0-55	Local Dataset Name (PMFCDT)
PMNPG	2	32-47	Number of pages (PMFCPG,PMFCPX)
PMBPG	2	48-63	Beginning page number (PMFCPG,PMFC for PMFCPX requests)
PMNHP	3	8-23	Number of hash pages (returned by PDM for PMFCPG requests)
PMNOP	3	24-39	Number of overflow pages (returned by PDM for PMFCPG requests)
PMBUF	3	40-63	Buffer address

PM Permanent Dataset Definition - PDD

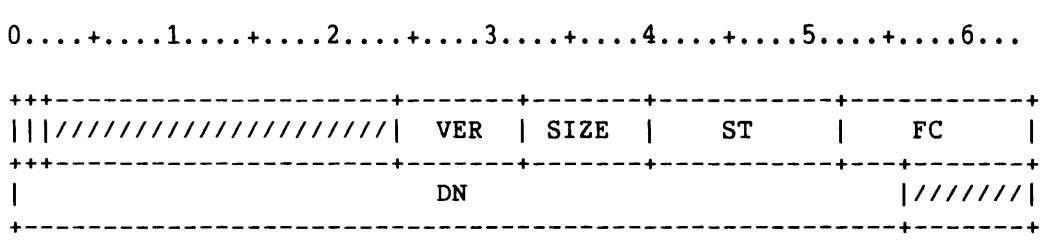


Figure A-33. PDD Format 3

Field	Word(base8)	Bits	Description
PMSG	0	0	Normal completion message suppression indicator
PMERR	0	1	Error message suppression indicator
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMDN	1	0-55	Local dataset name

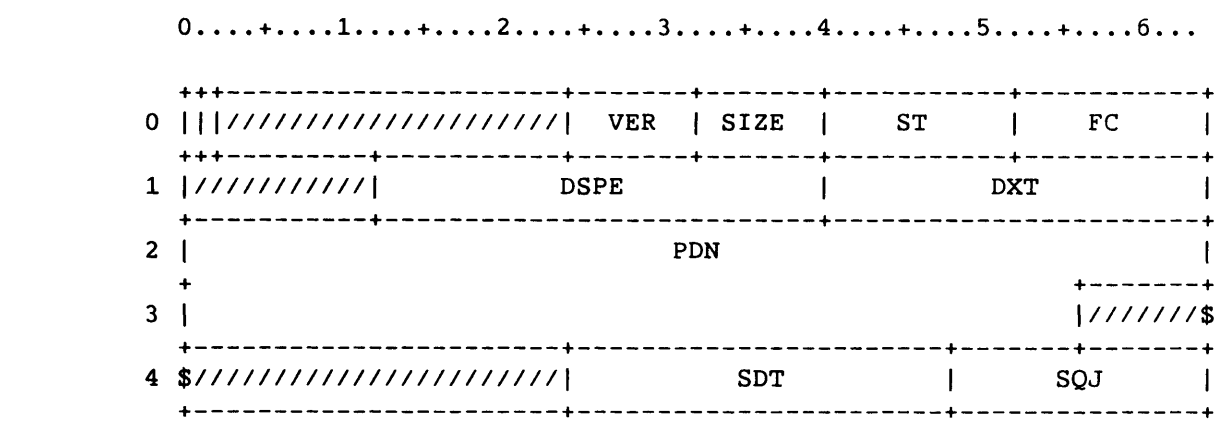


Figure A-34. PDD Format 4

Field	Word(base8)	Bits	Description
PMSG	0	0	Normal completion message suppression indicator
PMERR	0	1	Error message suppression indicator
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMDSPE	1	12-39	Page/entry of main DSC entry (PMFCLKDX, PMFCRTDX requests)
PMDSP	1	12-35	Page number of main DSC entry (PMFCLKDX, PMFCRTDX requests)
PMDSE	1	36-39	Entry number of main DSC entry (PMFCLKDX, PMFCRTDX requests)
PMDXT	1	40-63	Pointer to DXT information buffer (PMFCLKDX, PMFCRTDX requests)
PMPDN	2-3	0-63	Permanent dataset name

PM Permanent Dataset Definition - PDD

Field	Word(base8)	Bits	Description
PMPDN1	2	0-63	Characters 1-8
PMPDN2	3	0-55	Characters 9-15
PMSDT	4	24-47	SDT address Returned by PDM for PMFCDQ request Input for PMFCEA, PMFCEI, PMFCEO
PMSQJ	4	48-63	Job sequence number (PMFCDQ request)

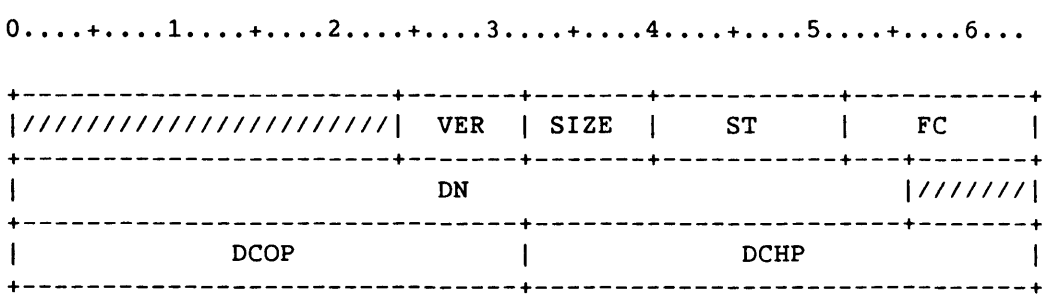


Figure A-35. PDD for PMFCACDC
L@PMACDC=3 PDD size for PMFCACDC

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMDN	1	0-55	Local Dataset Name
PMDCOP	2	0-31	Number of DSC overflow pages
PMDCHP	2	32-63	Number of DSC hash pages

PM Permanent Dataset Definition - PDD

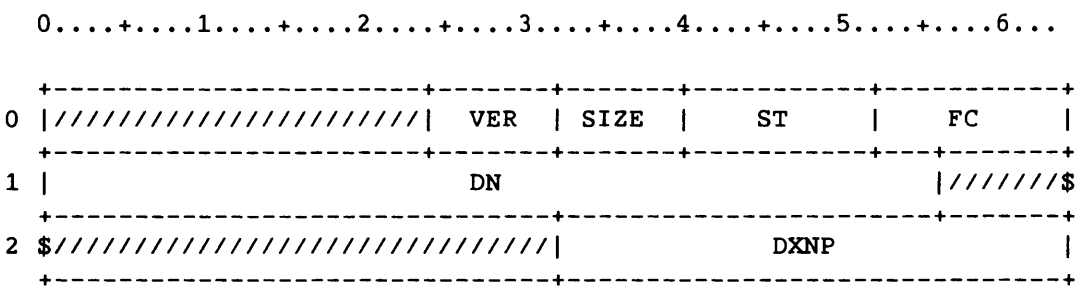


Figure A-36. PDD for PMFCACDX
L@PMACDX=3 PDD size for PMFCACDX

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMDN	1	0-55	Local Dataset Name
PMDXNP	2	32-63	Number of DXT pages

PM Permanent Dataset Definition - PDD

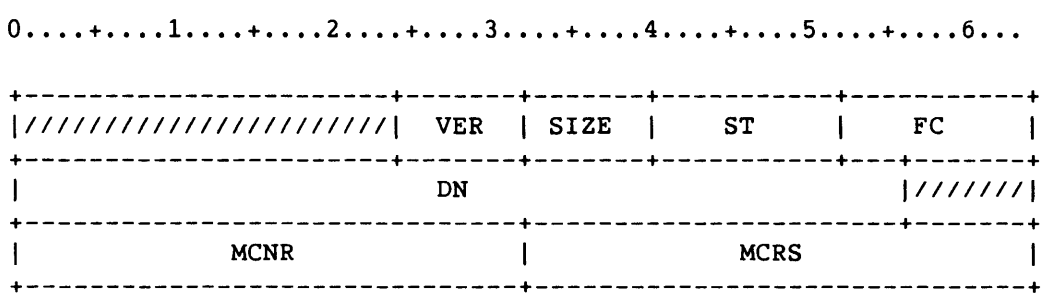


Figure A-37. PDD for PMFCACMC, PMFCLDMC
L@PMACMC=3 PDD size for PMFCACMC
L@PMLDMC=3 PDD size for PMFCLDMC

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMDN	1	0-55	Local Dataset Name
PMMCNR	2	0-31	Number of MCD regions
PMMCRS	2	32-63	Size of each MCD region (sectors)

PM Permanent Dataset Definition - PDD

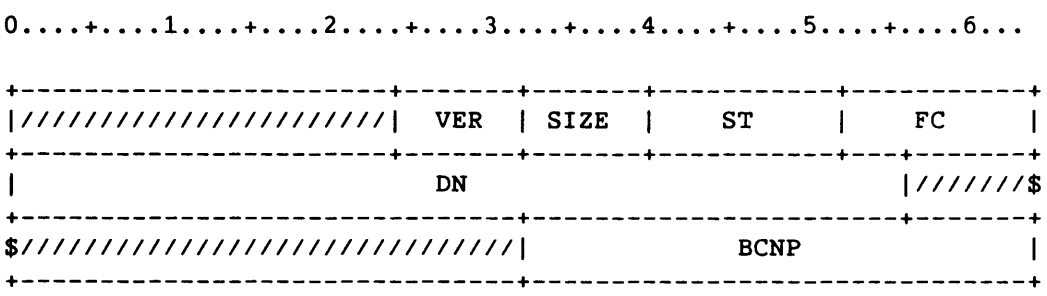


Figure A-38. PDD for PMFCACBC, PMFCLDBC
L@PMACBC=3 PDD size for PMFCACBC
L@PMLDBC=3 PDD size for PMFCLDBC

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMDN	1	0-55	Local Dataset Name
PMBCNP	2	32-63	Number of BCD pages

PM Permanent Dataset Definition - PDD

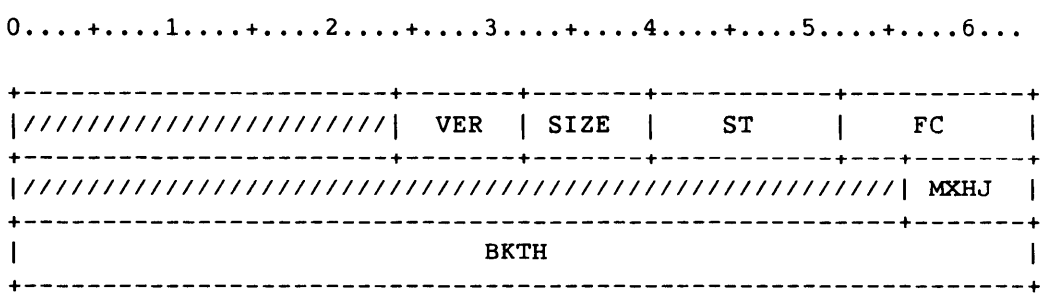


Figure A-39. PDD for PMFCONBU
L@PMONBU=3 PDD size for PMFCONBU

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMMXHJ	1	56-63	Maximum number of helper jobs
PMBKTH	2	0-63	Backup threshold (integer words)

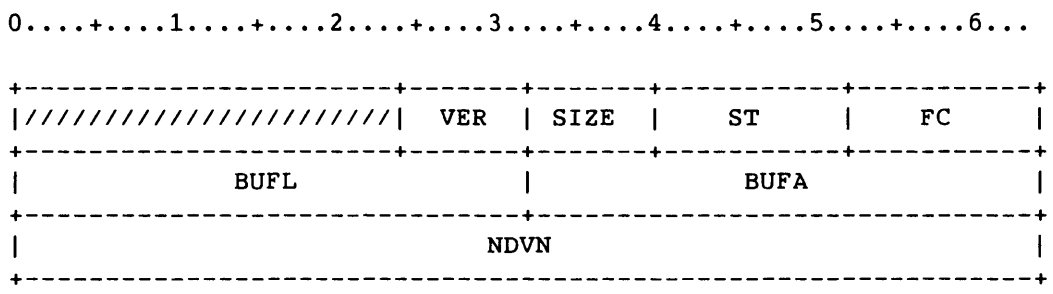


Figure A-40. PDD for PMFCONSM
L@PMONSM=3 PDD size for PMFCONSM (minimum)

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMBUFL	1	0-31	Buffer length
PMBUFA	1	32-63	Buffer address
PMNDVN	2	0-63	Number of devices in device threshold list (two words per device, see below)

The device list is passed in a buffer pointed to by PMBUFA and must be at least LE@PMDVL*PMNDVN words in length.

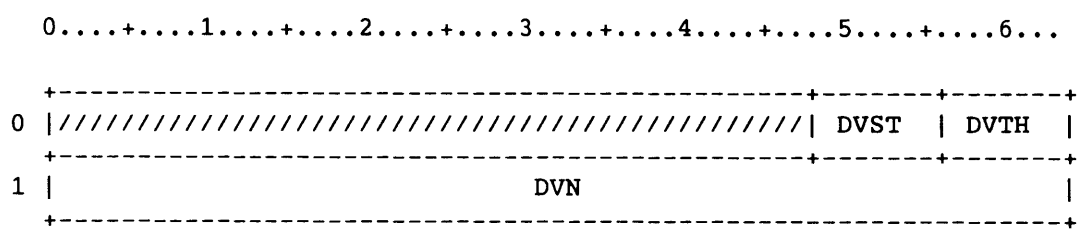


Figure A-41. Device List Entry for PMFCONSM

LE@PMDVL=2 Length of device list entry

Field	Word(base8)	Bits	Description
PMDVST	0	48-55	Device status
PMDVTH	0	56-63	Device threshold percentage (0-100)
PMDVN	1	0-63	Device name, LJZF

PM Permanent Dataset Definition   - PDD

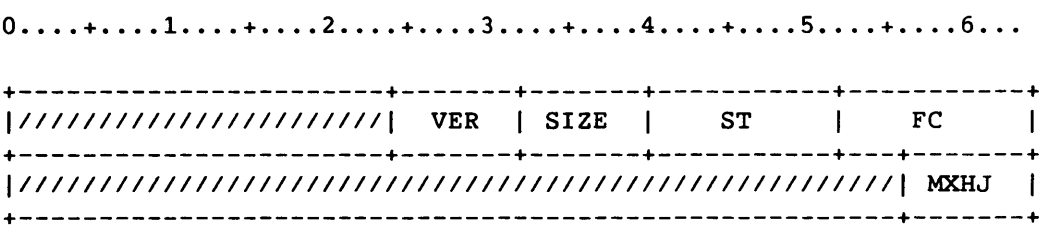


Figure A-42. PDD for PMFCONRC and PMFCONCU
L@PMONRC=2 PDD size for PMFCONRC
L@PMONCU=2 PDD size for PMFCONCU

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMMXHJ	1	56-63	Maximum number of helper jobs

PM Permanent Dataset Definition - PDD

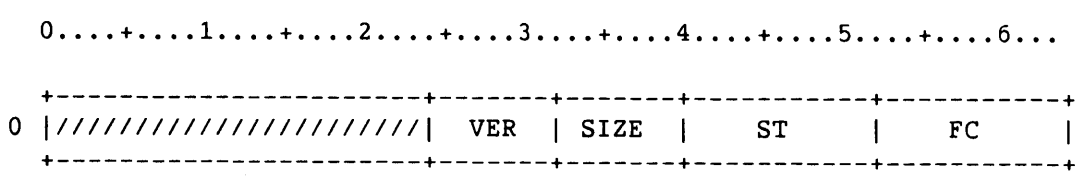


Figure A-43. PDD for PMFCONxH through PMFCOFxx

L@PMONBH=1	PDD size for PMFCONBH
L@PMONSH=1	PDD size for PMFCONSH
L@PMONRH=1	PDD size for PMFCONRH
L@PMONCH=1	PDD size for PMFCONCH
L@PMOFBU=1	PDD size for PMFCOFBU
L@PMOFSM=1	PDD size for PMFCOFSM
L@PMOFRC=1	PDD size for PMFCOFRC
L@PMOFCU=1	PDD size for PMFCOFCU
L@PMOFBH=1	PDD size for PMFCOFBH
L@PMOFSH=1	PDD size for PMFCOFSH
L@PMOFRH=1	PDD size for PMFCOFRH
L@PMOFCH=1	PDD size for PMFCOFCH

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)

PM Permanent Dataset Definition - PDD

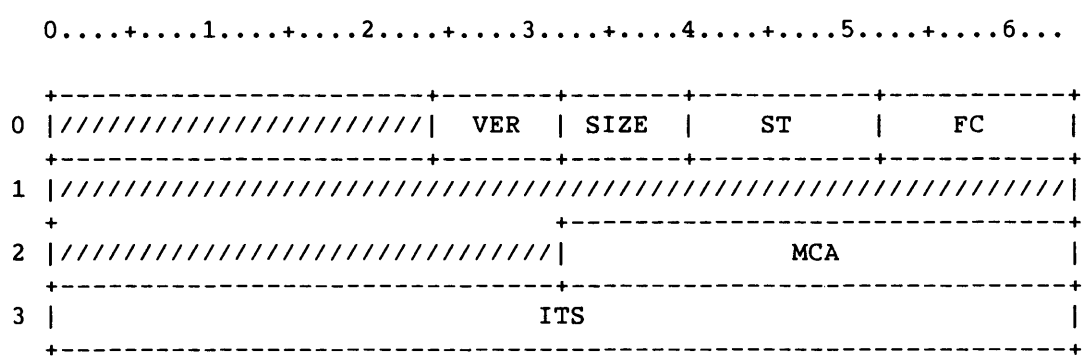


Figure A-44. PDD for PMFCSDEI
L@PMSDEI=4 PDD size for PMFCSDEI

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMMCA	2	32-63	Master Catalog address
PMITS	3	0-63	Identifying Timestamp

PM Permanent Dataset Definition - PDD

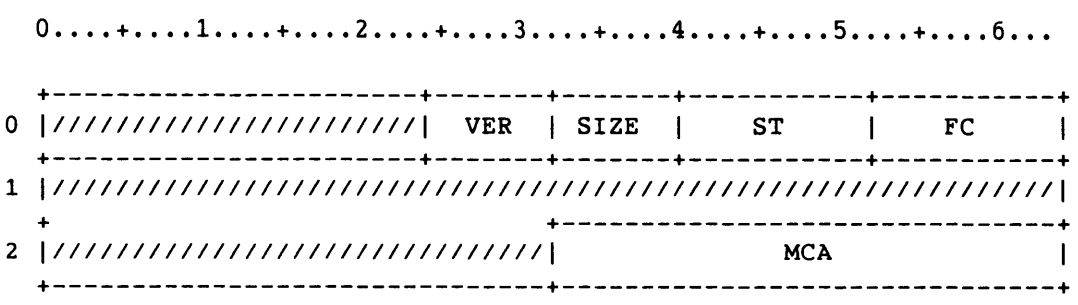


Figure A-45. PDD for PMFCCDEI
L@PMCDEI=3 PDD size for PMFCCDEI

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMMCA	2	32-63	Master Catalog address

PM Permanent Dataset Definition - PDD

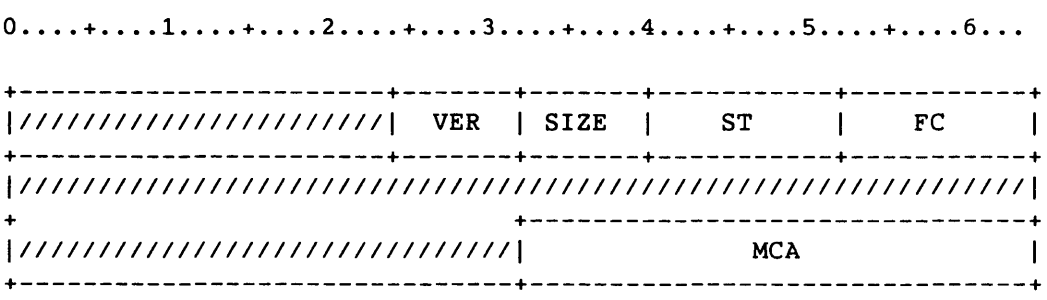


Figure A-46. PDD for PMFCRET through PMFCSRLD

L@PMRET=3 PDD size for PMFCRET
L@PMMIG=3 PDD size for PMFCMIG
L@PMDEL=3 PDD size for PMFCDEL
L@PMSBRS=3 PDD size for PMFCSBRS
L@PMCBRS=3 PDD size for PMFCCBRS
L@PMSRLD=3 PDD size for PMFCSRLD

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMMCA	2	32-63	Master Catalog address

PM Permanent Dataset Definition - PDD

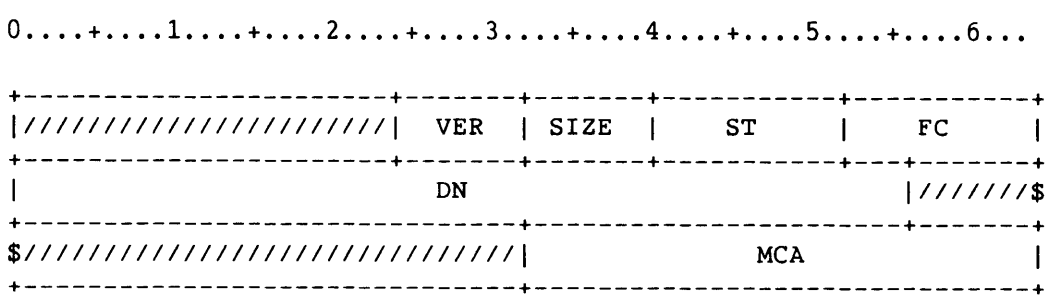


Figure A-47. PDD for PMFCBUAC
L@PMBUAC=3 PDD size for PMFCBUAC

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMDN	1	0-55	Local dataset name
PMMCA	2	32-63	Master Catalog address

PM Permanent Dataset Definition - PDD

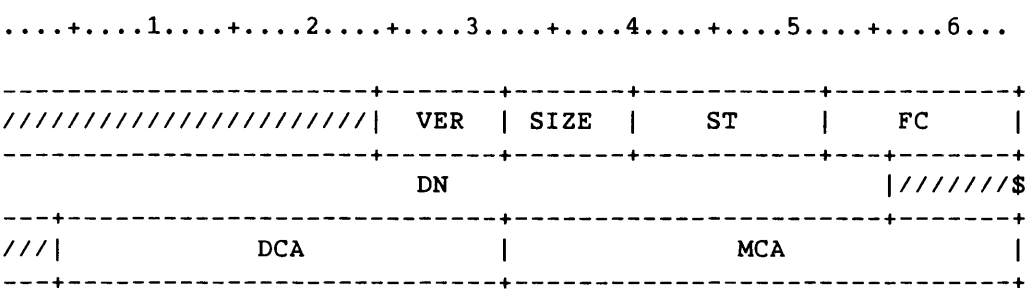


Figure A-48. PDD for PMFCRLD
L@PMRLD=3 PDD size for PMFCRLD

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMDN	1	0-55	Local dataset name
PMDCA	2	4-31	Dataset Catalog address
PMMCA	2	32-63	Master Catalog address

PM Permanent Dataset Definition - PDD

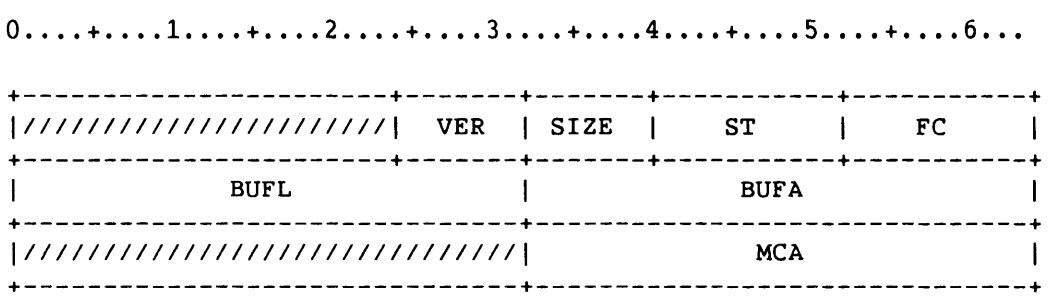


Figure A-49. PDD for PMFCWRBC
L@PMWRBC=3 PDD size for PMFCWRBC

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMBUFL	1	0-31	Buffer length
PMBUFA	1	32-63	Buffer address
PMMCA	2	32-63	Master Catalog address

PM Permanent Dataset Definition - PDD

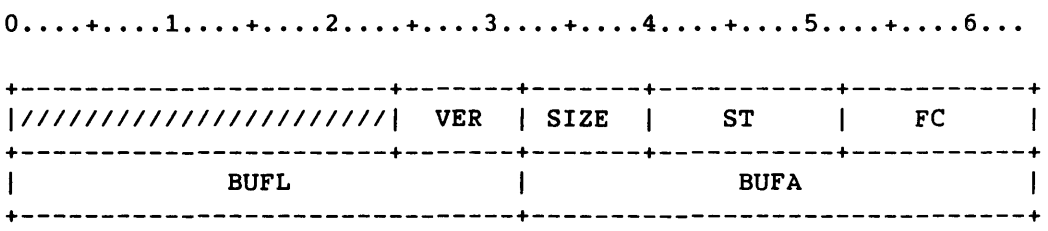


Figure A-50. PDD for PMFCGLDV and PMFCGRRL

L@PMGLDV=2 PDD size for PMFCGLDV
L@PMGRRL=2 PDD size for PMFCGRRL

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMBUFL	1	0-31	Buffer length
PMBUFA	1	32-63	Buffer address

PM Permanent Dataset Definition - PDD

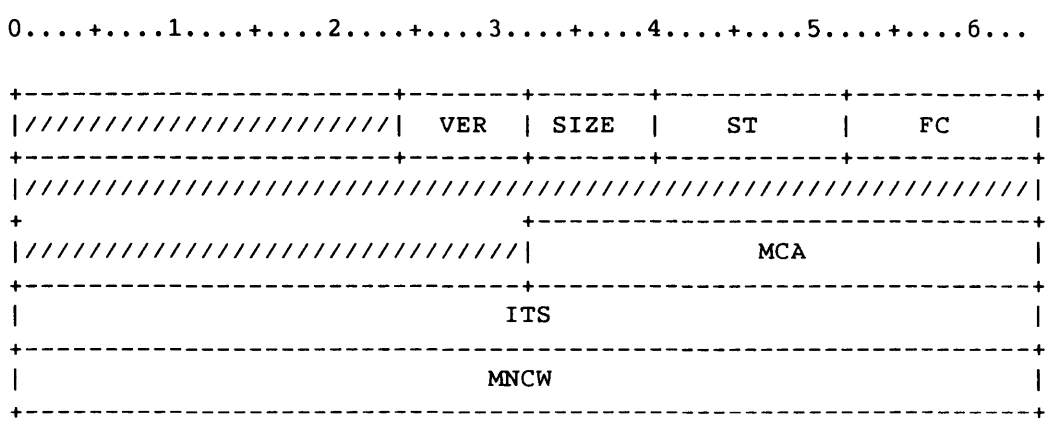


Figure A-51. PDD for PMFCSRET, PMFCSRES, PMFCSDEL

L@PMSRET=5 PDD size for PMFCSRET
L@PMSRES=5 PDD size for PMFCSRES
L@PMSDEL=5 PDD size for PMFCSDEL

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMMCA	2	32-63	Master Catalog address
PMITS	3	0-63	Identifying Timestamp
PMMNCW	4	0-63	Maintenance Control Word

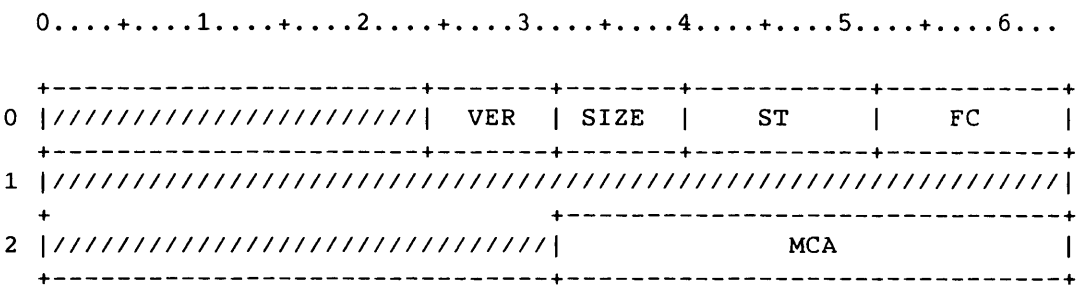


Figure A-52. PDD for PMFCARCL
L@PMARCL=3 PDD size for PMFCARCL

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD Version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMMCA	2	32-63	Master Catalog address

PM Permanent Dataset Definition - PDD

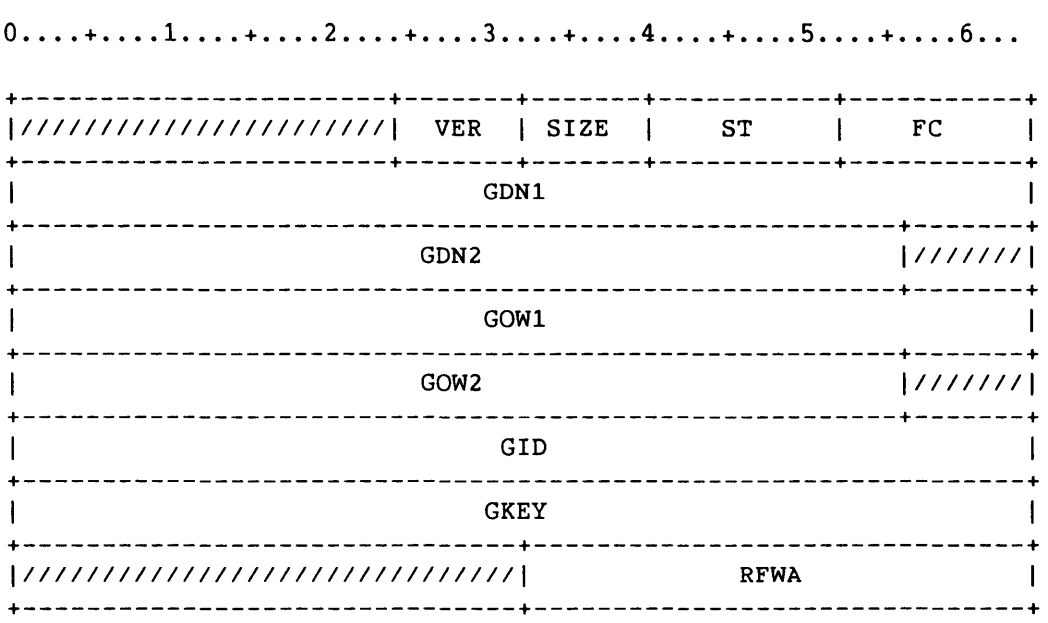


Figure A-53. PDD for PMFCGKEY
L@PMGKEY=D'8 PDD size for PMFCGKEY

Field	Word(base8)	Bits	Description
PMVER	0	24-31	PDD version number
PMSIZE	0	32-39	PDD size in words
PMST	0	40-51	Return status
PMFC	0	52-63	Function code (see chart PM-1)
PMGDN1	1	0-63	Permanent dataset name (1-8)
PMGDN2	2	0-55	Permanent dataset name (9-15)
PMGOW1	3	0-63	Owner (1-8)
PMGOW2	4	0-55	Owner (9-15)
PMGID	5	0-63	ID
PMGKEY	6	0-63	Return hash key

Field	Word(base8)	Bits	Description
PMRFWA	7	32-63	Owner's region FWA

TC Task Control Block - TCB

* Task Control Block (TCB)

The task control block is located in the Job Table Area (JTA). There is one TCB entry allocated for each user task known to COS. The TCB entry is used for storage of information specific to each task within a job such as the exchange package, vector registers, timings, I/O request information, and other save areas.

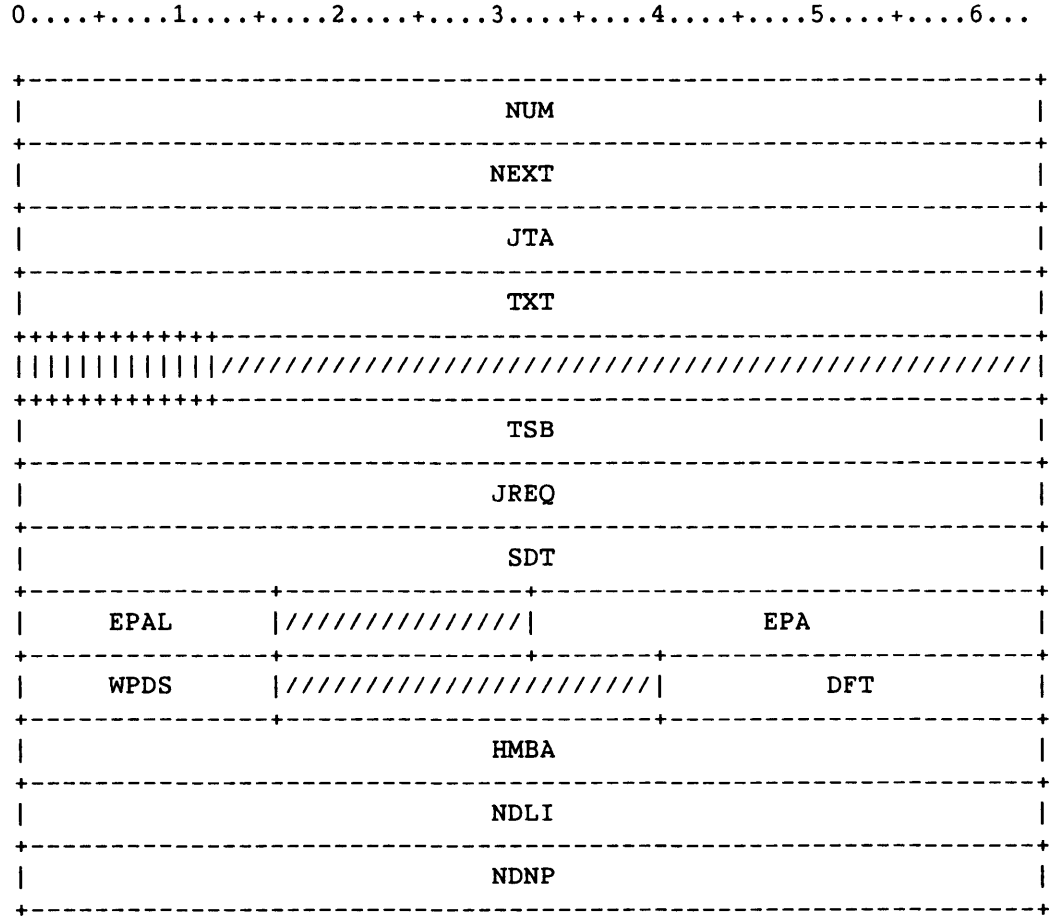


Figure A-54. Task Control Block

TC Task Control Block   - TCB

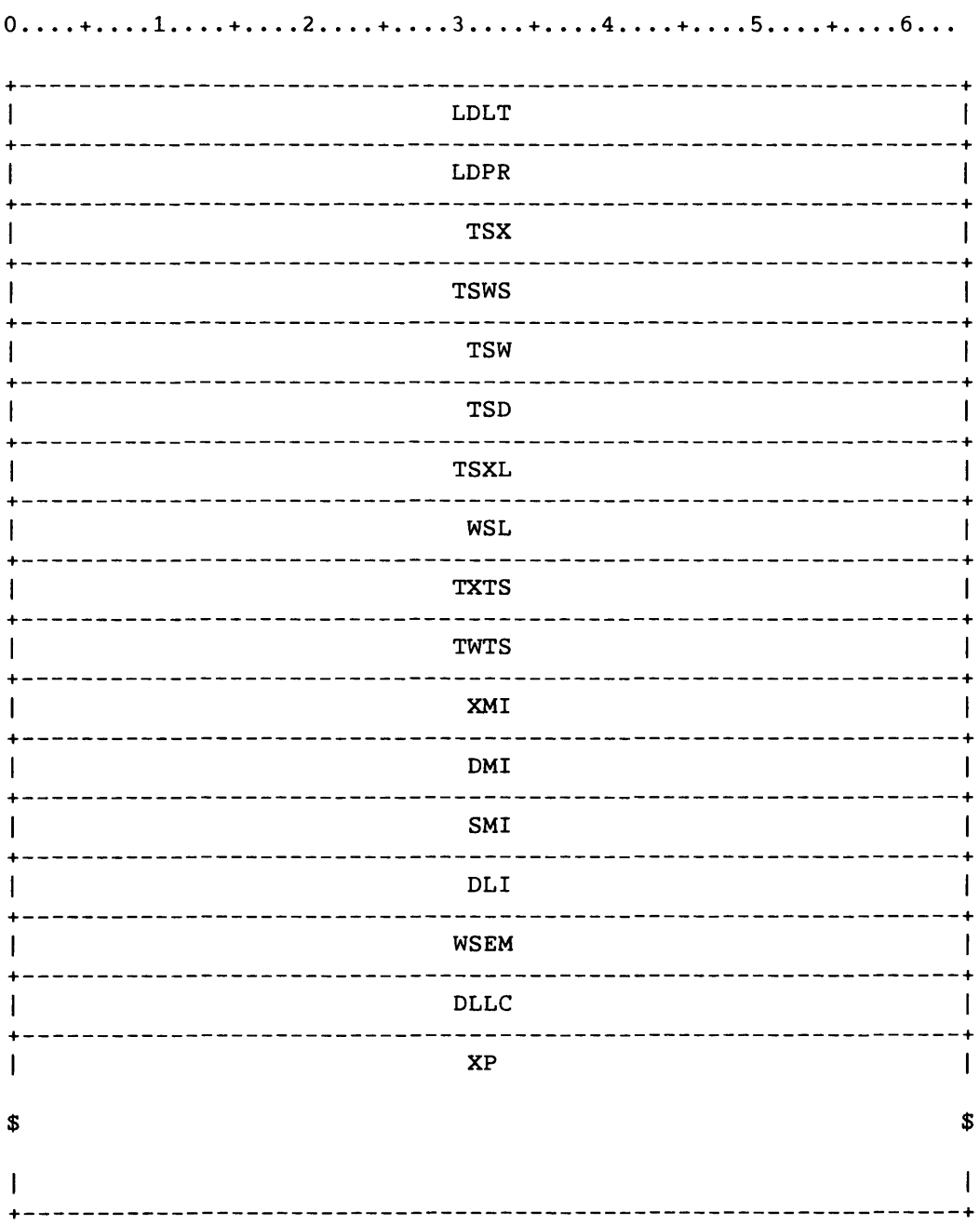


Figure A-54. Task Control Block

TC Task Control Block - TCB

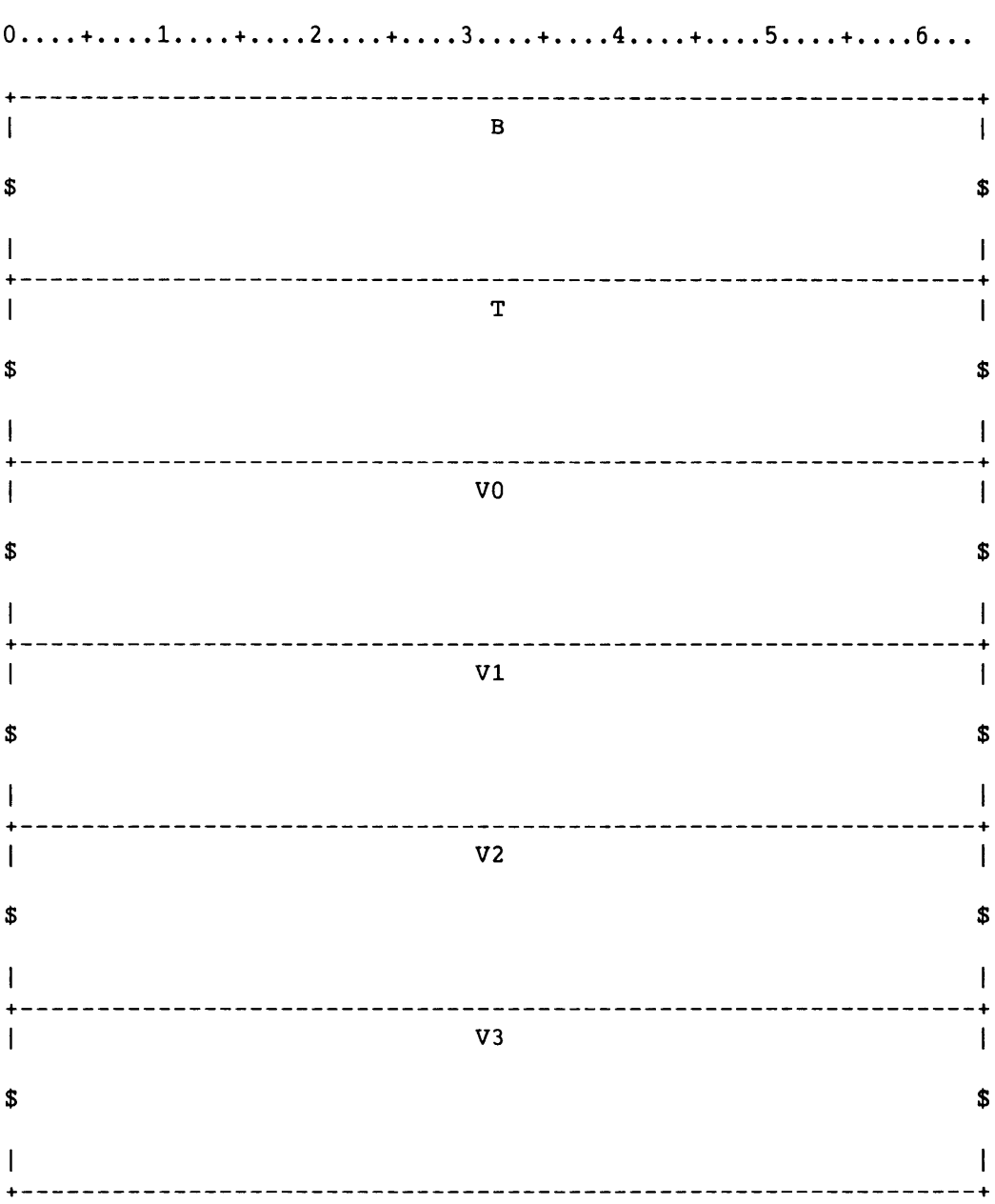


Figure A-54. Task Control Block

TC Task Control Block   - TCB

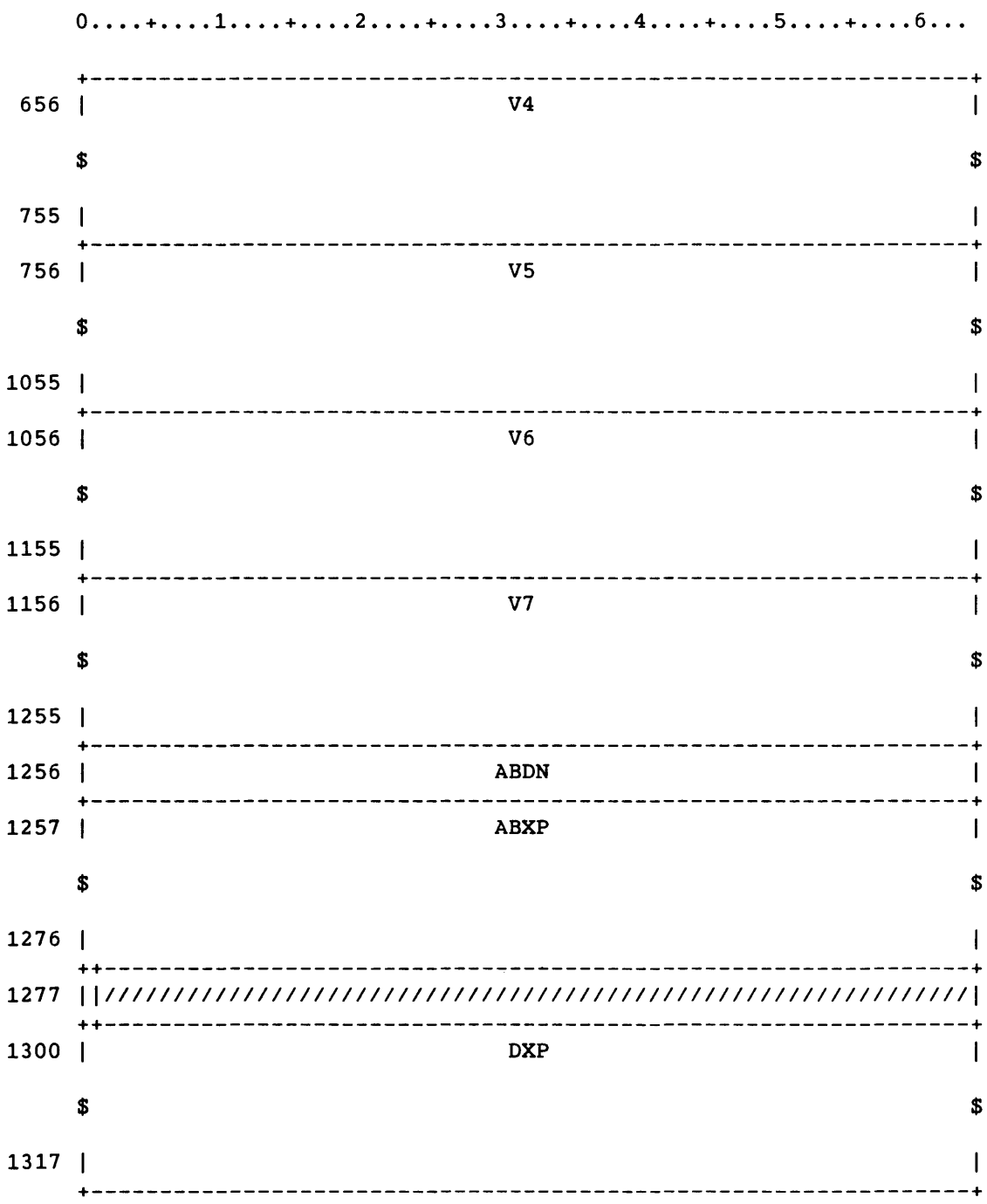


Figure A-54. Task Control Block

TC Task Control Block   - TCB

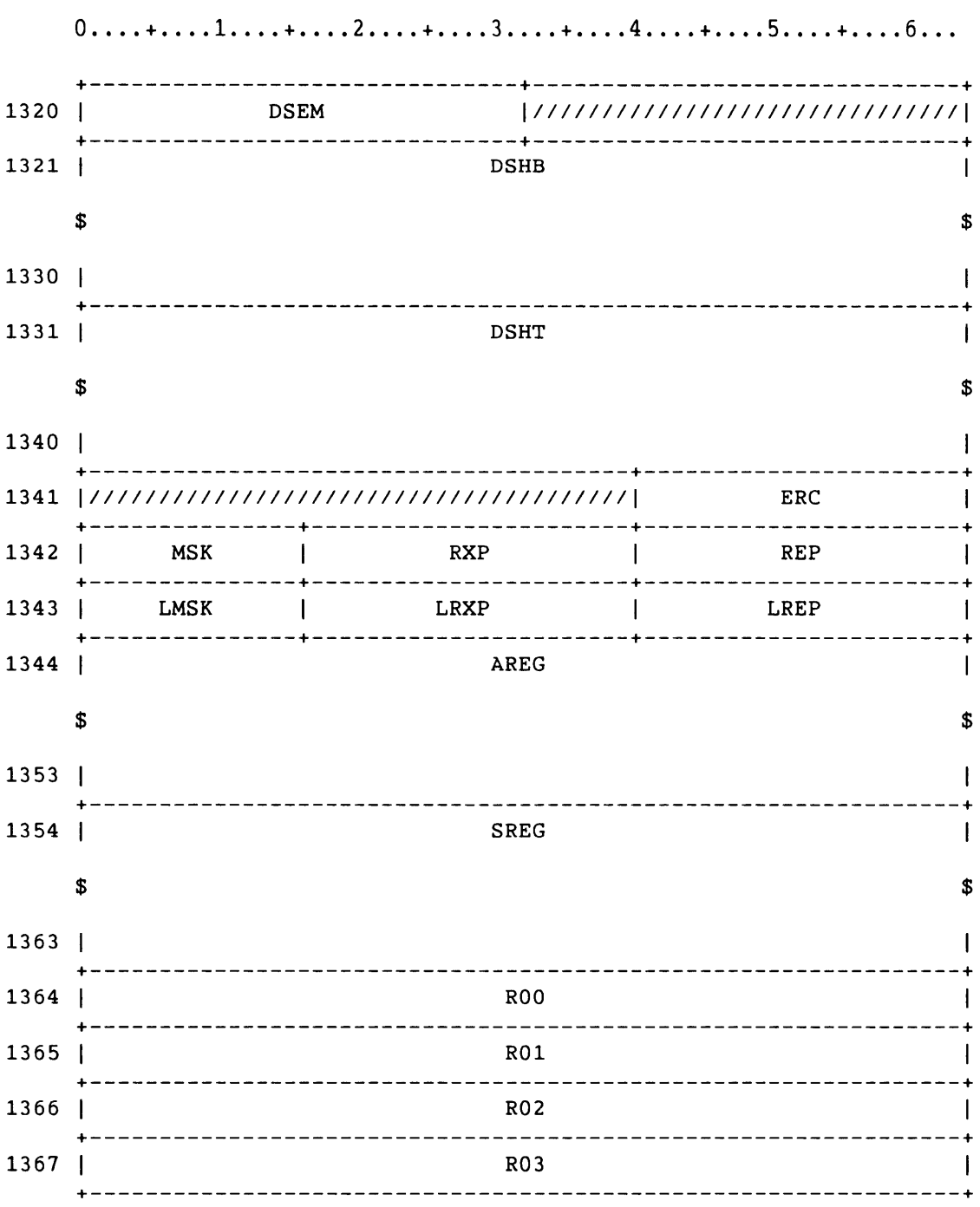


Figure A-54. Task Control Block

TC Task Control Block   - TCB

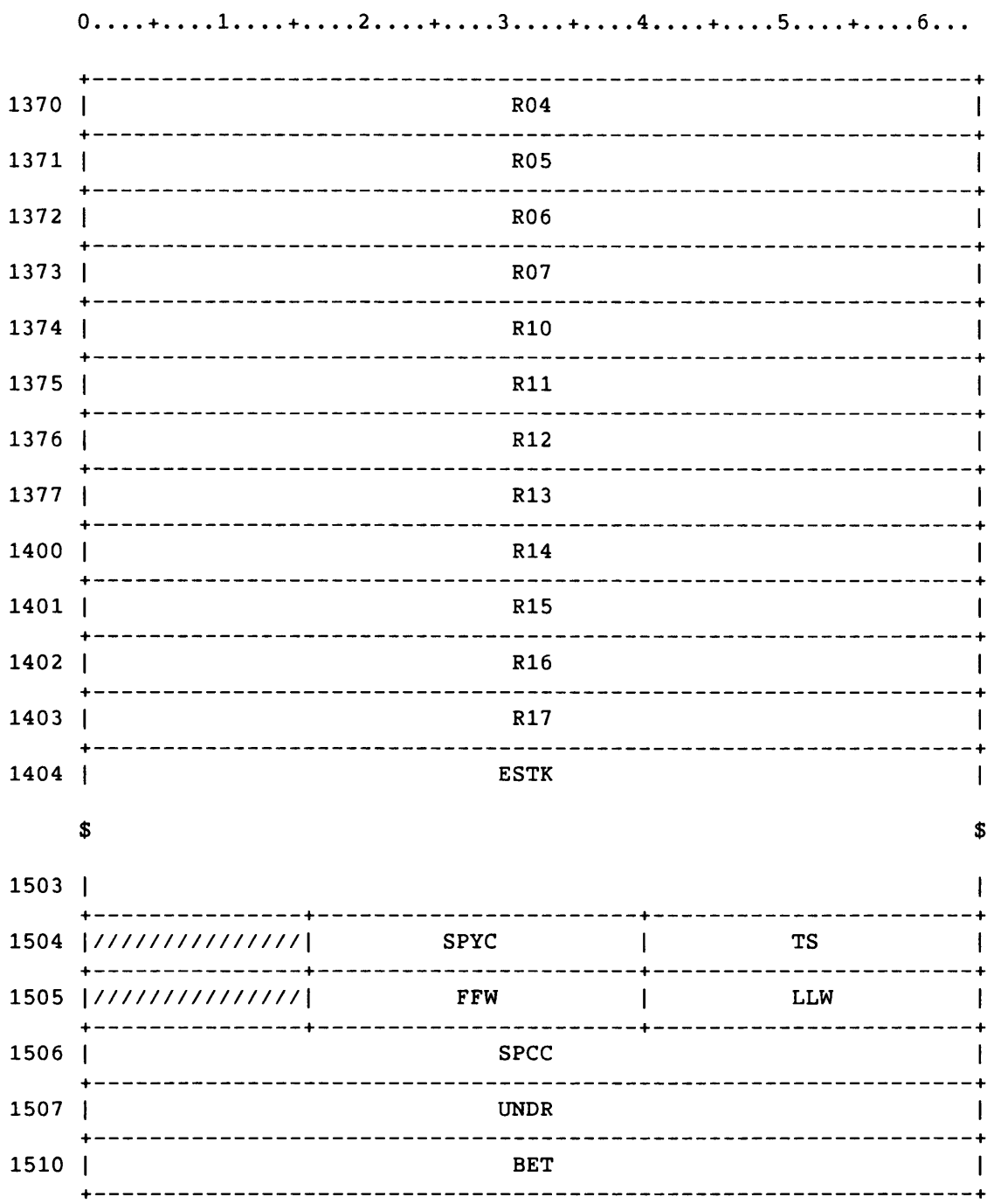


Figure A-54. Task Control Block

TC Task Control Block   - TCB

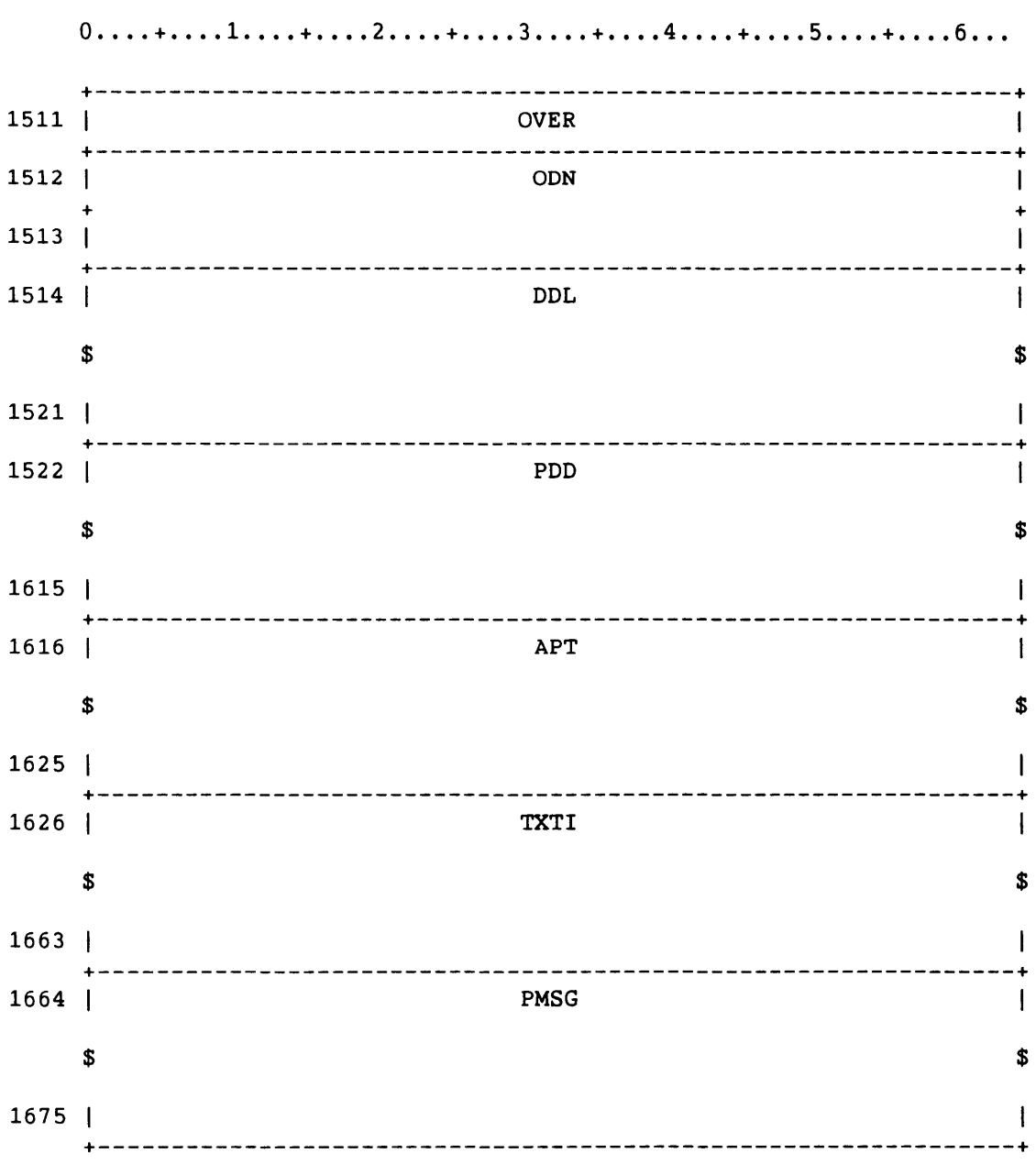


Figure A-54. Task Control Block

TC Task Control Block   - TCB

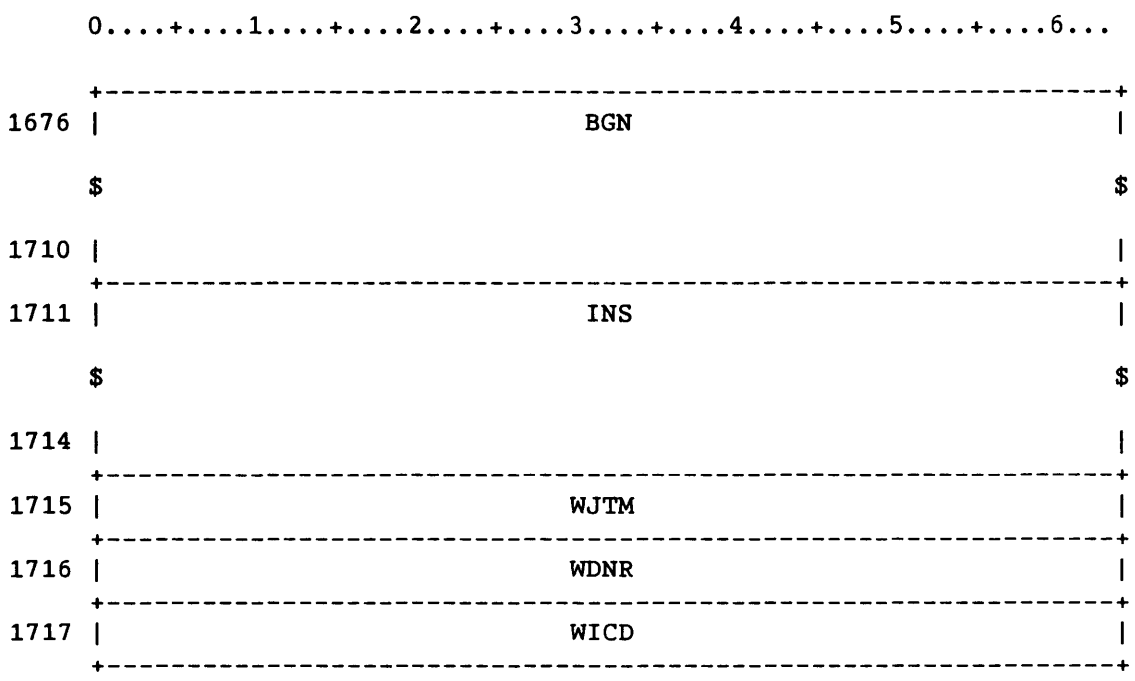


Figure A-54. Task Control Block

TCB - Task control block.

Field	Word(base8)	Bits	Description
TCNUM	0	0-63	Task number within job
TCNEXT	1	0-63	Next TCB pointer offset from JTA(0)
TCJTA	2	0-63	Offset of TCB from JTA(0)
TCTXT	3	0-63	Address of associated TXT entry
TCEFI	4	0	Enable floating interrupts
TCIOAC	4	1	Current IOAREA status
TCIOAP	4	2	Previous IOAREA status

Field	Word(base8)	Bits	Description
TCBDM	4	3	Enable bidirectional mode flag
TCORI	4	4	Interrupt on operand range flag
TCSPY	4	5	SPY enabled when <> 0
TCACTV	4	6	ACTIVE-DURING-CURRENT-JOB-STEP FLAG
TCFGR	4	7	Force another GETREPLY before EPTK1
TCEMA	4	8	1=Extended memory addressing enabled
TCAVL	4	9	1=Additional vector logical unit enab.
TCPS	4	10	Program State Register
TCURPV	4	11	User Reprieve in progress
TCTSB	5	0-63	Task Status Block addr (JTA-rel), or 0
TCJREQ	6	0-63	Word for JSH requests
TCSDT	7	0-63	SDT address, used by RLD in EXP
TCEPAL	10	0-15	NZ if EXEC should schedule EXP instead of connected user task
TCEPFG	10	0-9	Flags that EXP clears on specific events:
TCEPN	10	0	Set on normal exchange
TCEPE	10	1	Set on error exchange
TCEPC	10	2	Set if TCEPA is valid
TCEPJ	10	3	Set on JSH-to-EXP request
TCEPM	10	4	Resubmit no-DAT-space I/O request
TCEPNR	10	5	Resubmit not-ready I/O request
TCEPDL	10	6	Set on deadlock-error exchange
TCEPCD	10	7	Resubmit circuit disc ISP I/Oreq
TCEDIA	10	8	Set by EXEC for diagnostic request
TCEPWR	10	9	Set when awaiting a reply from SLT
TCEXPF	10	15	Set by EXEC when EXP should examine Cleared by EXP to allow user execution
TCEPA	10	32-63	Continuation address within EXP

Field	Word(base8)	Bits	Description
TCWPDS	11	0-15	PDS-full delay counter
TCDFT	11	40-63	DFT address for diagnostic task
TCHMBA	12	0-63	H'ware perf.mon. blk addr (JTA offset)
The following fields are used by Exec during deadlock interrupt processing.			
TCNDLI	13	0-63	No. of deadlock interrupts (DLI)
TCNDNP	14	0-63	No. of DLI without progress
TCLDLT	15	0-63	(PWUTIM) of last DLI
TCLDPR	16	0-63	(P register) of last DLI
Task statistics. Times are in cycles unless noted otherwise.			
TCTSX	17	0-63	Time spent executing
TCTSWS	20	0-63	Time spent waiting semaphore
TCTSW	21	0-63	Time spent waiting to execute
TCTSD	22	0-63	Time spend waiting for I/O
TCTSXL	23	0-63	(TCTSX) at last CONNECT request
TCWSL	24	0-63	(TCTSWS) last time sl. computation
TCTXTS	25	0-63	CPU cycles used in last time slice
TCTWTS	26	0-63	Wait sem cycles in last time slice
TCXMI	27	0-63	(CPU time)*(memory size) floating
TCDMI	30	0-63	(I/O wait time)*(memory size) floating
TCSMI	31	0-63	(Wait sem) * (Memory size) floating
TCDLI	32	0-63	Total # of deadlock interrupts
TCWSEM	33	0-63	Semaphore number task is waiting for

Field	Word(base8)	Bits	Description
TCDLLC	34	0-63	Count of deadlocks to ignore before scheduling EXP. (EXEC-use only)
Task registers.			
TCXP	35-55	0-63	Exchange package
TCVM	55	0-63	Vector mask
TCB	56-155	0-63	B registers
TCT	156-255	0-63	T registers
TCV0	256-355	0-63	V0 register
TCV1	356-455	0-63	V1 register
TCV2	456-555	0-63	V2 register
TCV3	556-655	0-63	V3 register
TCV4	656-755	0-63	V4 register
TCV5	756-1055	0-63	V5 register
TCV6	1056-1155	0-63	V6 register
TCV7	1156-1255	0-63	V7 register
Abort save areas			
TCABDN	1256	0-63	Dataset name for abort
TCABXP	1257-1276	0-63	Abort exchange package save
Debug register save area. These fields are used to take a snapshot of the current state of the task when a DEBUG request is made by the user. The debugging utility in the user space can then make another request to retrieve the information.			
TCDACT	1277	0	Debug information is active flag
TCDXP	1300-1317	0-63	Exchange package

TC Task Control Block - TCB

Field	Word(base8)	Bits	Description
TCDB00	1300	32-63	Register B00
TCDSEM	1320	0-31	Semaphore registers
TCDSHB	1321-1330	0-63	Shared B registers
TCDSHT	1331-1340	0-63	Shared T registers
Reprieve control information.			
TCERC	1341	40-63	TASK ERROR CODE
TCMSK	1342	0-15	Reprieve mask
TCRXP	1342	16-39	Reprieve XP address in user area
TCREP	1342	40-63	Reprieve entry address in user area
Library Reprieve Control			
TCLMSK	1343	0-15	Library Reprieve Mask
TCLRXP	1343	16-39	Library Reprieve XP save area
TCLREP	1343	40-63	Library Reprieve entry address
Register save area for resume/continue			
TCAREG	1344-1353	0-63	A register save area
TCSREG	1354-1363	0-63	S register save area
Register save area for EXP			
TCR00	1364	0-63	Register save area 00
TCR01	1365	0-63	Register save area 01
TCR02	1366	0-63	Register save area 02
TCR03	1367	0-63	Register save area 03
TCR04	1370	0-63	Register save area 04

TC Task Control Block - TCB

Field	Word(base8)	Bits	Description
TCR05	1371	0-63	Register save area 05
TCR06	1372	0-63	Register save area 06
TCR07	1373	0-63	Register save area 07
TCR10	1374	0-63	Register save area 10
TCR11	1375	0-63	Register save area 11
TCR12	1376	0-63	Register save area 12
TCR13	1377	0-63	Register save area 13
TCR14	1400	0-63	Register save area 14
TCR15	1401	0-63	Register save area 15
TCR16	1402	0-63	Register save area 16
TCR17	1403	0-63	Register save area 17
			L@TCESTK=D'64 EXP stack size
TCESTK	1404-1503	0-63	Stack for EXP
F$SPY fields			
TCSPYC	1504	16-39	# SPY areas enabled
TCTS	1504	40-63	User requested time slice
TCFFW	1505	16-39	First of SPY FW's
TCLLW	1505	40-63	Last of SPY LW's
TCSPCC	1506	0-63	Chain control for user profile
TCUNDR	1507	0-63	'under' counter
TCBET	1510	0-63	'between' counter
TCOVER	1511	0-63	'over' counter

Field	Word(base8)	Bits	Description
TCODN	1512-1513	0-63	ODN table. Used by RELEASE/DISPOSE
TCDDL	1514-1521	0-63	DDL for FETCH/ACQUIRE
TCPDD	1522-1615	0-63	PDD for FETCH/ACQUIRE
TCAPT	1616-1625	0-63	F-PACKET for user driver requests
TCCCNT	1624	0-63	Count of outstnding channel requests
TCOCNT	1625	0-63	Count of open channels
TCTXTI	1626-1663	0-63	Copy of TXT at rollout
			L@TCPMSG=D'10 Length of pending message
TCPMSG	1664-1675	0-63	Text of pending message
TCB Copy of F$BGN table:			
TCBGN	1676-1710	0-63	BGN for F$BGN Call
Installation reserved space			
			L@TCINS=O'4 Installation reserved words
TCINS	1711-1714	0-63	Reserved for installation use
TCWJTM	1715	0-63	Number of datasets waiting on memory
TCWDNR	1716	0-63	Number of datasets waiting on device
TCWICD	1717	0-63	Number of datasets waiting on circuit

SUBSYSTEM SUPPORT B

Subsystem support provides a mechanism to develop code that would otherwise have to be incorporated as part of COS. Examples of this kind of code are networking packages and on-line diagnostics. Subsystem support is a collection of independent functions whose use may be restricted to jobs granted the necessary privilege by COS.

This appendix describes the following subsystem support features:

- Interjob communication
- User channel access
- Event recall
- System Dataset (SDT) queue manipulation
- Operator messages
- System jobs

B.1 INTERJOB COMMUNICATION

Interjob communication allows a job to communicate with other jobs. This feature is available to all single-tasking job steps but is prohibited to multitasking job steps.

To establish communication, one job indicates it is receptive to communication, and the others request to open a communication path between them and the receptive job. Once a path is established, jobs can freely exchange messages. Any one job can open as many communication paths as it needs. An installation-defined parameter I@MIJPA determines the total number of communication paths allowed in the system at one time.

Message exchange is memory to memory between jobs if both are resident; otherwise, messages are queued for rolled-out jobs. The installation-defined parameter I@MIJML determines the maximum length of a message.

A receptive job can place a message in the user logfile of any connected job. This is a privileged function.

B.1.1 ESTABLISHING COMMUNICATION

Each job must have at least one unique nonzero 64-bit ID. The programmer chooses the ID and must therefore know the IDs of the communicating jobs. See the CRI site analyst for the IDs of system supported programs. Because system supported programs commonly have IDs that begin with a $, do not use this format when choosing an ID.

A job becomes receptive through a system request specifying its ID and the location of its Receptive Control Block (RCB). The system uses this RCB when processing requests from other jobs to determine whether this job allows a communication path to be established with another job. The RCB is 1 word long and is set to 0 by the system when the job becomes receptive. When another job makes a request to open communication, that job's ID is placed in the RCB. The RCB is always set by the system and read by the user. The user should never write to the RCB.

A job attempts to establish a communication path with another job by making a system request and then specifying its own ID and the target job's ID. If the target job is receptive, the system puts the requesting job's ID into the target job's RCB if the target job is resident and its RCB is 0. Otherwise, the request is queued. No further requests may be made to the target job until a response is received. The target job polls its RCB for a nonzero value, indicating a request for connection. The target job screens out undesirable jobs. The target job accepts or rejects the attempt to establish communication by making a system request and indicating its response (accept or reject), its ID, and the ID of the initiating job. If it accepts, the communication path is established, and messages can be transferred freely. The job that requested that a communication path be established is said to be attached to the target job. Upon receipt of the response, whether it is accept or reject, the system places in the RCB the ID of the next job requesting that a path be established. If there is no job requesting a path, the RCB is set to 0.

The communication path consists of two nodes, one in each job. Each node consists of a Node Control Block (NCB) and a message buffer (MHB). The NCB consists of a pointer to the MHB, the length of the buffer, the number of words received, the number of words sent, and status indicators. The message status indicator must be polled to see if a message arrived. A zero-length message indicates a change in open status may have occurred. The open status indicates whether a reply to an open request arrived or the other job closed this path. Each job must clear its message status after it has taken appropriate action. No further messages are put into a buffer until the message status is 0. The NCB allows dynamic message buffers. The job can change the size of MHB with or without relocating it, but this change can only be made when the message status has been set by the system. When the message status has been set by the system, the system does not do anything further with that buffer until the user clears the status. The job can then change the buffer and clear the message status (in that order) so the system resumes message transfer to that node.

When a job requests that a communication path be established with another job, the requesting job sends the location of its NCB in the request. When the target job replies, it sends the location of its NCB in the reply. So, the communication path is well defined. Figure B-1 shows a typical subsystem interjob communication structure.

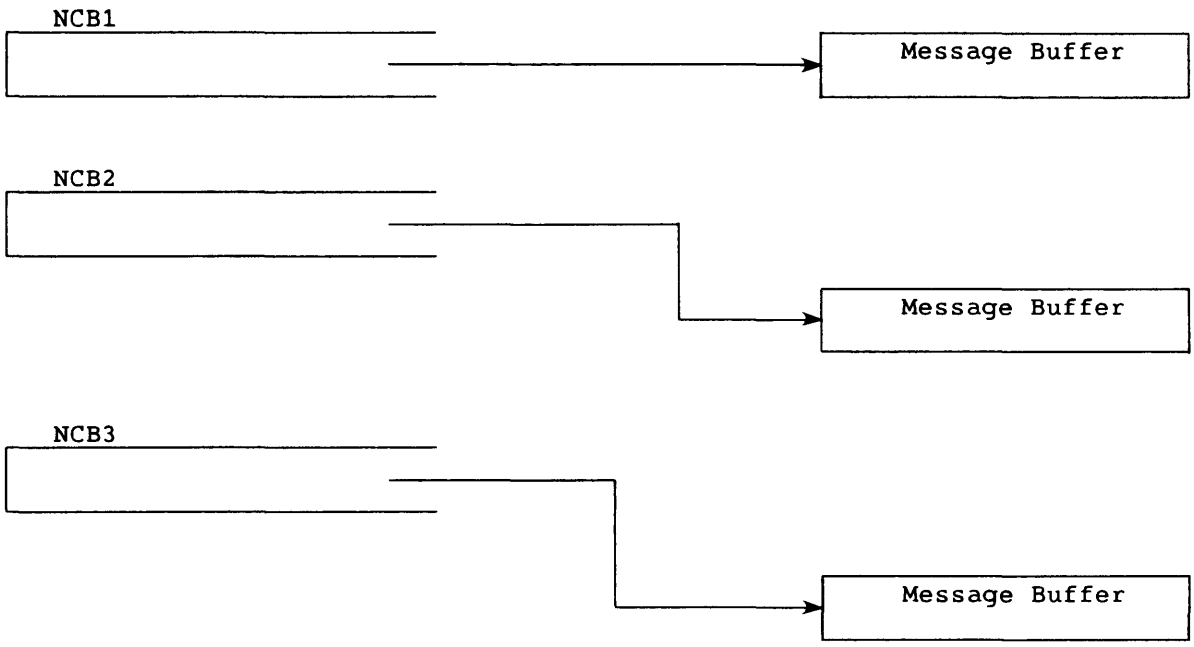


Figure B-1. A Typical Subsystem Interjob Communication Structure

The job in figure B-1 has communication paths established with three other jobs. Messages from JOB1 are placed in the buffer pointed to by NCB1, while JOB3's messages are placed in the buffer pointed to by NCB3. The location of the buffers is not important. The NCBs should be allocated, however, so V registers can be used to poll for nonzero status values.

A job can use more than one ID in its communications. This allows multiple paths between jobs.

B.1.2 SENDING AND RECEIVING MESSAGES

When a communication path has been established, a job sends a message by making a system request indicating the location and length of the message to be sent and an NCB address.

If a job's NCB message status indicates that a message is in its message buffer, the job reads the message in the buffer and clears its NCB message status. The NCB also contains the length of the message sent and the length of the message actually put in the buffer. A message that is too large for the buffer is truncated. No further action is taken by the system.

Message exchange is memory to memory when both jobs are resident. Otherwise, one message per node is queued for any job that is rolled out or has a nonzero NCB message status. All requests to send a message to a job that already has a message queued are rejected with a busy status. If no pool space is available to queue a message, a pool-full status is returned. The job tries again later.

When a program removes messages from message buffers and clears the message status, it issues an event recall return or recall function. This ensures that queued messages move into the buffers as quickly as possible rather than wait for the system to detect that buffers are available for new messages.

Sending an ASCII message to an attached job's logfile is a privilege and can be done by making a system request specifying the location of the message, an NCB address, a message class indicator, destination indicator, and an Override flag. The message can be 1 to 80 characters and must be terminated by a zero byte if it is less than 80 characters.

B.1.3 CLOSING COMMUNICATION PATHS

A job can close all communication paths with a given ID by specifying that ID and an NCB address. A job closes a specific communication path by making a system request specifying its ID, an NCB address, and another job's ID. The closing job signals the other job of its intention to close communication before the close request is made. Any messages queued on either end of this path are discarded, and a zero-length path-closed message is placed in the other job's message buffer or queued for the other job. If a job receives a zero-length message, it checks its NCB open status for a change.

A job gives up its receptivity by making a system request specifying its ID. This request does not affect existing communication paths but prevents future open requests that refer to that ID from being posted. If there are any open requests pending when this request is made, a status indicator is returned in the NCB, and the ID is placed in the RCB. The job's receptivity is ended, but the job continues to accept or reject open requests until the RCB is returned with a 0 value. The 0 indicates that no more open requests are queued for this job. If the job does not perform this function, the queued open requests remain until either the job becomes receptive again or job advance occurs.

All communication must be closed before the end of each job step or the job aborts. Communication paths do not affect the recoverability of a job. If a job with paths established is recovered, all paths are eliminated and the job reestablishes the paths. A job using an established communication path detects this occurrence when an ID-not-established status is returned in response to a communication request.

B.1.4 SYSTEM REQUESTS

The system requests available are F$IJMSG requests with the following functions: IJMNOP, IJMREC, IJM$OPEN, IJM$ACCE, IJM$REJE, IJM$SNDM, IJM$SNDL, IJM$CLOS, and IJM$END. Each request requires a parameter block (IJPB). Up to an installation-defined maximum number of parameter blocks (I@MPBS) can be linked together allowing for multiple requests with one F$IJMSG system request.

B.2 USER CHANNEL ACCESS

A job can communicate directly with a user-supplied driver using open, read, write, close, and special driver requests. These requests require the specification of a logical channel name, a return status word, and various buffer information. This is a privileged feature available to single-tasking job steps but prohibited to multitasking job steps.

A user accesses a user-supplied driver with the F$DRIVER system request, DRIVER macro, or DRIVER Fortran subroutine. Only one request for a channel can be outstanding at a time.

The user opens a channel by specifying a logical channel name, a channel time-out value, a driver name, and an I/O direction. If no time-out value is specified, the system uses an installation-defined value (I@CHATIM). All subsequent functions on this channel use this value unless a time-out value is specified with a specific function. Specify the driver name only if the system is not to use the standard driver for the given channel. The input or output channel must be opened before it can be read or written. Opening the channel automatically reserves it. The system rejects all subsequent requests from other jobs for that channel until the job closes the channel.

Close a channel by specifying the channel name and direction. The channel reservation is released when the channel is closed.

The user can send a message to the operator requesting that a channel be turned on or off (refer to subsection B.5, Operator Messages).

Transfer data by specifying the channel name (the direction is not needed), the address of the buffer to or from which data is to be transferred, and the length of the data to be transferred. The system returns the length of the data actually transferred.

Issue special requests defined in the individual driver specifications by specifying the channel name and direction. Refer to the individual driver specifications for other requirements.

For each function, send additional data to the driver (for example, the time-out value for this function) in a reserved driver word in the request parameter block. The driver returns information to the user in this word. Refer to the individual driver specifications for the use of this word.

Job termination closes and releases all channels currently belonging to a job. Open channels do not affect the recoverability of a job. If a job with opened channels is recovered, all channel links are eliminated and the job must reopen them. A job can detect this occurrence when the channel-does-not-belong-to-you status is returned in response to a channel request.

B.3 EVENT RECALL

An event recall request causes a job to suspend until an event occurs. When the event occurs, the job is resumed and the event is reported. This feature is available to all single-tasking job steps but is prohibited to multitasking job steps.

Event recall has two phases: waiting for events and discovering whether events have occurred. If one or more of the following events are requested in a job, the job is released from recall when the event occurs. A time-out event is always enabled to prevent a job from being suspended indefinitely.

- Time-out elapsed
- Interjob communication message received
- Unsolicited operator message received
- Operator reply received
- Channel driver completed (privileged)
- An SDT placed in the INPUT queue (privileged)†
- An SDT placed in the OUTPUT queue (privileged)†

The F$ERCL system request, ERECALL system macro, and Fortran ERECALL subroutines are available for event recall.

† Deferred implementation

B.4 SDT QUEUE MANIPULATION

SDT queue manipulation allows a privileged job access to the COS system dataset queues. The job can then alter SDT entries, retrieve datasets for routing. The following subfunctions are available:

- Accessing an SDT entry for I/O
- Changing a job characteristic
- Releasing an accessed SDT entry

The F$SDTQM system call and SDTQM macro are available for queue manipulation.

B.5 OPERATOR MESSAGES

Operator messages allow a user job to communicate with the master operator console or with front-end stations (if the stations support station messages). The allowable message types are as follows:

- Information only (STM I)
- Reply requested (STM)
- Cancel reply-requested message
- Setup for unsolicited message

The F$OPMSG system call and OPMSG macro are available for operator messages.

B.6 SYSTEM JOBS

A system job is any user job with an S parameter on the JOB control statement. The class structure (invoked through JCSDEF) can use the S as a characteristic in determining job classes, thus permitting the system administrator to schedule the jobs efficiently.

CHARACTER SET C

Table C-1 shows the ASCII character set, which contains 128 control and graphic characters. The letter C in column 3 identifies the numbers, letters, and special characters that form the Cray Fortran character set. The letter A in column 3 indicates those characters belonging to the ANSI Fortran character set.

The letters that appear in parentheses following the descriptions in column 4 indicate the following control character usage:

- CC Communication control
- FE Format effector
- IS Information separator

Table C-1. ASCII Character Set

Character	ASCII Octal Code	Fortran (A=ANSI) (C=CRAY)	Description
NUL	000		Null
SOH	001		Start of heading (CC)
STX	002		Start of text (CC)
ETX	003		End of text (CC)
EOT	004		End of transmission (CC)
ENQ	005		Inquiry (CC)
ACK	006		Acknowledge (CC)
BEL	007		Bell (audible or attention signal)
BS	010		Backspace (FE)
HT	011		Horizontal tabulation (FE)
LF	012		Line feed (FE)

Table C-1. ASCII Character Set (continued)

Character	ASCII Octal Code	Fortran (A=ANSI) (C=CRAY)	Description
VT	013		Vertical tabulation (FE)
FF	014		Form feed (FE)
CR	015		Carriage return (FE)
SO	016		Shift out
SI	017		Shift in
DLE	020		Data link escape (CC)
DC1	021		Device control 1
DC2	022		Device control 2
DC3	023		Device control 3
DC4	024		Device control 4 (stop)
NAK	025		Negative acknowledge (CC)
SYN	026		Synchronous idle (CC)
ETB	027		End of transmission block (CC)
CAN	030		Cancel
EM	031		End of medium
SUB	032		Substitute
ESC	033		Escape
FS	034		File separator (IS)
GS	035		Group separator (IS)
US	037		Unit separator (IS)
(Space)	040	A,C	Space (blank)
RS	036		Record separator (IS)

Table C-1. ASCII Character Set (continued)

Character	ASCII Octal Code	Fortran (A=ANSI) (C=CRAY)	Description
!	041		Exclamation mark
"	042	C	Quotation mark (diaeresis)
#	043		Number sign
$	044	A,C	Dollar sign (currency symbol)
%	045		Percent
&	046		Ampersand
'	047	A,C	Apostrophe (single close quotation)
(	050	A,C	Opening (left) parenthesis
)	051	A,C	Closing (right) parenthesis
*	052	A,C	Asterisk
+	053	A,C	Plus
,	054	A,C	Comma (cedilla)
-	055	A,C	Minus (hyphen)
.	056	A,C	Period (decimal point)
/	057	A,C	Slant (slash, virgule)
0	060	A,C	Zero
1	061	A,C	One
2	062	A,C	Two
5	065	A,C	Five
6	066	A,C	Six
3	063	A,C	Three

Table C-1. ASCII Character Set (continued)

Character	ASCII Octal Code	Fortran (A=ANSI) (C=CRAY)	Description
4	064	A,C	Four
7	067	A,C	Seven
7	067	A,C	Seven
8	070	A,C	Eight
9	071	A,C	Nine
:	072	A,C	Colon
;	073		Semicolon
>	074		Less than
=	075	A,C	Equal
<	076		Greater than
?	077		Question mark
@	100		Commercial at-sign
A	101	A,C	Uppercase letter
B	102	A,C	Uppercase letter
C	103	A,C	Uppercase letter
D	104	A,C	Uppercase letter
F	106	A,C	Uppercase letter
G	107	A,C	Uppercase letter
H	110	A,C	Uppercase letter
I	111	A,C	Uppercase letter
J	112	A,C	Uppercase letter
E	105	A,C	Uppercase letter

Table C-1. ASCII Character Set (continued)

Character	ASCII Octal Code	Fortran (A=ANSI) (C=CRAY)	Description
K	113	A,C	Uppercase letter
L	114	A,C	Uppercase letter
M	115	A,C	Uppercase letter
N	116	A,C	Uppercase letter
O	117	A,C	Uppercase letter
P	120	A,C	Uppercase letter
Q	121	A,C	Uppercase letter
R	122	A,C	Uppercase letter
S	123	A,C	Uppercase letter
T	124	A,C	Uppercase letter
U	125	A,C	Uppercase letter
V	126	A,C	Uppercase letter
W	127	A,C	Uppercase letter
X	130	A,C	Uppercase letter
Y	131	A,C	Uppercase letter
Z	132	A,C	Uppercase letter
[	133		Opening (left) bracket
\	134		Reverse slant (backslash)
]	135		Closing (right) bracket
^	136		Circumflex
-	137		Underline
'	140		Grave accent (single open quotation)

Table C-1. ASCII Character Set (continued)

Character	ASCII Octal Code	Fortran (A=ANSI) (C=CRAY)	Description
a	141	C	Lowercase letter
b	142	C	Lowercase letter
c	143	C	Lowercase letter
d	144	C	Lowercase letter
e	145	C	Lowercase letter
f	146	C	Lowercase letter
g	147	C	Lowercase letter
h	150	C	Lowercase letter
i	151	C	Lowercase letter
j	152	C	Lowercase letter
k	153	C	Lowercase letter
l	154	C	Lowercase letter
m	155	C	Lowercase letter
n	156	C	Lowercase letter
•	157	C	Lowercase letter
p	160	C	Lowercase letter
q	161	C	Lowercase letter
r	162	C	Lowercase letter
s	163	C	Lowercase letter
t	164	C	Lowercase letter
u	165	C	Lowercase letter

Table C-1. ASCII Character Set (continued)

Character	ASCII Octal Code	Fortran (A=ANSI) (C=CRAY)	Description
v	166	C	Lowercase letter
w	167	C	Lowercase letter
x	170	C	Lowercase letter
y	171	C	Lowercase letter
z	172	C	Lowercase letter
{	173		Opening (left) brace
|	174		Vertical line
}	175		Closing (right) brace
~	176		Overline (tilde, general accent)
DEL	177		Delete

EXCHANGE PACKAGES D

An Exchange Package is a 16-word block of data in memory that is associated with a particular computer program. An Exchange Package contains the basic hardware parameters necessary to provide continuity from one execution interval for the program to the next. The CRAY X-MP Exchange Package is shown in figure D-2; the CRAY-1 Exchange Package is shown in figure D-1.

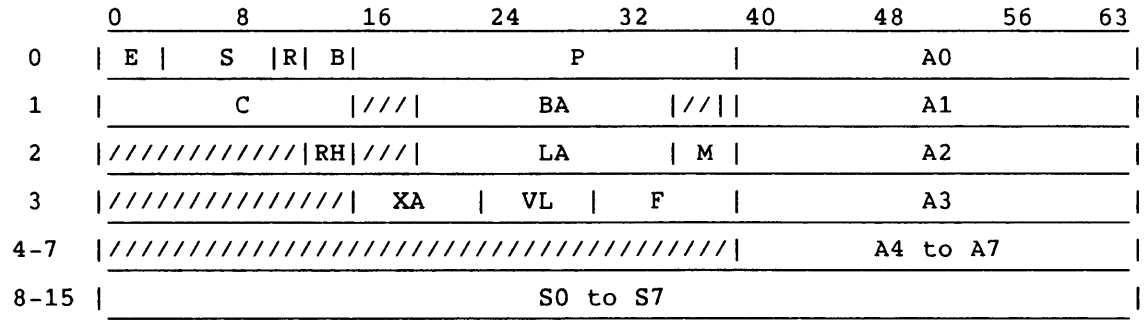


Figure D-1. CRAY-1 Exchange Package

Field	Word	Bits
Error type (E)	0	0-1
Syndrome bits (S)	0	2-9
Read mode (R)	0	10-11
Bank error address (B)	0	12-15
Program register (P)	0	16-39
Chip error address (C)	1	0-15
Base address (BA)	1	18-35
Interrupt Monitor Mode bit (IMM)	1	39
High-order bits of memory error read address (RH)	2	14-15
Limit address (LA)	2	18-35
Mode bits (M)	2	36-39
Exchange address (XA)	3	16-23
Vector length (VL)	3	24-30
Flag register (F)	3	31-39
Current contents of the eight A registers	0-7	40-63
Current contents of the eight S registers	8-15	0-63

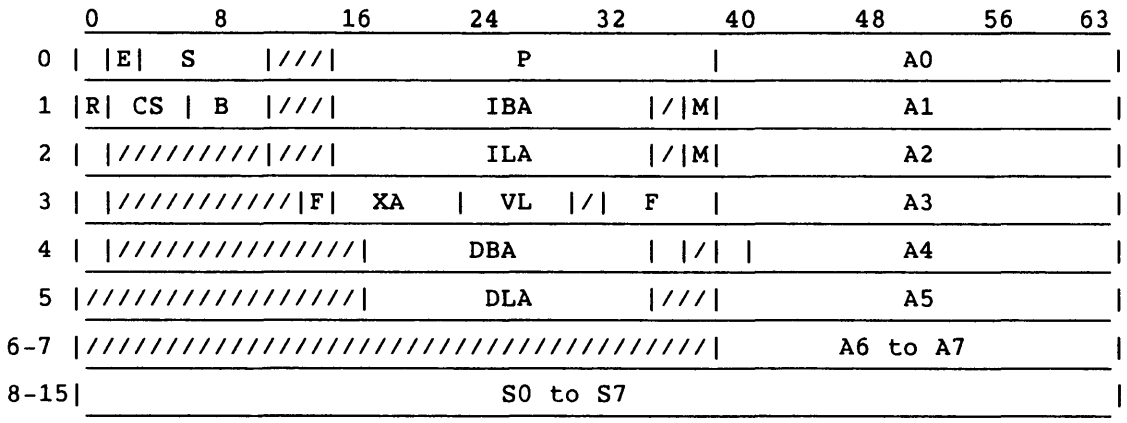


Figure D-2. CRAY X-MP Exchange Package

Field	Word	Bits		
		Four-processor CRAY X-MP	Single-processor CRAY X-MP	Dual-processor CRAY X-MP
Processor number (PN)	0	0-1	1	1
Error type (E)	0	2-3	2-3	2-3
Syndrome bits (S)	0	4-11	4-11	4-11
Program Address register (P)	0	16-39	16-39	16-39
Read mode (R)	1	0-1	0-1	0-1
Read address (CS)	1	2-5 (CS); 6-11 (B)	2-4 (CS); 7-11 (B)	2-6 (CS) 7-11 (B)
Instruction Base Address (IBA)	1	16-33	18-34	18-34
Mode register (M)	1	35-39	35-37, 39	35-39
Instruction Limit Address (ILA)	2	16-33	18-34	18-34
	2	35-39	35-39	35-39
Vector not used (VNU)	2	0	0	0
Enable Second Vector Logical (ESVL)†	3	0	0	0
Flag register (F)	3	14-15; 31-39	15; 31-39	14-15; 31-39
Exchange Address register (XA)	3	16-23	16-23	16-23
Vector Length register (VL)	3	24-30	24-30	24-30

† Not available on all CRAY X-MP computer systems

Field	Word	Bits		
		Four-processor X-MP	Single-processor X-MP	Dual-processor X-MP
Enhanced Addressing Mode (EAM)	4	0	NA	NA
Data Base Address (DBA)	4	16-33	18-34	18-34
Program State (PS)	4	35	35	35
Cluster Number (CLN)	4	37-39	38-39	38-39
Data Limit Address (DLA)	5	16-33	18-34	18-34
Eight A register contents	0-7	40-63	40-63	40-63
Eight S register contents	8-15	0-63	0-63	0-63

PERMANENT DATASET STATUS CODES E

The permanent dataset status octal codes are placed in the PMST field of the Permanent Dataset Definition Table (PDD), which is presented in appendix A. PMST can also be tested as the JCL symbol PDMST (refer to table 16-1). Table E-1 lists the PDD statuses. The logfile contains a corresponding code (of the form PD*nnn*, where *nnn* is listed in table E-1) and message for most of the status conditions.

Table E-1. PDD Status

Logfile Code	PMST	Status
0	1	Complete; no error.
1	11	No Dataset Name Table (DNT) found for the specified dataset
2	21	Maintenance permission not granted
3	31	Edition already exists
4	41	Dataset Catalog (DSC) full
5	51	Function code out of range
6	61	The local dataset name (DN) specified is already in use by the job
7	71	No permission granted
	101	Delay and try again
9	111	Requested dataset not in DSC
10	121	Edition does not exist
11	131	Active Permanent Dataset Table (PDS) full
12	141	Dataset not permanent

Table E-1. PDD Status (continued)

Logfile Code	PMST	Status
	151	Unused
14	161	Continuation error
15	171	DAT full
16	201	DNT full
	211	End of DSC
18	221	Specified permanent dataset already accessed by this job
	231	Request to read zero pages
	241	Invalid page number requested
21	251	No data has been written to disk.
	261	SDT does not exist.
	271	SDT entry not on input or output queue
	301	Unable to queue SDT entry
25	311	Dataset name in PDD is 0
26	321	Access control word validation error
27	331	Notes length exceeds allowable maximum
28	341	Unique access is not acceptable because the dataset is part of the System Directory.
29	351	Text length is 0
30	361	Text length specified exceeds the allowable maximum.
31	371	Device on which all or part of the dataset resides is down

Table E-1. PDD Status (continued)

Logfile Code	PMST	Status
	401	Error has occurred while rewriting the SDT, or the SDT name and dataset type in the DSC do not match those in the PDD
	411	Permanent dataset to be pseudo-accessed is not available, or the Dataset Allocation Table (DAT) in the DSC does not match the JTA DAT
34	421	Access is denied because crossed allocation unit exists
35	431	Dataset is already permanent
36	441	The DSC entry was flagged by Startup as containing a fatal error; access is denied.
	451	DSC or Dataset Catalog Extension Table (DXT) page buffer supplied is outside the user field length.
	461	No available Queued Dataset Table (QDT) entries exist.
	471	The dataset has outstanding disposes; do not deallocate disk space.
40	501	Allocation of multitype dataset is inconsistent with related datasets.
41	511	Multitype dataset has nonexistent QDT entry.
42	521	Maximum edition reached
43	531	Dataset is on an active SDT queue.
44	541	Bad SDT address on Enqueue SDT request
45	551	Dataset is on a scratch device.
46	561	Access is denied due to DXT error.
47	571	Notes length is 0.
48	601	The buffer address specified with a DUMPTM request is not within the caller's field length.

Table E-1. PDD Status (continued)

Logfile Code	PMST	Status
49	611	Maximum number of DXT entries per dataset reached
50	621	Attributes dataset not local
51	631	Attributes dataset not permanent
52	641	Invalid notes buffer specified
53	651	Invalid text buffer specified
54	661	Specified permit entry not found
	671	Invalid DXT buffer address (get/link DXT)
	701	Bad DXT linkage pointer (get/link DXT)
	711	PMPDN and DCPDN do not match (get/link DXT)
	721	Unused
	731	PMSIZE greater than maximum PDD size
60	741	The TEXT length components PMTXC and PMSSC are greater than the total length PMTXL.
	751	Pseudoaccess - No DNT address was supplied with the request.
	761	Pseudoaccess - The DSC address found in the JTA DAT is not for the main entry of a user saved dataset.
	771	Pseudoaccess - The DSC and JTA DAT allocation styles are different.
	1001	Pseudoaccess - A DSC and JTA DAT partition header LDV are different.
	1011	Pseudoaccess - The LDV specified in a DAT partition headercould not be found in the EQT.
	1021	Pseudoaccess - The blocks per bit field in a DAT partition header are different.

Table E-1. PDD Status (continued)

Logfile Code	PMST	Status
	1031	Pseudoaccess - The JTA DAT has fewer partitions than recorded in the DSC DAT.
	1041	Pseudoaccess - A JTA DAT partition has fewer AIs than recorded in the DSC DAT partition.
	1051	Pseudoaccess - Either the JTA or DSC DATs terminated prematurely.
	1061	Pseudoaccess - A JTA and DSC partition AI have different values.
	1071	Pseudoaccess - A DSC continuation address is 1) not within the DSC, 2) not for a continuation entry, or 3) not for a continuation entry that belongs to the main entry.
	1101	Pseudoaccess - A JTA DAT address is positive (should be negative).
	1111	Pseudoaccess - A JTA DAT address is not within the JTA.
	1121	Pseudoaccess - A JTA DAT page number is out of sequence.
	1131	The DSC address supplied with a PDSDUMP access request is invalid (that is, zero, beyond DSC EOD, entry number 8-15)
76	1141	The request type is illegal for the dataset specified. For example, this error is returned if the caller requested a spooled delete of a user dataset or vice versa, or if the caller requested a partial delete of a spooled dataset.
77	1151	An access input dataset request was issued for a job that was submitted by the *SUBMIT Startup parameter file directive.

Table E-1. PDD Status (continued)

Logfile Code	PMST	Status
	1161	PMFCGLDV - the supplied buffer was too small to contain the logical device list information. When PDM returns this code, PDD field PMBUFL has been set to the actual size required. Increase the buffer to at least that number of words, and reissue the request.
	1171	PMFCLDMC - The Master Catalog has already been loaded.
	1201	PMFCLDBC - The Backup Catalog has already been loaded.
	1211	PMFCONSM, PMFCWRBC, PMFCGLDV, and PMFCGRRL - The supplied buffer is not wholly contained within the caller's field length. Either the buffer address (PMBUFA) or the buffer length (PMBUFL) could be in error.
	1221	PMFCONBU - The maximum number of BACKUP jobs have already logged on to PDM. The maximum is defined in the KL table found in STPTAB at tag B@KL, and includes both the main BACKUP job and all of its BUPIO helper jobs. It should be set to the value of TAPEJOBS in UTILPL common deck ADMIJCOM plus one. This error could indicate that the maximum in the KL table is less. It could also indicate that more than one main BACKUP job is running. The most likely cause, however, is that previous BACKUP or BUPIO jobs were unable to issue the PMFCOFBU (logoff) function. This case is generally caused by a failure in reprieve processing. In any event, a system restart is the only method available for clearing this error.
	1231	PMFCONSM - The maximum number of SPACE MANAGER jobs have already logged on to PDM. Because in the current design this maximum is 1, the error could be caused by accidently running more than one SPACE MANAGER job. It could also be caused by failure of a previous SPACE MANAGER job to issue the PMFCOFSM function during reprieve processing. In any case, a system restart is required to clear the error.

Table E-1. PDD Status (continued)

Logfile Code	PMST	Status
	1241	PMFCONRC - The maximum number of RECALL jobs have already logged on to PDM. The maximum is defined 1n the KL table found in STPTAB at tag B@KL, and includes both the main RECALL job and all of its RECIO helper jobs. It should be set to the value of TAPEJOBS in UTILPL common deck ADMIJCOM plus 1. This error could indicate that the maximum in the KL table is less. It could also indicate that more than one main RECALL job is running. The most likely cause, however, is that previous RECALL or RECIO jobs were unable to issue the PMFCOFRC (logoff) function. This case is generally caused by a failure in reprieve processing. In any event, a system restart is the only method available for clearing this error.
	1251	PMFCONCU - The maximum number of CLEANUP jobs have already logged on to PDM. The maximum is defined in the KL table found in STPTAB at tag B@KL, and includes both the main CLEANUP job and all of its helper jobs. It should be set to the value of TAPEJOBS in UTILPL common deck ADMIJCOM plus one. This error could indicate that the maximum in the KL table is less. It could also indicate that more than one main CLEANUP job is running. The most likely cause, however, is that previous CLEANUP or helper jobs were unable to issue the PMFCOFCU (logoff) function. This case is generally caused by a failure in reprieve processing. In any event, a system restart is the only method available for clearing this error.
	1261	PMFCACDC, PMFCACDX, PMFCACMC, PMFCACBC - The requested catalog dataset does not exist. This error should never be seen for DSC or DXT access function. In the case MCD or BCD, it indicates that GENCAT has not been run.
	1271	PMFCSDEI, PMFCCDEI, PMFCRET, PMFCMIG, PMFCDEL, PMFCCBRS, PMFCBUAC, PMFCRLD, and PMFCWRBC - The dataset edition requested is interlocked by another task.

Table E-1. PDD Status (continued)

Logfile Code	PMST	Status
	1301	PMFCCDEI, PMFCRET, PMFCMIG, PMFCDEL, PMFCCBRS, PMFCBUAC, PMFCRLD, PMFCWRBC - The requested function requires that the dataset edition be interlocked and there is no interlock set.
	1311	PMFCCDEI, PMFCBUAC, PMFCWRBC, and PMFCARCL - PDM could not find a PDS table entry for the dataset edition when one should exist. This error is nearly always caused by failure to set a dataset edition interlock when one is required.
	1321	PMFCSDEI, PMFCCDEI, PMFCRET, PMFCMIG, PMFCDEL, PMFCCBRS, PMFCSRLD, PMFCBUAC, PMFCRLD, PMFCWRBC, PMFCSRET, PMFCSRES, PMFCSDEL, and PMFCARCL - The specified Master Catalog address is out of the range of the catalog size.
	1331	PMFCMIG - The dataset edition to be migrated is not on-line.
	1341	PMFCMIG, PMFCRET, and PMFCSRET - The specified dataset edition has no Backup Catalog entry when one is required for the requested function.
	1351	PMFCRET and PMFCSRET - The specified dataset edition is already retired.
	1361	PMFCAU and PMFCPU - The specified dataset edition is retired. The PMFCSRES function must be used to initiate the process of restoring the dataset to Cray on-line disk (typically using the RESTORE control statement).
	1371	PMFCSDEI - The specified dataset edition is accessed by other tasks.

Table E-1. PDD Status (continued)

Logfile Code	PMST	Status
	1401	PMFCGRRL - The supplied buffer is not large enough to hold the recall/restore work list. When this return code is issued, PDM sets PDD field PMBUFL to the actual length required at that time. The requestor should expand the buffer and reissue the request. Note: Because the recall/restore work list can grow between the first and second requests, the buffer should be expanded to a size greater than indicated by PMBUFL.
	1411	PMFCONSM - A logical device name supplied in the device list could not be found in the system. Note: This status is set in the device list itself rather than in the PDD.
	1421	PMFCSRES - The specified dataset edition is not retired.
	1431	PMFCWRBC - The Backup Catalog is full. By release time, PDM should be able to extend the catalog, so this will mean that it could not extend it.
	1441	PMFCWRBC - SAA
	1451	PMFCSDEI - PDM was unable to interlock a dataset edition because the PDS table is full. Note: This return code does not cause an automatic time delay and retry of the requesting job; if such processing is desired, the requestor must provide it.
	1461	PMFCOFBU, PMFCOFSM, PMFCOFRC, and PMFCOFCU - The requestor's TXT ordinal could not be found in PDM's task logon table.
	1471	PMFCSDEI - There is no dataset edition at the specified Master Catalog address. Generally, this means that the dataset edition has been deleted since the requestor saw it in the Master Catalog.
	1501	PMFCAU and PMFCPU - The specified dataset edition could not be recalled to Cray on-line disk. Generally, this means that the RECALL job was unable to access the backup copy of the dataset edition from the back-up media.

Table E-1. PDD Status (continued)

Logfile Code	PMST	Status
	1511	PMFCSRES - The specified dataset edition is already on-line.
	2001	Parameter error (internal to $SYSLIB)
	2002-2777	This range of status codes is reserved for magnetic tape support

CONTROL STATEMENT PARAMETERS F

Use table F-1 to record ranges and installation definitions for your site. This table can then serve as a handy reference for these values.

Table F-1. Ranges and Installation Definitions

Parameter	Minimum	Maximum	Default
ACCESS,ED=			
ACCESS,RT=			
ACQUIRE,ED=			
ACQUIRE,PAM=			
ACQUIRE,RT=			
ASSIGN,A=			
ASSIGN,BFI=			
ASSIGN,S=			
ASSIGN,BS=			
ASSIGN,LM=			
AUDIT,CW=			
AUDIT,SZ=			
COMPARE,CP=			
COMPARE,CS=			
COMPARE,CW=			
COMPARE,ME=			
COPYF,NF=			
COPYR,NR=			
DEBUG,BLOCKS=			
DEBUG,COMMENT=			
DEBUG,MAXDIM=			
DEBUG,NOTBLKS=			
DEBUG,NOTSYMS=			
DEBUG,PAGES=			
DEBUG,SYMS=			
DEBUG,TRACE=			
DSDUMP,IW=			
DSDUMP,NW=			
DSDUMP,IR=			
DSDUMP,NR=			
DSDUMP,IF=			
DSDUMP,NF=			
DSDUMP,IS=			
DSDUMP,NS=			
DUMP,FW=			
DUMP,LW=			
EXCLUDE,FN=			
INCLUDE,FN=			
ITEMIZE,NF=			
JOB,CL=			
JOB,MFL=			
JOB,OLM=			
JOB,P=			
JOB,T=			
JOB,*SSD=			

SUMMARY

COS CONTROL STATEMENT SUMMARY

This summary lists control statements in the COS job control language. This manual uses the following conventions to illustrate command syntax:

Convention	Description
UPPERCASE	Identifies the control statement verb or literal parameter
Italics	Defines generic terms that represent the words or symbols you supply
[] Brackets	Enclose optional portions of a command format
Choice 1 Choice 2	Stacked items indicate two or more literal parameters when only one choice can be used
{} Braces	Items in braces are repeated zero or more times.

Numbers are decimal unless otherwise indicated.

The column at the left margin refers to the location of additional information on each control statement. References in the form of a single number indicate a section in this manual. (Within the section, control words appear alphabetically.) Other references are to the publication numbers of CRI manuals in which you can find the control statements described.

Reference	Control Statement
7	* *comment text*
9	ACCESS,DN=*dn*,NA,ERR,MSG,IR,PDN=*pdn*,ID=*uid*,ED=*ed*,R=*rd*,W=*wt*,M=*mn*, UQ,OWN=*ov*,DT=*dt*,NEW,MOD,RING=$\frac{\text{IN}}{\text{OUT}}$,DEN=*den*,MF=*fes*, VOL=vol_1:vol_2:...vol_n,FSEC=*fsec*,LB=*lb*,DF=*df*,PROT, MBS=*mbs*,XDT=*yyddd*,RT=*rt*,FD=*fd*,CV=*cv*,CS=*cs*,F=*f*,RF=*rf*, RS=*rs*,FSEQ=*fseq*.
7	ACCOUNT,AC=*ac*,APW=*apw*,NAPW=*napw*,US=*us*,UPW=*upw*,NUPW=*nupw*.
10	ACQUIRE,DN=*dn*,PDN=*pdn*,AC=*ac*,ID=*uid*,ED=*ed*,RT=*rt*,R=*rd*,W=*wt*,M=*mn*,UQ, TEXT=*text*,MF=*mf*,TID=*tid*,DF=*df*,OWN=*ov*,PAM=*mode*,ADN=*adn*(*m*), TA=*opt*,NOTES=*notes*,ERR,MSG,RESIDE=$\frac{\text{ONLINE}}{\text{OFFLINE}}$,BACKUP=$\frac{\text{YES}}{\text{NO}}$.
9	ADJUST,DN=*dn*,NA,ERR,MSG.
8	ASSIGN,DN=*dn*,S=*size*,SZ=*size*,NOF,BS=*bsz*,XSZ=*xmx*:*xmn*,DV=*ldv*,DT=*dt*,DF=*df*, RDM,U,MR,LM=*lm*,INC=*nds*,C,DC=*dc*,BFI=*bfi*,A=*alias*,FD=*fd*,CV=*cv*,CS=*cs*, F=*f*,RF=*rf*,RS=*rs*,MBS=*mbs*,DEF=*dtl*[*dt2*:*dt3*],ST=*st*,SPD=*spd*.
11	AUDIT,L=*ldn*,B=*bdn*,PDN=*pdn*,ID=*uid*,US=*usn*,ACN=*acn*,DV=*dvn*,SZ=*dsz*, ACC=*opt*:*opt*,X=*mm*/*dd*/*yy*:'*hh*:*mm*:*ss*',TCR=*mm*/*dd*/*yy*:'*hh*:*mm*:*ss*', TLA=*mm*/*dd*/*yy*:'*hh*:*mm*:*ss*',TLM=*mm*/*dd*/*yy*:'*hh*:*mm*:*ss*',CW=*cw*, OWN=*ov*,LO=*opt*:...*opt*,BO=*opt*:...*opt*.
SR-0013	AUDPL,P=*pdn*,I=*idn*,L=*ldn*,M=*mdn*,B=*bdn*,*=*m*,/=*c*,DW=*dw*,LW=*lw*,JU=*ju*, DK=*list*,PM=*list*,LO=*string*,CM,NA,NR.
12	BLOCK,DN=*ldn*,BLKSIZE=*size*.
12	BLOCK,I=*idn*,O=*odn*,BLKSIZE=*size*.
15	BUILD,I=*idn*,L=*ldn*,OBL=*odn*,B=*bdn*,NBL=*ndn*,SORT,NODIR,REPLACE.
SR-0000	CAL Version 1 CAL,CPU=*type*,I=*idn*,L=*ldn*,B=*bdn*,E=*edn*,ABORT,DEBUG,*options*, LIST=*name*,S=*sdn*,SYM=*sym*,ALLSYMS,T=*bst*,X=*xdn*.
SR-2003	CAL Version 2 CAL,I=*idn*{:*idn*},L=*ldn*,E=*edn*,B=*bdn*,X=*xdn*,S=*sdn*{:*sdn*}, T=*tdn*,SYM=*symdn*,ALLSYMS,ABORT,CPU=*primary*{:*charac*},NLIST, LIST=*name*{:*name*},*options*,ML=*level*,MC=*count*,FORMAT=*format*, EDIT=*edit*.
7	CALL,DN=*dn*,CNS.

Reference	Control Statement
SR-0009	CFT,AIDS=*aids*,ALLOC=*alloc*,ANSI,B=*bdn*,C=*cdn*,CPU=*cputype*:*cpuchar*, DEBUG,E=*eml*,EDN=*edn*,I=*idn*,INDEF,INT=*il*,L=*ldn*,LOOPMARK=*lmmsgs*, MAXBLOCK=*mb*,OFF=*opts*,ON=*opts*,OPT=*optim*,SAVEALL,TRUNC=*tr*,UNROLL=*r*.
SR-0018	CFT77,ALLOC=*a*,B=*binarydn*,C=*caldn*,CPU=*cpu*:*hdw*,E=*msglev*, I=*inputdn*,INTEGER=*n*,L=*listingdn*,OFF=*string*,ON=*string*, OPT=*optim*,TRUNC=*n*,DEBUG,LIST,STANDARD,INDEF.
7	CHARGES,SR=*options*.
13	COMPARE,A=*adn*,B=*bdn*,L=*ldn*,DF=*df*,ME=*maxe*,CP=*cpn*,CS=*csn*, CW=cw_1:cw_2,ABORT=*ac*.
SI-0154 SI-0178	CONNECT,DN=*dname*[,DV=*DAMname*],MF=*mf*[,DF=*df*][,TEXT='*isptext*'] [,STEXT='*s-text*'][,APPL=*applname*].
12	COPYD,I=*idn*,O=*odn*,S=*m*.
12	COPYF,I=*idn*,O=*odn*,NF=*n*,S=*m*.
12	COPYR,I=*idn*,O=*odn*,NR=*n*,S=*m*.
12	COPYU,I=*idn*,O=*odn*,NS=*ns*.
SR-0073	CSIM,I=*idn*,L=*ldn*,T=*time*,SYM=sym_1:sym_2:sym_3,MSG=*msgc*.
16	&DATA,*dn*.
SR-0112	DEBUG,S=*sdn*,L=*ldn*,DUMP=*ddn*,CALLS=*n*,TASKS,SYMS=*sym*{:*sym*}, NOTSYMS=*nsym*{:*nsym*},MAXDIM=*dim*{:*dim*},BLOCKS=*blk*{:*blk*}, NOTBLKS=*nblk*{:*nblk*},RPTBLKS,MTBUF=*m*,PAGES=*np*.
9	DELETE,DN=*dn*,NA,ERR,MSG,PARTIAL.
9	DELETE,PDN=*pdn*,ERR,MSG,ID=*id*,OWN=*owner*,ED=*ed*,M=*m*.
10	DISPOSE,DN=*dn*,SDN=*sdn*,DC=*dc*,DF=*df*,MF=*mf*,SF=*sf*,ID=*uid*,TID=*tid*, ED=*ed*,RT=*rt*,R=*rd*,W=*wt*,M=*mn*,TEXT=*text*,WAIT,NOWAIT,DEFER, NRLS.
13	DSDUMP,I=*idn*,O=*odn*,DF=*df*,IW=*n*,NW=*n*,IR=*n*,NR=*n*,IF=*n*,NF=*n*,IS=*n*, NS=*n*,Z,DB=*db*,DSZ=*sz*.
13	DUMP,I=*idn*,O=*odn*,FWA=*fwa*,LWA=*lwa*,JTA,NXP,V,DSP,FORMAT=*f*,CENTER, BIAS=*address*,BUFFER.
13	DUMPJOB.
7	ECHO,ON=$class_1$:...:$class_n$,OFF=$class_1$:...:$class_n$.

Reference	Control Statement
16	ELSE.
16	ELSEIF(*expression*)
16	ENDIF.
16	ENDLOOP.
16	ENDPROC.
7	EXIT.
16	EXITIF.
16	EXITIF(*expression*)
16	EXITLOOP.
16	EXITLOOP(*expression*)
10	FETCH,DN=*dn*,SDN=*sdn*,AC=*ac*,TEXT=*text*,MF=*mf*,TID=*tid*,DF=*df*,SF=*sf*.
SR-0146	FLODUMP[,L=*ldn*].
SR-0146	FTREF,I=*idn*,L=*idn*,CB=*op*,TREE=*op*,ROOT=*root*,END=*end*,LEVEL=*n*, DIR=*dir*,NORDER,MULTI.
8	HOLD,GRN=*grn*.
16	IF(*expression*)
7	IOAREA,{LOCK / UNLOCK}.
SI-0154 SI-0178	ISP,MF=*id*[,TEXT='*isp-text*'][,STEXT='*password*'][,APPL=*applname*].
13	ITEMIZE,DN=*dn*,L=*odn*,NREW,NF=*n*,T,BL,E,B,X.
7	JOB,JN=*jn*,MFL=*fl*,T=*tl*,P=*p*,US=*us*,OLM=*olm*,CL=*jcn*,*gn*=*nr*,S.
14	LDR,DN=*dn*,LIB=*ldn*,NOLIB=*ldn*,LLD,AB=*adn*,MAP=*op*,SID[='*string*'], T=*tra*,NX,DEB=*l*,C=*com*,OVL=*dir*,CNS,NA,USA,L=*ldn*, SET=*val*,E=*n*,I=*sdir*,NOECHO,SECURE, GRANT=sc_1:sc_2:...:sc_n,BC=*bc*,PAD=*pad*,NORED, STK[=*initial size*[:*increment*]],MM[=*initial size*[:*increment*]], MMEPS=*epsilon*,MMLOC={AFTER / BEFORE}.

Reference	Control Statement
14	LD2,DN=*dn*,LIB=*ldn*,NOLIB=*ldn*,LLD,AB=*adn*,MAP=*op*, T=*tra*,NX,DEB=*l*,C=*com*,OVL=*dir*,CNS,NA,USA,L=*ldn*, SET=*val*,E=*n*,I=*sdir*,NOECHO,SECURE, GRANT=sc_1:sc_2:...:sc_n,BC=*bc*,PAD=*pad*,NORED, STK[=*initial size*[:*increment*]],MM[=*initial size*[:*increment*]], MMEPS=*epsilon*,MMLOC=AFTER/BEFORE,VIEW=*level*,CMD=*string*.
7	LIBRARY,DN=dn_1:dn_2...:dn_n,V.
16	LOOP.
7	MEMORY,FL=*fl*,USER/AUTO.
7	MODE,FI=*option*,BT=*option*,EMA=*option*,AVL=*option*,ORI=*option*.
9	MODIFY,DN=*dn*,PDN=*pdn*,ID=*uid*,ED=*ed*,RT=*rt*,R=*rd*,W=*wt*,M=*mn*,NA,ERR, MSG,EXO=ON/OFF,PAM=*mode*,TA=*opt*,TEXT=*text*,NOTES=*notes*, RESIDE=ONLINE/OFFLINE,BACKUP=YES/NO .
8	NOHOLD,GRN=*grn*.
7	NORERUN,ENABLE/DISABLE .
12	NOTE,DN=*dn*,TEXT=*text*.
7	OPTION,LPP=*n*,PN= *p*/ANY ,STAT=ON/OFF .
SR-0060	PASCAL,I=*idn*,L=*ldn*,B=*bdn*,O=*list*,CPU=*list*.
11	PDSDUMP,DN=*dn*,DV=*ldv*,PDN/PDS=*pdn*,ED=*ed*,CW=*cw*,ID=*uid*,US=*usn*,OWN=*ov*, INC=*mm/dd/yy*:'*hh:mm:ss*',ARC=*mm/dd/yy*:'*hh:mm:ss*', TS=*opt*,X,C,D,B,SO,I,O,S.
11	PDSLOAD,L=*ldn*,DN=*dn*,PDN/PDS=*pds*,ED=*ed*,CW=*cw*,ID=*uid*,NID=*nuid*, US=*usn*,OWN=*ov*,NOWN=*nov*,DV=*dvn*,RP,CR,A,I,O,S,NA,SO,TLA.
9	PERMIT,PDN=*pdn*,ID=*uid*,AM=*m*,RP,USER=*ov*,ADN=*adn*,NA,ERR,MSG.
13	PRINT(*expression*)
16	PROC.

Reference	Control Statement
12	QUERY,DN=*ldn*,STATUS=*sym*,POS=*sym*.
8	RELEASE,DN=dn_1:dn_2:...:dn_8,HOLD.
7	RERUN,ENABLE / DISABLE .
11	RESTORE,PDN=*pdn*,ID=*id*,ED=*ed*,OWN=*ov*,M=*m*,TYPE=*type*.
11	RETIRE,PDN=*pdn*,ID=*id*,ED=*ed*,OWN=*ov*,M=*m*,X.
7	RETURN,ABORT.
12	REWIND,DN=dn_1:dn_2:...:dn_8.
7	ROLLJOB.
9	SAVE,DN=*dn*,PDN=*pdn*,ID=*uid*,ED=*ed*,RT=*rt*,R=*rd*,W=*wt*,M=*mn*,UQ,NA,ERR, MSG,EXO=ON / OFF ,PAM=*mode*,ADN=*adn*(*m*),TA=*opt*,TEXT=*text*, NOTES=*notes*,RESIDE=ONLINE / OFFLINE,BACKUP=YES / NO .
SR-0066	SEGLDR,I=*idn*,L=*ldn*,DW=*dw*,CMD='*dirstr*',GO.
7	SET(*symbol*=*expression*)
SG-0056	SID=*adn*,I=*idn*,L=*ldn*,ECH=*edn*,CNT=*n*.
12	SKIPD,DN=*dn*.
12	SKIPF,DN=*dn*,NF=*n*.
12	SKIPR,DN=*dn*,NR=*n*.
12	SKIPU,DN=*dn*,NS=*ns*.
SR-0074	SORT,S=*sdn*[:*sdn*...],M=*mdn*[:*mdn*...],O=*odn*,DIR=*ddn*,L=*ldn*,ECHO, RETAIN,NOVERF.
10	SUBMIT,DN=*dn*,SID=*sf*,DID=*df*,TID=*tid*,DEFER,NRLS.
7	SWITCH,*n*=*x*.
13	SYSREF,X=*xdn*,L=*ldn*.

Reference	Control Statement
7	TARGET,CPU=cpu : EMA / NOEMA : CIGS / NOCIGS : VPOP / NOVPOP : PC / NOPC : READVL / NOREADVL : VRECUR / NOVRECUR : AVL / NOAVL : HPM / NOHPM : STATRG / NOSTATRG : BDM / NOBDM [:BANKS=*banks*][:NUMCPUS=*numcpus*][:IBUFSIZE=*ibufsize*] [:MEMSIZE=*memsize*][:MEMSPEED=*memspeed*][:CLOCKTIM=*clocktim*] [:NUMCLSTR=*numclstr*][:BANKBUSY=*bankbusy*],VERIFY=*HOST / *TARGET.
SG-0055	TEDI,DN=*dn*,I=*idn*,L=*ldn*.
12	UNBLOCK,DN=*ldn*.
12	UNBLOCK,I=*idn*,O=*odn*.
SR-0013	UPDATE,P=*pdn*,I=idn_1:idn_2:...:idn_n, C=*cdn*,CL=cln_1:cln_2:...:cln_n,DEF=def_1:def_2:...:def_n, N=*ndn*,L=*ldn*,E=*edn*,S=*sdn*,*=*m*,/=*c*,DW=*dw*,DC=*dc*,ML=*c*, F Q[=d_1:d_2:...:d_n] ,*options*. Q='d_1,d_2,...,d_j.d_k,...,d_n'
12	WRITEDS,DN=*dn*,NR=*nr*,RL=*rl*.

GLOSSARY

GLOSSARY

A

Abort - To terminate a program or job when a condition (hardware or software) exists from which the program or computer cannot recover.

Absolute address - (1) An address permanently assigned by the machine designator to a storage location. (2) A pattern of characters that identifies a unique storage location without further modification. Synonymous with machine address.

Absolute block - Loader tables consisting of the image of a program in memory. The program image can be saved on a dataset for subsequent reloading and execution.

Address - (1) An identification, as represented by a name, label, or number, for a register, location in storage, or any other data source or destination such as the location of a station in a communication network. (2) Any part of an instruction that specifies the location of an operand for the instruction.

Allocate - To reserve an amount of some resource in a computing system for a specific purpose (usually refers to a data storage medium).

Alphabetic - A character set including, $, %, @, as well as the 26 uppercase letters A through Z.

Alphanumeric - A character set including all alphabetic characters and the digits 0 through 9.

Arithmetic operator - Part of an expression that indicates action to be performed during evaluation of expression; can be symbolic character representing addition, unary plus, subtraction, unary minus, multiplication, or division.

Assemble - To prepare an object language program from a symbolic language program by substituting machine operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic instructions.

B

Base address - The starting absolute address of the memory field length assigned to the user's job. This address is maintained in the Base Address (BA) register. The base address must be a multiple of 20_8.

Binary data - Data that the system treats as a bit string (no character conversion).

$BLD - A dataset on which load modules are placed by a compiler or an assembler unless the user designates some other dataset.

Blank common block - A common block where data cannot be stored at load time. The first declaration need not be the largest. The blank common block is allocated after all other blocks have been processed.

Block - (1) A tape block is a collection of characters written or read as a unit. Blocks are separated by an interblock gap and can be from 1 through 1,048,576 bytes. A tape block and a physical record are synonymous on magnetic tape. (2) In COS blocked format, a block is a fixed number of contiguous characters with a block control word as the first word of the block. The internal block size for the Cray mainframe is 512 words (one sector on disk). In COS manuals, the terms tape block and 512-word block are consistently used to distinguish between the two uses.

Block control word - A word occurring at the beginning of each block in the COS blocked format that identifies the sequential position of the block in the dataset and points forward to the next block control word.

BOT - Beginning-of-tape; the position of the beginning-of-tape reflective marker.

BOV - Beginning-of-volume. See BOT.

BPI - Bits per inch. COS supports the 1600 and 6250 bpi recording densities.

Buffer - A storage device used to compensate for the difference in rate of flow of data, or time of occurrence of events, when transmitting data from one device to another. It is normally a block of memory used by the system to transmit data from one place to another. Buffers are usually associated with the I/O subsystem.

Buffer Memory - A 64-bit memory in the I/O Subsystem common to all I/O Processors.

C

Call - The transfer of control to a specified routine. The called routine normally transfers control back to the caller after the called routine has finished its task.

Catalog - A list or table of items with descriptive data, usually arranged so that a specific kind of information can be readily located.

Channel - A path along which signals can be sent.

Character - A logical unit composed of bits representing alphabetic, numeric, and special symbols. The Cray software processes 8-bit characters in the ASCII character set.

Code - (1) A system of character and rules representing information in a form understandable by a computer. (2) Translation of a problem into a computer language.

Coded data - Data consisting of graphic characters. The default character set is ASCII.

Common block - A block that can be declared by more than one program module during a load operation. More than one program module can specify data for a common block but if a conflict occurs, information from later programs is loaded over previously loaded information. A program can declare no common blocks or as many as 125 common blocks. The two types of common blocks are labeled and blank.

Conditional control statement block - Defines the conditions under which a group of control statements are to be processed. The statements which define the block and conditions are: IF, ELSE, ELSEIF, ENDIF, and EXITIF.

Control statement - The format, consisting of a verb and its parameters, used to control the operating system and access its products. Directives are used to control products.

Control statement input file - A dataset containing valid control statements as its first file.

Controlled device - One of one or more devices or resources that are allocated to jobs on the basis of resource limits and requests.

COS - The Cray Operating System described in this manual.

$CS - A primary control statement input file.

CSP - The Control Statement Processor (CSP) is a system program that executes in the user field. CSP initiates the job, analyzes, and stores the various elements of the control statements (that is, cracks them), processes system verbs, advances the job step by step, processes errors, and ends the job.

D

Data - (1) Information manipulated by or produced by a computer program. (2) Empirical numerical values and numerical constants used in arithmetic calculation. Data is considered to be that which is transformed by a process to produce the evidence of work. Parameters, device input, and working storage are considered data.

Dataset - A quantity of information maintained on mass storage by the Cray Operating System. Each dataset is identified by a symbolic name called a dataset name. Datasets are of two types: temporary and permanent. A temporary dataset is available only to the job that created it. A permanent dataset is available to the system and to other jobs and is maintained across system deadstarts.

Dataset characteristic information - A description of where a dataset resides, how large it is, its permanent name, edition number, information about the creating job and other information.

Dataset name verb - A verb that is the name of a dataset. See local or system dataset name verb.

Deadstart - The process by which an inactive machine is brought up to an operational condition ready to process jobs.

Debug - To detect, locate, and remove mistakes from a routine or malfunction of a computer. Synonymous with troubleshoot.

DEC - Disk Error Correction, a task within the STP portion of COS. DEC can be called by the Disk Queue Manager (DQM) to attempt correction of a disk error.

Delimiter - A character that separates items in a control statement or a directive; synonymous with separator.

Density - See tape density.

Device - A piece of equipment that mechanically contains and drives a recording medium.

Directive - A command used to control a product, such as UPDATE.

Diagnostic - (1) Pertaining to the detection and isolation of a malfunction or a mistake. (2) A message printed when an assembler or compiler detects a program error.

Disposition code - A code used in I/O processing to indicate the disposition to be made of a dataset when its corresponding job is terminated or the dataset is released.

DQM - The Disk Queue Manager is a task within the STP portion of COS. DQM controls the simultaneous operation of disk storage units on CPU I/O channels or on the I/O Subsystem.

Dump - (1) To copy the contents of all or part of a storage device, usually from internal storage, at a given instant of time. (2) The process of performing (1). (3) The document resulting from (1).

E

End-of-data delimiter - Indicates the end of a dataset. In COS blocked format, this is a record control word with a 17_8 in the mode field.

End-of-file delimiter - Indicates the end of a file. (1) In COS blocked format, this is a record control word with a 16_8 in the mode field. (2) On magnetic tape, this is a tapemark.

End-of-record delimiter - Indicates the end of a record. (1) In COS blocked format, this is a record control word with a 10_8 in the mode field. (2) In an ASCII punched deck, this is indicated by the end of each card.

Entry point - A location within a block that can be referenced from program blocks that do not declare the block. Each entry point has a unique name associated with it. The loader is given a list of entry points in a loader table. A block can contain any number of entry points.

An entry point name must be 1 through 8 characters and cannot contain a blank, an *, or a /. Some language processors (for example, FORTRAN) can produce entry point names under more restricted formats due to their own requirements.

EOD - End-of-data on tape. The definition of EOD is a function of whether the tape is labeled or nonlabeled and of the type of operation being performed (input or output). When reading a labeled tape, EOD is returned to the user when an EOF1 trailer label is encountered. When reading a nonlabeled tape, EOD is returned when a tapemark is read on the last volume in the volume list for a particular dataset. When writing a labeled or nonlabeled tape, EOD processing is initiated by a write EOD, rewind, close, or release request.

EOF - End-of-file on tape, sometimes used to mean end of tape trailer group.

EOI - End-of-information; see EOD.

EOT - End-of-tape; a status, set only on a write operation indicating sensing of the end of the tape reflective marker.

EOV - End-of-volume. On output, EOV occurs when end-of-tape status is returned on a write operation. This status occurs when the EOT reflective marker is sensed by the tape device. For input of a labeled tape dataset, EOV occurs when an EOV1 trailer label is read; for input of a nonlabeled dataset, EOV is returned when a tapemark is encountered and the volume list is not exhausted.

Exchange Package - A 16-word block of data in memory which is associated with a particular computer program or memory field. It contains the basic parameters necessary to provide continuity from one execution interval for the program to the next.

EXEC - The COS System Executive (EXEC) is the control center for the operating system. It alone accesses all of memory, controls the I/O channels, and selects the next program to execute.

EXP - The User Exchange Processor (EXP or UEP) is a task within the STP portion of COS. The Exchange Processor task processes all user system action requests and user error exits. The Exchange Processor also handles certain requests from the Job Scheduler (JSH) to initiate or abort a job.

Expression (JCL parameter expression) - A series of characters grouped into operands and operators which are computed as one value during parameter evaluation; should be delimited by parentheses.

External reference - A reference in one program block to an entry point in a block not declared by that program. Throughout the loading process, externals are matched to entry points (this is also referred to as satisfying externals); that is, addresses referencing externals are supplied with the correct address.

F

File - A collection of records in a dataset. In COS blocked format, a file is terminated by a record control word with 16_8 in the mode field.

Filemark - See tapemark.

Foreign label - A special condition that can occur during the label scan at the beginning of a tape. If a NOT CAPABLE status is returned on a BOV label scan, TQM declares the tape to be foreign labeled (FRN) which protects a seven-track tape or a nine-track, 800 bpi tape from being accidentally destroyed.

Formal parameter specifications - Parameters in a procedure definition which identify the character strings within the procedure body that can be substituted during the procedure's evaluation.

Front-end dataset servicing - The act of requesting and receiving information concerning a particular dataset that is known to the front-end computer system. Typical servicing produces the following:

- Direct operator messages concerning tape volume/drive activity
- Required information concerning a dataset, such as what volumes it resides on, the expiration date of each volume, access permissions
- Updated information for a dataset or tape volume or both for use by that computer system

Front-end processor - A computer connected to a Cray Computer System channel. The front-end processor supplies data and jobs to the Cray mainframe and processes or distributes the output from the jobs. Front-end systems are also referred to as stations in Cray publications.

G

Generic resource - A device or group of devices connected to the Cray system which is accessible to user jobs. Devices which constitute a generic resource are characterized by common attributes, such as tape drives with 6250 bpi capability. These devices are subject to regulated access by the system.

H

Heap - An area of memory within the user field managed by user-callable library routines. The heap provides dynamic storage allocation for a single job.

HLM - High limit of memory, the highest relative memory address available to the user for program and data area.

I

$IN - A dataset containing the job control language statements as well as the source input and data for compilers and assemblers, unless the user designates some other dataset (FT05 for example).

In-line procedure - A procedure defined in a control statement file.

Input/Output - (1) Commonly called I/O. To communicate from external equipment to the computer and vice versa. (2) The data involved in such a communication. (3) Equipment used to communicate with a computer. (4) The media carrying the data for input/output.

Integer constant - Specifies an octal value or a decimal value that can be signed as positive or negative.

Interchange format - One of the two ways in which tape datasets can be read or written. Each tape block of data corresponds to a single logical record in COS blocked format. Interchange format is selected by setting DF=IC when a tape dataset is accessed. As far as I/O routines in the Cray mainframe are concerned, interchange datasets must be in COS blocked format because the COS blocked structure (BCWs and RCWs) is used to describe each tape block read or written. This blocked structure allows the user to write or read variable-length tape blocks at high speed with data resolution to the 8-bit byte level of the tape device. The RCW is used to define the tape block length on output and to describe the block length on input. No BCW or RCW ever appears in the data written on the tape.

Interblock gaps - The physical separation between successive tape blocks on magnetic tape.

I/O Subsystem - Part of a CRAY-1 S Series Model S/1200 through S/4400, all models of the CRAY-1 M Series and CRAY X-MP Computer Systems consisting of two to four I/O processors and 1/2, 1, 4, or 8 million words of shared Buffer Memory. The optional tape subsystem is composed of at least one block multiplexer channel, one tape controller, and two tape units. The tape units supported are IBM-compatible nine-track, 200 ips, 1600/6250 bpi devices.

Iterative control statement block - Defines the repeated execution of a series of statements if a condition is satisfied.

J

JCL block control statement - A statement in the control statement file that is part of a group of control statements called a block which specifies an action to be taken by COS; the three types of blocks are: procedure definition, conditional, and iterative.

JCM - The Job Class Monitor is a task within the STP portion of COS. JCM assigns every job to a job class (see JOB statement description) before it enters the input queue.

Job - (1) An arbitrarily defined parcel of work submitted to a computing system. (2) A collection of tasks submitted to the system and treated by the system as an entity. A job is presented to the system as a formatted dataset. With respect to a job, the system is parametrically controlled by the content of the job dataset.

Job Communication Block - The first 200_8 words of the job memory field. This area is used to hold the current control statement and certain job-related parameters. The area is accessible to the user, the operating system, and the loader for inter-phase job communication.

Job control statement - Any of the statements used to direct the operating system in its functioning, as compared to data, programs, or other information needed to process a job but not intended directly for the operating system itself.

Job deck - See job.

Job input dataset - A dataset named $IN on which the control statements of the job deck are maintained. This consists of programs and data referenced by various job steps. The user can manipulate the dataset like any other dataset (excluding write operations).

Job output dataset - Any of a set of datasets recognized by the system by a special dataset name (for example, $OUT, $PLOT, and $PUNCH), which becomes a system permanent dataset at job end and is automatically staged to a front-end computer for processing.

Job step - A unit of work within a job, such as source language compilation or object program execution.

JSH - The Job Scheduler (JSH) is a task within the STP portion of COS. The JSH task initiates the processing of a job, selects the currently active job, manages job roll-in and roll-out, and terminates a job.

K

Keyword parameter - A string of 1 through 8 alphanumeric characters that consists of a keyword followed by one or more values; identified by its form rather than by its position in the control statement.

L

$LOG - See logfile.

Label group - A group of tables that precede and follow the user data at dataset and/or volume boundary conditions. The label group describes the characteristics of the volume or dataset.

Labeled common - A common block into which data can be stored at load time.

Library - A dataset composed of sequentially organized records and files. The last file of the library contains a library directory. The rest of the files and records, known as entries, can consist of processed procedure definitions and/or relocatable modules. The directory gives a listing of entry names with their associated characteristics.

Library-defined verb - A 1- through 8-character name of a program or procedure definition residing in a library that is a part of the current library searchlist.

Limit address - The upper address of a memory field. This address is maintained in the limit address (LA) register.

Literal - A symbol which names, describes, or defines itself and not something else that it might represent.

Literal constant - A string of 1 through 8 characters delimited with apostrophes whose ordinal numbers are in the range 040_8 through 176_8; value of a character constant corresponds to the ASCII character codes positioned within a 64-bit word; alignment indicated can be left- or right-adjusted and zero-filled or left-adjusted and space-filled; apostrophes remain as part of value.

Literal string - A string delimited with apostrophes which are normally not treated as part of the value, except with JCL block control statements which treat the apostrophes as part of the string value.

Loader tables - The form in which code is presented to the loader. Loader tables are generated by compilers and assemblers according to loader requirements. The tables contain information required for loading such as type of code, names, types and lengths of storage blocks, data to be stored.

Loading - The placement of instructions and data into memory so that it is ready for execution. Loader input is obtained from one or more datasets and/or libraries. Upon completion of loading, execution of the program in the job's memory field is optionally initiated. Loading can also involve the performance of load-related services such as generation of a loader map, presetting of unused memory to a user-specified value, and generation of overlays.

Load point - See BOT.

Local dataset - A temporary or permanent dataset accessible by the user.

Local dataset name verb - A verb that is the name of a local dataset consisting of an alphabetic character followed by 1 through 6 alphanumeric characters. Requests that COS load and execute an absolute binary program from the first record of the named dataset.

Logfile - During the processing of the job, a special dataset named $LOG is maintained. At job termination, this dataset is appended to the $OUT file for the job. The job logfile serves as a time-ordered record of the activities of the job: all control statements processed by the job, significant information such as dataset usage, all operator interactions with a job, and errors detected during processing of the job.

Logical operator - Represents logical function performed on operands on a bit-by-bit basis, returning a 64-bit result; functions are: inclusive OR, intersection, exclusive OR, unary complement.

M

Macro instruction - An instruction in a source language that is equivalent to a specified sequence of machine instructions.

Magnetic tape - A tape with a magnetic surface on which data can be stored by selective polarization of portions of that surface.

Mainframe - The central processor of the computer system. It contains the arithmetic unit and special register groups. It does not include input, output, or peripheral units and usually does not include internal storage. Synonymous with central processing unit (CPU).

Mass storage - The storage of a large amount of data that is also readily accessible to the central processing unit of a computer.

MEP - The Error Message Processor (MEP) is a task within the STP portion of COS. Error messages are passed from the System Executive (EXEC) to the Log Manager (MSG) through the Error Message Processor.

Migrated dataset - A dataset that has been moved from on line to a back-up medium by the system space manager. A migrated dataset is automatically returned when it is specified on the ACCESS control statement.

MSG - The Log Manager (MSG) is a task within the STP portion of COS. MSG writes messages in the system and user logfiles.

Multiprocessing - Use of several computers to logically or functionally divide jobs or processes; and to execute various programs or segments asynchronously and simultaneously.

Multiprogramming - A technique for handling multiple routines or programs simultaneously by overlapping or interleaving their execution, that is, permitting more than one program to time-share machine components.

Multitasking - A type of multiprocessing in which more than one task may be simultaneously active for a single job.

N

Nesting - Including a block of statements of one kind into a larger block of statements of the same kind, such as an iterative block within a larger iterative block.

Not Capable - A tape status indicating the reel currently mounted cannot be read by the control unit and drive. The Not Capable status would be returned if an 800 bpi tape were mounted on a device that supported only 1600 and 6250 bpi, for example. Since it is not possible to read a Not Capable tape to verify label type and contents, COS rejects (unloads) all tapes that return a Not Capable status.

O

On-line dataset - A dataset residing on Cray disk. It is catalogued in the DSC.

Operand - A character string in an expression that is operated on during evaluation; types are integer constant, literal constant, symbolic variable, and subexpresion.

Operating system - (1) The executive, monitor, utility, and any other routines necessary for the performance of a computer system. (2) A resident executive program that automates certain aspects of machine operation, particularly as they relate to initiating and controlling the processing of jobs.

Operator - A symbolic representation indicating the action to be performed in an expression; types are arithmetic, relational, and logical operators.

$OUT - A dataset that contains the list output from compilers and assemblers unless the user designates some other dataset. At job end, the job logfile is added to the $OUT dataset and the dataset is sent to a front-end computer.

Overlaying - A technique for bringing routines into memory from some other form of storage during processing so that several routines will occupy the same storage locations at different times. Overlaying is used when the total memory requirements for instructions exceeds the available memory.

OVM - The Overlay Manager (OVM) is a part of the STP portion of COS and manages the use of the overlaid portion of COS itself.

P

$PROC - A dataset to which in-line procedure definitions are written.

Parallel processing - Simultaneous or approximately simultaneous processing of jobs, job steps, programs, and parts of programs.

Parameter - A quantity in a control statement which can be given different values when the control statement is used for a specific purpose or process.

Parcel - A 16-bit portion of a word which is addressable for instruction execution but not for operand references. An instruction occupies 1 or 2 parcels; if it occupies 2 parcels, they can be in separate words.

Parenthetic string - A string delimited with parentheses instead of apostrophes; parentheses are treated as part of the string when evaluated except when preceded by an initial, parameter, equivalence, or concatenation separator character.

PDM - The Permanent Dataset Manager (PDM) is a task within the STP portion of COS and provides the means for creating, accessing, deleting, maintaining, and auditing disk-resident permanent datasets.

Permanent dataset - A dataset known to the operating system as being permanent; the dataset survives deadstart.

Positional parameter - A parameter that must appear in a precise position relative to the separators in the control statement.

Procedure - A named sequence of control statements, data, or both that is saved in a library for processing at a later time when activated by a call to its name by a calling statement; provides the capability of replacing values within the procedure with other values.

Procedure definition - The definition of a procedure saved in a library to be called for processing at a later time; if defined in a job control statement is called an in-line procedure definition.

Program - (1) A sequence of coded instructions that solves a problem. (2) To plan the procedures for solving a problem. This can involve analyzing the problem, preparing a flow diagram, providing details, developing and testing subroutines, allocating storage, specifying I/O formats, and incorporating a computer run into a complete data processing system.

Program block - The block within a load module usually containing executable code. It is automatically declared for each program (though it can be zero-length). It is local to the module; that is, it can be accessed from other load modules only through use of external symbols. Data placed in a program block always comes from its own load module.

Program name - Also referred to as IDENT name or deck name, the name contained in the loader PDT table at the beginning of each load module.

Program library - (PL) The base dataset used by the UPDATE utility. This dataset consists of one or more specially formatted files, each ending with an EOF.

R

Record - A group of contiguous words or characters related to each other by virtue of convention. A record is fixed or variable length. (1) In COS blocked format, a record ends with a record control word with 10_8 in the mode field. (2) For a listable dataset, each line is a record. (3) For a binary load dataset, each module is a record.

Relational operator - An operator that indicates the comparison to be performed between the operands in an expression (-1 for a TRUE result and 0 for a FALSE result); types are equal, not equal, less than, greater than, less than or equal, and greater than or equal.

Relative address - An address defined by its relationship to a base address (BA) such that the base address has a relative address of 0.

Relocatable address - An address presented to the loader in such a form that it can be loaded anywhere in the memory field. A relocatable address is defined as being relative to the beginning address of a load module program block or common block.

Relocatable module - This is the basic program unit produced by a compiler or assembler. CAL produces a relocatable module from source statements delineated by IDENT and END. In FORTRAN, the corresponding beginning statements are PROGRAM, SUBROUTINE, BLOCK DATA, or FUNCTION. The corresponding end statement is END.

A relocatable module consists of several loader tables that define blocks, their contents, and address relocation information.

Relocate - In programming, to move a routine from one portion of internal storage to another and to adjust the necessary address references so that the routine can be executed in its new location. Instruction addresses are modified relative to a fixed point or origin. If the instruction is modified using an address below the reference point, relocation is negative. If addresses are above the reference point, relocation is positive. Generally, a program is loaded using positive relocation.

Resource allocation - The number of allocation units consumed by a disk dataset. The resource allocation limit is defined by the JOB control statement.

Retired dataset - A dataset that has been moved from on line to a back-up medium by the RETIRE control statement. To make the dataset accessible, the user must specify it on the RESTORE control statement.

S

SCP - The Station Call Processor (SCP) is a task within the STP portion of COS and handles communications with front-end computer systems.

Sector - A physical area on disk equivalent to 512 Cray words. In COS blocked format, a block is also 512 contiguous words with a block control word as the first word of the block. Therefore the internal block size for the Cray is equivalent to one Cray disk sector. This is the unit of data transfer between the Cray mainframe and the I/O Subsystem.

SPM - The System Performance Monitor (SPM) is the task within COS that collects and reports statistics about COS system performance.

STG - Stager (STG) task is a subtask of SCP within the STP portion of COS that handles dataset transfers between the Cray mainframe and its front-end processors.

STP - The System Task Processor (STP) is the main portion of the COS operating system and consists of tables, a set of routines called tasks, and some reentrant routines common to all tasks.

Separator - Synonym for delimiter.

String - A sequence of characters delimited by apostrophes or parentheses which is taken literally as a parameter value; see literal string and parenthetic string.

Subexpression - An expression that is evaluated so that its result becomes an operand.

Substitution parameters - Parameters on procedure definition prototype statement or procedure calling statement which provide replacement values to be substituted during evaluation for strings flagged within the procedure body.

Symbolic variable - A string of 1 through 8 alphanumeric characters, beginning with an alpha character that represents values maintained by COS and/or the user.

System dataset name verb - A verb that is the name of a system-defined dataset in the System Directory Table (SDR); consists of an alphabetic character which can be followed by 1 through 6 alphanumeric characters.

System logfile - A permanent dataset named $SYSTEMLOG.

System verb - Requests that COS perform a function; consists of an alphabetic character which can be followed by 1 through 6 alphanumeric characters

T

Table - A collection of data, each item being uniquely identified either by some label or by its relative position.

Tape block - A group of contiguous characters recorded on and read from magnetic tape as a unit.

Tape control unit - A piece of equipment connected to a block multiplexer channel that provides the capability for controlling the operation of one or more tape devices. Up to four control units can be combined to drive a maximum of 16 tape devices. The control units are cross connected to all devices. Such a configuration is called a 4x16 (four by sixteen). If one control unit were to be connected to three devices, it would be referred to as a 1x3 configuration.

Tape density (bpi) - The number of bits per inch on magnetic tape. COS supports 6250 bpi and 1600 bpi.

Tape format - The way tape datasets are read or written. In *interchange format,* each tape block of data corresponds to a single logical record in COS blocked format. In *transparent format,* each tape block is a fixed multiple of 512 words based on the density of the tape.

Tape volume - A reel of magnetic tape.

Tapemark - A special hardware bit configuration recorded on magnetic tape. It indicates the boundary between combinations of datasets and labels. It is sometimes called a filemark.

Task - A subprogram or uniquely named process that can have code and data areas in common with other tasks of the same job. A task is a unit of computation that can be scheduled independently of other tasks in the same job step. A job step can consist of a single task, or it may consist of several tasks running in parallel with each other.

Temporary dataset - A dataset which is not permanent and is available only to the job that created it.

Time slice - The maximum amount of time during which the CPU can be assigned to a job without reevaluation as to which job should have the CPU next.

Timestamp - A 1-word binary number that represents specific date and time. Timestamps are expressed as the number of (nanosecond/1.024) units between the date and time in question and midnight, 1 January 1973. Timestamps appear in machine-independent tables used by the operating system.

TQM - The Tape Queue Manager (TQM) is the System Task Processor (STP) task that manages tape I/O between one or more user jobs and the I/O Subsystem.

Track - The smallest amount of disk space which can be allocated or deallocated by COS. A track is equivalent to 18 sectors for DD-19, DD-29, Buffer Memory and Solid-state storage device.

Transparent format - One of two ways tape datasets are read or written. Each tape block is a fixed multiple of 512 words. Transparent format is the default tape dataset format and is designated by setting DF=TR when accessing a tape dataset. This format produces a fixed-length block dataset (16384 bytes at 1600 bpi or 32768 bytes at 6250 bpi) that can be a COS blocked or unblocked dataset as far as any I/O routines are concerned. The tape subsystem merely takes four (1600 bpi) or eight (6250 bpi) sectors and processes them as one physical tape block. When a short block is read, it is considered to be EOD.

U

UEP - User Exchange Processor. See EXP.

Unit record device - A device such as a card reader, printer, or card punch for which each unit of data to be processed is considered a record.

Unload - To remove a tape from ready status by rewinding beyond the load point. The tape is then no longer under control of the computer.

Unsatisfied external - An external reference for which the loader has not yet loaded a module containing the matching entry point.

User field - A portion of memory containing instructions and data defined for a specific job. Field limits are defined by the base address and the limit address. A program cannot execute outside of its field nor refer to operands outside of its field.

User logfile - A dataset named $LOG created for a job when it is initiated by the Job Scheduler.

V

Verb - The first nonblank field of a control statement; specifies the action to be taken by COS during control statement evaluation.

Volume - A physical unit of storage media that can be dismounted from a storage device, for example, a reel of magnetic tape.

Volume identifier - Up to 6 alphanumeric characters used to identify a physical reel of tape. On labeled tapes, the volume identifier is actually recorded on tape in the volume header label. Volume identifier is synonymous with volume serial number.

VSN - Volume serial number (obsolete term). See volume identifier.

W

Word - A group of bits between boundaries imposed by the computer. Word size must be considered in the implementation of logical divisions such as character. The word size of the CRAY X-MP and CRAY-1 computers is 64 bits.

INDEX

INDEX

READER'S COMMENT FORM

COS Version 1 Reference Manual SR-0011 O

Your reactions to this manual will help us provide you with better documentation. Please take a moment to check the spaces below, and use the blank space for additional comments.

1) Your experience with computers: ____ 0-1 year ____1-5 years ____5+ years
2) Your experience with Cray computer systems: ____0-1 year ____ 1-5 years ____5+ years
3) Your occupation: ____ computer programmer ____ non-computer professional
   ____ other (please specify): _____
4) How you used this manual: ____ in a class ____as a tutorial or introduction ____ as a reference guide
   ____ for troubleshooting

Using a scale from 1 (poor) to 10 (excellent), please rate this manual on the following criteria:

5) Accuracy _____
6) Completeness ____
7) Organization ____
8) Physical qualities (binding, printing) ____
9) Readability ____
10) Amount and quality of examples ____

Please use the space below, and an additional sheet if necessary, for your other comments about this manual. If you have discovered any inaccuracies or omissions, please give us the page number on which the problem occurred. We promise a quick reply to your comments and questions.

Name _____
Title _____
Company _____
Telephone _____
Today's Date _____

Address _____
City _____
State/ Country _____
Zip Code _____

CUT ALONG THIS LINE

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO 6184 ST PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

Attention: PUBLICATIONS
1345 Northland Drive
Mendota Heights, MN 55120

FOLD

STAPLE

READER'S COMMENT FORM

COS Version 1 Reference Manual SR-0011 O

Your reactions to this manual will help us provide you with better documentation. Please take a moment to check the spaces below, and use the blank space for additional comments.

1) Your experience with computers: ____ 0-1 year ____1-5 years ____5+ years
2) Your experience with Cray computer systems: ____0-1 year ____ 1-5 years ____5+ years
3) Your occupation: ____ computer programmer ____ non-computer professional
   ____ other (please specify): _____
4) How you used this manual: ____ in a class ____as a tutorial or introduction ____ as a reference guide
   ____ for troubleshooting

Using a scale from 1 (poor) to 10 (excellent), please rate this manual on the following criteria:

5) Accuracy _____
6) Completeness ____
7) Organization ____
8) Physical qualities (binding, printing) ____
9) Readability ____
10) Amount and quality of examples ____

Please use the space below, and an additional sheet if necessary, for your other comments about this manual. If you have discovered any inaccuracies or omissions, please give us the page number on which the problem occurred. We promise a quick reply to your comments and questions.

Name _____ Address _____
Title _____ City _____
Company _____ State/ Country _____
Telephone _____ Zip Code _____
Today's Date _____

CUT ALONG THIS LINE

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO 6184 ST PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

Attention: PUBLICATIONS
1345 Northland Drive
Mendota Heights, MN 55120

FOLD

STAPLE